ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ / ΜΕΣΟΛΟΓΓΙ

Πτυχιακή εργασία

ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΜΙΚΡΟΕΦΑΡΜΟΓΗΣ JAVA ΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥΣ ΣΚΟΠΟΥΣ

Αθανάσιος Παπαζαχαρίας, Α.Μ. 15028

Ευανθία Τζώτζου, Α.Μ. 15031

Μεσολόγγι 2015

ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ / ΜΕΣΟΛΟΓΓΙ

Πτυχιακή εργασία

ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΜΙΚΡΟΕΦΑΡΜΟΓΗΣ JAVA ΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥΣ ΣΚΟΠΟΥΣ

Αθανάσιος Παπαζαχαρίας, Α.Μ. 15028

Ευανθία Τζώτζου, Α.Μ. 15031

Επιβλέπων καθηγητής

Ιωάννης Ντόκας

Μεσολόγγι 2015

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Διοίκησης Επιχειρήσεων/Μεσολογγίου του ΤΕΙ Δυτικής Ελλάδας δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Στις οικογένειές μας

Ευχαριστίες

Θα θέλαμε να εκφράσουμε τις ευχαριστίες μας στον υπεύθυνο καθηγητή μας κ. Ιωάννη Ντόκα, για την αποδοχή του και την υποστήριξη που μας παρείχε καθώς και τις σημαντικές συμβουλές του και την καθοδήγησή μας στην εκπόνηση της παρούσας εργασίας.

Οφείλουμε όμως ένα μεγάλο ευχαριστώ και στις οικογένειές μας για την συμπαράστασή τους και την κατανόησή τους κατά τη διάρκεια όλων των σπουδών μας.

Ο Αθανάσιος Παπαζαχαρίας επιθυμεί να ευχαριστήσει την σύζυγό του Ειρήνη για την κατανόηση και τη βοήθεια που του παρείχε, το γιο του Δημήτριο για τις αμέτρητες ώρες που θα μπορούσαν να είχαν περάσει μαζί αλλά και τον νεογέννητο γιό τους για τις αγκαλιές που του στέρησε το πρώτο μήνα της ζωής του. Τέλος θέλει να ευχαριστήσει από καρδιάς και την κα. Ζαρζάνη Βασιλική που ήταν εκεί για την οικογένειά του όταν ό ίδιος δεν μπορούσε.

Η Τζώτζου Ευανθία επιθυμεί να ευχαριστήσει τον σύζυγό της Αλέξανδρο για την κατανόηση και την υποστήριξη που της παρείχε, τις κόρες της Ιουλία, Βασιλική, Αικατερίνη και Μαρία στις οποίες οφείλει πολλές ώρες απουσίας από την καθημερινότητά τους, καθώς και τον πεθερό της Νικόλαο και την πεθερά της Ιουλία για τη συμπαράστασή τους.

Αθανάσιος Παπαζαχαρίας

Ευανθία Τζώτζου

ΠΕΡΙΛΗΨΗ

Αυτή η εργασία είναι μια προσπάθεια να καταδειχθεί η δυνατότητα ανάπτυξης εκπαιδευτικών μικροεφαρμογών Java από εκπαιδευτικούς που έχουν ένα βασικό υπόβαθρο στον αντικειμενοστραφή προγραμματισμό. Ειδικότερα μια μικροεφαρμογή σχεδιάστηκε και αναπτύχθηκε με απώτερο σκοπό να χρησιμοποιηθεί για τη διδασκαλία ενός διδακτικού αντικειμένου σε συγκεκριμένο τύπο σχολείου. Η μικροεφαρμογή αυτή οπτικοποιεί το θεώρημα της δειγματοληψίας και επιτρέπει στους χρήστες να αλληλεπιδράσουν με αυτή, ορίζοντας τόσο τις συχνότητες του σήματος εισόδου, όσο και τη συχνότητα δειγματοληψίας. Οι μαθητές καλούνται να εξαγάγουν μόνοι τους το θεώρημα της δειγματοληψίας κερδίζοντας με αυτόν τον τρόπο την γνώση μέσω της διερευνητικής μάθησης.

Χρησιμοποιώντας τα κατάλληλα εργαλεία και με βασικές γνώσεις στον αντικειμενοστραφή προγραμματισμό, οι σπουδαστές σχεδίασαν και υλοποίησαν την μικροεφαρμογή. Αρχικός στόχος ήταν να χρησιμοποιηθεί από τους μαθητές της Β΄ και της Γ΄ τάξης του τμήματος Ηλεκτρονικής του ΕΠΑ.Λ. Μεσολογγίου, οι οποίοι συναντούν το θεώρημα αυτό σε 2 διαφορετικά μαθήματα. Η σχεδίαση παρουσιάζεται με αρκετά επεξηγηματικό τρόπο ώστε να μπορεί να λειτουργήσει ως οδηγός για την ανάπτυξη παρόμοιων μικροεφαρμογών.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ΕΠΑ.Λ.:	Επαγγελματικό Λύκειο
Τ.Π.Ε:	Τεχνολογίες της Πληροφορίας και των Επικοινωνιών
JVM:	Java Virtual Machine
Α.Π.Σ.:	Αναλυτικό Πρόγραμμα Σπουδών
ADC:	Analog to Digital Converter
DAC:	Digital to Analog Converter
S/H:	Sample and Hold
AM:	Amplitude Modulation
FM:	Frequency Modulation
GUI:	Graphical User Interface
API:	Application Programming Interface

ΑΠΟΔΟΣΗ ΟΡΩΝ

Java applet	Μικροεφαρμογή Java
Compiler	Μεταγλωττιστής
Interpreter	Διερμυνευτής
Bytecode	Ψηφιοκώδικας
Browser	Φυλλομετρητής
Plugin	Πρόσθετο
Aliasing	Αναδίπλωση
Slider	Ολισθητής
Interface	Διεπαφή
Pane	Επίπεδο
Container	Περιέχων
Layout	Χωροθέτηση
Event	Γεγονός
Listener	Ακροατής
Class	Κλάση
Constructor	Δημιουργός

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια στην Ελλάδα (και νωρίτερα σε άλλες χώρες) υπάρχει μια ραγδαία ανάπτυξη των Τεχνολογιών της Πληροφορίας και των Επικοινωνιών (ΤΠΕ). Η ανάπτυξη αυτή επηρεάζει κάθε πτυχή της ζωής των ανθρώπων. Από προσωπικό επίπεδο έως συλλογικό και από απλά θέματα καθημερινότητας έως εξειδικευμένα θέματα. Η εκπαίδευση, ως ζωτικό στοιχείο της κοινωνίας και της ανάπτυξης αλλά και κομμάτι της κοινωνικής ζωής των παιδιών και των εφήβων δεν ήταν δυνατόν αλλά και ούτε θεμιτό να μην επηρεαστεί από αυτή την εξέλιξη. Σε μια εποχή όπου η γνώση παράγεται με γοργούς ρυθμούς και η αγορά εργασίας γίνεται πιο απαιτητική ο ρόλος του εκπαιδευτικού δεν μπορεί να είναι αυτός του ανθρώπου που κατέχει τη γνώση την οποία πρέπει να την μεταδώσει στους μαθητές του. Είναι περισσότερο ο ρόλος του καθοδηγητή ο οποίος θα βοηθήσει τον μαθητή να ανακαλύψει τη γνώση και με αυτόν τον τρόπο να την κάνει κτήμα του. Η εποχή του μαυροπίνακα αντικαθίσταται πλέον από την εποχή της οθόνης του υπολογιστή. Χωρίς να σημαίνει ότι το πρώτο δεν είναι χρήσιμο, το δεύτερο όμως είναι απαραίτητο για τον μαθητή ώστε να καλλιεργήσει τις δεξιότητές του και τα ιδιαίτερα χαρακτηριστικά του.

Οι εκπαιδευτικοί λοιπόν είναι αυτοί που πρώτοι πρέπει να εξοικειωθούν με την χρήση των ΤΠΕ ώστε να μπορέσουν να βοηθήσουν τους μαθητές τους να τις χρησιμοποιήσουν για όφελός τους. Και πράγματι έχουν γίνει αρκετές επιμορφώσεις στους εκπαιδευτικούς για το αντικείμενο αυτό και η χρήση των ΤΠΕ στην εκπαιδευτική διαδικασία έχει αυξηθεί κατά πολύ. Οι εκπαιδευτικοί έχουν πλέον στη διάθεσή τους μια πληθώρα εργαλείων ώστε να βελτιώσουν την εκπαιδευτική διαδικασία και να βοηθήσουν τους μαθητές τους να ανακαλύψουν τη γνώση. Εποπτικά μέσα διδασκαλίας, εξομοιώσεις, πλατφόρμες μάθησης είναι μόνο μερικά από αυτά. Ο παγκόσμιος ιστός αποτελεί μια τεράστια εργαλειοθήκη όπου μπορεί ο εκπαιδευτικός να αναζητήσει και να βρει το καταλληλότερο εργαλείο που θα τον βοηθήσει να κάνει καλύτερα τη δουλειά του. Και υπάρχουν πολλά.

Έχει γίνει πολλή δουλειά από πολλούς (εκπαιδευτικούς και μη) με αποτέλεσμα να υπάρχει διαθέσιμο πολύ υλικό. Όμως αυτό δεν σημαίνει ότι ο εκπαιδευτικός δεν μπορεί ή δεν πρέπει να δημιουργήσει και αυτός υλικό το οποίο θα τον βοηθήσει στη διδασκαλία. Υπάρχουν πολλές δυνατότητες για τους εκπαιδευτικούς να αναπτύξουν το δικό τους υλικό, ό,τι και αν είναι αυτό. Ανάλογα με τις δεξιότητές τους μπορούν να φτιάξουν δικά τους φύλλα εργασίας, σημειώσεις, παρουσιάσεις, online μαθήματα και άλλα πολλά. Μέρος του εκπαιδευτικού

υλικού που μπορεί να αναπτυχθεί αποτελούν και οι μικροεφαρμογές Java (Java applets). Αν και φαντάζει κάτι δύσκολο, στην ουσία είναι κάτι εφικτό για όσους διαθέτουν βασικές γνώσεις προγραμματισμού και καλή διάθεση. Η παρούσα εργασία θέλει να καταδείξει αυτό ακριβώς το σημείο. Τον βαθμό δυσκολίας (ή ευκολίας) τον οποίο παρουσιάζει η ανάπτυξη μιας μικροεφαρμογής Java. Σκοπός της εργασίας λοιπόν είναι η σχεδίαση και η ανάπτυξη μιας μικροεφαρμογής σε Java για εκπαιδευτικούς σκοπούς. Θα μπορούσε κάλλιστα η μικροεφαρμογή να ήταν κάτι διαφορετικό, π.χ. ένα παιχνίδι, αλλά προτιμήθηκε κάτι πιο πρακτικό και το οποίο θα μπορούσε να αξιολογηθεί με τη χρήση του σε Σχολεία.

1. Μικροεφαρμογές Java (Java applets)

1.1. Εισαγωγή

Οι τεχνολογικές εξελίξεις γενικά και η εξάπλωση του παγκόσμιου ιστού ειδικότερα, συνέβαλαν στη δημιουργία νέων τρόπων ανάπτυξης και διανομής λογισμικού. Αποτέλεσμα αυτής της εξέλιξης ήταν και η δημιουργία της γλώσσας αντικειμενοστραφούς προγραμματισμού με το όνομα Java. Η Java εισήχθηκε το 1995 από την εταιρεία Sun Microsystems σαν διάδοχος της γλώσσας Oak που ανέπτυσσε η ίδια ομάδα. Στόχος της ήταν η ανάπτυξη μιας γλώσσας προγραμματισμού που θα έτρεχε σε ετερογενή δικτυακά περιβάλλοντα (Γεωργίου, 2008) (ΕΜΠ, 2013). Η Java διαθέτει συγγένεια με την C++, εφόσον στα αρχικά στάδια ανάπτυξής της επιχειρήθηκε η δημιουργία ενός κατάλληλου εργαλείου που θα μπορούσε να χρησιμοποιηθεί ως πλατφόρμα ανάπτυξης λογισμικού για έξυπνες μικροσυσκευές κάνοντας χρήση της C++. Έτσι, διατήρησε μεν αρκετά από τα χαρακτηριστικά της C++, εξελίχθηκε δε με πρόσθετα χαρακτηριστικά που απαιτούνταν για τον προγραμματισμό των μικροσυσκευών.

Σήμερα, η γλώσσα προγραμματισμού Java αποτελεί μια σύγχρονη αντικειμενοστραφή γλώσσα με χαρακτηριστικά τα οποία δεν συναντώνται σε άλλες γλώσσες προγραμματισμού. Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία λειτουργικού συστήματος και πλατφόρμας. Η ασφάλεια εκτέλεσης των προγραμμάτων, όπως η δημιουργία applets στο διαδίκτυο, είναι άλλο ένα πλεονέκτημα της Java. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (σύντομα θα τρέχουν και σε Playstation καθώς και σε άλλες κονσόλες παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.

Σημαντικά χαρακτηριστικά της Java αποτελούν (Digital Academy, 2013):

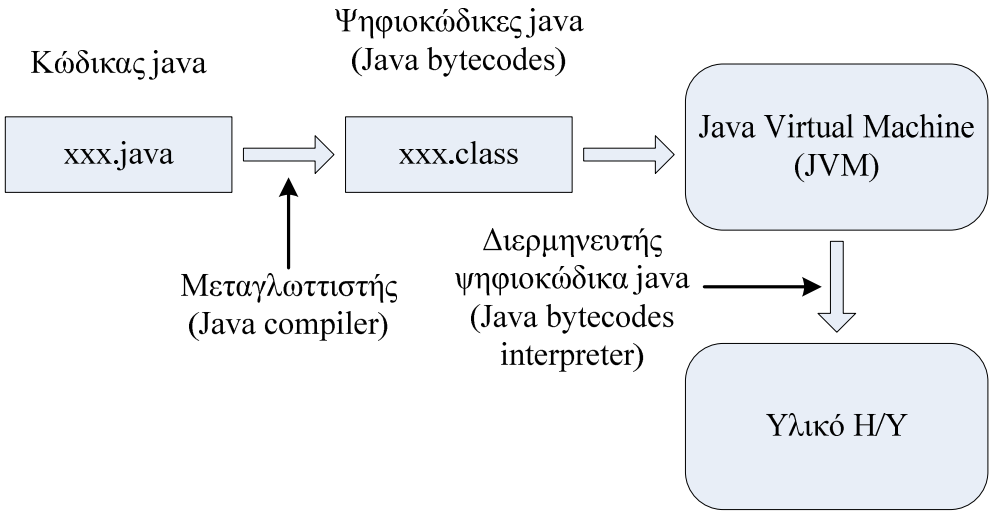
- είναι αμιγώς αντικειμενοστραφής (δεν περιέχει διαδικαστικές δομές)
- είναι ανεξάρτητη πλατφόρμας (το ίδιο πρόγραμμα που γράφηκε για Windows θα τρέξει και σε Linux)
- είναι σχετικά απλή (ή τουλάχιστον πιο απλή από τη C++ καθώς δεν έχει δείκτες και η διαχείριση της μνήμης γίνεται από την γλώσσα και όχι τον χρήστη)

- είναι γλώσσα υψηλού επιπέδου
- είναι μεταγλωττιζόμενη (μπορεί να γίνει compile) αλλά και διερμηνευόμενη (μπορεί να γίνει interpret)
- παρέχει έμφυτη υποστήριξη πολυμέσων
- παρέχει έμφυτη υποστήριξη πολυνηματικών εφαρμογών
- είναι κατάλληλη για προγραμματισμό δικτυακών εφαρμογών
- είναι δυναμική

κ.α.

Τέλος, αξίζει να σημειωθεί ότι τα προγράμματα Java εκτελούνται στην εικονική μηχανή Java όχι στον πραγματικό Η/Υ. Τα προγράμματα Java γράφονται για την εικονική μηχανή Java και η μηχανή αυτή λειτουργεί σαν ένα firewall μεταξύ Η/Υ και προγραμμάτων Java παρέχοντας έτσι μεγαλύτερη ασφάλεια. Ένα πρόγραμμα Java δεν έχει άμεση προσπέλαση στους πόρους του Η/Υ (συσκευές εισόδου/εξόδου, σύστημα αρχείων) παρά μόνο στην εικονική μηχανή Java (JVM) .

Στο παρακάτω σχήμα φαίνεται το διάγραμμα μεταγλώττισης – εκτέλεσης κώδικα σε Java.

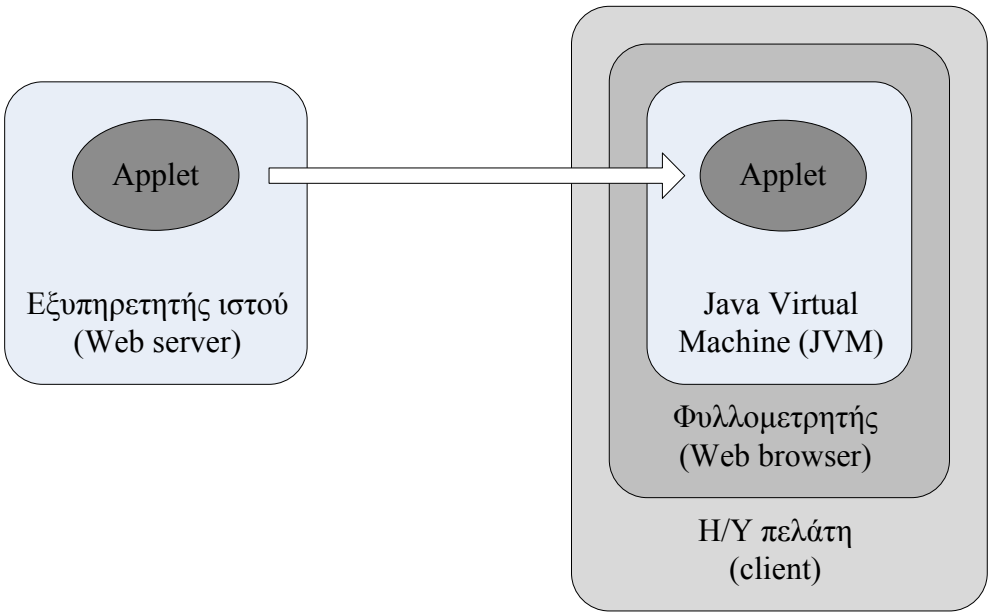


Εικόνα 1.1 Διάγραμμα μεταγλώττισης – εκτέλεσης κώδικα σε Java

1.2. Μικροεφαρμογές Java applets

Μια μικροεφαρμογή (applet) είναι ένα πρόγραμμα της Java που εκτελείται μέσα από ένα οποιοδήποτε φυλλομετρητή του ιστού (web browser) του internet. Τα αντικειμενικά προγράμματα των μικροεφαρμογών μπορούν να συμπεριλαμβάνονται σε έγγραφα Γλώσσας Μορφοποίησης Υπερκειμένου (Hypertext Markup Language (HTML) documents), δηλαδή σε ιστοσελίδες (web pages).

Όλοι οι σημερινοί φυλλομετρητές έχουν τη δυνατότητα εκτέλεσης μικροεφαρμογών της Java. Η διαδικασία εκτέλεσης είναι αρκετά απλή. Όταν ο φυλλομετρητής φορτώσει μια ιστοσελίδα που περιέχει μια μικροεφαρμογή τότε η μικροεφαρμογή κατεβαίνει τοπικά στον υπολογιστή του χρήστη (client) από έναν απομακρυσμένο υπολογιστή (server) και ξεκινά η εκτέλεσή της.



Εικόνα 1.2 Εκτέλεση Applets

Γενικά στην πληροφορική ο όρος applet αναφέρεται σε μια μικρή εφαρμογή η οποία εκτελεί μια πολύ συγκεκριμένη λειτουργία. Συχνά χρησιμοποιείται στο πλαίσιο ενός μεγαλύτερου προγράμματος, κυρίως ως plugin. Χαρακτηριστικά παραδείγματα από τέτοιες μικροεφαρμογές αποτελούν τα movie player και media player applets τα οποία αναπαράγουν αρχεία video σε έναν φυλλομετρητή (Γαβαλάς Δ.).

Επιπρόσθετα, ο όρος applet αναφέρεται και στα Java applets, τα οποία είναι προγράμματα γραμμένα στη γλώσσα προγραμματισμού Java. Τα Java applets συνήθως περιλαμβάνονται σε ιστοσελίδες και σε συνδυασμό με το χαρακτηριστικό της ανεξαρτησίας της από την πλατφόρμα εκτέλεσης, κατέστησαν τη Java πολύ δημοφιλή. Συγχρόνως, παρέχουν web εφαρμογές με αλληλεπιδραστικά χαρακτηριστικά τα οποία δεν μπορεί να δώσει η (X)HTML και μπορούν να εκτελεστούν από φυλλομετρητές (browsers) που τρέχουν σε πολλές πλατφόρμες (π.χ. Windows, Unix, Mac Os και Linux). Ουσιαστικά ο ενδιάμεσος κώδικας .class (εικόνα 1.1) ο οποίος παράγεται είναι αυτός που είναι ανεξάρτητος πλατφόρμας. Βέβαια το δυνατό σημείο της Java αποτελεί ταυτόχρονα και αδύναμο σημείο καθώς η διαδικασία αυτή είναι χρονοβόρα και έτσι η Java υστερεί σε ταχύτητα σε σχέση με άλλες γλώσσες. Τέλος, και στα Java applets η εκτέλεση των προγραμμάτων λαμβάνει χώρα μέσω της εικονικής μηχανής Java (EMJ) (Java Virtual Machine - JVM), όπως συμβαίνει και στα αυτόνομα προγράμματα Java (όπως αναφέρθηκε παραπάνω).

Μεταξύ κανονικών εφαρμογών και μικροεφαρμογών Java υφίστανται και αρκετές διαφορές, οι οποίες συνοψίζονται ως εξής:

- οι μικροεφαρμογές (applets) δεν έχουν μέθοδο main() όπως οι αυτόνομες εφαρμογές (applications). Έχουν όμως ένα σύνολο μεθόδων προερχομένων από την κλάση Applet, π.χ.:
    - *public void init()*: κώδικας αρχικοποίησης της μικροεφαρμογής. Εκτελείται πρώτος.
    - *public void start()*: κώδικας που εκτελείται μετά την αρχικοποίηση ή αν η μικροεφαρμογή ξεκινήσει πάλι μετά από διακοπή
    - *public void stop()*: κώδικας που εκτελείται όταν διακοπεί η εκτέλεση του applet
    - *public void destroy()*: κώδικας που εκτελείται μόλις κλείσει ο φυλλομετρητής αν (σπανιότατα) υπάρχει λόγος να απελευθερωθεί η δεσμευμένη μνήμη με χειροκίνητο τρόπο
    - *public void paint(Graphics g)*: εδώ μπαίνει ο κώδικας που τυπώνει / ζωγραφίζει πράγματα στο παράθυρο της μικροεφαρμογής

- Τα applets, λόγω της web φύσης τους, υπακούουν σε ένα σύνολο περιορισμών που δεν ισχύουν για τα Java applications.
- Τα applets υποχρεωτικά κληρονομούν την κλάση Java.applet.Applet

public class MyApplet extends Java.applet.Applet{....}

Η διαδικασία υλοποίησης μικροεφαρμογών applets γενικά είναι η εξής:

- Συγγραφή του πηγαίου κώδικα σε ένα αρχείο με επέκταση .Java με χρήση text editor ή ενός περιβάλλοντος ανάπτυξης όπως το NetBeans ή το Eclipse
- Μεταγλώττιση του πηγαίου κώδικα σε bytecode (προκύπτει ένα .class αρχείο)
- Δημιουργία (X)HTML σελίδας με αναφορά στο παραγόμενο .class αρχείο
- Εκτέλεση του applet με άνοιγμα του applet σε έναν φυλλομετρητή

Κατά την εκτέλεση των applets τίθενται αυστηροί περιορισμοί ως προς το τι επιτρέπεται να κάνει ένα applet. Το γεγονός αυτό οφείλεται στο ότι τα applets εκτελούνται "τοπικά", στον Η/Υ του χρήστη μέσω πρόσβασης σε έναν απομακρυσμένο διακομιστή από τον οποίο μεταφορτώνεται το applet. Σε περίπτωση που η πρόσβαση δεν υπόκειται σε περιορισμούς δεν είναι δύσκολο για έναν κακόβουλο προγραμματιστή να επέμβει στον υπολογιστή του χρήστη και π.χ. να διαγράψει τα αρχεία του. Στην πραγματικότητα η ασφάλεια που παρέχει η Java ήταν ένα αντικείμενο συζητήσεων και παραφιλολογίας για πολύ καιρό μετά την εισαγωγή της.

Οι απαραίτητοι, λοιπόν, περιορισμοί ασφαλείας σε applets συνοψίζονται στους ακόλουθους:

Ένα applet δεν μπορεί να:

- διαβάσει ή γράψει / δημιουργήσει αρχεία στο τοπικό σύστημα αρχείων του χρήστη (local filesystem)
- επικοινωνήσει με άλλη Internet τοποθεσία εκτός αυτής που φιλοξενεί την ιστοσελίδα στην οποία περιλαμβάνεται το applet
- εκτελέσει προγράμματα στο σύστημα του χρήστη
- φορτώσει προγράμματα αποθηκευμένα στο σύστημα του χρήστη.

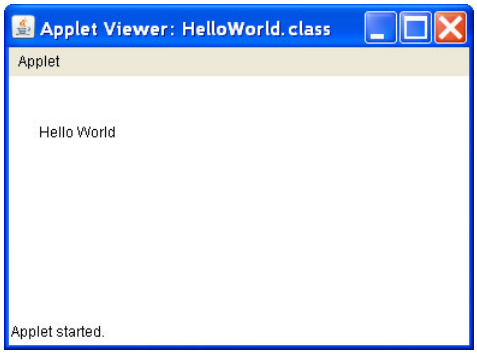
1.2.1. Παραδείγματα Java Applet

Δυο απλά παραδείγματα Java Applet κρίνονται απαραίτητα για να στηριχθεί η άποψη ότι η γλώσσα Java είναι σχετικά απλή. Βέβαια η απλότητα και η ευκολία μιας γλώσσας προγραμματισμού δεν εξαρτάται μόνο από την ίδια αλλά και από τις εμπειρίες και προτιμήσεις του χρήστη. Άρα είναι κάτι υποκειμενικό.

Παρακάτω φαίνεται ένα απλό παράδειγμα applet που τυπώνει τη φράση “Hello World”:

```
import Java.applet.*;
import Java.awt.*;

public class HelloWorld extends Applet {

      public void paint(Graphics g) {
            g.drawString("Hello World", 25, 50);
      }
}
```

Η έξοδος του πιο πάνω κώδικα, όπως φαίνεται με το applet viewer του Eclipse είναι αυτή της εικόνας 1.3.



Εικόνα 1.3 Η έξοδος του πιο πάνω προγράμματος

Για να ενσωματωθεί το applet αυτό σε κώδικα HTML, αρκεί η εξής δήλωση:

<applet code= ¨ HelloWorld.class ¨ width=600 height=100> </applet>

σε ένα αρχείο με προέκταση .html (ή .htm). Για να φορτωθεί το applet στο υπολογιστή του χρήστη θα πρέπει το αρχείο HelloWorld.class να βρίσκεται στον ίδιο διακομιστή με την ιστοσελίδα (ή στον ίδιο φάκελο αν φορτώνεται τοπικά).

Όπως φαίνεται και στο πιο πάνω παράδειγμα, υπάρχουν ειδικά applet tags που υποστηρίζονται από την (X)HTML (<applet>...</applet>). Η έξοδος του applet εμφανίζεται σε ένα τμήμα της περιοχής του παραθύρου του φυλλομετρητή και έχει το μέγεθος που ορίστηκε με τη δήλωση width=xxx height=xxx.

Τα Java applets παρέχουν στο χρήστη και την δυνατότητα να δίνει αρχικές τιμές σε μεταβλητές του Java κώδικα μέσα από τον HTLM κώδικα, έτσι ώστε κάθε φορά να προκύπτει διαφορετικό αποτέλεσμα. Τα προγράμματα αυτά επαναχρησιμοποιούνται καθόσον ο χρήστης του applet δεν είναι κατ' ανάγκη ο προγραμματιστής, αλλά έχει το .*class αρχείο και γνωρίζει με ποιο τρόπο χρησιμοποιούνται οι παράμετροι. Για παράδειγμα, αν στον κώδικα Java και στην μέθοδο init() υπάρχει μια δήλωση της μορφής:

text = getParameter("text")

τότε γράφοντας στον κώδικα HTML κάτι τέτοιο:

<PARAM NAME="text" VALUE="Hello there!">

Όταν θα εκτελούνταν η μέθοδος init(), η μεταβλητή text του κώδικα Java θα έπαιρνε την τιμή "Hello there!".

Τα Java applets μπορούν να χρησιμοποιηθούν και για τη δημιουργία γραφικών. Γραφικά μπορούν να χρησιμοποιηθούν και σε παραθυρικές εφαρμογές (application GUIs), όχι όμως σε εφαρμογές γραμμής εντολών

Ο πρωταρχικός και απλούστερος τρόπος για εισαγωγή γραφικών γίνεται με χρήση των μεθόδων της κλάσης Java.awt.* ή της Java.swing.*

Ένα παράδειγμα χρήσης γραφικών φαίνεται παρακάτω:

```
import Java.applet.Applet;
import Java.awt.*;

public class SimpleGraphics extends Applet{
      public void paint(Graphics graph) {
            // Ορισμός Γραμματοσειράς (Τύπος, Στύλ, Μέγεθος)
            Font f = new Font("TimesRoman", Font.BOLD, 16);
```

```
            graph.setFont(f);
            /* Ορισμός χρώματος της τρέχουσας πένας που ζωγραφίζουμε,
            χρησιμοποιούμε το έτοιμο χρώμα Color.red (ιδιότητα red της
κλάσης Color)*/
            graph.setColor(Color.red);
            /* Ζωγραφίζεται το String Ellipse στη θέση (185, 75) του
καμβά της
            μικροεφαρμογής μας
            μας */
            graph.drawString("Ellipse", 70, 70);
            graph.drawOval(50, 50, 90, 30);
      }
}
```

Η έξοδος του πιο πάνω κώδικα, όπως φαίνεται με το applet viewer του Eclipse είναι αυτή της εικόνας 1.4.

Εικόνα 1.4 Η έξοδος του πιο πάνω προγράμματος

Πιο ωραία γραφική σχεδίαση μπορεί να επιτευχθεί με τις κλάσεις Java2D στις οποίες περιλαμβάνονται ειδικά μοτίβα σχεδίασης, πένες διαφορετικού πλάτους και στυλ, δυνατότητα εξομάλυνσης των πλευρών των σχεδιαζόμενων σχημάτων (anti-aliasing), ορισμός συντεταγμένων με ακρίβεια float ή double κ.α.

1.3.Εξελίξεις στις μικροεφαρμογές Java

Η ανάπτυξη του διαδικτύου και κυρίως η αύξηση της ταχύτητας διαμοιρασμού δεδομένων έχει βοηθήσει στην εμφάνιση όλο και πιο πρωτοποριακών τεχνολογιών (κυρίως για το διαμοιρασμό πολυμεσικών εφαρμογών). Τέτοιες είναι οι εφαρμογές τύπου RIA (rich internet applications) με κυριότερους εκπροσώπους τους το Adobe Flex και την JavaFX. Το χαρακτηριστικό των προγραμμάτων τύπου RIA είναι ότι αποτελούν ολοκληρωμένες εφαρμογές που εγκαθίστανται αυτόματα μέσα από την προβολή τους από ένα φυλλομετρητή, χωρίς να υπάρχει ανάγκη να ακολουθηθούν οι κλασικές μέθοδοι εγκατάστασης λογισμικού, αποφεύγοντας έτσι κάποια από τα προβλήματα που οι τελευταίες εμφάνιζαν (δικαιώματα, αλληλεξαρτήσεις κ.λ.π.). Επίσης, μπορούν να έχουν πρόσβαση σε ένα σύνολο δεδομένων, που μπορεί να φιλοξενούνται σε κάποιον web server, επιτρέποντας την αλληλεπίδραση του χρήστη με τα πιο πρόσφατα δεδομένα και δυνατότητες που ο δημιουργός της κάθε εφαρμογής προσφέρει (Κουϊμτζής, 2009).

Η τεχνολογία JavaFX αποτελεί την τελευταία και πιο ολοκληρωμένη επέκταση της τεχνολογίας Java. Μέχρι σήμερα η γλώσσα/τεχνολογία Java επέτρεπε τη δημιουργία εφαρμογών για κάθε είδους ηλεκτρονική συσκευή (κινητά τηλέφωνα, smartphones, υπολογιστές, pda's, tv sets) επεκτείνοντας τις δυνατότητές τους σε σημαντικό βαθμό. Με την τεχνολογία JavaFX, πλέον επιδιώκεται η δημιουργία εφαρμογών που ανταποκρίνονται σε απαιτήσεις χρήσης υψηλής ποιότητας video, γραφικών, ήχου και περιεχομένου. Η JavaFX εγκαθίσταται πάνω στην υπάρχουσα αρχιτεκτονική της Java, επιτρέποντας τη δημιουργία πολυμεσικών εφαρμογών υψηλής ποιότητας. Η αρχιτεκτονική της δομή περιλαμβάνει τα εξής μέρη:

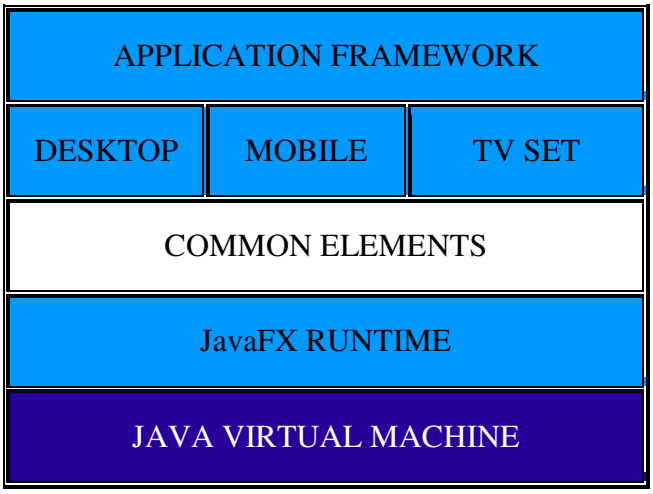
α) Το κοινό πλαίσιο εκτέλεσης των προγραμμάτων της Java (JVM)

β) Τις απαραίτητες επεκτάσεις για την εκτέλεση πολυμεσικών εφαρμογών (JavaFX Runtime)

γ) Το κοινό προγραμματιστικό περιβάλλον (API), όπου με τη χρήση του, όλα τα προγράμματα μπορούν να εκτελούνται σε οποιαδήποτε συσκευή υποστηρίζει την πλατφόρμα

δ) Προγραμματιστικές επεκτάσεις που αφορούν τη δημιουργία εφαρμογών για συγκεκριμένες συσκευές, λαμβάνοντας υπόψη τα ιδιαίτερα χαρακτηριστικά τους

ε) Το πλήθος των εργαλείων που θα κάνουν χρήση οι προγραμματιστές



Εικόνα 1.5 Αρχιτεκτονική πλατφόρμας JavaFX

2. Οι μικροεφαρμογές Java στην εκπαίδευση

2.1. Εισαγωγή

Η ανάγκη χρήσης της συμβολικής αναπαράστασης μιας συγκεκριμένης κατάστασης που μιμείται ή αναπαριστά στοιχεία ή συστήματα του πραγματικού κόσμου στη σχολική πραγματικότητα και στην εκπαίδευση εν γένει, κατέστησε απαραίτητη την εισαγωγή των Τεχνολογιών Πληροφορίας και Επικοινωνιών (ΤΠΕ) στο χώρο της εκπαίδευσης (Θεοφανέλλης, 2008).

Η εισαγωγή αυτών των τεχνολογιών στην εκπαίδευση είναι πλέον μια αναγκαιότητα. Η σωστή χρήση τους επιφέρει ουσιαστικές καινοτομίες, τόσο στα μέσα διδασκαλίας, όσο και στη διαδικασία της μάθησης (Θεοφανέλλης, 2008). Ήδη, νέα εποπτικά μέσα χρησιμοποιούνται από όλο και μεγαλύτερη μερίδα εκπαιδευτικών ενώ ταυτόχρονα η παραγωγή εκπαιδευτικού υλικού είχε μια σημαντική ανάπτυξη τα τελευταία χρόνια.

Μια κατηγορία εκπαιδευτικού υλικού που χρησιμοποιείται ευρέως στη μαθησιακή-διδακτική διαδικασία είναι οι μικροεφαρμογές Java (Java applets). Η ένταξη αυτής της τεχνολογίας στη διαδικασία της μάθησης προσφέρει δυνατότητες διαδραστικής αναζήτησης της γνώσης επιτρέποντας να επιτευχθεί η «διερευνητική μάθηση» (Κασούτσας, 2012). Επιπλέον, η χρήση τέτοιων μέσων εμπλουτίζει τη διδασκαλία και την καθιστά πιο ελκυστική για τους μαθητές. Εξάλλου η ύπαρξη εικόνας, ήχου, χρώματος στην οθόνη του υπολογιστή, όπως και η απεικόνιση με γραφικό τρόπο διάφορων εννοιών προσελκύουν την προσοχή των μαθητών και τους βοηθούν να εμπεδώσουν αυτές τις έννοιες καλύτερα (Κασούτσας, 2012).

Η χρήση, λοιπόν, των μικροεφαρμογών Java στην εκπαίδευση μπορεί να βοηθήσει τους μαθητές να αντιληφθούν καλύτερα νόμους, σχέσεις, θεωρήματα ή ακόμα και συμπεριφορές (Θεοφανέλλης, 2008). Πιο συγκεκριμένα, με αυτές τις μικροεφαρμογές, μαθηματικές σχέσεις μπορούν να γίνουν επεξηγηματικές γραφικές παραστάσεις και οι φυσικές έννοιες μπορούν να γίνουν ένα παιχνίδι αλληλεπίδρασης. Τέλος, η κατάκτηση νέας γνώσης επιτυγχάνεται ευκολότερα και με πιο αποδοτικό τρόπο.

Η χρήση των μικροεφαρμογών Java στην εκπαίδευση βοηθούν τους μαθητές να αντιλαμβάνονται νόμους, μοντέλα ή σχήματα συμπεριφοράς με μεγάλη ευκολία, να ανακαλύπτουν νέες σχέσεις, να κατακτούν νέες βεβαιότητες και να ανατρέπουν άλλες.

2.2.Οφέλη των μικροεφαρμογών Java στην εκπαίδευση

Υπάρχουν κάποιες έννοιες - ανάλογα με τη φύση της πληροφορίας που εμπεριέχουν – των οποίων το νόημα δε μπορεί να αποδοθεί επιτυχώς στον πίνακα διδασκαλίας ή ακόμα και με τη χρήση διαφανειών. Για παράδειγμα υπάρχουν οι ακόλουθοι τύποι πληροφοριών:

- Δυναμική πληροφορία: η προσομείωση ενός πειράματος στη φυσική (π.χ. ελεύθερη πτώση σώματος) ή μιας χημικής αντίδρασης
- Πολυμεσική πληροφορία: video εκπαιδευτικού περιεχομένου σχετικά με έννοιες των μαθημάτων
- Αλληλεπιδραστική πληροφορία: η αλλαγή παραμέτρων σε ένα σύστημα επιφέρει διαφορετικά αποτελέσματα στην έξοδο (π.χ. βολή υπό γωνία).

Η χρήση μικροεφαρμογών Java μπορεί να βοηθήσει στην αναπαράσταση όλων αυτών των ειδών πληροφορίας. Είναι,όμως, ιδιαίτερα χρήσιμη στην περίπτωση της αλληλεπίδρασης του μαθητή με το σύστημα που προσομειώνεται καθώς και την ανακάλυψη της γνώσης. Η αλληλεπίδραση είναι το χαρακτηριστικό που λείπει από άλλες τεχνικές διδασκαλίας (Παπαστεργίου, 2005)

Εκτός από τη δυνατότητα υλοποίησης διαδραστικών εφαρμογών, η χρήση των μικροεφαρμογών Java παρουσιάζουν και αρκετά άλλα πλεονεκτήματα (Kamthan 2009).

- Επίδειξη στη τάξη και συσχετισμός: η επίδειξη στην τάξη είναι σημαντικό στοιχείο για θέματα τα οποία έχουν πρακτικό προσανατολισμό και απαιτούν και πρακτική εφαρμογή έτσι ώστε να γίνουν πιο κατανοητά από τους μαθητές.
- Αξιολόγηση: οι παραδοσιακές μέθοδοι αξιολόγησης, όπως τα tests στην τάξη μπορούν να αντικατασταθούν ή να εμπλουτιστούν με απλά tests τύπου πολλλαπλών επιλογών βασισμένα σε Java applets. Με αυτό τον τρόπο οι μαθητές αναπτύσσουν την αυτοπεποίθησή τους, καθώς μπορούν να ελέγξουν μόνοι τους το βαθμό στον οποίο έχουν κατανοήσει ένα αντικείμενο. Τέτοιου είδους tests οι μαθητές μπορούν να εκπονήσουν και από το δικό τους χώρο στο δικό τους χρόνο.
- Κόστος: πολλές εφαρμογές οι οποίες χρησιμοποιούνται στην εκπαίδευση και παρέχονται από τα σχολεία ή τα εκπαιδευτικά ιδρύματα αποτελούν ουσιαστικά εμπορικές εφαρμογές (π.χ. Mathematica, MATLAB) τις οποίες όμως οι μαθητές δεν

έχουν τη δυνατότητα να αποκτήσουν. Τα Java applets, από την άλλη, είναι δωρεάν και ο χρήστης μπορεί να τα εκτελέσει μέσα από έναν από τους δωρεάν διαθέσιμους φυλλομετρητές που έχουν τη δυνατότητα εκτέλεσης μικροεφαρμογών Java (π.χ. Internet Explorer, Mozzila Firefox, Google Chrome)

- Μεταβιβασιμότητα και προσαρμοστικότητα: η διαθεματικότητα είναι πλέον ένα ζητούμενο στην εκπαίδευση. Πολλά μαθήματα έχουν Αναλυτικά Προγράμματα Σπουδών που συχνά αλληλεπικαλύπτονται (π.χ. αγγλικά και πληροφορική ή γεωγραφία και ιστορία) και ιδιαίτερα στα ΕΠΑ.Λ. υπάρχουν μαθήματα ειδικότητας που σε ορισμένες περιπτώσεις χρησιμοποιούν ίδιες έννοιες. Έτσι, μικροεφαρμογές Java οι οποίες έχουν αναπτυχθεί για ένα αντικείμενο μπορούν να χρησιμοποιηθούν και για κάποιο άλλο.

2.3.Προϋποθέσεις χρήσης των μικροεφαρμογών Java στην εκπαίδευση

Προκειμένου να χρησιμοποιηθούν οι μικροεφαρμογές Java στον εκπαιδευτικό χώρο θα πρέπει να ληφθούν υπόψη ορισμένα θέματα – προϋποθέσεις (Kamthan 2009).

- Καταλληλότητα: οι μικροεφαρμογές Java πρέπει να βασίζονται σε προσεκτικά επιλεγμένες ενότητες του Α.Π.Σ. (Αναλυτικό Πρόγραμμα Σπουδών) ενός μαθήματος. Δε χρειάζεται να χρησιμοποιηθούν μικροεφαρμογές Java για όλες τις ενότητες ενός μαθήματος. Κάτι τέτοιο θα αποτελούσε υπερβολή και σπατάλη χρόνου, ιδιαίτερα εάν οι μικροεφαρμογές δεν υπάρχουν και πρέπει να αναπτυχθούν από την αρχή. Σε πολλές περιπτώσεις η απλή κλασική μέθοδος του μαυροπίνακα είναι παραπάνω από αρκετή για τους μαθητές προκειμένου να εμπεδώσουν κάποιες έννοιες. Ακόμα και σε αυτές τις περιπτώσεις, όμως, η συμπληρωματική χρήση ενός applet συμβάλλει θετικά στη μάθηση.

- Εκπαίδευση: η χρήση των μικροεφαρμογών Java μπορεί να είναι απλή, ειδικά αν το περιβάλλον διεπαφής του χρήστη έχει σχεδιαστεί προσεκτικά ή αν υπάρχει ικανή βοήθεια. Από την άλλη, όμως, η σχεδίαση και υλοποίηση της ίδιας της μικροεφαρμογής σε Java μπορεί να είναι επίπονη διαδικασία, ειδικά αν ο εκπαιδευτικός - δημιουργός δεν έχει επαρκείς γνώσεις προγραμματισμού. Η παρούσα

εργασία έχει και αυτό το σκοπό, να δείξει δηλαδή ότι η σχεδίαση και υλοποίηση μικροεφαρμογών Java είναι εφικτή αν υπάρχει το σωστό υπόβαθρο.

- Διαθεσιμότητα: Αν η ανάπτυξη μιας μικροεφαρμογής Java είναι εφικτή, τότε πρέπει να υλοποιηθεί από τον εκπαιδευτικό ή μήπως ο εκπαιδευτικός πρέπει να αναζητήσει ήδη έτοιμες μικροεφαρμογές; Η απάντηση σε αυτό το ερώτημα δεν είναι εύκολη. Εάν υπάρχει αρκετός χρόνος και διάθεση ίσως ήταν καλύτερο να υλοποιηθεί η εφαρμογή από το μηδέν. Τα οφέλη από μια τέτοια προσέγγιση είναι πολλαπλά. Καταρχάς ο εκπαιδευτικός θα αποκτήσει σημαντική εμπειρία ενώ η εφαρμογή θα είναι πιο κοντά στον τρόπο διδασκαλίας του εκπαιδευτικού επιτυγχάνοντας τη λεγόμενη «εξατομικευμένη μάθηση». Και αν και υπάρχει ο κίνδυνος να μην είναι έτοιμη η μικροεφαρμογή εγκαίρως, προκειμένου να χρησιμοποιηθεί στην πράξη, θα μπορούσε να εφαρμοστεί τα επόμενα σχολικά έτη. Από την άλλη, οι έτοιμες εφαρμογές Java είναι διαθέσιμες όταν κάποιος τις χρειάζεται αλλά δεν είναι απαραίτητα προσαρμόσιμες στην εκάστοτε εκπαιδευτική διαδικασία που ακολουθείται. Επιπλέον, η γλώσσα είναι ένα άλλο πρόβλημα καθώς μικροεφαρμογές στα Γερμανικά ή τα Γαλλικά ή ακόμα και τα Αγγλικά μπορεί να μην είναι εύχρηστες από μαθητές στη Ελλάδα.

Η χρήση των μικροεφαρμογών Java ως εργαλεία διερεύνησης δεν απαιτεί τεχνικές γνώσεις, αλλά μπορούν αυτές να χρησιμοποιηθούν όπως τα πειράματα. Το γεγονός αυτό αντικρούει τα επιχειρήματα μεγάλου ποσοστού εκπαιδευτών που διστάζουν να χρησιμοποιήσουν το διαδίκτυο κατά τη διδασκαλία, προβάλλοντας ως λόγους, τόσο τις τεχνικές γνώσεις που απαιτούνται, όσο και την έλλειψη πλαισίου για το πως η τεχνολογία θα χρησιμοποιηθεί σε συνδυασμό με τις παραδοσιακές μεθόδους. Τα οφέλη, άλλωστε, από τη χρήση των μικροεφαρμογών Java επεκτείνονται και στους εκπαιδευτικούς, εφόσον με τον τρόπο αυτό πετυχαίνουν καλύτερα αποτελέσματα σε λιγότερο χρόνο και με λιγότερο κόπο. Τις ίδιες εφαρμογές τις χρησιμοποιεί και σε άλλες τάξεις, οι μαθητές κατανοούν τις έννοιες πιο γρήγορα και αυξάνει η απόδοση της εκάστοτε τάξης πιο ξεκούραστα.

2.4. Εκπαιδευτικοί τομείς χρήσης των μικροεφαρμογών Java.

Υπάρχει ήδη στο διαδίκτυο ένα αρκετά ευρύ φάσμα της εκπαιδευτικής πραγματικότητας για την οποία έχουν δημιουργηθεί και χρησιμοποιούνται, ίσως δειλά ακόμη, μικροεφαρμογές Java για εκπαιδευτικούς σκοπούς.

Ενδεικτικά αναφέρουμε τους πιο διαδεδομένους εκπαιδευτικούς τομείς για τους απoίους έχουν δημιουργηθεί διάφορες μικροεφαρμογές Java:

- *μαθηματικά:* π.χ. "Mathlets" που είναι μικρά ανεξάρτητα εργαλεία εκμάθησης τα οποία εστιάζουν σε ένα συγκεκριμένο μαθηματικό θέμα ή πρόβλημα και είναι έτοιμα προς χρήση για επιδείξεις απ' τους καθηγητές ή για εξατομικευμένη μάθηση απ' τους ίδιους τους μαθητές. Βρίσκονται είτε σε ιστοσελίδες στο διαδίκτυο είτε είναι οργανωμένα σε βιβλιοθήκες όπως είναι το Math Forum, Library of Virtual Manipulatives, Digital Classroom Resources κ.α. (Κασούτσας, 2012)
- *φυσική:* παραδείγματα συνδέσμων ιστοτόπων είναι (ενδεικτικά αναφερόμενοι):
    - Εικονικό εργαστήριο με μικροεφαρμογές φυσικής από το ΕΚΦΕ Κιλκίς http://ekfe.kil.sch.gr/applets/spin/Physics/phys.html
    - Εικονικό εργαστήριο Φυσικής ΙΙ,

      http://www.physics.ntua.gr/~konstant/p102/vlab/
    - Java applets on Physics, http://www.walter-fendt.de/ph14gr/
    - Phet από το πανεπιστήμιο του Colorado, https://phet.colorado.edu/el/
    - Πολλοί άλλοι Ελληνικοί και ξένοι σχετικοί ιστότοποι.
- *χημεία:* σε έκθεση των Εκπαιδευτικών Α. Τζαμτζή, Α. Γεωργιάδου και Π. Γιαννακουδάκη / 2ο Συνέδριο Σύρου με θέμα ΤΠΕ στην Εκπαίδευση αναφέρεται ότι εντοπίστηκαν πάνω από 200 μικροεφαρμογές Java στο αντικείμενο της χημείας.
- *Πληροφορική:* ορισμένα από τα θέματα που καλύπτουν είναι : υλικό, μετατροπείς, πίνακες αληθείας, κρυπτογραφία, αλγόριθμοι, διαδίκτυο, κυκλώματα.
- *Βιολογία:* παραδείγματα σχετικών ιστοσελίδων είναι (ενδεικτική αναφορά):
    - http://e-yliko.gr (Η εκπαιδευτική πύλη του ΥΠΕΠΘ)

- http://e-slate.cti.gr/ (Ιστοσελίδα που μπορεί να κατεβάσει κανείς το αβάκιο, έτοιμους μικρόκοσμους ή να δημιουργήσει δικούς του μικρόκοσμους.)
- http://www.bioplek.org/bioplek.html (Ιστοσελίδα για θέματα Γενετικής)
- πλήθος άλλων σχετικών ιστότοπων

- *Γεωγραφία- Γεωλογία:* παραδείγματα σχετικών ιστοσελίδων είναι (ενδεικτική αναφορά):
  - .http://www.google.com/earth/index.html (Ιστοσελίδα που δίνει τη δυνατότητα πτήσης πάνω από την επιφάνεια της Γης)
  - http://seismo.geology.upatras.gr/ (Ιστοσελίδα του Πανεπιστημίου Πάτρας σχετική με τους σεισμούς)
  - http://www.civilprotection.gr (Σελίδα πολιτικής προστασίας)
  - και πλήθος άλλων σχετικών ιστοσελίδων.
- *Γλώσσα:* για παράδειγμα υφίστανται μικροεφαρμογές Java που παρουσιάζουν κάποιο κείμενο και ζητούν από τον εκπαιδευόμενο να δώσει κατάλληλες απαντήσεις. Ανάλογα με την απάντηση εμφανίζεται μήνυμα λάθους και ο εκπαιδευόμενος μπορεί να εξασκηθεί και να έχει συνολική εικόνα της απόδοσής του.
- *Αστρονομία, ηλεκτρονική – ηλεκτρολογία, μηχανολογία, οικονομίας και άλλοι.*

Στο Β΄ μέρος της παρούσας εργασίας θα δημιουργηθεί και θα αναλυθεί ένα παράδειγμα μικροεφαρμογής Java χρήσιμο για την κατανόηση της λειτουργίας των τεχνικών αυτών στον εκπαιδευτικό χώρο. Συγκεκριμένα, θα δημιουργηθεί μια μικροεφαρμογή java που αφορά τη Δευτεροβάθμια Τεχνική Εκπαίδευση και συγκεκριμένα τους μαθητές της δευτέρας και της τρίτης τάξης του τομέα Ηλεκτρονικής. Το δε παράδειγμα που θα χρησιμοποιηθεί είναι το θεώρημα δειγματοληψίας του Nyquist.

3. Δειγματοληψία σήματος και το θεώρημα του Nyquist

3.1. Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει αρχικά μια παρουσίαση της ανάγκης της δειγματοληψίας ενός σήματος. Η διαδικασία μετάβασης από τον αναλογικό κόσμο στον ψηφιακό θα περιγραφεί σε αδρές γραμμές, χωρίς να γίνει εκτενής αναφορά σε όλα τα πεδία εφαρμογής της δειγματοληψίας. Θα υπάρχουν οι απαραίτητες πληροφορίες που θα διευκολύνουν τον αναγνώστη να αφομοιώσει τις βασικές έννοιες της δειγματοληψίας και της ψηφιοποίησης ενός σήματος. Η ανάκτηση της αρχικής πληροφορίας από ένα δειγματοληπτημένο σήμα, στο βαθμό που αυτό είναι δυνατόν, θα οδηγήσει στην διατύπωση του ζητούμενου θεωρήματος δειγματοληψίας, του λεγόμενου και θεωρήματος Nyquist. Οι μαθηματικές σχέσεις που παρατίθενται στο κεφάλαιο αυτό είναι απαραίτητες μεν για την διατύπωση των εννοιών με τον απαιτούμενο μαθηματικό φορμαλισμό όχι όμως και δεσμευτικές για την παρουσίαση του θεωρήματος δειγματοληψίας. Εξάλλου η εφαρμογή που αναπτύχθηκε στηρίζεται στη δυνατότητα να αντιληφθεί κανείς τις βασικές έννοιες και τις διαδικασίες πίσω από την δειγματοληψία μέσω της οπτικής αναπαράστασης της διαδικασίας. Σε αντίθετη περίπτωση, το όλο εγχείρημα θα ήταν άτοπο δεδομένου ότι ο στόχος είναι να αντιληφθούν μαθητές της Β΄ και Γ΄ τάξης ΕΠΑ.Λ. το γιατί υπάρχει το θεώρημα αυτό. Ο μαθηματικός φορμαλισμός θα ήταν κάτι τελείως ξένο για τους μαθητές αυτών των τάξεων (οποιουδήποτε τύπου λυκείου) καθώς δεν έχουν το απαραίτητο γνωστικό υπόβαθρο.

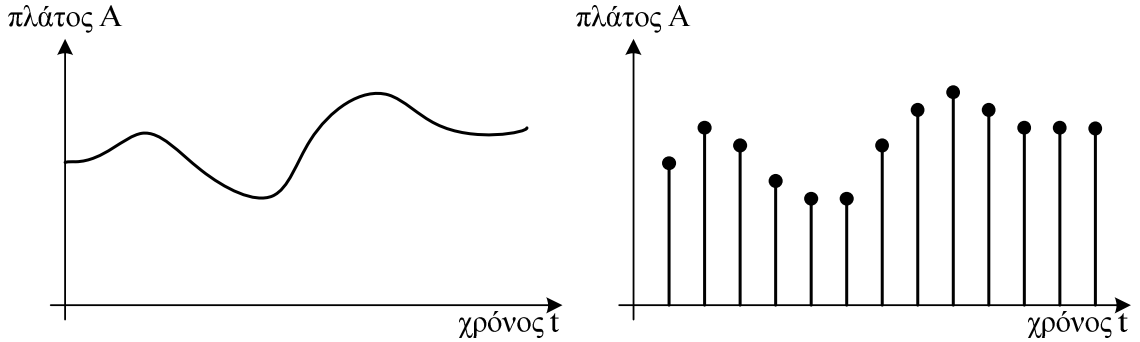
3.2. Από τον αναλογικό κόσμο στον ψηφιακό

Ο κόσμος μέσα στον οποίο κινούμαστε και όπως τον αντιλαμβανόμαστε κυριαρχείται από την αναλογικότητα. Ο όρος αναλογικό εδώ χρησιμοποιείται σε αντιδιαστολή με τον όρο ψηφιακό. Έτσι όταν λέμε ότι ο κόσμος μας είναι αναλογικός αναφερόμαστε στο γεγονός ότι αντιλαμβανόμαστε τις περισσότερες (αν όχι όλες) φυσικές παραμέτρους σαν συνεχείς συναρτήσεις του χρόνου αλλά και με συνεχείς τιμές στο πεδίο τιμών της κάθε μιας. Έτσι υπάρχουν άπειρες τιμές της θερμοκρασίας στη διάρκεια της ημέρας, οι οποίες μπορούν να μετρηθούν με πολύ μεγάλη ακρίβεια (όση επιτρέπει η μέθοδος μέτρησής τους). Το ίδιο και η ένταση του ήχου, μπορεί να λάβει άπειρες τιμές με πλάτος το οποίο περιγράφεται σαν μια

συνεχή συνάρτηση στο χρόνο. Από την άλλη στον ηλεκτρονικό υπολογιστή αλλά και σε κάθε ψηφιακή συσκευή (και όχι μόνο) είναι γνωστό ότι όλη η πληροφορία αποδομείται σε μια σειρά από bits (0 και 1) τα οποία δημιουργούνται, επεξεργάζονται ή ταξιδεύουν με μεγάλους μεν αλλά συγκεκριμένους ρυθμούς. Έτσι ένα τραγούδι αποθηκεύεται σε μορφή mp3 με ποιότητα 192Kbps ή η πρόβλεψη της θερμοκρασίας ενημερώνεται ανά τρίωρο. Υπάρχει λοιπόν μια σαφής διαφοροποίηση ανάμεσα στο συνεχές του φυσικού κόσμου και το διακριτό των τεχνολογιών επικοινωνίας και πληροφοριών.

Στο σημείο αυτό θα πρέπει οι επιμέρους τις ειδικές αναφορές να αφεθούν στο πλάι και να ξεκινήσει πλέον η γενίκευση. Έτσι δεν θα γίνεται αναφορά πλέον σε τιμές θερμοκρασίας ή σε ένταση ήχου αλλά σε σήματα πληροφορίας. Τα σήματα πληροφορίας αναπαριστούν δεδομένα του φυσικού ή τεχνητού κόσμου τα οποία έχουν για εμάς κάποιο νόημα και θέλουμε να αποθηκεύσουμε, επεξεργαστούμε, αναλύσουμε κ.α. Έτσι λοιπόν, σύμφωνα με τη διαφοροποίηση που αναφέρθηκε πιο πριν, ορίζονται 2 ειδών σήματα, τα αναλογικά και τα ψηφιακά.

Αναλογικό σήμα είναι ένα σήμα που έχει συνεχείς τιμές τόσο στο πεδίο του χρόνου (ή οποιαδήποτε άλλη παράμετρος απεικονίζεται στον άξονα x) όσο και στο πεδίο τιμών του πλάτους του. Αντιθέτως, ψηφιακό σήμα είναι ένα σήμα που μπορεί να πάρει διακριτές τιμές τόσο στο πεδίο του χρόνου αλλά και στο πεδίο τιμών του πλάτους του. Παράδειγμα αναλογικού σήματος είναι (κλασικά) ή ένταση του ήχου, ενώ παράδειγμα ψηφιακού σήματος είναι η βαθμολογία μια ομάδας ποδοσφαίρου στη διάρκεια των 34 αγωνιστικών.



(α) Αναλογικό σήμα (β) Ψηφιακό σήμα

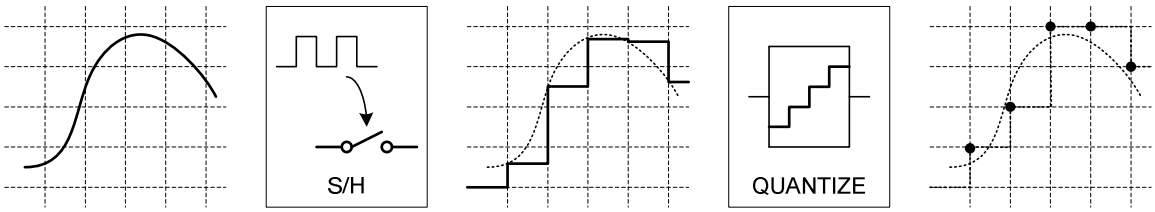
Εικόνα 3.1 Παραδείγματα αναλογικού και ψηφιακού σήματος

3.2.1. Μετατροπή από αναλογικό σήμα σε ψηφιακό

Η διαδικασία κατά την οποία ένα αναλογικό σήμα πληροφορίας μετατρέπεται σε ψηφιακό ονομάζεται ψηφιοποίηση και εκτελείται από εξειδικευμένα κυκλώματα, τους μετατροπείς από αναλογικό σε ψηφιακό (analog/digital converters – ADC). Η μετατροπή αυτή περιλαμβάνει 2 βασικά βήματα, την δειγματοληψία (σε συνδυασμό με την συγκράτηση) και την κβάντιση. Η μεν δειγματοληψία, στην οποία θα γίνει πιο εκτενής αναφορά στην παράγραφο 3.3, είναι η διαδικασία κατά την οποία λαμβάνονται δείγματα από το σήμα πληροφορίας, τα οποία κατά προτίμηση ισαπέχουν στο χρόνο. Δηλαδή ανά τακτά χρονικά διαστήματα λαμβάνεται ένα δείγμα του σήματος πληροφορίας. Η συγκράτηση υλοποιείται και αυτή από το κύκλωμα δειγματοληψίας και συγκράτησης (sample and hold – S/H). Συγκράτηση σημαίνει να διατηρείται το πλάτος του δείγματος που λήφθηκε, σταθερό μέχρι να ληφθεί το επόμενο δείγμα. Ο λόγος που συμβαίνει αυτό είναι να διατηρείται σταθερή η τιμή του δείγματος στις εισόδους της επόμενης βαθμίδας του μετατροπέα ώστε να μπορέσει να λειτουργήσει και η κβάντιση. Τι είναι η κβάντιση όμως και γιατί είναι απαραίτητη;

Μετά τη δειγματοληψία ενός αναλογικού σήματος αυτό το οποίο προκύπτει είναι ένα σήμα με διακριτές τιμές στο χρόνο (τα δείγματα), αλλά με άπειρες τιμές στο πλάτος. Η δειγματοληψία από μόνη της δεν παράγει ψηφιακό σήμα. Αυτό που δίνει είναι ένα διακριτό στο χρόνο, αναλογικό, όμως πάλι, σήμα. Για να μετατραπεί το σήμα αυτό σε καθαρά ψηφιακό θα πρέπει κάθε δείγμα να αναπαρασταθεί από ένα πεπερασμένο αριθμό από bits (0 ή 1). Η πρακτική λέει ότι όλα τα δείγματα θα πρέπει να αναπαρασταθούν με το ίδιο αριθμό από bits. Το πόσα, το καθορίζει η εκάστοτε εφαρμογή ή τα διαθέσιμα κυκλώματα. Τα 16bit είναι μια συνηθισμένη επιλογή χωρίς όμως να σημαίνει ότι δεν μπορεί να χρησιμοποιηθούν λιγότερα (4 ή 8) ή περισσότερα (32 ή 64). Θεωρώντας ότι τα δείγματα θα αναπαρασταθούν με n bits το κάθε ένα, τότε οι διαφορετικές στάθμες (πλάτη) που μπορούν να απεικονιστούν είναι 2^n. Έτσι για n = 16 bits θα υπάρχουν 2^{16} = 65536 διαφορετικές και ισαπέχουσες στάθμες. Το πλάτος λοιπόν κάθε δείγματος δεν μπορεί να πάρει οποιαδήποτε τιμή αλλά απαραίτητα θα πρέπει να πάρει μια από αυτές τις 2^n διαθέσιμες τιμές, φυσικά την πιο κοντινή στην πραγματική του τιμή. Αυτή η διαδικασία ονομάζεται κβάντιση.

Το ψηφιακό σήμα που παράγεται μπορεί πλέον να αποθηκευτεί, υποστεί επεξεργασία, να αποσταλεί μέσω του διαδικτύου ή οποιαδήποτε άλλη διαδικασία εφαρμόζεται σε ψηφιακά σήματα.



Εικόνα 3.2 Αναπαράσταση της διαδικασίας μετατροπής ενός αναλογικού σήματος σε ψηφιακό.

Πρέπει να επισημανθεί ότι η όλη διαδικασία μετατροπής από αναλογικό σήμα σε ψηφιακό δεν είναι πλήρως αναστρέψιμη. Δηλαδή δεν είναι δυνατόν να ανακτηθεί το αρχικό αναλογικό σήμα σε κάθε λεπτομέρειά του. Το πλάτος κατ' αρχάς έχει υποστεί τη μη αναστρέψιμη διαδικασία της κβάντισης και άρα έχει χαθεί ένα κομμάτι πληροφορίας (αυτό ονομάζεται και σφάλμα κβάντισης). Αλλά και η διαδικασία της δειγματοληψίας δεν είναι πάντα αναστρέψιμη. Μπορεί λοιπόν να γίνει ανάκτηση του αρχικού σήματος αλλά το ανακτημένο σήμα δεν θα είναι 100% ίδιο με το αρχικό. Υπάρχει μάλιστα και η πιθανότητα μεγάλης απόκλισης αν δεν τηρηθεί η συνθήκη που επιβάλει το θεώρημα της δειγματοληψίας.

3.2.2. Μετατροπή από ψηφιακό σήμα σε αναλογικό

Η διαδικασία ανάκτησης του αναλογικού σήματος από το ψηφιακό γίνεται με τη βοήθεια των μετατροπέων από ψηφιακό σε αναλογικό (digital/analog converters – DAC). Διαδικασία είναι αναστροφή της διαδικασίας S/H. Οι ψηφιακές λέξεις εισέρχονται στον μετατροπέα, ο οποίος αντιστοιχεί κάθε λέξη σε συγκεκριμένη τάση στην έξοδό του, την οποία διατηρεί σταθερή μέχρι να έρθει η επόμενη ψηφιακή λέξη. Το αποτέλεσμα είναι ένα συνεχές αναλογικό σήμα που όμως έχει μια μορφή σαν σκαλοπάτια. Ένα χαμηλοπερατό φίλτρο βοηθά ώστε το σήμα αυτό να πάρει μια πιο ομαλή μορφή.

3.3.Δειγματοληψία

Η δειγματοληψία λοιπόν, όπως περιγράφηκε πιο πάνω και όπως μαρτυρά και το όνομά της είναι η διαδικασία κατά την οποία από ένα συνεχές (στο χρόνο) σήμα πληροφορίας λαμβάνονται δείγματα για να προκύψει εντέλει ένα ψηφιακό σήμα. Τα δείγματα αυτά

λαμβάνονται ανά τακτά χρονικά διαστήματα που ισαπέχουν χρόνο T_s ο οποίος ονομάζεται περίοδος δειγματοληψίας. Ο χρόνος αυτός είναι σημαντικό χαρακτηριστικό της δειγματοληψίας. Συνήθως χρησιμοποιείται το αντίστροφό του, που είναι η συχνότητα δειγματοληψίας f_s:

$$f_s = \frac{1}{T_s}$$

Έτσι αν το σήμα πληροφορίας είναι το $x_c(t)$, τότε το δειγματοληπτημένο σήμα θα έχει την εξής έκφραση:

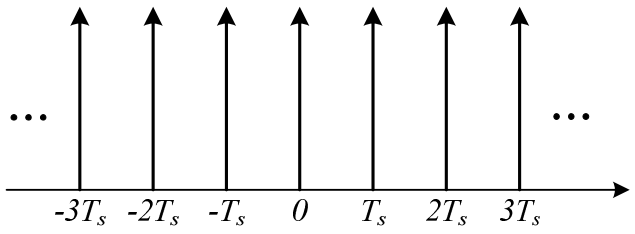
$$x[n] = x_c(n \cdot T_s), \text{ με } -\infty < n < \infty$$

Πρέπει να επισημανθεί ότι η μαθηματική αναπαράσταση της διαδικασίας δειγματοληψίας η οποία θα παρουσιαστεί εδώ είναι ακριβώς αυτό, μια αναπαράσταση. Δεν έχει σχέση με την πραγματική διαδικασία και τα κυκλώματα που παίρνουν μέρος σε αυτή. Απλώς αναπαριστά τη διαδικασία και βοηθά στην κατανόηση των εννοιών.

Για παράδειγμα, στην μαθηματική αναπαράσταση της δειγματοληψίας τα δείγματα λαμβάνονται σαν το γινόμενο του σήματος πληροφορίας με μια σειρά από συναρτήσεις δ του Dirac, που απέχουν μεταξύ τους χρόνο T_s. Η μαθηματική έκφραση αυτής της ακολουθίας είναι:

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - n \cdot T_s)$$

Η γραφική απεικόνιση της πιο πάνω ακολουθίας φαίνεται στην εικόνα 3.3.



Εικόνα 3.3 Ακολουθία συναρτήσεων δ Dirac για αναπαράσταση της δειγματοληψίας

Από καθαρά μαθηματική πλευρά υπάρχει σαφής διαχωρισμός του σήματος $x_s(t)$ που προκύπτει από το γινόμενο του σήματος πληροφορίας με την ακολουθία $s(t)$, από το διακριτό σήμα $x[n]$ που προκύπτει από τη δειγματοληψία.

$$x_s(t) = x_c(t)s(t) \Rightarrow x_s(t) = x_c(t)\sum_{n=-\infty}^{\infty}\delta(t - n\cdot T_s) \Rightarrow x_s(t) = \sum_{n=-\infty}^{\infty} x_c(n\cdot T_s)\delta(t - n\cdot T_s)$$

Από πλευράς όμως σχηματικής απεικόνισης και αντίληψης δεν έχει σημασία αν το σήμα δειγματοληψίας θα είναι μια ακολουθία συναρτήσεων δ Dirac (impulse unit response – κρουστική συνάρτηση) ή δ του Kronecker. Για το λόγο αυτό, στη συνέχεια δεν θα απεικονίζουμε τη συνάρτηση δειγματοληψίας με βέλη αλλά με στελέχη με κύκλους στην άκρη.

Μέχρι εδώ παρουσιάστηκε η διαδικασία της δειγματοληψίας σαν μαθηματική αναπαράσταση στο πεδίο του χρόνου. Μπορούμε όμως να την δούμε και στο πεδίο των συχνοτήτων. Έτσι, το φάσμα του σήματος πληροφορίας όπως προκύπτει από το μετασχηματισμό Fourier είναι:

$$F\{x_c(t)\} = X_c(j2\pi f)$$

Ενώ το φάσμα της ακολουθίας συναρτήσεων δ είναι:

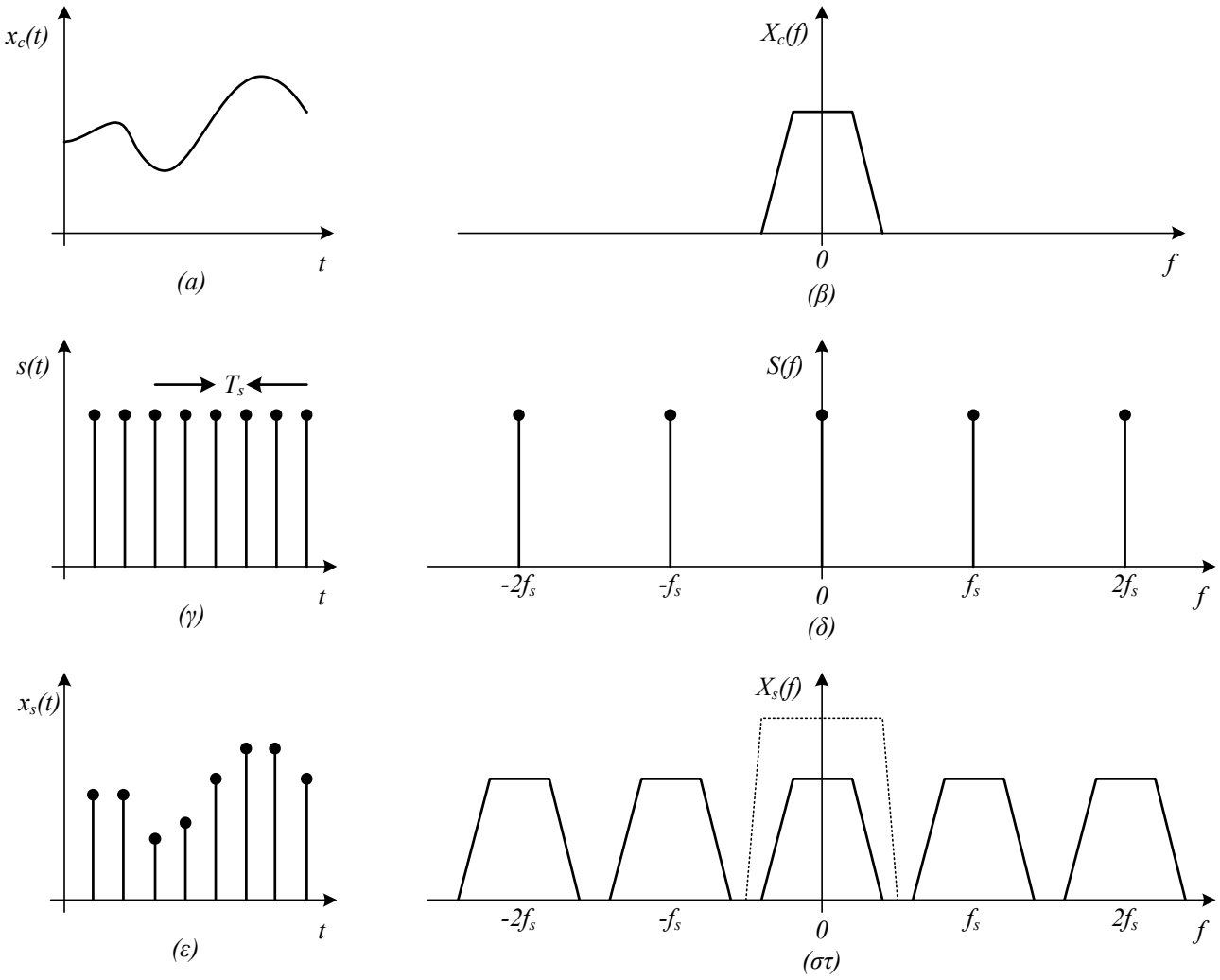
$$F\{s(t)\} = \frac{1}{T_s}\sum_{k=-\infty}^{\infty}\delta(j2\pi(f - kf_s))$$

Το φάσμα του δειγματοληπτημένου σήματος θα είναι η συνέλιξη του φάσματος του σήματος πληροφορίας με το φάσμα του σήματος δειγματοληψίας (γενικά ισχύει ότι ο πολλαπλασιασμός στο πεδίο του χρόνου μετατρέπεται σε συνέλιξη στο πεδίο των συχνοτήτων). Άρα αν $X_s(f)$ το φάσμα του δειγματοληπτημένου σήματος, ισχύει ότι:

$$X_s(f) = \frac{1}{T_s}X_c(j2\pi f) * \sum_{k=-\infty}^{\infty}\delta(j2\pi(f - kf_s)) = \frac{1}{T_s}X_c(j2\pi(f - kf_s))$$

Η μορφή του φάσματος του δειγματοληπτημένου σήματος είναι μια επανάληψη του φάσματος του αρχικού σήματος πληροφορίας, γύρω από κάθε ακέραιο πολλαπλάσιο της συχνότητας f_s

Στο επόμενο σχήμα φαίνεται η διαδικασία που μόλις περιγράφηκε (τόσο στο πεδίο του χρόνου όσο και στο πεδίο των συχνοτήτων) για ένα τυχαίο σήμα πληροφορίας που έχει συνεχές φάσμα στις χαμηλές συχνότητες.



Εικόνα 3.4 Αναπαράσταση της διαδικασίας της ικανοποιητικής δειγματοληψίας με τη βοήθεια μαθηματικών σχέσεων

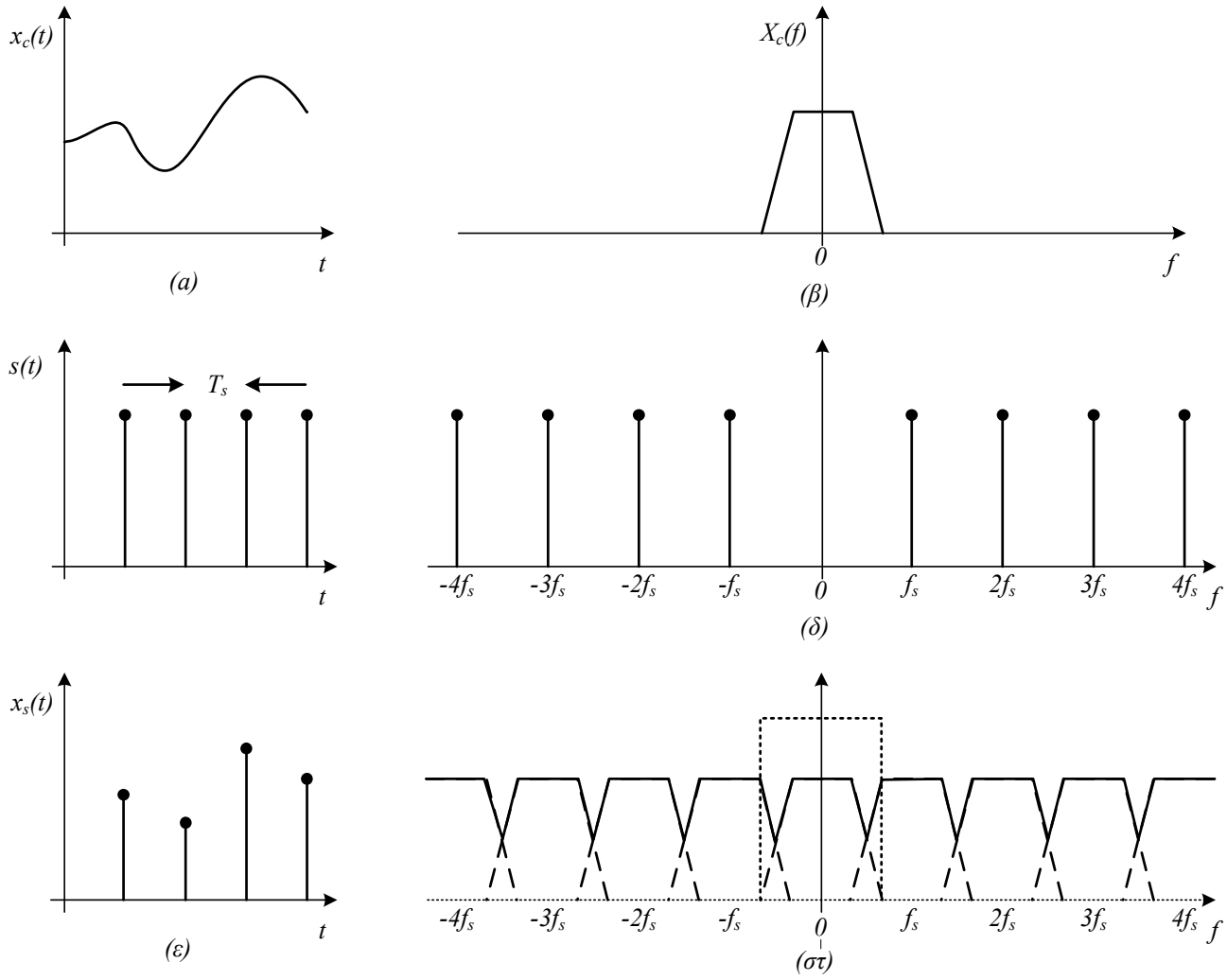
Στο πάνω τμήμα της εικόνας 3.4 φαίνεται το σήμα πληροφορίας στο πεδίο του χρόνου (α) και στο πεδίο των συχνοτήτων (β). Το σήμα δειγματοληψίας είναι μια ακολουθία από κρουστικές συναρτήσεις τόσο στο πεδίο του χρόνου (γ) όσο και στο πεδίο των συχνοτήτων (δ). Επειδή η σχέση μεταξύ περιόδου και συχνότητας είναι ¨ένα προς¨, δηλαδή $f_s = \frac{1}{T_s}$, όσο πιο μικρή η περίοδος δειγματοληψίας και άρα πιο κοντά οι κρουστικές συναρτήσεις στο (γ), τόσο μεγαλύτερη η συχνότητα δειγματοληψίας και άρα πιο απομακρυσμένες μεταξύ τους οι κρουστικές συναρτήσεις στο (δ), και αντίστροφα. Στο τμήμα (ε) φαίνεται το δειγματοληπτημένο σήμα και στο (στ) το φάσμα του. Είναι εμφανές αυτό που αναφέρθηκε

και πιο πριν, ότι δηλαδή πολλαπλασιασμός στο πεδίο του χρόνου σημαίνει συνέλιξη στο πεδίο των συχνοτήτων.

Κατά τη διαδικασία της ανάκτησης του αρχικού σήματος, θα χρησιμοποιηθεί και ένα χαμηλοπερατό φίλτρο, η απόκριση του οποίου έχει σχεδιαστεί στο τμήμα (στ) με διακεκομμένες γραμμές. Το φίλτρο αυτό θα βοηθήσει να απορριφθούν όλες οι εικόνες του βασικού φάσματος (αυτές που βρίσκονται γύρω από της συχνότητες ...,$-2f_s$, $-f_s$, f_s, $2f_s$, …) και να μείνει μόνο η βασική ζώνη. Με αυτόν τον τρόπο θα ανακτηθεί όλο το αρχικό φασματικό περιεχόμενο του σήματος (όχι απαραίτητα με τα ίδια πλάτη ή διαφορά φάσης βέβαια).

3.3.1. Το θεώρημα της δειγματοληψίας

Θα παρουσιαστεί και μια δεύτερη περίπτωση δειγματοληψίας, με διαφορετική συχνότητα f_s αυτή τη φορά. Η διαδικασία φαίνεται στην εικόνα 3.5. Το σήμα πληροφορίας και το φάσμα του είναι το ίδιο με αυτό, της εικόνας 3.4. Επιλέγουμε όμως μεγαλύτερη περίοδο δειγματοληψίας (ή αντίστοιχα μικρότερη συχνότητα δειγματοληψίας). Τα τμήματα (γ) και (δ) εμφανίζονται διαφορετικά απ' ότι στην εικόνα 3.4. Πράγματι στο (γ) οι κρουστικές συναρτήσεις έχουν απομακρυνθεί ενώ στο (δ) έχουν πλησιάσει η μια την άλλη. Παίρνοντας την συνέλιξη του (β) με το (δ) προκύπτει το (στ). Πλέον το φάσμα της βασικής ζώνης (γύρω από το 0) αλληλεπικαλύπτεται με τις εικόνες του γύρω από τις συχνότητες $-f_s$ και f_s. Το ίδιο συμβαίνει σε όλο το μήκος του φάσματος. Αυτό σημαίνει ότι το αρχικό φάσμα αλλοιώθηκε. Όσο καλό να είναι το φίλτρο που θα τοποθετηθεί στην έξοδο (ακόμα και ιδανικό να ήταν) δεν θα είναι δυνατόν να ανακτηθεί το αρχικό φασματικό περιεχόμενο και άρα ούτε το αρχικό σήμα στο χρόνο. Το φαινόμενο αυτό ονομάζεται aliasing και προκύπτει όταν έχουμε υπο-δειγματοληψία.



Εικόνα 3.5 Αναπαράσταση της διαδικασίας της υπό-δειγματοληψίας με τη βοήθεια μαθηματικών σχέσεων

Υπό-δειγματοληψία συμβαίνει όταν η συχνότητα δειγματοληψίας δεν είναι μεγάλη αρκετά ώστε να είναι δυνατή η ανάκτηση του αρχικού σήματος. Ποια όμως είναι η ικανή συχνότητα δειγματοληψίας ώστε να μην εμφανίζεται αυτό το φαινόμενο; Το θεώρημα δειγματοληψίας ή θεώρημα Nyquist – Shannon, ή θεώρημα Whittaker–Nyquist–Kotelnikov–Shannon ή απλώς (και εσφαλμένα) θεώρημα Nyquist δίνει την απάντηση. Σύμφωνα με αυτό θα πρέπει η συχνότητα δειγματοληψίας f_s να είναι μεγαλύτερη από το διπλάσιο της μέγιστης συχνότητας F_{max} του σήματος πληροφορίας, δηλαδή:

$$f_s > 2 \cdot F_{\max}$$

Έτσι, για παράδειγμα, ένα αναλογικό σήμα με μέγιστη συχνότητα τα 8KHz θα πρέπει να περάσει από δειγματοληψία με συχνότητα μεγαλύτερη από 16KHz ώστε να μετατραπεί σε ψηφιακό. Σε αυτή την περίπτωση οι εικόνες του βασικού φάσματος βρίσκονται σε ικανή απόσταση από αυτό ώστε με ένα φίλτρο να είναι δυνατή η επιλογή αυτού και μόνο. Στην πράξη η συχνότητα δειγματοληψίας είναι λίγο μεγαλύτερη από το όριο που θέτει το θεώρημα δειγματοληψίας.

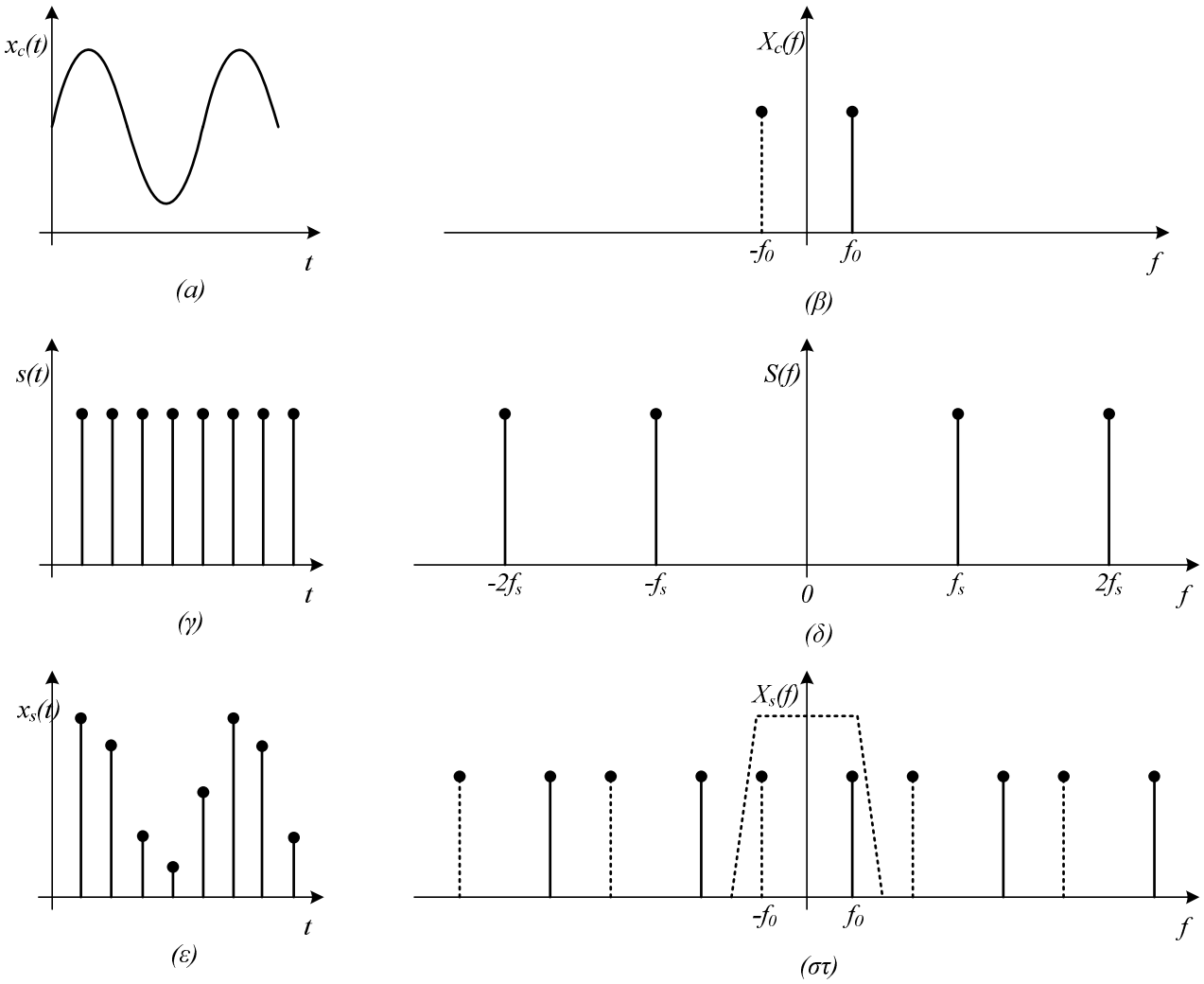
Τυπικές συχνότητες δειγματοληψίας είναι:

- Τηλεφωνία και ήχος τηλεφωνικής ποιότητας. Η μέγιστη συχνότητα περιορίζεται κάτω από 4000Hz και μια συχνότητα δειγματοληψίας 8KHz κρίνεται ικανοποιητική.
- CD ήχου και εφαρμογές ψηφιακού ήχου. Οι περισσότεροι άνθρωποι αντιλαμβάνονται ακουστικές συχνότητες μέχρι 16KHz ενώ κάποιοι μπορεί να φτάσουν μέχρι τα 20KHz. Τα CD ήχου αλλά και άλλες εφαρμογές χρησιμοποιούν τυπικά τα 44,1KHz σαν συχνότητα δειγματοληψίας.

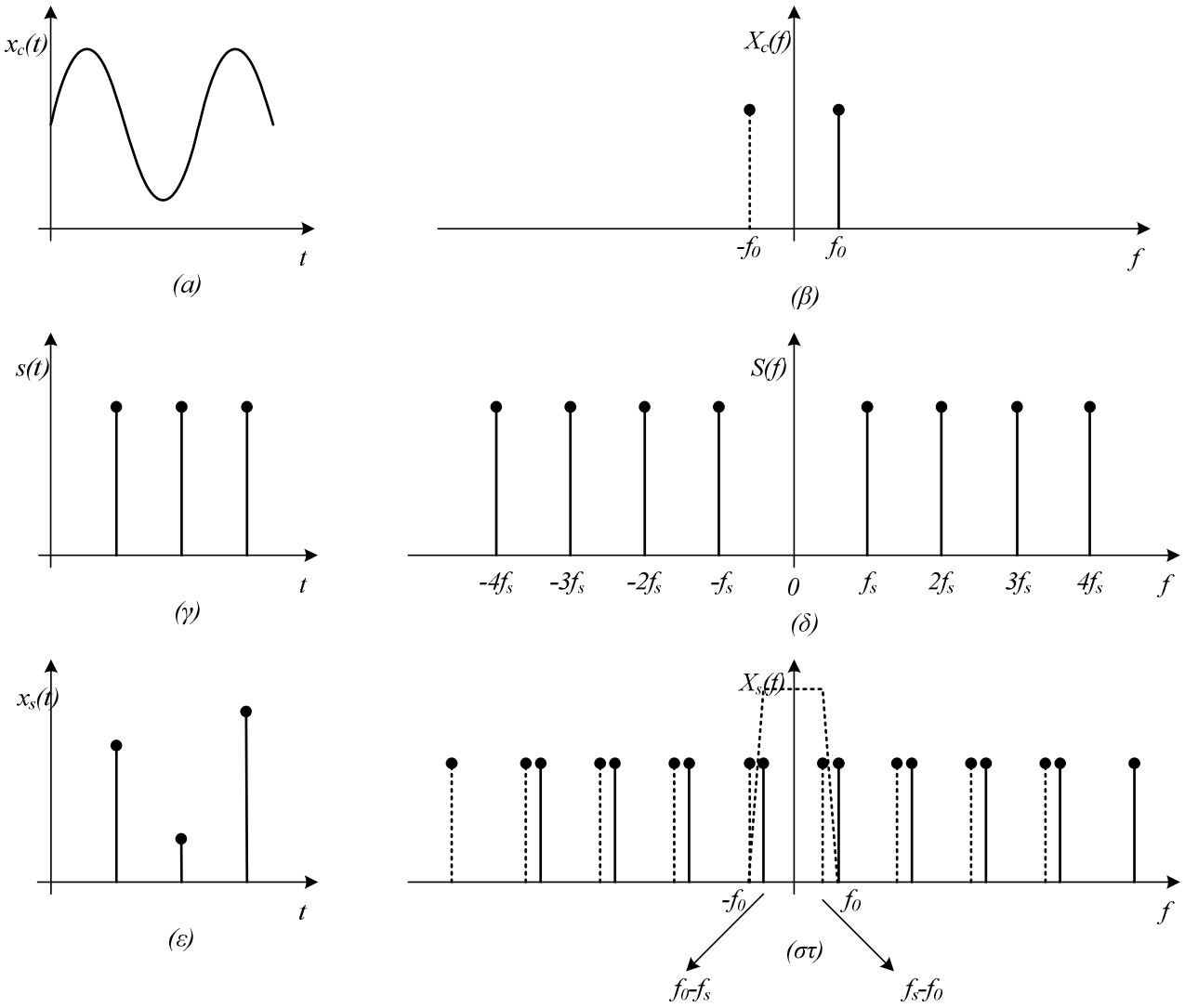
3.3.2. Δειγματοληψία ημιτονικού σήματος

Κλείνοντας το κεφάλαιο θα παρουσιαστεί η ειδική περίπτωση δειγματοληψίας ενός ημιτονικού σήματος συχνότητας f_0. Αυτή η ανάλυση θα χρησιμοποιηθεί στα επόμενα κεφάλαια. Ένα ημιτονικό σήμα έχει φάσμα όπως αυτό που φαίνεται στην εικόνα 3.6 (β). Αποτελείται δηλαδή από 2 φασματικές γραμμές, μία στη συχνότητα f_0 και μια στην - f_0. Η ύπαρξη 2 συχνοτήτων οφείλεται στην μιγαδική μαθηματική έκφραση του ημιτόνου. Αν γίνεται ικανοποιητική δειγματοληψία, τότε τα σήματα και τα φάσματά τους είναι αυτά που φαίνονται στην εικόνα 3.6.

Αν όμως συμβαίνει υπό-δειγματοληψία τότε το αποτέλεσμα φαίνεται στην εικόνα 3.7. Στο φάσμα του δειγματοληπτημένου σήματος παρουσιάζεται και πάλι το φαινόμενο του aliasing. Αν χρησιμοποιηθεί ένα χαμηλοπερατό φίλτρο για να κρατήσει τις φασματικές γραμμές οι οποίες είναι πιο κοντά στην αρχή των αξόνων και να αποκόψει τις υψηλότερες, τότε το αποτέλεσμα είναι ένα ημιτονικό πάλι σήμα με μικρότερη συχνότητα από το αρχικό, την f_s-f_0.



Εικόνα 3.6 Αναπαράσταση της διαδικασίας της ικανοποιητικής δειγματοληψίας ημιτονικού σήματος συχνότητας f_0.



Εικόνα 3.7 Αναπαράσταση της διαδικασίας της υπο-δειγματοληψίας ημιτονικού σήματος συχνότητας f_0.

4. Σχεδίαση της μικροεφαρμογής

4.1. Εισαγωγή

Σε κάθε έργο πληροφορικής (μικρό ή μεγάλο), πριν την υλοποίησή του προηγείται η φάση της σχεδίασης. Ανάλογα με το είδος του έργου και την έκτασή του, η φάση αυτή μπορεί να είναι μια επίπονη και χρονοβόρα διαδικασία ανάλυσης αναγκών και παραμέτρων ή ένα απλό σχέδιο σε ένα χαρτί ή στο μυαλό του προγραμματιστή. Ειδικά στην ανάπτυξη διεπαφών, όπως στην περίπτωση της υλοποίησης μιας μικροεφαρμογής Java, είναι απαραίτητο να υπάρχει μια ξεκάθαρη εικόνα του πώς θα πρέπει να διευθετηθούν τα διάφορα στοιχεία τα οποία θα απαρτίζουν την διεπαφή. Στο κεφάλαιο αυτό θα παρουσιαστεί η ανάλυση η οποία έγινε για την ανάπτυξη της μικροεφαρμογής και το προκαταρκτικό σχέδιο που επιλέχθηκε για την διεπαφή που θα υλοποιηθεί.

4.2. Βασική ανάλυση αναγκών

Όπως έχει ήδη αναφερθεί στην εισαγωγή της εργασίας, στόχος της είναι η εξοικείωση με την ανάπτυξη μικροεφαρμογών java για εκπαιδευτικούς σκοπούς μέσω της υλοποίησης συγκεκριμένου παραδείγματος για μαθητές συγκεκριμένης βαθμίδας εκπαίδευσης. Η βαθμίδα αυτή είναι η Δευτεροβάθμια Τεχνική Εκπαίδευση και συγκεκριμένα οι μαθητές της δευτέρας και της τρίτης τάξης του τομέα Ηλεκτρονικής. Το δε παράδειγμα που θα χρησιμοποιηθεί είναι το θεώρημα δειγματοληψίας του Nyquist, το οποίο παρουσιάστηκε λεπτομερώς στο κεφάλαιο 3. Αυτά όλα θέτουν το πλαίσιο μέσα στο οποίο βρίσκονται οι ανάγκες των χρηστών της μικροεφαρμογής. Πιο συγκεκριμένα μπορούμε να πούμε ότι οι χρήστες έχουν κάποια χαρακτηριστικά τα οποία θα πρέπει να ληφθούν υπόψη για την σχεδίαση της μικροεφαρμογής:

i. Ανάγκη οπτικοποίησης όσο το δυνατόν περισσότερων εννοιών
ii. Ανάγκη ανάλυσης σε επιμέρους βασικά δομικά στοιχεία ή βήματα
iii. Μη επαρκής εξοικείωση με την έννοια του φάσματος συχνοτήτων
iv. Μη επαρκής (σε μεγάλο ποσοστό) εμπέδωση της αντίστροφης σχέσης μεταξύ περιόδου και συχνότητας

Επιπλέον, η μικροεφαρμογή θα πρέπει να τονίζει κάποια στοιχεία του θεωρήματος της δειγματοληψίας του Nyquist τα οποία οι μαθητές θα πρέπει να εμπεδώσουν, όπως:

v. Η συχνότητα δειγματοληψίας θα πρέπει να είναι μεγαλύτερη του διπλάσιου της **μέγιστης** συχνότητας του σήματος πληροφορίας

vi. Η δειγματοληψία μπορεί να θεωρηθεί σαν το γινόμενο του σήματος εισόδου με μια σειρά από συναρτήσεις δ, με περίοδο T_s ίση με $1/f_s$.

vii. Η δειγματοληψία με περίοδο μικρότερη της f_s οδηγεί σε λάθος απεικόνιση του σήματος πληροφορίας

viii. Η λάθος απεικόνιση οφείλεται στο φαινόμενο του aliasing το οποίο είναι εμφανές αν κάνουμε αναπαράσταση των σημάτων στο πεδίο των συχνοτήτων

Με βάση τις παραπάνω επισημάνσεις θα πρέπει αρχικά να γίνει ο σχεδιασμός της μικροεφαρμογής.

Η επισήμανση (i) μας επιβάλει την ανάγκη ύπαρξης γραφικών παραστάσεων καθώς αυτός είναι ο πιο άμεσος τρόπος για τους μαθητές οποιασδήποτε βαθμίδας να αντιληφθούν τον τρόπο με τον οποίο επιδρά η διαδικασία της δειγματοληψίας στο σήμα πληροφορίας s(t).

Πόσες γραφικές παραστάσεις όμως θα χρειαστούν για την αναπαράσταση της διαδικασίας της δειγματοληψίας; Θα μπορούσε κανείς να χρησιμοποιήσει μόνο ένα γράφημα στο οποίο θα απεικονίζονται το σήμα πληροφορίας s(t) (για το οποίο θα χρησιμοποιείται και ο όρος ***σήμα εισόδου*** καθώς αποτελεί την είσοδο για το κύκλωμα δειγματοληψίας), τα δείγματα που προκύπτουν από τη δειγματοληψία καθώς και το σήμα το οποίο θα προέκυπτε από την ανασύσταση του δειγματοληπτημένου σήματος πληροφορίας (για το οποίο θα χρησιμοποιείται και ο όρος ***σήμα εξόδου*** καθώς αποτελεί την έξοδο της διαδικασίας ανάκτησης του σήματος εισόδου), όλα με διαφορετικό χρώμα. Μια τέτοια αναπαράσταση είναι πολύ συμπαγής και επεξηγηματική αλλά όχι για μαθητές της Β΄ και της Γ΄ τάξης των ΕΠΑ.Λ. Η μη διάκριση του σήματος πληροφορίας, του σήματος δειγματοληψίας, του δειγματοληπτημένου σήματος πληροφορίας αλλά και του σήματος εξόδου που προκύπτει από τα δείγματα που λήφθηκαν, δεν θα βοηθήσει στην εύκολη αντίληψη του θεωρήματος του Nyquist από τους μαθητές. Υπάρχει η ανάγκη τουλάχιστον το σήμα πληροφορίας s(t) και το σήμα δειγματοληψίας να είναι ξεχωριστά έτσι ώστε ο μαθητής να αντιληφθεί την διαδικασία δειγματοληψίας σαν το γινόμενο των δυο αυτών σημάτων.

Τελικά επιλέχθηκαν να χρησιμοποιηθούν 4 γραφικές παραστάσεις για την καλύτερη απεικόνιση της διαδικασίας. Στην πρώτη γραφική παράσταση θα απεικονίζεται το σήμα πληροφορίας s(t), στη δεύτερη το σήμα δειγματοληψίας, στην τρίτη το δειγματοληπτημένο σήμα και στην τέταρτη το σήμα που προκύπτει από την ανασύσταση του δειγματοληπτημένου σήματος. Με αυτόν τον τρόπο καλύπτονται οι ανάγκες οι οποίες περιγράφονται από τις επισημάνσεις (ii) και (vi). Πράγματι, με αυτή την προσέγγιση, η διαδικασία της δειγματοληψίας αναλύεται στα βασικά βήματα που την αποτελούν ενώ ταυτόχρονα είναι ξεκάθαρο στους μαθητές ότι ουσιαστικά πρόκειται για μια διαδικασία που μπορεί να αναπαρασταθεί σαν το γινόμενο των δυο σημάτων (πληροφορίας και δειγματοληψίας).

Η επόμενη απαίτηση η οποία πρέπει να ικανοποιηθεί είναι η (v) όπου επισημαίνεται ότι θα πρέπει ο μαθητής να σχετίσει τη συχνότητα δειγματοληψίας με τη μέγιστη συχνότητα του σήματος πληροφορίας. Συνήθως για να γίνει επίδειξη της επίδρασης μιας διαδικασίας σε ένα συνεχές αναλογικό σήμα, χρησιμοποιείται σαν παράδειγμα ένα ημίτονο, η μορφή του οποίου είναι γνωστή στους μαθητές από τα Μαθηματικά αλλά και από άλλα μαθήματα. Έτσι και εδώ μπορεί να χρησιμοποιηθεί σαν σήμα πληροφορίας ένα ημίτονο το οποίο υποβάλλεται στη διαδικασία της δειγματοληψίας και σαν αποτέλεσμα προκύπτει ένα ίδιο ημίτονο ή ένα ημίτονο μικρότερης συχνότητας (αν δεν ικανοποιείται η συνθήκη $f_s>2f_{max}$). Μάλιστα ένα ημίτονο δείχνει καλύτερα το φαινόμενο του aliasing. Τι γίνεται όμως αν το σήμα πληροφορίας περιέχει περισσότερες συχνότητες; Το ζητούμενο είναι ο μαθητής να μπορέσει να εξάγει το συμπέρασμα ότι θα πρέπει η δειγματοληψία να γίνει με συχνότητα μεγαλύτερη από το διπλάσιο της μέγιστης συχνότητας του σήματος πληροφορίας. Για να γίνει αυτό θα πρέπει σαν είσοδο να έχουμε ένα σήμα πληροφορίας με 2 ή περισσότερες συχνότητες. Μπορούμε για παράδειγμα να έχουμε το συνδυασμό 2 ημιτόνων διαφορετικών συχνοτήτων. Αυτός είναι ίσως και ο πιο απλός τρόπος να ικανοποιηθεί η απαίτηση (v). Θα πρέπει λοιπόν στην μικροεφαρμογή να υπάρχουν τα απαραίτητα στοιχεία ελέγχου ώστε να είναι δυνατή η επιλογή του σήματος πληροφορίας μεταξύ ενός απλού ημίτονου ή ενός σήματος που να αποτελείται από 2 ή τρία ημίτονα (επιλέχθηκε τελικά η δυνατότητα να υπάρχουν έως και 3 ημίτονα στο σήμα εισόδου αν και με 2 ικανοποιούνταν πλήρως η απαίτηση (v)).

Τα στοιχεία ελέγχου που αναφέρθηκαν πιο πριν για το σήμα πληροφορίας s(t), το οποίο αποτελείται από ένα έως τρία ημίτονα, δεν είναι τίποτε άλλο από βασικά στοιχεία ελέγχου μιας διεπαφής, όπως ολισθητές (sliders) και κουτιά επιλογής (check boxes). Οι sliders θα

επιτρέπουν στον χρήστη (μαθητή) να μεταβάλλει την συχνότητα του κάθε ημιτόνου, πράγμα που θα έχει αντίκτυπο στην γραφική παράσταση του σήματος πληροφορίας, του δειγματοληπτημένου σήματος αλλά και του σήματος εξόδου (από την ανασύσταση του δειγματοληπτημένου σήματος). Τα check boxes θα χρησιμοποιούνται για να επιλέγονται τα ημίτονα που θα αποτελούν το σήμα πληροφορίας. Ένας ολισθητής θα χρησιμοποιείται και για την επιλογή της συχνότητας δειγματοληψίας.

Η απαίτηση (vii) ικανοποιείται με όσα περιγράφηκαν πιο πάνω. Αν ο χρήστης επιλέξει συχνότητα δειγματοληψίας τέτοια ώστε να εμφανίζεται το φαινόμενο του aliasing, τότε το σήμα εξόδου θα είναι διαφορετικό από το σήμα εισόδου. Στην περίπτωση όπου το σήμα εισόδου είναι ένα απλό ημίτονο, ο μαθητής θα μπορεί να εξάγει και τη σχέση που συνδέει την συχνότητα του σήματος εισόδου, τη συχνότητα του σήματος δειγματοληψίας και τη συχνότητα του σήματος εξόδου, αν και κάτι τέτοιο δεν είναι στους στόχους των μαθημάτων του ΕΠΑ.Λ. για τα οποία αναπτύσσεται η μικροεφαρμογή.

Η απαίτηση (viii) είναι και αυτή εκτός των στόχων των μαθημάτων του ΕΠΑ.Λ. Αν και οι μαθητές της Γ΄ τάξης του Ηλεκτρονικού τομέα των ΕΠΑ.Λ. διδάσκονται για το φάσμα ενός σήματος (ημιτονικού ή πιο σύνθετου) η χρήση της γνώσης αυτής περιορίζεται στα συστήματα αναλογικής διαμόρφωσης (AM ή FM). Για το λόγο αυτό η αναπαράσταση του φάσματος κάθε σήματος δεν αποτελεί ουσιαστική απαίτηση από την μικροεφαρμογή. Η παρουσίαση όμως των φασμάτων μπορεί να βοηθήσει στην κατανόηση της αλλοίωσης της συχνότητας στο σήμα εξόδου λόγω του φαινομένου του aliasing. Λόγω όμως της επισήμανσης (iii), επαφίεται στον διδάσκοντα αν θα προσπαθήσει να εξηγήσει το φαινόμενο ή απλώς θα περιοριστεί σε αυτά που προβλέπονται από το αναλυτικό πρόγραμμα σπουδών.

Πάντως η μικροεφαρμογή μπορεί να βοηθήσει στην εμπέδωση της σχέσης $f = \frac{1}{T}$ (iv) καθώς αυξάνοντας της συχνότητα του σήματος εισόδου ή του σήματος δειγματοληψίας ο μαθητής θα παρατηρεί ότι η περίοδός τους μικραίνει και το αντίστροφο.

4.3. Σχεδίαση της διεπαφής

Σύμφωνα με την προηγούμενη παράγραφο, η διεπαφή χρήστη για την μικροεφαρμογή θα πρέπει να περιλαμβάνει:

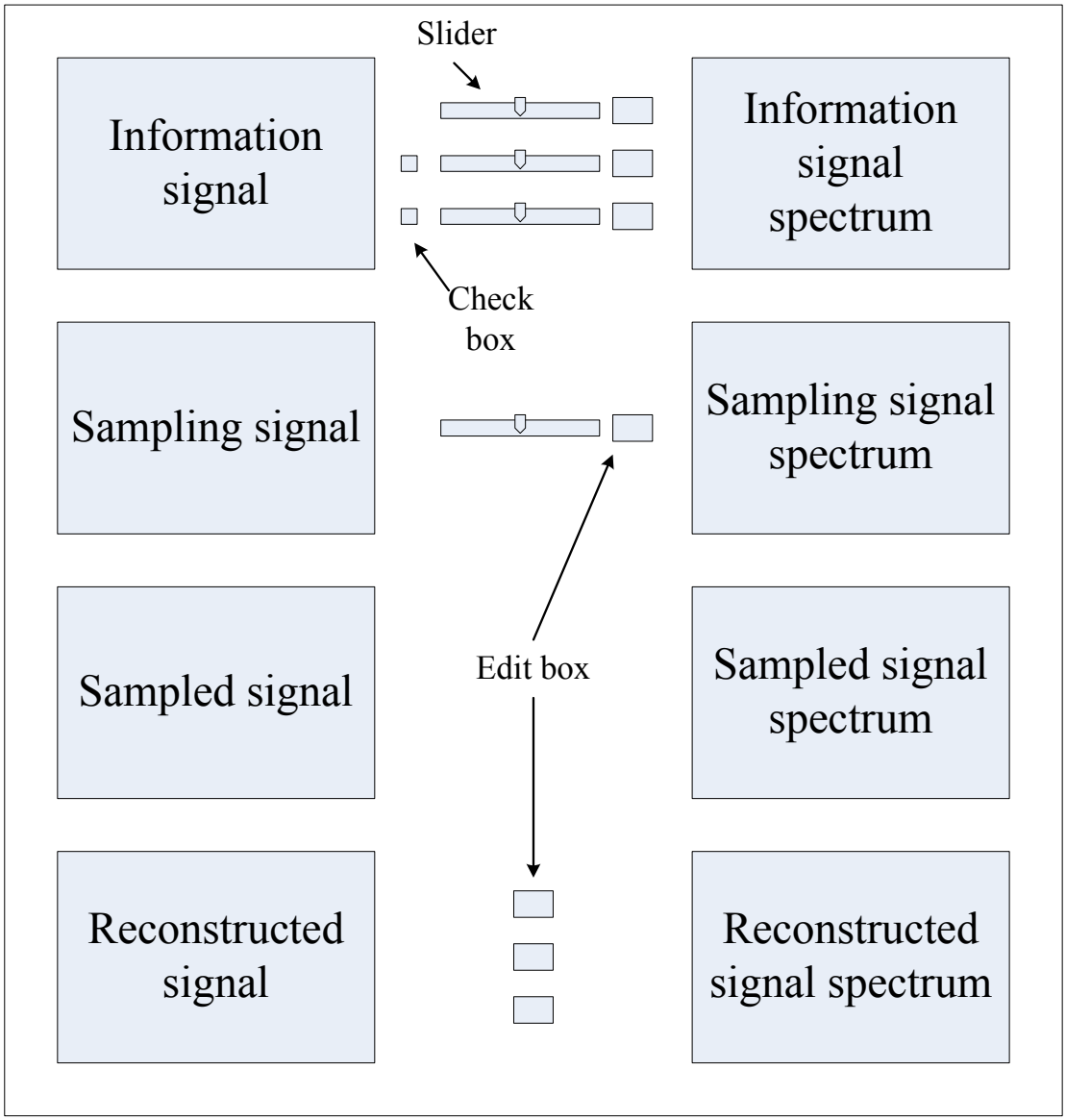
- μια γραφική παράσταση για το σήμα πληροφορίας
- μια γραφική παράσταση για το σήμα δειγματοληψίας
- μια γραφική παράσταση για το δειγματοληπτημένο σήμα
- μια γραφικά παράσταση για το σήμα εξόδου που προκύπτει από την ανασύσταση του δειγματοληπτημένου σήματος
- 3 sliders για τον έλεγχο του σήματος πληροφορίας (3 ημίτονα το πολύ)
- 2 check boxes με τα οποία θα επιλέγεται κάθε φορά αν το σήμα πληροφορίας θα αποτελείται από 1, 2 ή 3 συχνότητες
- 1 slider για την επιλογή της συχνότητας δειγματοληψίας

Επιπλέον αυτών, μπορεί να περιλαμβάνει ακόμα 4 γραφικές παραστάσεις όπου θα απεικονίζονται τα φάσματα των 4 σημάτων που αναφέρθηκαν πιο πριν καθώς και edit boxes (πλαίσια κειμένου) όπου θα αναγράφονται οι συχνότητες του σήματος πληροφορίας, του σήματος δειγματοληψίας και του σήματος εξόδου.

Το ζητούμενο τώρα είναι το πώς θα τοποθετηθούν τα στοιχεία αυτά στην διεπαφή ώστε η τελευταία να είναι λειτουργική. Η τοποθέτηση μπορεί να γίνει κατά δυο τρόπους: οριζόντια και κατακόρυφα.

4.3.1. Κατακόρυφη διάταξη

Στην κατακόρυφη διάταξη τα γραφήματα του σήματος πληροφορίας, του σήματος δειγματοληψίας, του δειγματοληπτημένου σήματος και του σήματος εξόδου, τοποθετούνται το ένα κάτω από το άλλο. Αντίστοιχα και τα γραφήματα για τα φάσματά τους. Τα στοιχεία ελέγχου (sliders, check boxes, edit boxes) τοποθετούνται ενδιάμεσα, όπως στο επόμενο σχήμα.



Εικόνα 4.1 Κατακόρυφη διάταξη της διεπαφής

Η διάταξη αυτή παρουσιάζει δυο σημαντικά πλεονεκτήματα αλλά δυστυχώς και ένα ουσιαστικό μειονέκτημα. Τα πλεονεκτήματα είναι:

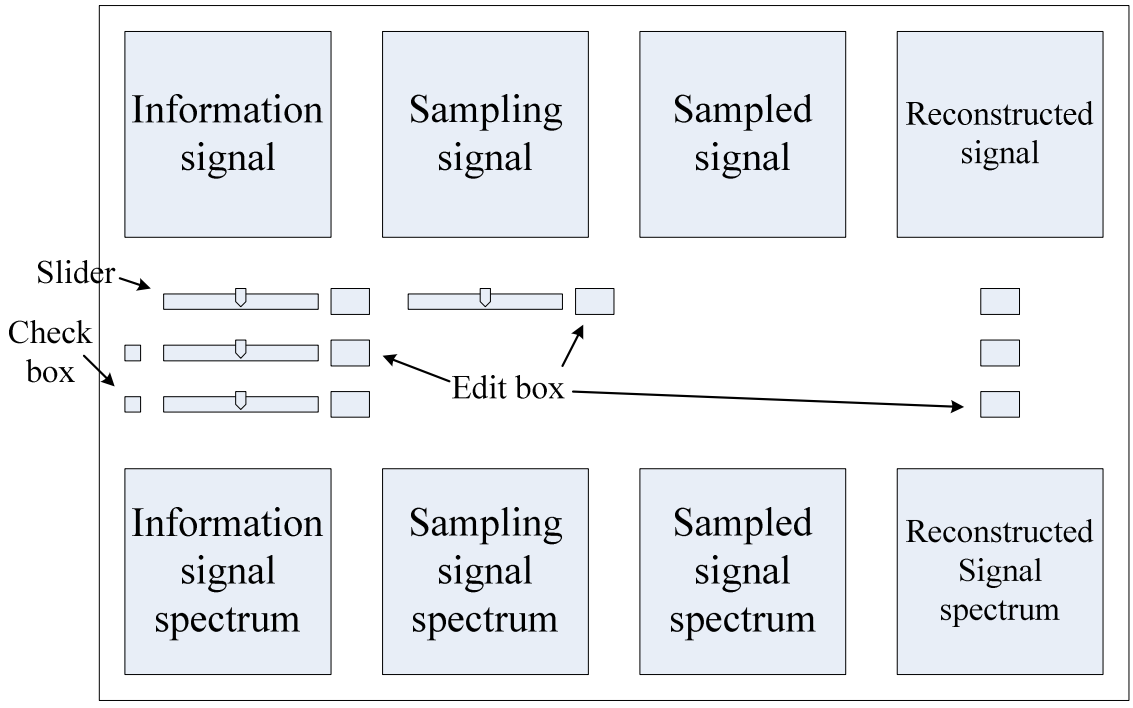
- Το πλάτος του κάθε γραφήματος είναι σημαντικό (μπορεί να φτάσει στο 1/3 του πλάτους της οθόνης). Έτσι μπορούν να απεικονιστούν περισσότεροι κύκλοι του κάθε περιοδικού σήματος. Επιπλέον πιο ευρύ γράφημα σημαίνει καλύτερη σχεδίαση καθώς ο οριζόντιος άξονας έχει περισσότερα σημεία (pixel).

- Η διάταξη από πάνω προς τα κάτω δίνει καλύτερη απεικόνιση των σχέσεων μεταξύ των σημάτων. Έτσι το δειγματοληπτημένο σήμα φαίνεται καλύτερα ότι προκύπτει από το γινόμενο του σήματος πληροφορίας με το σήμα δειγματοληψίας. Και στη συνέχεια το σήμα εξόδου προκύπτει από τα δείγματα του δειγματοληπτημένου σήματος (όπως στις εικόνες 3.4-3.7)

Το μειονέκτημα όμως που παρουσιάζει η διάταξη αυτή και το οποίο την κάνει δύσχρηστη για την μικροεφαρμογή είναι η μεγάλη διάσταση στον κατακόρυφο άξονα. Έτσι αυτή η διάταξη θα καταλαμβάνει περισσότερες από μια οθόνες στον κατακόρυφο άξονα, καθιστώντας την εφαρμογή δύσχρηστη κα ακαλαίσθητη.

4.3.2. Οριζόντια διάταξη

Στην οριζόντια διάταξη τα γραφήματα του σήματος πληροφορίας, του σήματος δειγματοληψίας, του δειγματοληπτημένου σήματος και του σήματος εξόδου, τοποθετούνται το ένα δίπλα από το άλλο. Αντίστοιχα και τα γραφήματα για τα φάσματά τους. Τα στοιχεία ελέγχου (sliders, check boxes, edit boxes) τοποθετούνται ενδιάμεσα, όπως στο επόμενο σχήμα.



Εικόνα 4.2 Οριζόντια διάταξη της διεπαφής

Η διάταξη αυτή έχει το πλεονέκτημα ότι μπορεί να χωρέσει σε μια οθόνη αλλά υστερεί στα 2 σημεία που είχε σαν πλεονέκτημα η κατακόρυφη διάταξη. Έτσι το πλάτος κάθε γραφήματος περιορίζεται σημαντικά (ώστε να χωρέσουν και τα 4 στην οριζόντια διάσταση της οθόνης). Σαν αποτέλεσμα όμως περιορίζεται ο αριθμός των κύκλων των περιοδικών σημάτων οι οποίοι μπορούν να απεικονιστούν. Επιπλέον υπάρχει και απώλεια ακρίβειας λόγω περιορισμένου αριθμού pixels στον οριζόντιο άξονα.

Αν και η οριζόντια διάταξη υστερεί σε 2 σημαντικά σημεία, είναι αυτή που τελικά υιοθετήθηκε καθώς οδήγησε στην σχεδίαση μιας διεπαφής η οποία να μπορεί να προβάλλεται ολόκληρη σε μια οθόνη χωρίς να χρειάζεται ο χρήστης να κάνει scroll αριστερά/δεξιά ή πάνω/κάτω.

5. Ανάπτυξη πρωτοτύπου σε MATLAB®

5.1. Εισαγωγή

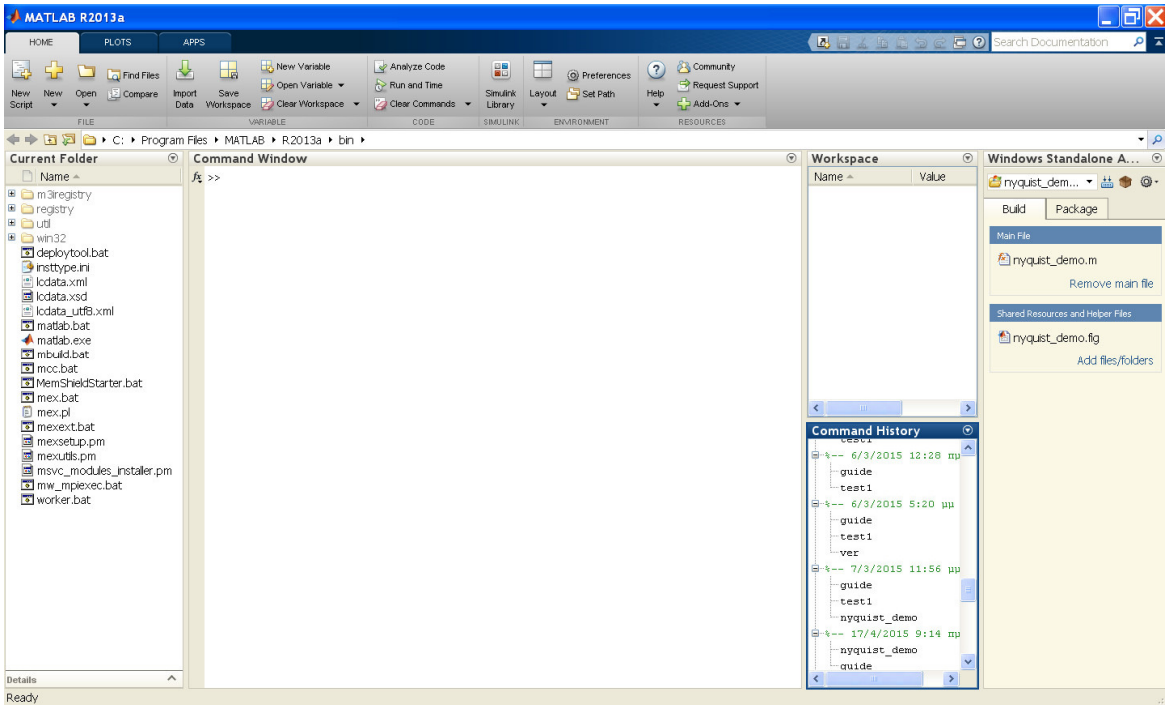
Μετά τη σχεδίαση ενός έργου πληροφορικής και πριν την υλοποίησή του, αρκετές φορές παρεμβάλλεται μια φάση δημιουργίας ενός πρωτοτύπου. Η δημιουργία του πρωτοτύπου μπορεί να είναι μέρος του μοντέλου ανάπτυξης του έργου όπου το πρωτότυπο αυτό συνεχώς βελτιώνεται και προσαρμόζεται στις ανάγκες του περιβάλλοντος μέχρι να ικανοποιεί πλήρως όλες τις απαιτήσεις. Υπάρχουν φορές όμως που η δημιουργία του πρωτότυπου είναι ξεχωριστή από αυτή της υλοποίησης του έργου. Σε αυτές τις περιπτώσεις το πρωτότυπο εξυπηρετεί συγκεκριμένους σκοπούς του σχεδιαστή ή του προγραμματιστή. Για παράδειγμα μπορεί να είναι μια πλατφόρμα δοκιμής αλγορίθμων σε κάποιο μαθηματικό εργαλείο πριν την υλοποίησή τους σε μια γλώσσα προγραμματισμού. Έτσι και στην περίπτωση που μελετάται εδώ, πριν την υλοποίηση της μικροεφαρμογής Java σε κάποιο περιβάλλον ανάπτυξης προγραμμάτων Java, δημιουργήθηκε ένα πρωτότυπό της με τη βοήθεια του εργαλείου MATLAB® της MathWorks. Φυσικά, το μέγεθος της εφαρμογής που αναπτύσσεται δεν επιβάλλει μια τέτοια προσέγγιση, τη δημιουργία δηλαδή ενός πρωτοτύπου σε άλλη γλώσσα περιγραφής. Η μεγαλύτερη άνεση όμως που είχαν οι σπουδαστές με την χρήση του MATLAB®, απ' ότι με την υλοποίηση μικροεφαρμογών Java οδήγησε στην απόφαση να δημιουργηθεί ένα πρωτότυπο όπου θα δοκιμάζονταν τόσο η χωροθέτηση των διάφορων στοιχείων που θα απαρτίζουν την μικροεφαρμογή όσο και ο τρόπος με τον οποίο θα αλληλεπιδρούσε ο χρήστης με αυτήν.

5.2. Το πρόγραμμα MATLAB®

Το MATLAB αποτελεί μια γλώσσα προγραμματισμού τέταρτης γενιάς που αναπτύχθηκε και εξελίσσεται από τη MathWorks. Το όνομά του προέρχεται από τις λέξεις **MAT**rix **LAB**oratory. Όπως μαρτυρεί και το όνομά του είναι εξειδικευμένο στην διαχείριση πινάκων αλλά ταυτόχρονα προσφέρει πολλές άλλες δυνατότητες (γραφικές παραστάσεις, δημιουργία διεπαφών χρήστη, προγραμματισμός και πολλά άλλα).

Αυτό που έκανε το MATLAB χρήσιμο στην δημιουργία του πρωτοτύπου είναι η δυνατότητα που παρέχει για την κατασκευή εφαρμογών με GUI (Graphical User Interface). Επειδή αυτός

ακριβώς είναι και ο στόχος της εργασίας, η σκέψη να χρησιμοποιηθεί το συγκεκριμένο λογισμικό ήρθε σχεδόν αβίαστα. Στην επόμενη εικόνα φαίνεται το περιβάλλον εργασίας του MATLAB.

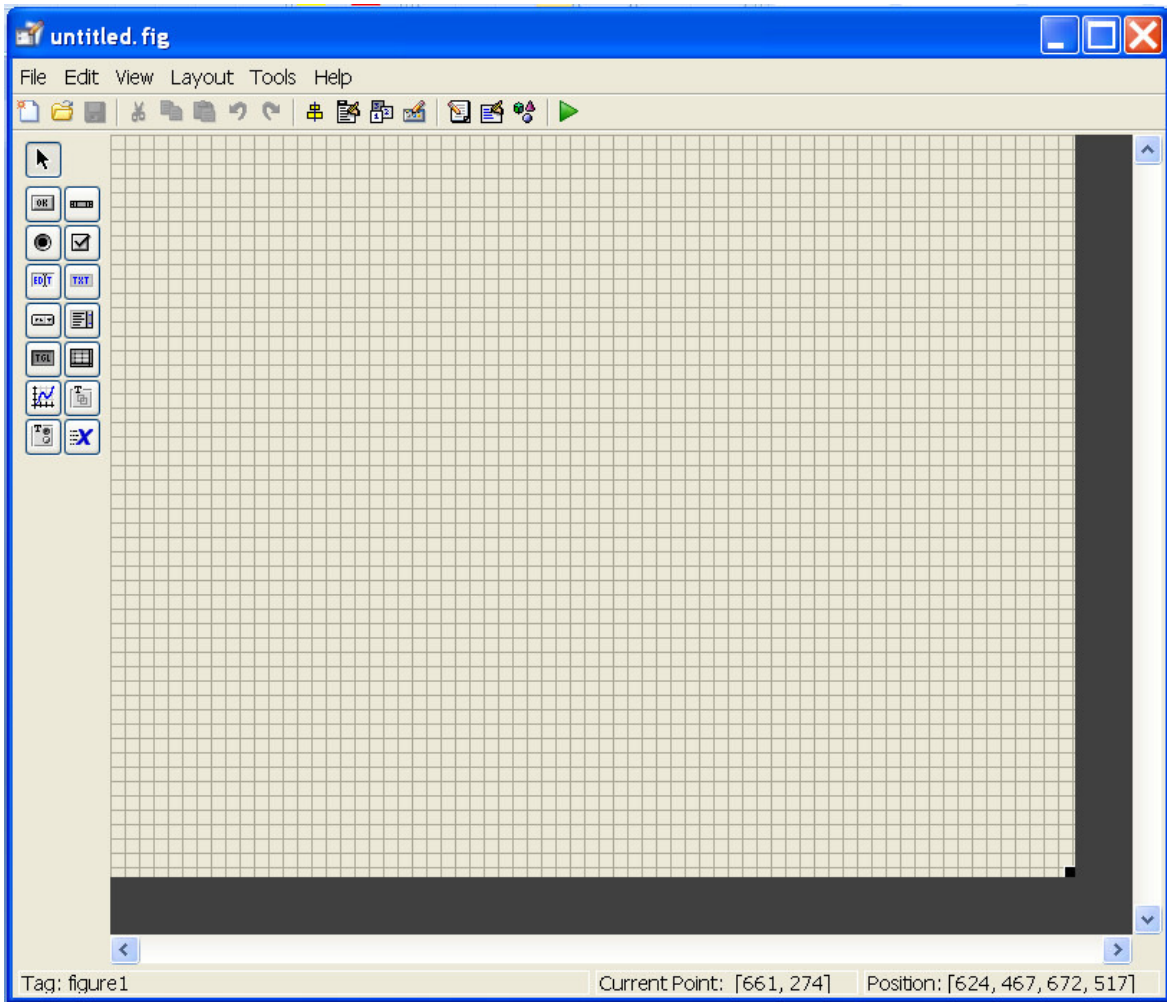


Εικόνα 5.1 Το περιβάλλον εργασίας του MATLAB

Στην κονσόλα του MATLAB μπορεί κανείς να γράψει εντολές και να τις δει να εκτελούνται με το πάτημα του Enter.

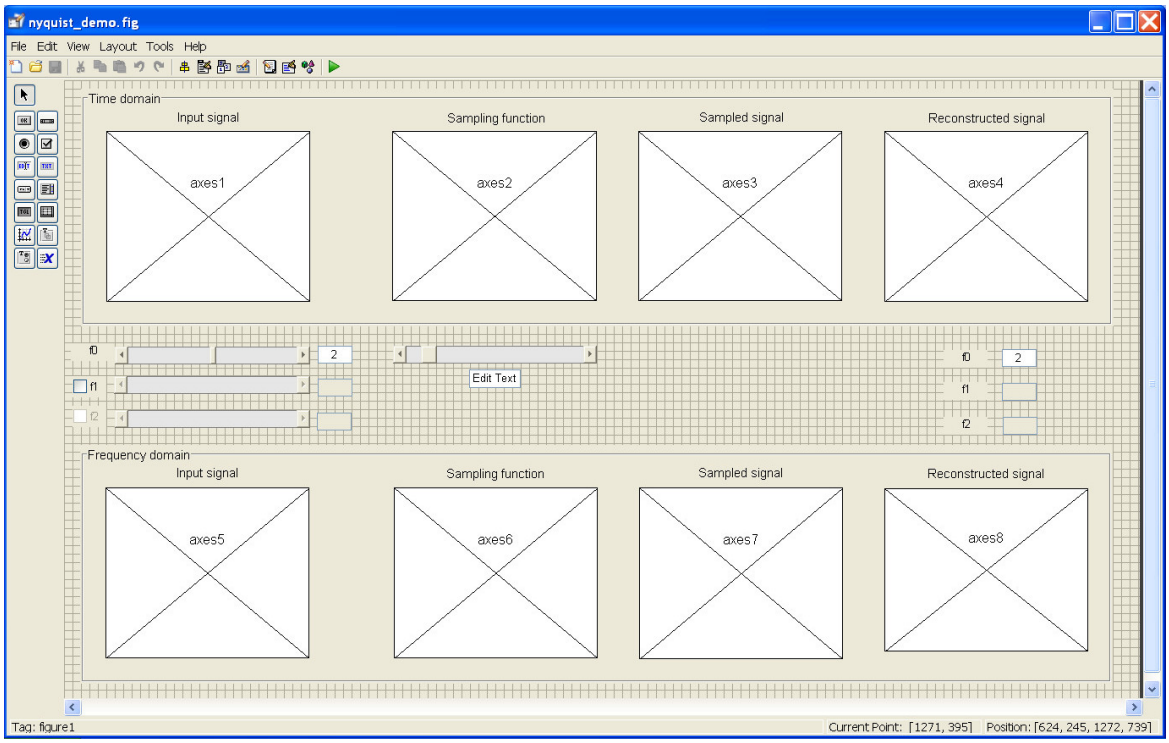
5.2.1. Ο σχεδιαστής διεπαφών του MATLAB

Το εργαλείο του MATLAB που χρησιμοποιήθηκε για την σχεδίαση του πρωτοτύπου είναι ο σχεδιαστής διεπαφών (GUI Design Editor) ή αλλιώς GUIDE το οποίο εκτελείται γράφοντας guide στην κονσόλα του MATLAB. Εάν κανείς επιλέξει να δημιουργήσει ένα κενό GUI, εμφανίζεται το επόμενο παράθυρο.



Εικόνα 5.2 Το παράθυρο του GUI Design Editor

Με τα εργαλεία που παρέχει ο GUI Design Editor κατασκευάστηκε η διεπαφή χρήστη για το πρωτότυπο της μικροεφαρμογής. Η διεπαφή είναι αυτή που φαίνεται στην επόμενη εικόνα και αποτελεί ουσιαστικά μεταφορά στο περιβάλλον του MATLAB του σχήματος 4.2.



Εικόνα 5.3 Η διεπαφή χρήστη όπως σχεδιάστηκε στο GUIDE

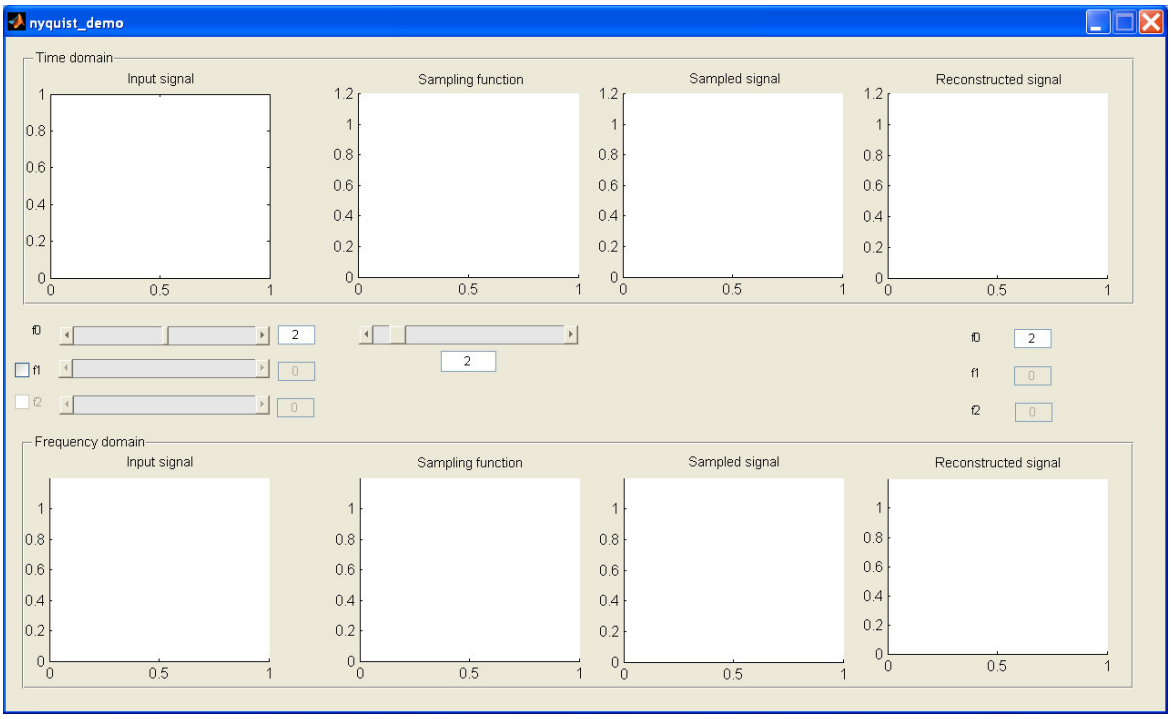
Η διεπαφή περιλαμβάνει όλα τα γραφήματα που προβλέπονται από το σχήμα 4.2 καθώς και τους ολισθητές, τα πλαίσια κειμένου και τα check boxes. Η διεπαφή αποθηκεύεται σαν figure του MATLAB (επέκταση .fig). Επιλέγοντας εκτέλεση του figure παράγεται το αποτέλεσμα που φαίνεται στην εικόνα 5.4. Το αρχείο αποθηκεύθηκε με το όνομα nyquist_demo.fig.

Παρατηρεί κανείς ότι τα γραφήματα είναι κενά ενώ μετακινώντας τους ολισθητές δεν θα παραγόταν κανένα αποτέλεσμα. Αυτό είναι φυσικά αναμενόμενο καθώς θα πρέπει να γραφεί ο απαραίτητος κώδικας ώστε η διεπαφή να καταστεί λειτουργική.

Παρόλα αυτά υπάρχουν κάποιες τιμές σε ορισμένα πεδία της διεπαφής. Για παράδειγμα η θέση του ολισθητή που αντιστοιχεί στην συχνότητα f0 και το αντίστοιχο πλαίσιο κειμένου. Το πλαίσιο κειμένου που αντιστοιχεί στην συχνότητα εξόδου f0 έχει πάλι κάποια τιμή. Οι τιμές αυτές τέθηκαν μέσω των ιδιοτήτων των αντίστοιχων αντικειμένων. Για παράδειγμα στην εικόνα 5.5 φαίνεται ότι ο ολισθητής slider0 (αντιστοιχεί στην συχνότητα f0) παίρνει τιμές από 0 (Min) έως 4 (Max) και η αρχική του τιμή (value) είναι το 2.

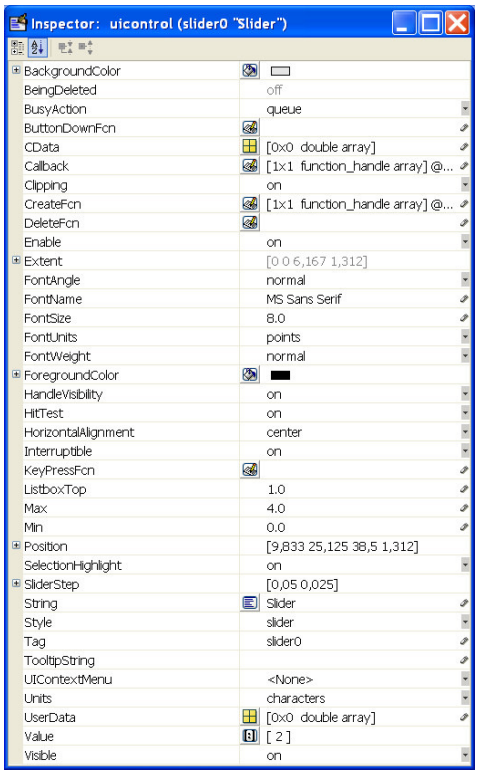
Αντίστοιχα στο πλαίσιο κειμένου που αντιστοιχεί στη συχνότητα f0 υπάρχει η τιμή 2 που έχει τεθεί μέσω της ιδιότητας String του αντίστοιχου αντικειμένου, όπως φαίνεται και στην εικόνα 5.6.

Αυτά βέβαια θα μπορούσαν να είχαν παραληφθεί καθώς όταν εκτελείται το figure θα υπερκεραστούν μέσω του κώδικα που θα γραφεί.

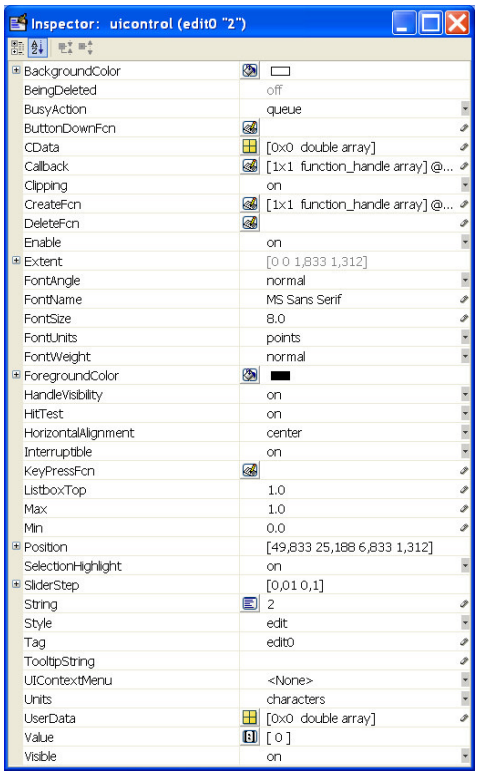


Εικόνα 5.4 Η διεπαφή χρήστη μετά την εκτέλεση του αντίστοιχου figure.

Η μορφή του κώδικα που χρησιμοποιείται στο MATLAB μοιάζει με τη γλώσσα προγραμματισμού C++ της οποίας έχει πολλές δομές και χαρακτηριστικά. Όμως το MATLAB διατηρεί γενικά το δικό του συντακτικό. Ο κώδικας αποθηκεύεται σε αρχεία με την επέκταση .m (m files). Για κάθε figure του MATLAB υπάρχει και το αντίστοιχο αρχείο .m το οποίο έχει το ίδιο όνομα με το figure και περιέχει τον κώδικα που εκτελείται όταν χρησιμοποιείται η διεπαφή.



Εικόνα 5.5 Οι ιδιότητες του ολισθητή slider0 που αντιστοιχεί στην συχνότητα f0.



Εικόνα 5.6 Οι ιδιότητες του πλαισίου κειμένου edit0 που αντιστοιχεί στην συχνότητα f0.

5.3. Ο κώδικας σε MATLAB

Όπως αναφέρθηκε ήδη, για να είναι λειτουργική η διεπαφή που κατασκευάστηκε θα πρέπει να γραφεί και ο αντίστοιχος κώδικας. Το MATLAB από μόνο του έχει δημιουργήσει ένα .m αρχείο με το ίδιο όνομα με το figure (δηλαδή nyquist_demo.m). Το αρχείο αυτό περιλαμβάνει ένα κομμάτι αρχικοποίησης καθώς και διάφορες συναρτήσεις.

Το κομμάτι της αρχικοποίησης καθώς οι συναρτήσεις `nyquist_demo_OpeningFcn` και `nyquist_demo_OutputFcn` ακολουθούν. Να σημειωθεί ότι από τις δυο συναρτήσεις η πιο χρήσιμη είναι η opening function καθώς σε αυτή θα προστεθεί ο κώδικας ο οποίος πρέπει να εκτελείται όταν ξεκινά η εφαρμογή.

```
function varargout = nyquist_demo(varargin)
% NYQUIST_DEMO MATLAB code for nyquist_demo.fig
%      NYQUIST_DEMO, by itself, creates a new NYQUIST_DEMO or raises the existing
%      singleton*.
%
%      H = NYQUIST_DEMO returns the handle to a new NYQUIST_DEMO or the handle to
%      the existing singleton*.
%
%      NYQUIST_DEMO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in NYQUIST_DEMO.M with the given input arguments.
%
%       NYQUIST_DEMO('Property','Value',...) creates a new NYQUIST_DEMO or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before nyquist_demo_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to nyquist_demo_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help nyquist_demo
```

```
% Last Modified by GUIDE v2.5 08-Mar-2015 00:32:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @nyquist_demo_OpeningFcn, ...
                   'gui_OutputFcn',  @nyquist_demo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before nyquist_demo is made visible.
function nyquist_demo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to nyquist_demo (see VARARGIN)

% Choose default command line output for nyquist_demo
handles.output = hObject;
clc;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes nyquist_demo wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = nyquist_demo_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

Επίσης στον πιο πάνω κώδικα υπάρχει μια δομής δεδομένων με το όνομα handles. Αυτή περιλαμβάνει τα handles, τα οποία είναι ουσιαστικά αναφορές σε διάφορους πόρους ή αντικείμενα, και δεδομένα που μπορεί να ορίσει ο χρήστης. Η δομή handles είναι πολύ χρήσιμη καθώς μπορεί να χρησιμοποιηθεί για τον ορισμό κοινόχρηστων μεταβλητών (κάτι σαν global variables) οι οποίες μπορούν να προσπελαύνονται από διάφορες συναρτήσεις ή αντικείμενα. Φυσικά απαιτείται ιδιαίτερη προσοχή ώστε οι τιμές των μεταβλητών να μην τροποποιούνται όταν δεν είναι απαραίτητο.

Πέρα από τις πιο πάνω συναρτήσεις, το MATLAB δημιουργεί αυτόματα και μια ή δυο συναρτήσεις ακόμα για κάθε αντικείμενο που προστίθεται στο συγκεκριμένο figure. Οι συναρτήσεις αυτές είναι συναρτήσεις δημιουργίας (create functions) και συναρτήσεις callback. Οι τελευταίες δημιουργούνται μόνο για στοιχεία ελέγχου στα οποία ο χρήστης μπορεί να επέμβει μέσω της διεπαφής (δηλαδή, στη συγκεκριμένη εφαρμογή, στους ολισθητές, στα check boxes και στα πλαίσια κειμένου). Αν ο χρήστης μεταβάλει κατά οποιονδήποτε τρόπο κάποιο από αυτά τα στοιχεία, τότε εκτελείται η αντίστοιχη συνάρτηση callback.

Για παράδειγμα, οι συναρτήσεις που δημιουργήθηκαν για τον ολισθητή slider0 (ο οποίος αντιστοιχεί στην συχνότητα f0) είναι οι ακόλουθες:

```
% --- Executes on slider movement.
function slider0_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to slider0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider



% --- Executes during object creation, after setting all properties.
function slider0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if                                          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

Η συνάρτηση callback για το check box που αντιστοιχεί στη συχνότητα f1 είναι η :

```
% --- Executes on button press in freq1.
function freq1_Callback(hObject, eventdata, handles)
% hObject    handle to freq1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of freq1
% freq1_chk = get(hObject,'Value');
```

Τέλος, οι συναρτήσεις που δημιουργήθηκαν για το πλαίσιο κειμένου edit1 (αντιστοιχεί στην συχνότητα f1) είναι οι ακόλουθες:

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.

function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Προσθέτοντας τον κατάλληλο κώδικα σε κάθε συνάρτηση μπορεί να επιτευχθεί η επιθυμητή λειτουργία από την εφαρμογή. Πιο κάτω θα παρουσιαστεί ο κώδικας που γράφτηκε στο MATLAB. Πρώτα όμως θα αναφερθούν τα ονόματα των αντικειμένων που χρησιμοποιήθηκαν. Τα ονόματα αυτά φαίνονται στον επόμενο πίνακα όπου καταχωρείται και το όνομα του αντίστοιχου handle για κάθε αντικείμενο.

Πίνακας 5.1 Τα αντικείμενα που περιλαμβάνει το figure της εφαρμογής

Αντικείμενο	Περιγραφή	Handle
slider0	Ο ολισθητής (slider) για την συχνότητα f0 του σήματος πληροφορίας (εισόδου).	handles.h_sld_0
slider1	Ο ολισθητής (slider) για την συχνότητα f1 του σήματος πληροφορίας (εισόδου)	handles.h_sld_1
slider2	Ο ολισθητής (slider) για την συχνότητα f2 του σήματος πληροφορίας (εισόδου)	handles.h_sld_2

slider_d	Ο ολισθητής (slider) για την συχνότητα δειγματοληψίας f_s	handles.h_sld_s
edit0	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα f0 του σήματος πληροφορίας (εισόδου).	handles.h_edit_0
edit1	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα f1 του σήματος πληροφορίας (εισόδου).	handles.h_edit_1
edit2	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα f2 του σήματος πληροφορίας (εισόδου).	handles.h_edit_2
edit_s	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα δειγματοληψίας f_s .	handles.h_edit_s
edit5	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα f0 του σήματος (εξόδου) που προκύπτει από την ανασύσταση του δειγματοληπτημένου σήματος.	handles.h_edit_o0
edit6	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα f1 του σήματος (εξόδου) που προκύπτει από την ανασύσταση του δειγματοληπτημένου σήματος.	handles.h_edit_o1
edit7	Το πλαίσιο κειμένου όπου γράφεται η συχνότητα f2 του σήματος (εξόδου) που προκύπτει από την ανασύσταση του δειγματοληπτημένου σήματος.	handles.h_edit_o2
freq1	Το check box που ενεργοποιεί/απενεργοποιεί την συχνότητα f1 του σήματος πληροφορίας.	-
freq2	Το check box που ενεργοποιεί/απενεργοποιεί την συχνότητα f2 του σήματος πληροφορίας.	-
axis1	Οι άξονες όπου σχεδιάζεται το σήμα πληροφορίας (εισόδου).	handles.h_axes1
axis2	Οι άξονες όπου σχεδιάζεται το σήμα δειγματοληψίας.	handles.h_axes2
axis3	Οι άξονες όπου σχεδιάζεται το δειγματοληπτημένο	handles.h_axes3

	σήμα.	
axis4	Οι άξονες όπου σχεδιάζεται το σήμα (εξόδου) που προκύπτει από την ανασύσταση του δειγματοληπτημένου σήματος.	handles.h_axes4
axis5	Οι άξονες όπου σχεδιάζεται το φάσμα του σήματος πληροφορίας (εισόδου).	handles.h_axes5
axis6	Οι άξονες όπου σχεδιάζεται το φάσμα του σήματος δειγματοληψίας.	handles.h_axes6
axis7	Οι άξονες όπου σχεδιάζεται το φάσμα του δειγματοληπτημένου σήματος.	handles.h_axes7
axis8	Οι άξονες όπου σχεδιάζεται το φάσμα του σήματος εισόδου.	handles.h_axes8
figure1	Το figure που περιλαμβάνει όλα τα παραπάνω.	handles.h_figure

Επιπλέον των αντικειμένων που φαίνονται στον προηγούμενο πίνακα και των handles σε αυτά (τα οποία handles είναι καταχωρημένα στην ομώνυμη δομή), χρησιμοποιήθηκαν και κάποιες μεταβλητές οι οποίες ορίστηκαν σαν μέλη της δομής handles ώστε να είναι προσπελάσιμες από όλες τις συναρτήσεις του προγράμματος. Οι μεταβλητές αυτές είναι οι ακόλουθες:

Πίνακας 5.2 Οι μεταβλητές που περιλαμβάνονται στη δομή handles

Όνομα μεταβλητής	Περιγραφή
handles.freq0	Αποθηκεύει την συχνότητα f0 του σήματος πληροφορίας (εισόδου).
handles.freq1	Αποθηκεύει την συχνότητα f1 του σήματος πληροφορίας (εισόδου).
handles.freq2	Αποθηκεύει την συχνότητα f2 του σήματος πληροφορίας (εισόδου).
handles.freq_s	Αποθηκεύει την συχνότητα δειγματοληψίας

	f_s.
handles.freq0_f	Αποθηκεύει την συχνότητα f0 του σήματος εξόδου.
handles.freq1_f	Αποθηκεύει την συχνότητα f1 του σήματος εξόδου.
handles.freq2_f	Αποθηκεύει την συχνότητα f2 του σήματος εξόδου.
handles.num_of_sinuses	Ο αριθμός των ημιτόνων από τα οποία αποτελείται το σήμα εισόδου
handles.xmax	Η μέγιστη τιμή του οριζόντιου άξονα όλων των γραφικών παραστάσεων

Μπορούμε τώρα να δούμε τα επιμέρους τμήματα του κώδικα του πρωτοτύπου σε MATLAB.

5.3.1. Η αρχικοποίηση

Η αρχικοποίηση όλων των μεταβλητών και των handles της δομής handles γίνεται στην opening function:

```
% --- Executes just before nyquist_demo is made visible.
function nyquist_demo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to nyquist_demo (see VARARGIN)

% Choose default command line output for nyquist_demo
handles.output = hObject;
clc;

%set initial values
```

```
handles.freq0 = 2;
handles.freq1 = 0;
handles.freq2 = 0;
handles.freq_s = 2;
handles.freq0_f = 0;
handles.freq1_f = 0;
handles.freq2_f = 0;
handles.num_of_sinuses = 1;
handles.xmax = 1;

%update sliders and editboxes
handles.h_sld_0 = findobj('Tag','slider0');
set(handles.h_sld_0, 'Value', handles.freq0);
handles.h_sld_1 = findobj('Tag','slider1');
set(handles.h_sld_1, 'Value', handles.freq1);
handles.h_sld_2 = findobj('Tag','slider2');
set(handles.h_sld_2, 'Value', handles.freq2);
handles.h_edit_0 = findobj('Tag','edit0');
set(handles.h_edit_0, 'String', num2str(handles.freq0));
handles.h_edit_1 = findobj('Tag','edit1');
set(handles.h_edit_1, 'String', num2str(handles.freq1));
handles.h_edit_2 = findobj('Tag','edit2');
set(handles.h_edit_2, 'String', num2str(handles.freq2));
handles.h_sld_s = findobj('Tag','slider_d');
set(handles.h_sld_s, 'Value', handles.freq_s);
handles.h_edit_s = findobj('Tag','edit_s');
set(handles.h_edit_s, 'String', num2str(handles.freq0));

handles.h_edit_o0 = findobj('Tag','edit5');
set(handles.h_edit_o0, 'String', num2str(handles.freq0));
handles.h_edit_o1 = findobj('Tag','edit6');
set(handles.h_edit_o1, 'String', num2str(handles.freq1));
handles.h_edit_o2 = findobj('Tag','edit7');
set(handles.h_edit_o2, 'String', num2str(handles.freq2));

%get axes handles
handles.h_axes1 = findobj('Tag','axes1');
handles.h_axes2 = findobj('Tag','axes2');
```

```
handles.h_axes3 = findobj('Tag','axes3');
handles.h_axes4 = findobj('Tag','axes4');
handles.h_axes5 = findobj('Tag','axes5');
handles.h_axes6 = findobj('Tag','axes6');
handles.h_axes7 = findobj('Tag','axes7');
handles.h_axes8 = findobj('Tag','axes8');

handles.h_figure = findobj('Tag','figure1');
% Update handles structure
guidata(hObject, handles);

update_input_signal(hObject, handles);

update_input_spectrum(hObject, handles)

delta_x = update_sampling_signal(handles.h_figure, handles);

delta_x_f = update_sampling_spectrum(handles.h_figure, handles);

update_sampled_signal(handles.h_figure, handles, delta_x)

update_sampled_spectrum(handles.h_figure, handles, delta_x_f)
```

Για να αποθηκευθούν τα handles των αντικειμένων έπρεπε πρώτα να βρεθούν με κάποιο τρόπο. Ο τρόπος που επιλέχθηκε ήταν μέσω της συνάρτησης findobj(attribute, value) η οποία επιστρέφει το handle του αντικειμένου του οποίου η συγκεκριμένη ιδιότητα (attribute) έχει τη συγκεκριμένη τιμή (value), για παράδειγμα η εντολή:

```
handles.h_sld_0 = findobj('Tag','slider0');
```

βρίσκει το handle του ολισθητή με όνομα slider0 και το αποθηκεύει στη μεταβλητή handles.h_sld_0.

Μετά την απόδοση τιμής σε ένα πεδίο της δομής handles καλό είναι να αποθηκεύεται η δομή και να ενημερώνεται με την εντολή

```
guidata(hObject, handles);
```

ώστε να μπορούν οι υπόλοιπες συναρτήσεις να διαβάζουν την νέα τιμή του συγκεκριμένου πεδίου.

Στο τέλος του κώδικα αρχικοποίησης υπάρχουν και κάποιες συναρτήσεις οι οποίες γράφτηκαν για την ανανέωση των γραφικών παραστάσεων και οι οποίες θα παρουσιαστούν πιο κάτω.

5.3.2. Οι συναρτήσεις για τους ολισθητές

Για κάθε ολισθητή τροποποιήθηκαν οι συναρτήσεις callback ώστε να ενημερώνονται οι γραφικές παραστάσεις και τα αντίστοιχα πλαίσια κειμένου, όταν αυτοί μετακινούνται.

```
% --- Executes on slider movement.
function slider0_Callback(hObject, eventdata, handles)
% hObject    handle to slider0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

    % Retrieve GUI data (the handles structure)
    handles = guidata(hObject);
    % Read slider value
    handles.freq0 = get(hObject,'Value');
    % Update handles structure
    guidata(hObject, handles);

    set(handles.h_edit_0,'String', num2str(handles.freq0));

    update_input_signal(hObject, handles);
    update_input_spectrum(hObject, handles)

    slider_d_Callback(handles.h_sld_s,[],handles);


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

    % Retrieve GUI data (the handles structure)
    handles = guidata(hObject);
    % Read slider value
    handles.freq1 = get(hObject,'Value');
    % Update handles structure
    guidata(hObject, handles);

    set(handles.h_edit_1,'String', num2str(handles.freq1));

    update_input_signal(hObject, handles);
    update_input_spectrum(hObject, handles)

    slider_d_Callback(handles.h_sld_s,[],handles);


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

    % Retrieve GUI data (the handles structure)
    handles = guidata(hObject);
    % Read slider value
    handles.freq2 = get(hObject,'Value');
    % Update handles structure
    guidata(hObject, handles);
```

```
    set(handles.h_edit_2,'String', num2str(handles.freq2));

    update_input_signal(hObject, handles);
    update_input_spectrum(hObject, handles)

    slider_d_Callback(handles.h_sld_s,[],handles);


% --- Executes on slider movement.
function slider_d_Callback(hObject, eventdata, handles)
% hObject    handle to slider_d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

handles.freq_s = get(hObject,'Value');
set(handles.h_edit_s, 'String', num2str(handles.freq_s));

delta_x = update_sampling_signal(hObject, handles);

delta_x_f = update_sampling_spectrum(hObject, handles);


% Update handles structure
guidata(hObject, handles);

drawnow

update_sampled_signal(hObject, handles, delta_x)
update_sampled_spectrum(hObject, handles, delta_x_f)
```

5.3.3. Οι συναρτήσεις για τα check boxes

Όταν τα check boxes για την επιλογή των συχνοτήτων f1 και f2 στο σήμα πληροφορίας απενεργοποιούνται, θα πρέπει να απενεργοποιούνται και οι αντίστοιχοι ολισθητές και τα

πλαίσια κειμένου. Επίσης εκτελούνται και οι callback συναρτήσεις των ολισθητών που παραμένουν ενεργοί ώστε να ξανασχεδιαστούν οι γραφικές παραστάσεις.

```
% --- Executes on button press in freq1.
function freq1_Callback(hObject, eventdata, handles)
% hObject    handle to freq1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of freq1
freq1_chk = get(hObject,'Value');
if (freq1_chk == 1)
    set(handles.h_sld_1,'enable', 'on');
    set(handles.h_edit_1,'enable', 'on');
    set(handles.h_edit_o1,'enable', 'on');
    h_freq2_chk = findobj('Tag','freq2');
    set(h_freq2_chk,'enable', 'on');
    handles.num_of_sinuses = 2;

    % Update handles structure
    guidata(hObject, handles);

    slider1_Callback(handles.h_sld_1,[],handles);

else
    set(handles.h_sld_1,'enable', 'off');
    set(handles.h_edit_1,'enable', 'off');
    set(handles.h_edit_o1,'enable', 'off');
    h_freq2_chk = findobj('Tag','freq2');
    set(h_freq2_chk,'enable', 'off');
    handles.freq1 = 0;
    handles.num_of_sinuses = 1;

    % Update handles structure
    guidata(hObject, handles);

    slider0_Callback(handles.h_sld_0,[],handles);
```

```
end

% --- Executes on button press in freq2.
function freq2_Callback(hObject, eventdata, handles)
% hObject    handle to freq2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of freq2
freq2_chk = get(hObject,'Value');
if (freq2_chk == 1)
    set(handles.h_sld_2,'enable', 'on');
    set(handles.h_edit_2,'enable', 'on');
    set(handles.h_edit_o2,'enable', 'on');
    handles.num_of_sinuses = 3;

    % Update handles structure
    guidata(hObject, handles);

    slider2_Callback(handles.h_sld_2,[],handles);

else
    set(handles.h_sld_2,'enable', 'off');
    set(handles.h_edit_2,'enable', 'off');
    set(handles.h_edit_o2,'enable', 'off');
    handles.freq2 = 0;
    handles.num_of_sinuses = 2;

    % Update handles structure
    guidata(hObject, handles);

    slider1_Callback(handles.h_sld_1,[],handles);
end
```

5.3.4. Οι συναρτήσεις των χρηστών

Οι συναρτήσεις που γράφηκαν από τους σπουδαστές έχουν στόχο την υλοποίηση συγκεκριμένων ενεργειών, κυρίως όμως τον υπολογισμό των συχνοτήτων αναδίπλωσης

(aliasing frequencies) και την ανανέωση των γραφικών παραστάσεων. Ας τις δούμε όμως μια-μια.

Υπάρχει μια συνάρτηση για την σχεδίαση του σήματος πληροφορίας. Η συνάρτηση αυτή διαβάζει τις τιμές των τριών συχνοτήτων του σήματος εισόδου και το σχεδιάζει. Να σημειωθεί ότι θα πρέπει κάθε φορά να ορίζονται τα όρια του x άξονα.

```
function update_input_signal(hObject, handles)


    t = linspace(0,handles.xmax,200);
    y                                                                    =
sin(handles.freq0*2*pi*t)+sin(handles.freq1*2*pi*t)+sin(handles.freq2*2*pi*
t);
    axes(handles.h_axes1)
    plot(t,y);
    set(handles.h_axes1,'XLim', [0 handles.xmax]);
    set(handles.h_axes1,'XLimMode', 'manual');
    drawnow
    % Update handles structure
    guidata(hObject, handles);
```

Μια δεύτερη συνάρτηση σχεδιάζει το φάσμα του σήματος εισόδου. Ανάλογα με το πόσα ημίτονα αποτελούν το σήμα εισόδου ορίζεται και το μέγεθος του πίνακα f_x όπου αποθηκεύονται οι τετμημένες των φασματικών γραμμών.

```
function update_input_spectrum(hObject, handles)


    axes(handles.h_axes5)
    f_x = zeros(1,handles.num_of_sinuses);
    f_y = ones(1,handles.num_of_sinuses);
    f_x(1,1) = handles.freq0;
    if (handles.num_of_sinuses >= 2)
        f_x(1,2) = handles.freq1;
    end
    if (handles.num_of_sinuses == 3)
        f_x(1,3) = handles.freq2;
    end


    stem(f_x,f_y);
```

```
    hold on
    stem(-f_x,f_y,'--','r');
    hold off
    set(handles.h_axes5,'XLim', [-5 5]);
    set(handles.h_axes5,'XLimMode', 'manual');
    set(handles.h_axes5,'YLim', [0 1.05]);
    set(handles.h_axes5,'YLimMode', 'manual');
    drawnow
    % Update handles structure
    guidata(hObject, handles);
```

Η επόμενη συνάρτηση υπολογίζει και σχεδιάζει το σήμα δειγματοληψίας. Καθώς οι τετμημένες του σήματος αυτού χρειάζονται και σε άλλες συναρτήσεις, η συνάρτηση update_sampling_signal, επιστρέφει τον πίνακα με τις τετμημένες.

```
function delta_x = update_sampling_signal(hObject, handles)

    delta_x = ones(1,floor(handles.freq_s));
    delta_y = ones(1,floor(handles.freq_s));
    for i = 1:floor(handles.freq_s)
        delta_x(floor(handles.freq_s) + 1 - i ) = i/handles.freq_s;
    end

    axes(handles.h_axes2)
    stem(delta_x,delta_y);
    set(handles.h_axes2,'XLim', [0 1]);
    set(handles.h_axes2,'XLimMode', 'manual');
    set(handles.h_axes2,'YLim', [0 1.05]);
    set(handles.h_axes2,'YLimMode', 'manual');
    drawnow

    % Update handles structure
    guidata(hObject, handles);
```

Αντίστοιχα υπάρχει και συνάρτηση που υπολογίζει και σχεδιάζει το φάσμα του σήματος δειγματοληψίας. Η τιμή 12 που χρησιμοποιείται στην συνάρτηση αυτή είναι η μέγιστη τιμή του ολισθητή slider_d.

```
function delta_x_spectrum = update_sampling_spectrum(hObject, handles)


    delta_x_spectrum = ones(1,2*floor(12/handles.freq_s)+1);
    delta_y_spectrum = ones(1,2*floor(12/handles.freq_s)+1);
    for i = -floor(12/handles.freq_s):floor(12/handles.freq_s)
        delta_x_spectrum(floor(12/handles.freq_s)   +   1   -   i   )   =
i*handles.freq_s;
    end
    axes(handles.h_axes6)
    stem(delta_x_spectrum,delta_y_spectrum);
    set(handles.h_axes6,'XLim', [-12 12]);
    set(handles.h_axes6,'XLimMode', 'manual');
    set(handles.h_axes6,'YLim', [0 1.05]);
    set(handles.h_axes6,'YLimMode', 'manual');
    drawnow


    % Update handles structure
    guidata(hObject, handles);
```

Η επόμενη συνάρτηση υπολογίζει και σχεδιάζει το δειγματοληπτημένο σήμα. Στην ίδια γραφική παράσταση φαίνεται τόσο το σήμα πληροφορίας, όσο και το σήμα που πρόκειται να εμφανιστεί στην έξοδο. Αν το θεώρημα του Nyquist δεν ικανοποιείται πλήρως, τότε τα δυο αυτά σήματα είναι διαφορετικά και διέρχονται και τα δυο από τα ίδια δείγματα στην γραφική παράσταση.

```
function update_sampled_signal(hObject, handles, delta_x)


    t = linspace(0,handles.xmax,200);
    y                                                                    =
sin(handles.freq0*2*pi*t)+sin(handles.freq1*2*pi*t)+sin(handles.freq2*2*pi*
t);
    y_s                                                                  =
sin(handles.freq0*2*pi*delta_x)+sin(handles.freq1*2*pi*delta_x)+sin(handles
.freq2*2*pi*delta_x);


    if handles.freq_s < 2*handles.freq0
        handles.freq0_f = handles.freq0-handles.freq_s;
        set(handles.h_edit_o0, 'String', num2str(handles.freq0_f));
    else
```

```
        handles.freq0_f = 0;
        set(handles.h_edit_o0, 'String', num2str(handles.freq0));
    end
    if handles.freq_s < 2*handles.freq1
        handles.freq1_f = handles.freq1-handles.freq_s;
        set(handles.h_edit_o1, 'String', num2str(handles.freq1_f));
    else
        handles.freq1_f = 0;
        set(handles.h_edit_o1, 'String', num2str(handles.freq1));
    end
    if handles.freq_s < 2*handles.freq2
        handles.freq2_f = handles.freq2-handles.freq_s;
        set(handles.h_edit_o2, 'String', num2str(handles.freq2_f));
    else
        handles.freq2_f = 0;
        set(handles.h_edit_o2, 'String', num2str(handles.freq2));
    end


    axes(handles.h_axes3);
    stem(delta_x,y_s);
    hold on
    plot(t,y,'r:');


    if (handles.freq0_f ~= 0) & (handles.freq1_f ~= 0) & (handles.freq2_f
~= 0)
        y_r                                                              =
sin(handles.freq0_f*2*pi*t)+sin(handles.freq1_f*2*pi*t)+sin(handles.freq2_f
*2*pi*t);
    elseif   (handles.freq0_f   ~=   0)   &   (handles.freq1_f   ~=   0)   &
(handles.freq2_f == 0)
        y_r                                                              =
sin(handles.freq0_f*2*pi*t)+sin(handles.freq1_f*2*pi*t)+sin(handles.freq2*2
*pi*t);
    elseif   (handles.freq0_f   ~=   0)   &   (handles.freq1_f   ==   0)   &
(handles.freq2_f ~= 0)
        y_r                                                              =
sin(handles.freq0_f*2*pi*t)+sin(handles.freq1*2*pi*t)+sin(handles.freq2_f*2
*pi*t);
    elseif   (handles.freq0_f   ==   0)   &   (handles.freq1_f   ~=   0)   &
(handles.freq2_f ~= 0)
        y_r                                                              =
sin(handles.freq0*2*pi*t)+sin(handles.freq1_f*2*pi*t)+sin(handles.freq2_f*2
```

```
*pi*t);
    elseif (handles.freq0_f ~= 0) & (handles.freq1_f == 0) &
(handles.freq2_f == 0)
        y_r =
sin(handles.freq0_f*2*pi*t)+sin(handles.freq1*2*pi*t)+sin(handles.freq2*2*p
i*t);
    elseif (handles.freq0_f == 0) & (handles.freq1_f == 0) &
(handles.freq2_f ~= 0)
        y_r =
sin(handles.freq0*2*pi*t)+sin(handles.freq1*2*pi*t)+sin(handles.freq2_f*2*p
i*t);
    elseif (handles.freq0_f == 0) & (handles.freq1_f ~= 0) &
(handles.freq2_f == 0)
        y_r =
sin(handles.freq0*2*pi*t)+sin(handles.freq1_f*2*pi*t)+sin(handles.freq2*2*p
i*t);
    end
    if (handles.freq0_f ~= 0) | (handles.freq1_f ~= 0) | (handles.freq2_f
~= 0)
        plot(t,y_r,'g:');
    end
    hold off

    axes(handles.h_axes4);
    if (handles.freq0_f ~= 0) | (handles.freq1_f ~= 0) | (handles.freq2_f
~= 0)
        plot(t,y_r,'r');
    else
        plot(t,y,'r');
    end
    drawnow

    % Update handles structure
    guidata(hObject, handles);
```

Τέλος, υπάρχει και μια συνάρτηση για τον υπολογισμό και τη σχεδίαση του φάσματος του δειγματοληπτημένου σήματος. Για ευκολία στον υπολογισμό του υπάρχει και μια συνάρτηση που εξομοιώνει τη διαδικασία της συνέλιξης.

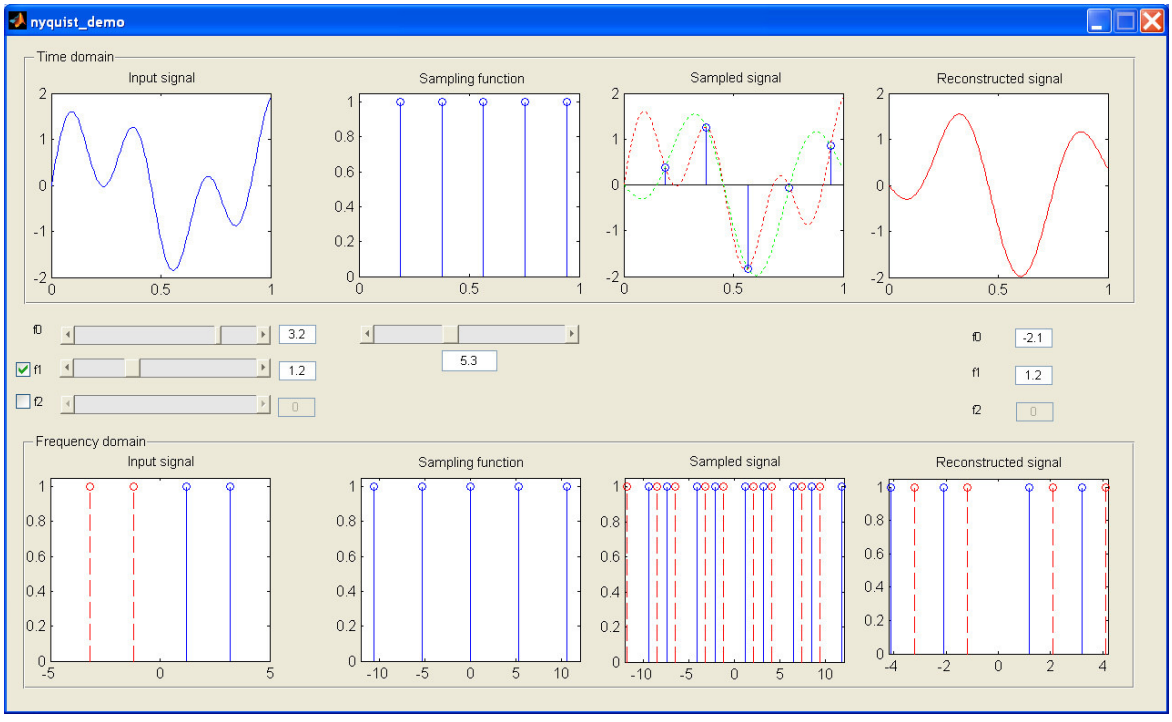
```
function update_sampled_spectrum(hObject, handles, delta_x_f)

    f_x = zeros(1,handles.num_of_sinuses);
```

```
    f_x(1,1)=handles.freq0;
    if handles.num_of_sinuses == 2
        f_x(1,2)=handles.freq1;
    elseif handles.num_of_sinuses == 3
        f_x(1,2)=handles.freq1;
        f_x(1,3)=handles.freq2;
    end

    cvfake = my_fake_conv(delta_x_f,f_x);
    cvfake_img = my_fake_conv(delta_x_f,-f_x);
    axes(handles.h_axes7)
    set(handles.h_axes7,'XLim', [-12 12]);
    stem(cvfake,ones(1,size(cvfake,2)));
    hold on;
    stem(cvfake_img,ones(1,size(cvfake_img,2)),'--','r');
    hold off;
    set(handles.h_axes7,'XLim', [-12 12]);
    set(handles.h_axes7,'XLimMode', 'manual');
    set(handles.h_axes7,'YLim', [0 1.05]);
    set(handles.h_axes7,'YLimMode', 'manual');

    zoom_x_limit = 0.2 + ceil(max(handles.freq0,max(handles.freq1,
handles.freq2)));
    axes(handles.h_axes8)
    set(handles.h_axes8,'XLim', [-zoom_x_limit zoom_x_limit]);
    stem(cvfake,ones(1,size(cvfake,2)));
    hold on;
    stem(cvfake_img,ones(1,size(cvfake_img,2)),'--','r');
    hold off;
    set(handles.h_axes8,'XLim', [-zoom_x_limit zoom_x_limit]);
    set(handles.h_axes8,'XLimMode', 'manual');
    set(handles.h_axes8,'YLim', [0 1.05]);
    set(handles.h_axes8,'YLimMode', 'manual');

    drawnow

    % Update handles structure
    guidata(hObject, handles);
```

```
function c = my_fake_conv(x,y)


    d_x = ones(1,size(x,2)*size(y,2));
    d_y = ones(1,size(y,2));
      for i = 1:size(x,2)
        for j = 1:size(y,2)
            d_x(size(y,2)*i+j-size(y,2)) = x(i)+ y(j);
        end
     end
     c = d_x;
```

5.4. Εκτελώντας τον κώδικα

Κλείνοντας αυτό το κεφάλαιο παρατίθεται και μια εικόνα από τη λειτουργία του πρωτοτύπου που κατασκευάστηκε σε MATLAB. Σε αυτή φαίνεται ότι υπάρχουν 2 συχνότητες στο σήμα πληροφορίας, η 3.3Hz και η 1.2Hz ενώ η συχνότητα δειγματοληψίας είναι 5.3Hz. Στην έξοδο η δεύτερη συχνότητα λαμβάνεται σωστά (1.2Hz) αλλά η πρώτη λαμβάνεται ως -2.1Hz λόγω του φαινομένου της αναδίπλωσης.



Εικόνα 5.7 Το πρωτότυπο σε λειτουργία

6. Υλοποίηση της μικροεφαρμογής Java

6.1. Εισαγωγή

Στο 6^{ο} κεφάλαιο θα περιγραφεί αναλυτικά η υλοποίηση της μικροεφαρμογής σε Java. Αρχικά αναφέρονται οι λόγοι που επιλέχθηκε σαν περιβάλλον υλοποίησης το Eclipse. Ακολουθεί μια περιγραφή της μορφής της μικροεφαρμογής και πως αυτή υλοποιήθηκε μέσα από τη Java. Βασικό ρόλο στην μικροεφαρμογή έχουν τα στοιχεία ελέγχου (ολισθητές, check boxes). Το πώς αυτά αλληλεπιδρούν με την υπόλοιπη εφαρμογή περιγράφεται στην παράγραφο 6.5. Η υλοποίηση των γραφημάτων αλλά και των πλαισίων και των αξόνων είναι το αντικείμενο των υπόλοιπων παραγράφων. Τέλος υπάρχει και μια περιγραφή της λειτουργίας της μικροεφαρμογής σαν σύνολο και πως αυτή επιτυγχάνει τους στόχους της.

6.2. Επιλογή του περιβάλλοντος ανάπτυξης

Όταν κανείς ξεκινά την υλοποίηση ενός έργου πληροφορικής, είτε αυτό είναι η ανάπτυξη μιας εφαρμογής ή η υλοποίηση μιας βάσης δεδομένων ή οτιδήποτε άλλο, θα πρέπει να επιλέξει τα εργαλεία που θα χρησιμοποιήσει. Η ραγδαία ανάπτυξη της τεχνολογίας όσον αφορά τις επιστήμες της πληροφορικής αλλά και η ευρεία διάδοση που έχουν, έχει οδηγήσει στην ανάπτυξη πολλών προγραμμάτων και εργαλείων για την διευκόλυνση των προγραμματιστών και των χρηστών γενικότερα που ασχολούνται με την ανάπτυξη εφαρμογών. Υπάρχει μια πληθώρα από εργαλεία. Άλλα δωρεάν και άλλα εμπορικά. Άλλα για λειτουργικά συστήματα ανοικτού κώδικα και άλλα για τα κυρίαρχα εμπορικά λειτουργικά συστήματα. Άλλα αναπτύσσονται από ιδιώτες ή ομάδες χρηστών και άλλα από εταιρείες. Έτσι και στην ανάπτυξη εφαρμογών Java, ο χρήστης έχει να διαλέξει μεταξύ πολλών επιλογών. Κατ' αρχάς μπορεί κανείς να χρησιμοποιήσει έναν απλό κειμενογράφο για να γράφει τον κώδικά του και έναν απλό compiler (ή ότι άλλο απαιτείται) για να τον μετατρέπει σε εκτελέσιμο κώδικα. Αυτό όμως είναι αρκετά οπισθοδρομικό και σίγουρα όχι ότι καλύτερο για χρήστες με μικρή εμπειρία. Ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE – Integrated Development Environment) προσφέρει πολλά εργαλεία και διευκολύνσεις στον προγραμματιστή βοηθώντας και επιταχύνοντας την διαδικασία ανάπτυξης μιας εφαρμογής.

Τα τρία κυρίαρχα, αυτή τη στιγμή, IDEs για την ανάπτυξη εφαρμογών Java είναι τα Eclipse, Netbeans και IntelliJ. Στα 2 πρώτα από αυτά υπήρχε και εμπειρία ανάπτυξης εφαρμογών..

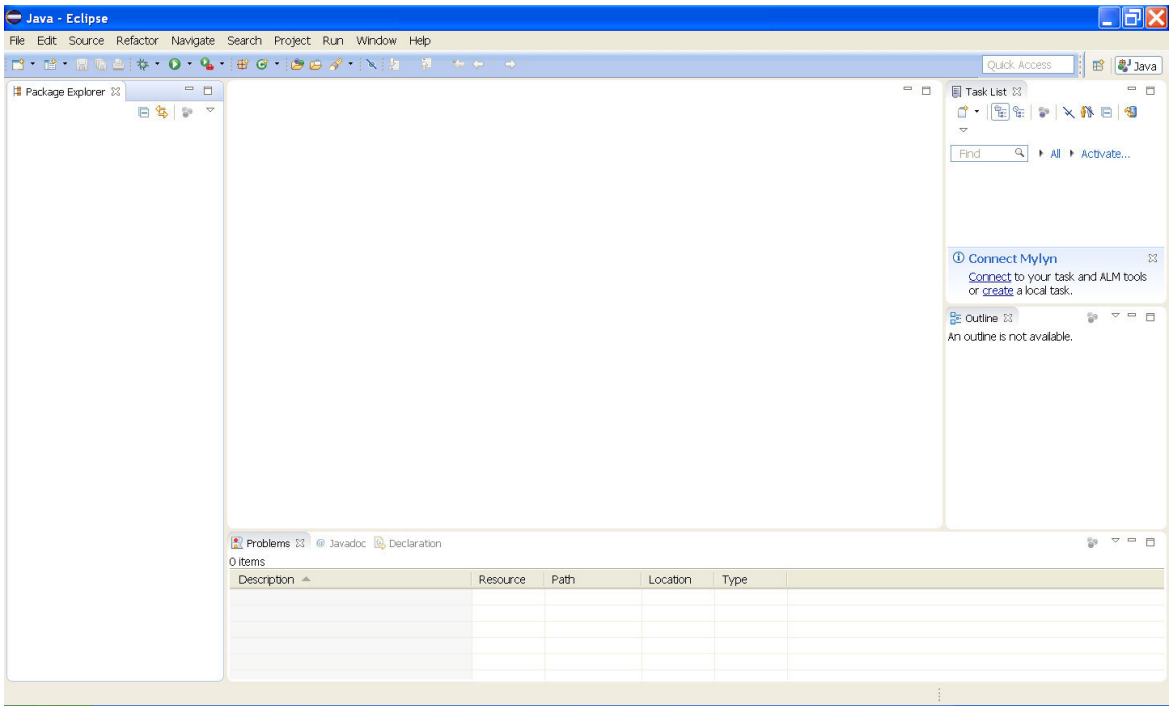
Τελικά σαν περιβάλλον ανάπτυξης επιλέχθηκε το Eclipse. Η επιλογή έγινε λόγω των ακόλουθων παραγόντων:

- Η ποιο πρόσφατη προγραμματιστική εμπειρία ήταν ανάπτυξη απλών εφαρμογών Java στο Eclipse. Πρόκειται για το μάθημα Αντικειμενοστραφής Προγραμματισμός του δεύτερου έτους σπουδών του τμήματος Ε.Π.Δ.Ο. του Τ.Ε.Ι. Μεσολογγίου.
- Η μεγαλύτερη online κοινότητα χρηστών η οποία παρέχει υποστήριξη και βοήθεια είναι αυτή του Eclipse. Σαν αποτέλεσμα υπάρχει μεγαλύτερη επιλογή για πρόσθετα (plugins) να και δεν ήταν σίγουρο ότι θα χρειαστεί κάποιο από αυτά στην ανάπτυξη της παρούσας εργασίας.
- Το Eclipse έχει τους περισσότερους υποστηρικτές, οι οποίοι το επιλέγουν σαν την πλατφόρμα ανάπτυξης εφαρμογών που προτιμούν.

6.3.Δημιουργώντας την μικροεφαρμογή

Από τη στιγμή που αποφασίστηκε το εργαλείο που θα χρησιμοποιηθεί για την υλοποίηση της μικροεφαρμογής, ξεκίνησε ουσιαστικά και η διαδικασία της υλοποίησης. Η ανάπτυξη έγινε βήμα-βήμα, προσθέτοντας κάθε φορά στην εφαρμογή ένα καινούργιο χαρακτηριστικό και δοκιμάζοντάς το. Υπήρχαν πισωγυρίσματα, προσπάθειες να παρακαμφθούν προβλήματα αλλά και απόπειρες να υιοθετηθούν προσεγγίσεις που τελικά δεν ευδοκίμησαν. Η εφαρμογή όμως ολοκληρωνόταν ημέρα με την ημέρα. Είναι φυσικό ότι στην παράγραφο αυτή αλλά και στις επόμενες δεν είναι δυνατόν να αναλυθούν όλα τα βήματα που έγιναν ή όλες οι διαφορετικές προσεγγίσεις που δοκιμάστηκαν. Αυτό που θα γίνει είναι μια παρουσίαση της τελικής εκδοχής τους μικροεφαρμογής στην οποία παρουσίαση θα επισημανθούν όσα σημεία της ανάπτυξης θεωρούνται σημαντικά.

Το περιβάλλον του Eclipse, έκδοση Luna, το οποίο επιλέχθηκε για την ανάπτυξη είναι αυτό που φαίνεται στην εικόνα 6.1.



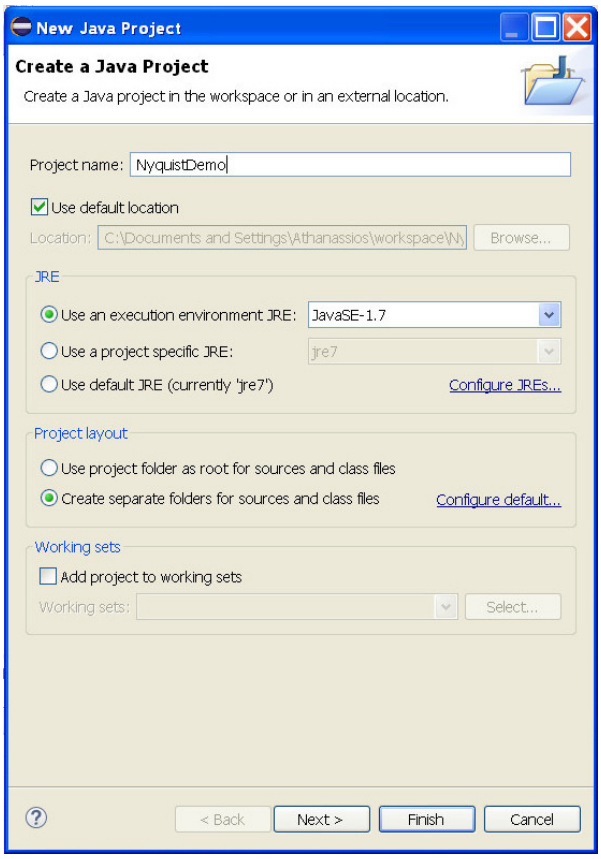
Εικόνα 6.1 Το περιβάλλον του Eclipse

Για να ξεκινήσει η υλοποίηση της μικροεφαρμογής έπρεπε πρώτα από όλα να δημιουργηθεί ένα project στο Eclipse. Το όνομα του project είναι NyquistDemo, όπως φαίνεται και στην εικόνα 6.2.

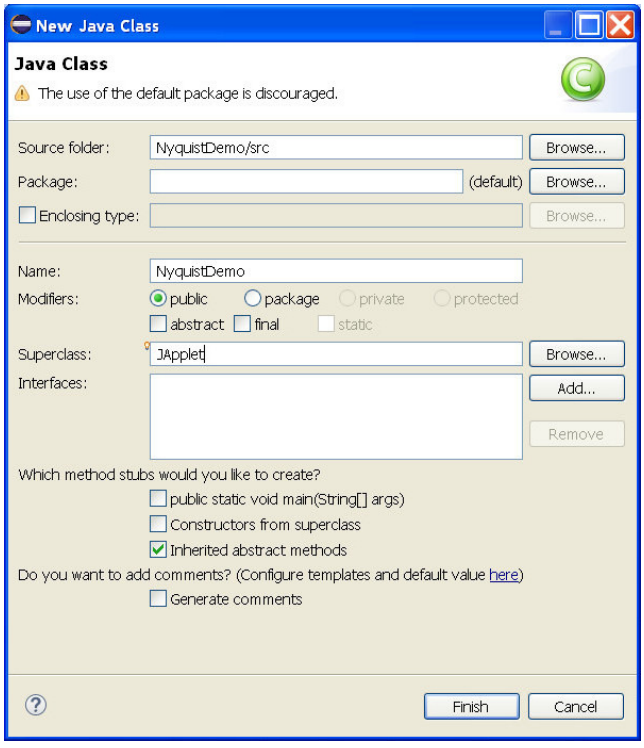
Στη συνέχεια εισήχθη μια νέα κλάση java η οποία ονομάστηκε επίσης NyquistDemo και η οποία κληρονομεί την JApplet. Το παράθυρο εισαγωγής μιας νέας κλάσης είναι αυτό της εικόνας 6.3.

Το αποτέλεσμα των διαδικασιών αυτό είναι ακόλουθος κώδικας.

```
import javax.swing.JApplet;

public class NyquistDemo extends JApplet
{

    public void init()
    {

    }
}
```

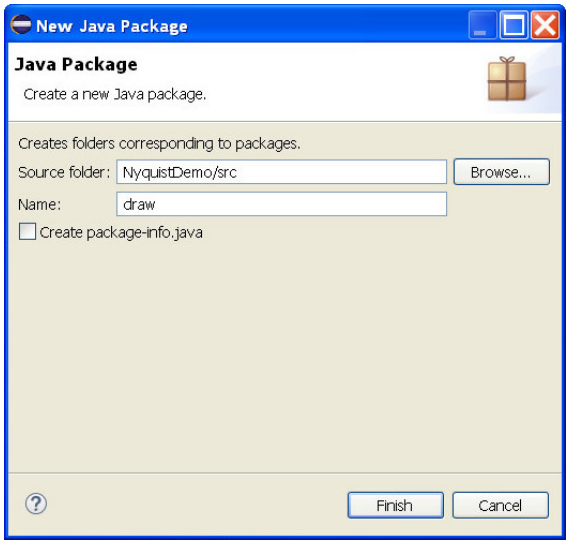


Εικόνα 6.2 Η δημιουργία του project



Εικόνα 6.3 Η δημιουργία της κλάσης NyquistDemo

Κατά τη διαδικασία της υλοποίησης της μικροεφαρμογής ο πιο πάνω κώδικας φυσικά επεκτάθηκε και εμπλουτίστηκε. Στη συνέχεια θα παρουσιαστεί ο τελικός κώδικα ανά τμήματα και κάθε φορά θα γίνεται επεξήγηση της πίσω από την υλοποίηση των διαφόρων λειτουργιών. Θα πρέπει να αναφερθεί επίσης ότι δημιουργήθηκε και ένα package με το όνομα draw και στο οποίο αποθηκεύθηκαν οι κλάσεις που δημιουργήθηκαν για την σχεδίαση των γραφημάτων της μικροεφαρμογής.



Εικόνα 6.4 Η δημιουργία του package draw

6.3.1. Οι κλάσεις και τα πακέτα που χρησιμοποιήθηκαν

Πέρα από τη κλάση `javax.swing.JApplet` που εισήχθηκε αυτόματα, για την εκτέλεση του τελικού προγράμματος απαιτούνται και μια σειρά από άλλες κλάσεις όπως και όλες οι κλάσεις του πακέτου (package) draw που δημιουργήθηκε για τις ανάγκες τις εργασίας. Έτσι το πρώτο κομμάτι του κώδικα είναι το ακόλουθο:

```
import javax.swing.JApplet;
import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JLayeredPane;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
```

```
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Hashtable;
import java.math.BigDecimal;
import java.math.RoundingMode;

import draw.*;
```

Έγινε προσπάθεια να μην αναμειχθούν αντικείμενα από το πακέτο SWING και το πακέτο AWT. Αυτό είναι μια καλή πρακτική σύμφωνα με τους ειδήμονες σε διάφορα φόρα. Έτσι όλα τα αντικείμενα είναι από το πακέτο SWING. Μόνο το χρώμα, η γραμματοσειρά και κάποια events που δεν ορίζονται στο SWING χρησιμοποιήθηκαν από το πακέτο AWT. Κάποια utilities για την δημιουργία των τιμών στους ολισθητές και κάποιες μαθηματικές συναρτήσεις συμπληρώνουν τα imports.

6.3.2. Ορισμός αντικειμένων

Μέσα στην κλάση NyquistDemo ορίζονται αρχικά οι μεταβλητές που χρησιμοποιούνται σαν global και αρχικοποιούνται και κάποιες από αυτές.

```
public class NyquistDemo extends JApplet
{
    //declare all Global vars and initialize some of them
    public double[] freqs = {1.0, 0.0, 0.0};
    public double samplingFreq = 1.0;
    public double[] aliasingFreqs = {0.0, 0.0, 0.0};
    Color bckgColor;
```

Ο πίνακας freq κρατά τις τιμές των τριών συχνοτήτων του σήματος εισόδου, η μεταβλητή samplingFreq τη συχνότητα δειγματοληψίας και ο πίνακας aliasingFreq τις τιμές των

αναδιπλωμένων (aliased) συχνοτήτων του σήματος εξόδου. Όσο για το bckgColor είναι το χρώμα του background του applet.

Κατόπιν ορίζονται τα αντικείμενα που θα χρησιμοποιηθούν στην μικροεφαρμογή

```
    //panel containing text fields, sliders and .......
    JSlider sliderSine1 = new JSlider(SwingConstants.HORIZONTAL, 10, 40, 10);
    JSlider sliderSine2 = new JSlider(SwingConstants.HORIZONTAL, 10, 40, 10);
    JSlider sliderSine3 = new JSlider(SwingConstants.HORIZONTAL, 10, 40, 10);
    JSlider sliderSampler = new JSlider(SwingConstants.HORIZONTAL, 10, 80, 10);

    JTextField sine1Text = new JTextField(2);
    JTextField sine2Text = new JTextField(2);
    JTextField sine3Text = new JTextField(2);
    JTextField samplerText = new JTextField(2);
    JTextField outSine1Text = new JTextField(2);
    JTextField outSine2Text = new JTextField(2);
    JTextField outSine3Text = new JTextField(2);

    DrawingPanel inputSignal = new DrawingPanel(freqs);
    DrawingPanel samplingSignal = new DrawingPanel(samplingFreq);
    DrawingPanel sampledSignal = new DrawingPanel(freqs, samplingFreq);
    DrawingPanel outputSignal = new DrawingPanel(aliasingFreqs, Color.red);

    GlassPanel glassInputSignal = new GlassPanel();
    GlassPanel glassSamplingSignal = new GlassPanel();
    GlassPanel glassSampledSignal = new GlassPanel();
    GlassPanel glassOutputSignal = new GlassPanel();

    DrawingPanel inputSignalSpectrum = new DrawingPanel(freqs, "spectrum");
    GlassPanel glassInputSignalSpectrum = new GlassPanel();

    DrawingPanel samplingSignalSpectrum = new DrawingPanel(samplingFreq,
"diracCombSpectrum");
    GlassPanel glassSamplingSignalSpectrum = new GlassPanel();

    DrawingPanel sampledSignalSpectrum = new DrawingPanel(freqs, samplingFreq,
"sampledSignalSpectrum");
    GlassPanel glassSampledSignalSpectrum = new GlassPanel();
```

```
    DrawingPanel outputSignalSpectrum = new DrawingPanel(aliasingFreqs,
"spectrum");
    GlassPanel glassOutputSignalSpectrum = new GlassPanel();

    JCheckBox chkSine2 = new JCheckBox("");
    JCheckBox chkSine3 = new JCheckBox("");

    JLabel inputSignalLabel = new JLabel("Input Signal", JLabel.CENTER);
    JLabel samplingSignalLabel = new JLabel("Sampling Signal", JLabel.CENTER);
    JLabel sampledSignalLabel = new JLabel("Sampled Signal", JLabel.CENTER);
    JLabel outputSignalLabel = new JLabel("Output Signal", JLabel.CENTER);
    JLabel inputSignalLabelF = new JLabel("Input Signal", JLabel.CENTER);
    JLabel samplingSignalLabelF = new JLabel("Sampling Signal", JLabel.CENTER);
    JLabel sampledSignalLabelF = new JLabel("Sampled Signal", JLabel.CENTER);
    JLabel outputSignalLabelF = new JLabel("Output Signal", JLabel.CENTER);

    JLabel inputFreq0Label = new JLabel("f0", JLabel.CENTER);
    JLabel outputFreq0Label = new JLabel("f0", JLabel.CENTER);
    JLabel outputFreq1Label = new JLabel("f1", JLabel.CENTER);
    JLabel outputFreq2Label = new JLabel("f2", JLabel.CENTER);



    JPanel timeDomainPanel = new JPanel();
    JPanel frequencyDomainPanel = new JPanel();
```

Κάποια από τα αντικείμενα αυτά είναι αντικείμενα που έχουν γίνει import. Αυτά είναι:

- 3 οριζόντιοι ολισθητές (αντικείμενα της κλάσης JSlider) με ονόματα sliderSine1, sliderSine2 και sliderSine3 που ρυθμίζουν τις συχνότητες των τριών ημιτόνων του σήματος εισόδου και που έχουν ελάχιστη τιμή το 10, μέγιστη το 40 και αρχική τιμή το 10.
- 1 οριζόντιος ολισθητής (αντικείμενο της κλάσης JSlider) με όνομα sliderSampler που ρυθμίζει τη συχνότητα δειγματοληψίας και που έχει ελάχιστη τιμή το 10, μέγιστη το 80 και αρχική τιμή το 10.
- 7 text boxes (αντικείμενα της κλάσης JTextField) με πλάτος 2 στήλες τα οποία έχουν ονόματα sine1Text, sine2Text, sine3Text (για τις συχνότητες του σήματος στην

είσοδο), samplerText (για τη συχνότητα δειγματοληψίας), outSine1Text, outSine2Text και outSine3Text (για τις συχνότητες του σήματος στην έξοδο).

- 2 check boxes (αντικείμενα της κλάσης JCheckBox) με ονόματα chkSine2 και chkSine3 για την ενεργοποίηση ή απενεργοποίηση του $2^{ου}$ και του $3^{ου}$ ημιτόνου στο σήμα εισόδου.
- 4 labels (αντικείμενα της κλάσης JLabel) για τα γραφήματα στο πεδίο του χρόνου (inputSignalLabel, samplingSignalLabel, sampledSignalLabel και outputSignalLabel) με τις αντίστοιχες τιμές.
- 4 labels (αντικείμενα της κλάσης JLabel) για τα γραφήματα στο πεδίο των συχνοτήτων (inputSignalLabelF, samplingSignalLabelF, sampledSignalLabelF και outputSignalLabelF) με τις αντίστοιχες τιμές.
- 3 labels (αντικείμενα της κλάσης JLabel) για τις συχνότητες εξόδου, τα outputFreq0Label, outputFreq1Label και outputFreq2Label με τιμές αντίστοιχα f0, f1 και f2.
- 1 label (αντικείμενο της κλάσης JLabel) για τη συχνότητα εισόδου f0 με όνομα inputFreq0Label. (για τις άλλες 2 συχνότητες χρησιμοποιήθηκε το κείμενο των check boxes)
- 2 panels (αντικείμενα της κλάσης JPanel) για την αισθητική της εφαρμογής τα οποία περικλείουν τα γραφήματα του πεδίου του χρόνου (timeDomainPanel) και τα γραφήματα του πεδίου των συχνοτήτων (frequencyDomainPanel).

Πέρα από τα αντικείμενα που είναι στιγμιότυπα κλάσεων που έχουν γίνει import από έτοιμα πακέτα της Java, υπάρχουν και αντικείμενα τα οποία ανήκουν σε κλάσεις που έχουν σχεδιαστεί και υλοποιηθεί μόνο για την εργασία αυτή και τα οποία βρίσκονται στο πακέτο draw. Τα αντικείμενα αυτά είναι τα:

- inputSignal, samplingSignal, sampledSignal και outputSignal τα οποία είναι της κλάσης DrawingPanel και τα οποία χρησιμοποιούνται για τη σχεδίαση των σημάτων στο πεδίο του χρόνου

- inputSignalSpectrum, samplingSignalSpectrum, sampledSignalSpectrum, outputSignalSpectrum τα οποία είναι και αυτά της κλάσης DrawingPanel και τα οποία χρησιμοποιούνται για τη σχεδίαση των σημάτων στο πεδίο των συχνοτήτων
- glassInputSignal, glassSamplingSignal, glassSampledSignal και glassOutputSignal τα οποία είναι διάφανα αντικείμενα σχεδίασης στα οποία υπάρχουν οι άξονες και τα περιθώρια για κάθε σήμα στο πεδίο του χρόνου
- glassInputSignalSpectrum, glassSamplingSignalSpectrum, glassSampledSignalSpectrum και glassOutputSignalSpectrum τα οποία είναι διάφανα αντικείμενα σχεδίασης στα οποία υπάρχουν οι άξονες και τα περιθώρια για κάθε σήμα στο πεδίο των συχνοτήτων

Το πώς λειτουργούν τα αντικείμενα αυτά θα περιγραφεί στην παράγραφο 6.6 όπου αναλύονται τα γραφήματα.

6.4.Η μορφή της μικροεφαρμογής

Από την σχεδίαση της μικροεφαρμογής στο χαρτί και την εξομοίωσή της με χρήση του MATLAB®, είχαμε κατασταλάξει στην μορφή της διεπαφής χρήστη της εφαρμογής. Η εικόνα 4.2 και η εικόνα 5.4 ήταν ο οδηγός για την κατασκευή της διεπαφής.

Το Eclipse δεν διαθέτει ενσωματωμένο εργαλείο για την κατασκευή παραθύρων ή διεπαφών όπως το NetBeans, αλλά υπάρχει διαθέσιμος ο WindowBuilder το οποίο είναι ένα πρόσθετο (plugin) για το Eclipse που παρέχει τη δυνατότητες ενός GUI Builder. Η εγκατάσταση και η χρήση του συγκεκριμένου πρόσθετου είναι εύκολη αλλά επιλέχθηκε να γίνει η δημιουργία της διεπαφής μέσω του κώδικα καθώς έτσι υπάρχει μεγαλύτερος έλεγχος πάνω στον τρόπο που τοποθετούνται τα αντικείμενα στο παράθυρο. Για να γίνει όμως η περιγραφή του τρόπου με τον οποίο διευθετήθηκαν τα διάφορα αντικείμενα στο παράθυρο της μικροεφαρμογής, καλό θα ήταν πρώτα να περιγραφούν εν συντομία κάποιες σχετικές βασικές έννοιες της Java.

6.4.1. Τα επίπεδα (panes) της Java

Κάθε εφαρμογή σε Java, είτε αυτή είναι standalone είτε είναι ένα applet, έχει έναν τουλάχιστον περιέχων (στο εξής θα χρησιμοποιείται ο πιο δόκιμος αγγλικός όρος container)

στο ανώτερο επίπεδο (top-level). Ο container αυτός είναι η ρίζα από μια ιεραρχία από containers. Η ιεραρχία αυτή αποτελεί τη δομή η οποία περιλαμβάνει όλα τα αντικείμενα που εμπεριέχονται στην εφαρμογή.

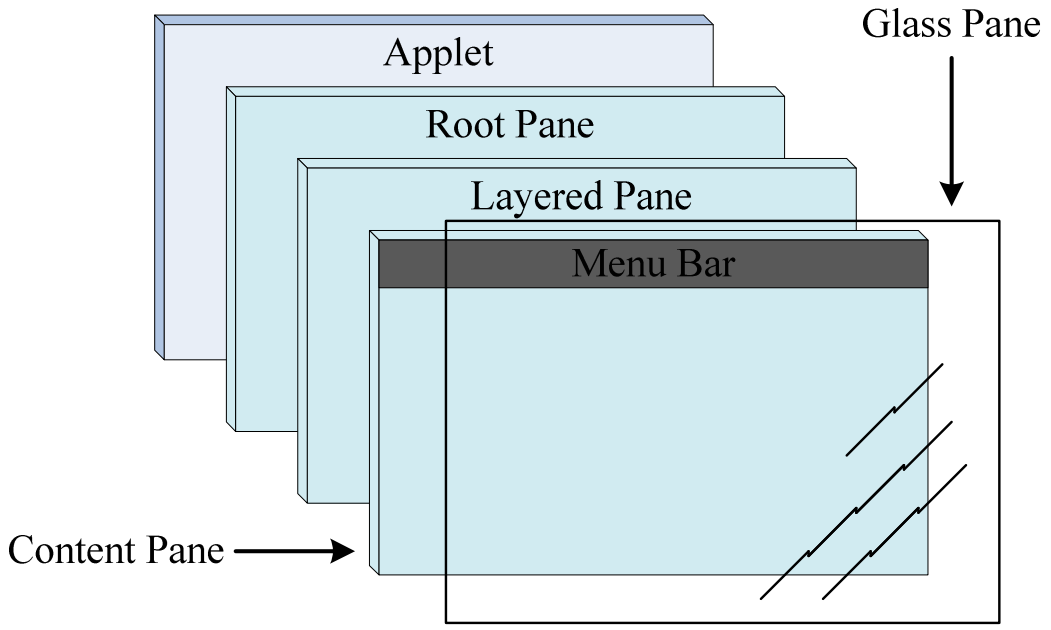
Αν πρόκειται για απλή εφαρμογή, ο container αυτός μπορεί να είναι ένα αντικείμενο Frame ή ένα αντικείμενο Dialog. Αν πρόκειται για μικροεφαρμογή ο container είναι ένα αντικείμενο Applet (JApplet). Κάθε τέτοιος top-level container βασίζεται σε ένα ενδιάμεσο container (το root pane ή επίπεδο root) το οποίο βρίσκεται πιο κοντά στον χρήστη και τον προγραμματιστή. Βασικά δεν χρειάζεται κανείς να γνωρίζει πολλά για το επίπεδο root για να μπορέσει να προγραμματίσει. Το επίπεδο αυτό δημιουργείται (είτε το θέλουμε είτε όχι) όταν δημιουργείται ένα αντικείμενο JApplet. Το root pane περιλαμβάνει κάποιους containers χαμηλότερων επιπέδων οι οποίοι όμως είναι αυτοί που μας ενδιαφέρουν πιο άμεσα.

Έτσι κάτω από το root pane υπάρχει το layered pane το οποίο περιέχει το content pane και μπορεί να περιέχει και μια προαιρετική μπάρα μενού. Το σημαντικό στο layered pane είναι ότι μπορούν να τοποθετηθούν αντικείμενα, το ένα πάνω στο άλλο, ακολουθώντας μια διάταξη Z (Z order) όπως περίπου έχουμε τον άξονα z στις καρτεσιανές συντεταγμένες. Για παράδειγμα μπορεί όταν επιλέγουμε ένα drop down menu, αυτό να εμφανίζεται πάνω από ένα άλλο στοιχείο στο layered pane.

Το content pane είναι το container το οποίο περιέχει όλα τα ορατά αντικείμενα του root pane, εκτός φυσικά της menu bar. Αυτή η μπάρα η οποία είναι προαιρετική περιλαμβάνει τα στοιχεία μενού του root pane. Αν το container έχει μια menu bar, τότε, εν γένει, χρησιμοποιούμε την μέθοδο `setJMenuBar()` για να την τοποθετήσουμε σωστά.

Τέλος υπάρχει και το glass pane, το οποίο είναι κρυφό από προεπιλογή. Αν γίνει ορατό τότε συμπεριφέρεται σαν ένα φύλλο γυαλιού πάνω από όλα τα άλλα αντικείμενα και στοιχεία του root pane. Είναι εντελώς διάφανο εκτός αν γίνει υπερφόρτωση της μεθόδου `paintComponent()` που περιλαμβάνει.

Στο επόμενο σχήμα φαίνεται η διάταξη των επιπέδων αυτών.



Εικόνα 6.5 Η ιεραρχία των επιπέδων σε ένα JApplet

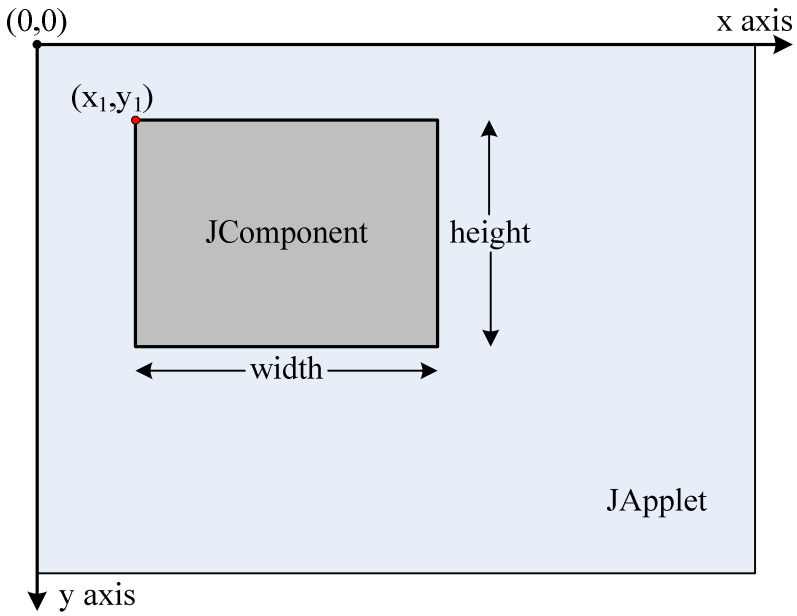
6.4.2. Το σύστημα συντεταγμένων της Java

Όπως θα περιγραφεί και παρακάτω, όταν ένα αντικείμενο τοποθετείται στον δισδιάστατο χώρο επιπέδου (εδώ του layered pane) είναι απαραίτητο να προσδιοριστούν, πέρα από το όνομα του αντικειμένου, η θέση του και το μέγεθός του. Η θέση του αντικειμένου προσδιορίζεται από ένα ζεύγος τιμών που είναι οι συντεταγμένες του. Βασικά δίνονται οι συντεταγμένες της μιας γωνίας του αντικειμένου, της πάνω αριστερά. Ο λόγος που δίνονται αυτές οι συντεταγμένες είναι ο τρόπος που ορίζονται οι άξονες x και y στην Java. Έτσι, αν έχουμε σχεδίαση σε 2 διαστάσεις (σε ένα JFrame ή σε ένα JApplet για παράδειγμα), η αρχή των αξόνων είναι η πάνω αριστερή γωνία του χώρου που θα χρησιμοποιήσει ο χρήστης. Ο άξονας x κατευθύνεται προς τα δεξιά και ο άξονας y προς τα κάτω. Έτσι αντίστοιχα η τετμημένη x αυξάνεται προς τα δεξιά και η τεταγμένη y προς τα κάτω. Στο επόμενο σχήμα οι συντεταγμένες του αντικειμένου JComponent είναι οι (x_1,y_1).

Επιπλέον των συντεταγμένων του αντικειμένου, απαιτείται να δοθούν και οι διαστάσεις του. Πρώτα δίνεται το πλάτος και κατόπιν το ύψος. Έτσι μια δήλωση για το πώς θα τοποθετηθεί γεωμετρικά ένα αντικείμενο έχει τη μορφή:

```
componentName.setBounds(x,y,width,height)
```

όπου `componentName` το όνομα του αντικειμένου και `setBounds(int x, int y, int width, int height)` είναι μια μέθοδος για να τοποθετείται ένα αντικείμενο ή να μεταφέρεται ή να του αλλάζει το μέγεθός του.



Εικόνα 6.6 Οι συντεταγμένες στην Java 2D

Τα παραπάνω θα διευκολύνουν την κατανόηση του πως τοποθετήθηκαν τα διάφορα στοιχεία στην μικροεφαρμογή και κυρίως πως τοποθετήθηκαν τα γραφήματα και οι άξονες.

6.4.3. Τα layout styles στη Java

Ένα πολύ ενδιαφέρον και χρήσιμο χαρακτηριστικό της java είναι και τα layouts. Layout είναι ο τρόπος με τον οποίο η java τοποθετεί τα διάφορα αντικείμενα στο παράθυρο της διεπαφής. Για διευκόλυνση του προγραμματιστή, η ίδια η γλώσσα παρέχει Layout Managers για την καλύτερη χωροθέτηση των αντικειμένων. Να σημειωθεί ότι στους Layout Managers πρέπει να γραφεί κώδικας με το χέρι. Αντίθετα, αν κανείς χρησιμοποιήσει GUI builders, τότε ο κώδικας αυτός παράγεται αυτόματα από το IDE.

Υπάρχουν layout managers γενικού σκοπού που παρέχουν κάποιες από τις κλάσεις AWT και Swing. Τέτοιοι είναι οι εξής

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

Οι δυο τελευταίοι είναι οι πιο ευέλικτοι από όλους του managers που αναφέρονται εδώ.

Η περιγραφή όλων αυτών των layout managers ξεφεύγει από τους σκοπούς της εργασίας και καθώς τελικά δεν χρησιμοποιήθηκε κανείς τους, εδώ γίνεται απλά μια αναφορά σε αυτούς.

Επίσης πρέπει να αναφερθεί ότι όταν χρησιμοποιείται ένας layout manager, συνήθως υπάρχει η δυνατότητα αλλαγής μεγέθους των αντικειμένων με δυναμικό τρόπο, όταν αλλάζει το μέγεθος του παραθύρου που τα περικλείει. Κάτι τέτοιο δεν συμβαίνει όταν τα αντικείμενα απλώς τοποθετούνται σε κάποιες θέσεις χωρίς τη χρήση κάποιου manager.

6.4.4. Αρχικοποιώντας τις μεταβλητές

Μετά τη δήλωση των διάφορων αντικειμένων που αναφέρθηκαν πιο πάνω, θα πρέπει τα αντικείμενα αυτά να τοποθετηθούν στην μικροεφαρμογή. Αυτή η διαδικασία, καθώς και οποιαδήποτε άλλη σχετίζεται με το τι θα κάνει η μικροεφαρμογή, περιγράφεται μέσα στη συνάρτηση `init()` η οποία είναι για τα applets ότι είναι η `main()` για τις κανονικές εφαρμογές Java.

```
    public void init()
    {
//initialize all ...., background info, and panel
        resize(1350, 760);   //size of applet window

        setLayout(null);  //set layout style to null, groupLayout is another good
idea
//        GroupLayout layout = new GroupLayout(getContentPane());
//        getContentPane().setLayout(layout);
```

```
// Setting background color of root pane for getting it with getRootPane().getBackground()
        bckgColor = getContentPane().getBackground();
        setBackground(bckgColor);

// getting layered pane in order to be able to place panels on top of each other
        JLayeredPane lp = getLayeredPane();

// set font for labels
        Font font = new Font("TimesRoman", Font.PLAIN, 13);

// set titles for panels
        TitledBorder titleT, titleF;
        titleT = BorderFactory.createTitledBorder("Time Domain");
        titleT.setTitleFont(font);
        titleF = BorderFactory.createTitledBorder("Frequency Domain");
        titleF.setTitleFont(font);
```

Όπως φαίνεται και στον κώδικα που παρατίθεται, αρχικά ορίζεται το μέγεθος του παραθύρου του applet. Αυτό ορίστηκε σε 1350 x 760 pixels. Το μέγεθος αυτό είναι ικανό για να φαίνονται όλα τα στοιχεία της μικροεφαρμογής. Αν η ανάλυση της οθόνης του υπολογιστή είναι μικρότερη από αυτό το μέγεθος, τότε η μικροεφαρμογή δεν θα χωράει ολόκληρη σε μια οθόνη και στον φυλλομετρητή θα εμφανίζονται μπάρες ολίσθησης αριστερά/δεξια ή πάνω/κάτω (ανάλογα με το ποια διάσταση δεν χωράει). Αυτό δεν είναι ότι καλύτερο αλλά δυστυχώς το μέγεθος της μικροεφαρμογής είναι ένα από τα μειονεκτήματά της.

Κατόπιν ορίζεται ότι δεν θα χρησιμοποιηθεί κάποιο από τα έτοιμα layout στυλ της Java αλλά τα αντικείμενα θα τοποθετούνται σε συγκεκριμένες θέσεις με την εντολή `setBounds`. Αυτό παρέχει πλήρη ελευθερία στην διάταξη των αντικειμένων αλλά, όπως αναφέρεται και στην προηγούμενη παράγραφο, δεν υπάρχει η δυνατότητα να μεταβάλλεται εύκολα το μέγεθος των αντικειμένων όταν το παράθυρο του φυλλομετρητή θα γίνεται resize.

Ένα άλλο στυλ που θα μπορούσε να χρησιμοποιηθεί είναι το groupLayout. Στην αρχική φάση της υλοποίησης χρησιμοποιήθηκε αυτό το στυλ αλλά όταν προέκυψαν δυσκολίες στην τοποθέτηση των αντικειμένων glassXxxxxSignal και glassXxxxxSignalSpectrum, τα οποία έπρεπε να τοποθετηθούν πάνω από άλλα αντικείμενα, εγκαταλείφθηκε αυτή η προσέγγιση.

Στη συνέχεια ορίζεται το background το root pane να είναι ίδιο με αυτό του content pane. Ο λόγος που έγινε αυτό ήταν ότι όταν θα ορισθούν τα περιθώρια γύρω από κάθε γράφημα, τότε θα μπορεί να επιλεγεί το χρώμα τους να είναι ίδιο με το background χρησιμοποιώντας την εντολή `getRootPane().getBackground()`.

Το αντικείμενο lp το οποίο είναι της κλάσης JLayeredPane ορίζεται και αρχικοποιείται να δείχνει στο layer pane της μικροεφαρμογής. Η χρησιμότητα της δήλωσης αυτής θα φανεί πιο κάτω όπου κάποια αντικείμενα θα πρέπει να τοποθετηθούν το ένα πάνω από το άλλο σε διάταξη z order.

Η γραμματοσειρά που χρησιμοποιήθηκε σε όλες τις ετικέτες (labels) αλλά και τους τίτλους, είναι η ίδια και ορίζεται με την δήλωση:

```
Font font = new Font("TimesRoman", Font.PLAIN, 13);
```

Πρόκειται γα την απλή Times Roman (δηλαδή όχι πλαγιαστή ή έντονη) και το μέγεθος που επιλέχθηκε είναι το 13.

Τέλος ορίζονται οι τίτλοι για τα δυο panels που θα περικλείουν τα γραφήματα στο πεδίο του χρόνου και στο πεδίο των συχνοτήτων. Οι τίτλοι είναι αντίστοιχα “Time Domain” και “Frequency Domain”.

6.4.5. Χωροθέτηση των αντικειμένων

Όλα είναι έτοιμα πλέον για να τοποθετηθούν τα αντικείμενα στο παράθυρο του applet. Για κάθε αντικείμενο θα ορισθούν τα όριά του με τη δήλωση

```
componentName.setBounds(x,y,width,height)
```

Οι θέσεις των διαφόρων αντικειμένων είναι αυτές που φαίνονται στο ακόλουθο κώδικα. Για κάποια αντικείμενα από αυτά δεν ορίζεται μόνο τη θέση τους αλλά κάποιες ιδιότητές τους. Έτσι για panels ορίζονται και οι τίτλοι τους ενώ για labels η γραμματοσειρά. Η γραμματοσειρά ορίζεται και για τα check boxes όπως και το κείμενο που εμφανίζεται δεξιά από αυτά. Τέλος για τα text boxes ορίζεται η οριζόντια στοίχιση να είναι στο κέντρο.

Για τα αντικείμενα των κλάσεων που δημιουργήθηκαν αποκλειστικά για την εργασία (δηλαδή των DrawingPanel και GlassPanel) είναι δυνατόν να ορίσθεί η μέγιστη τιμή του x

άξονα και αν ο άξονας έχει συμμετρία (είναι δηλαδή από –x έως x) ή όχι (ο άξονας είναι από 0 έως x).

```
// place everything on the window
        // first set bounds an other attributes for...
        // panels
        timeDomainPanel.setBounds(20, 0, 1320, 290);
        timeDomainPanel.setBorder(titleT);

        frequencyDomainPanel.setBounds(20, 460, 1320, 290);
        frequencyDomainPanel.setBorder(titleF);

        //input signal and related controls
        inputSignal.setBounds(50,20,301,260);
        glassInputSignal.setBounds(50,20,301,260);

        inputSignalLabel.setBounds(50,10,301,20);
        inputSignalLabel.setFont(font);
        sliderSine1.setBounds(65,310,250,50);
        sine1Text.setBounds(325,315,30,25);
        sine1Text.setHorizontalAlignment(JTextField.CENTER);
        inputFreq0Label.setBounds(40,310,20,20);
        inputFreq0Label.setFont(font);
        sliderSine2.setBounds(65,360,250,50);
        sine2Text.setBounds(325,365,30,25);
        sine2Text.setHorizontalAlignment(JTextField.CENTER);
        chkSine2.setBounds(20,360,40,20);
        chkSine2.setFont(font);
        chkSine2.setText(" f1");
        sliderSine3.setBounds(65,410,250,50);
        sine3Text.setBounds(325,415,30,25);
        sine3Text.setHorizontalAlignment(JTextField.CENTER);
        chkSine3.setBounds(20,410,40,20);
        chkSine3.setFont(font);
        chkSine3.setText(" f2");

        // sampling signal and related controls
```

```
        samplingSignal.setBounds(370,20,300,260);
        glassSamplingSignal.setBounds(370,20,300,260);

        samplingSignalLabel.setBounds(370,10,300,20);
        samplingSignalLabel.setFont(font);

        sliderSampler.setBounds(385,310,250,50);
        samplerText.setBounds(645,315,30,25);
        samplerText.setHorizontalAlignment(JTextField.CENTER);

        // sampled signal
        sampledSignal.setBounds(690,20,300,260);
        glassSampledSignal.setBounds(690,20,300,260);

        sampledSignalLabel.setBounds(690,10,300,20);
        sampledSignalLabel.setFont(font);
        // output signal
        outputSignal.setBounds(1010,20,300,260);
        glassOutputSignal.setBounds(1010,20,300,260);

        outputSignalLabel.setBounds(1010,10,300,20);
        outputSignalLabel.setFont(font);

        outSine1Text.setBounds(1150,315,30,25);
        outSine1Text.setHorizontalAlignment(JTextField.CENTER);
        outputFreq0Label.setBounds(1130,320,20,20);
        outputFreq0Label.setFont(font);
        outSine2Text.setBounds(1150,365,30,25);
        outSine2Text.setHorizontalAlignment(JTextField.CENTER);
        outputFreq1Label.setBounds(1130,370,20,20);
        outputFreq1Label.setFont(font);
        outSine3Text.setBounds(1150,415,30,25);
        outSine3Text.setHorizontalAlignment(JTextField.CENTER);
        outputFreq2Label.setBounds(1130,420,20,20);
        outputFreq2Label.setFont(font);

        // input signal spectrum
        inputSignalSpectrum.setBounds(50,480,301,260);
```

```
        glassInputSignalSpectrum.setBounds(50,480,301,260);
        glassInputSignalSpectrum.setAxesSymmetry(true);
        glassInputSignalSpectrum.setXAxesMaxValue(4);
        inputSignalLabelF.setBounds(50,470,301,20);
        inputSignalLabelF.setFont(font);
        // sampling signal spectrum
        samplingSignalSpectrum.setBounds(370,480,301,260);
        glassSamplingSignalSpectrum.setBounds(370,480,301,260);
        glassSamplingSignalSpectrum.setAxesSymmetry(true);
        glassSamplingSignalSpectrum.setXAxesMaxValue(8);
        samplingSignalLabelF.setBounds(370,470,301,20);
        samplingSignalLabelF.setFont(font);
        // sampled signal spectrum
        sampledSignalSpectrum.setBounds(690,480,301,260);
        glassSampledSignalSpectrum.setBounds(690,480,301,260);
        glassSampledSignalSpectrum.setAxesSymmetry(true);
        glassSampledSignalSpectrum.setXAxesMaxValue(8);
        sampledSignalLabelF.setBounds(690,470,301,20);
        sampledSignalLabelF.setFont(font);
        // output signal spectrum
        outputSignalSpectrum.setBounds(1010,480,301,260);
        glassOutputSignalSpectrum.setBounds(1010,480,301,260);
        glassOutputSignalSpectrum.setAxesSymmetry(true);
        glassOutputSignalSpectrum.setXAxesMaxValue(4);
        outputSignalLabelF.setBounds(1010,470,301,20);
        outputSignalLabelF.setFont(font);
```

Ο ορισμός όμως των ορίων καθώς και άλλων ιδιοτήτων των αντικειμένων δεν σημαίνει ότι έχουν τοποθετηθεί στο παράθυρο του applet. Για να γίνει αυτό θα πρέπει να χρησιμοποιηθεί η μέθοδος add(), όπως παρακάτω:

```
                // then add all above components
        lp.add(timeDomainPanel);
        lp.add(frequencyDomainPanel);

        lp.add(sliderSine1);
```

```
        lp.add(sine1Text);
        lp.add(inputFreq0Label);
        lp.add(sliderSine2);
        lp.add(sine2Text);
        lp.add(chkSine2);
        lp.add(sliderSine3);
        lp.add(sine3Text);
        lp.add(chkSine3);
        lp.add(inputSignal, Integer.valueOf(1));
        lp.add(glassInputSignal, Integer.valueOf(2));
        lp.add(inputSignalLabel, Integer.valueOf(3));

        lp.add(inputSignalSpectrum, Integer.valueOf(1));
        lp.add(glassInputSignalSpectrum, Integer.valueOf(2));
        lp.add(inputSignalLabelF, Integer.valueOf(3));

        lp.add(sliderSampler);
        lp.add(samplerText);
        lp.add(samplingSignal, Integer.valueOf(1));
        lp.add(glassSamplingSignal, Integer.valueOf(2));
        lp.add(samplingSignalLabel, Integer.valueOf(3));

        lp.add(samplingSignalSpectrum, Integer.valueOf(1));
        lp.add(glassSamplingSignalSpectrum, Integer.valueOf(2));
        lp.add(samplingSignalLabelF, Integer.valueOf(3));

        lp.add(sampledSignal, Integer.valueOf(1));
        lp.add(glassSampledSignal, Integer.valueOf(2));
        lp.add(sampledSignalLabel, Integer.valueOf(3));

        lp.add(sampledSignalSpectrum, Integer.valueOf(1));
        lp.add(glassSampledSignalSpectrum, Integer.valueOf(2));
        lp.add(sampledSignalLabelF, Integer.valueOf(3));

        lp.add(outputSignal, Integer.valueOf(1));
        lp.add(glassOutputSignal, Integer.valueOf(2));
        lp.add(outputSignalLabel, Integer.valueOf(3));
```

```
        lp.add(outputSignalSpectrum, Integer.valueOf(1));
        lp.add(glassOutputSignalSpectrum, Integer.valueOf(2));
        lp.add(outputSignalLabelF, Integer.valueOf(3));

        lp.add(outSine1Text);
        lp.add(outputFreq0Label);
        lp.add(outSine2Text);
        lp.add(outputFreq1Label);
        lp.add(outSine3Text);
        lp.add(outputFreq2Label);
```

Από τις παραπάνω εντολές, αυτές που δέχονται ένα όρισμα στην μέθοδο add() θα μπορούσαν αν γραφούν και χωρίς τον προσδιορισμό του layered pane. Δηλαδή η εντολή

```
lp.add(timeDomainPanel);
```

και η εντολή

```
add(timeDomainPanel);
```

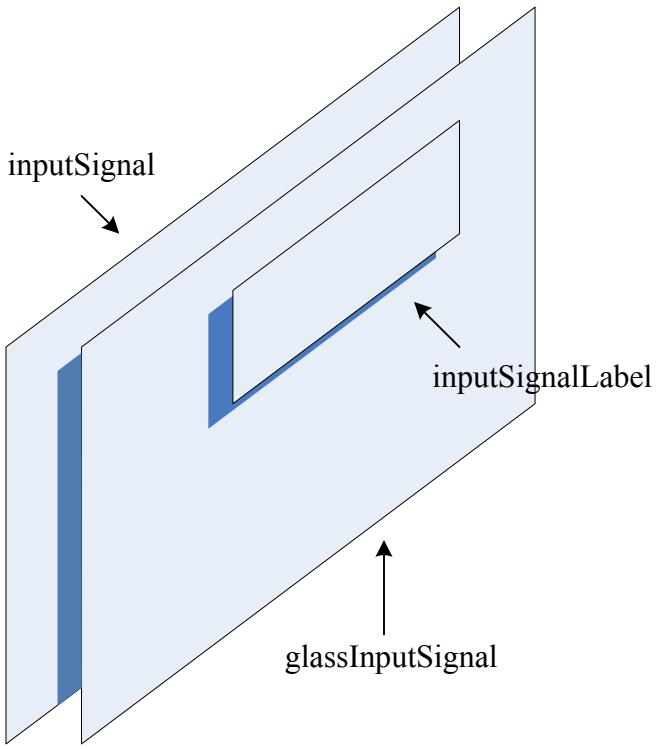
θα παράγουν το ίδιο αποτέλεσμα.

Στις υπόλοιπες εντολές ορίζεται η διάταξη (z order) που θα έχουν τα αντικείμενα που τοποθετούνται το ένα πάνω από το άλλο. Για παράδειγμα για το σήμα εισόδου ορίζονται τρία επίπεδα. Στο πρώτο επίπεδο τοποθετείται το γράφημα του σήματος εισόδου (inputSignal), στο δεύτερο επίπεδο (το οποίο είναι πάνω ακριβώς από το πρώτο) τοποθετείται ένα διάφανο panel στο οποίο θα σχεδιαστούν οι άξονες και τα περιθώρια (περιγράφονται αναλυτικά στην παράγραφο 6.6) και στο τρίτο επίπεδο τοποθετείται η ετικέτα (label), όπως στο επόμενο σχήμα.

Οι εντολές που χρησιμοποιήθηκαν στο συγκεκριμένο παράδειγμα του σήματος εισόδου είναι οι

```
lp.add(inputSignal, Integer.valueOf(1));
lp.add(glassInputSignal, Integer.valueOf(2));
lp.add(inputSignalLabel, Integer.valueOf(3));
```

Επειδή η μέθοδος add() δέχεται σαν όρισμα βάθους z μια τιμή της κλάσης Integer, μετατρέπουμε την int τιμή x (1 ή 2 ή 3) σε Integer με τη δήλωση `Integer.valueOf(x)`.



Εικόνα 6.7 Η διάταξη των επιπέδων για το σήμα εισόδου

6.4.6. Ρυθμίζοντας τις ενδείξεις των ολισθητών

Στο τμήμα του κώδικα που παρουσιάστηκε στην παράγραφο 6.4.2 ορίζονται 4 ολισθητές (sliders) με τις δηλώσεις:

```
JSlider sliderSine1 = new JSlider(SwingConstants.HORIZONTAL, 10, 40, 10);
JSlider sliderSine2 = new JSlider(SwingConstants.HORIZONTAL, 10, 40, 10);
JSlider sliderSine3 = new JSlider(SwingConstants.HORIZONTAL, 10, 40, 10);
JSlider sliderSampler = new JSlider(SwingConstants.HORIZONTAL, 10, 80, 10);
```

Οι ολισθητές αυτοί έχουν σαν ελάχιστη τιμή το 10 και σαν μέγιστη το 40 (οι τρεις πρώτοι) ή το 80 (ο τέταρτος). Το βήμα τους είναι η μονάδα. Το βήμα των ολισθητών πρέπει πάντα να είναι ακέραια τιμή. Η εφαρμογή όμως που σχεδιάζεται απαιτεί από τους ολισθητές να δίνουν και μη ακέραιες τιμές (π.χ. 1.1 ή 3.7). Για χάριν ευκολίας περιοριζόμαστε σε τιμές με ένα δεκαδικό ψηφίο. Η απεικόνιση (mapping) των ακέραιων τιμών σε δεκαδικές είναι μια εύκολη διαδικασία. Αρκεί να διαιρεθεί η ακέραια τιμή που δίνει ο ολισθητής με το 10. Έτσι το διάστημα από 10 έως 40 με βήμα 1 των πρώτων ολισθητών γίνεται από 1 έως 4 με βήμα 0.1. Αντίστοιχα για την συχνότητα δειγματοληψίας το διάστημα από 10 έως 80 με βήμα 1

γίνεται 1 έως 8 με βήμα 0.1. Με αυτόν τον τρόπο θα προκύπτει πάντα μια τιμή για τις συχνότητες από 1 έως 4 (για το σήμα εισόδου) και από 1 έως 8 (για το σήμα δειγματοληψίας).

Η προσέγγιση αυτή είναι λειτουργική (και ίσως και η πιο άμεση) και η μικροεφαρμογή δίνει τα αναμενόμενα αποτελέσματα. Υπάρχει όμως μια λεπτομέρεια που πρέπει να διορθωθεί. Οι ενδείξεις οι οποίες είναι γραμμένες κάτω από τους ολισθητές στο παράθυρο της μικροεφαρμογής θα είναι από 10 το ελάχιστο έως 40 (ή 80) το μέγιστο. Οι τιμές όμως που θα φαίνονται στα text boxes και οι οποίες θα χρησιμοποιούνται στις πράξεις θα είναι διαφορετικές (διαιρεμένες με το 10). Αυτή η ασυμφωνία δεν χρειάζεται να υφίσταται καθώς υπάρχει τρόπος να αλλαχθούν οι ενδείξεις που φέρουν οι ολισθητές. Για να γίνει αυτό θα χρησιμοποιηθεί ένας hash table, ένας πίνακας δηλαδή που αντιστοιχεί κάποια δεδομένα (κλειδιά) σε κάποια άλλα (τιμές) με αντιστοιχία ένα προς ένα.

Για το λόγο αυτό ορίζονται αρχικά 2 hash tables, ο `sld1LabelTable` και ο `sld2LabelTable`. Ο μεν πρώτος για τους τρεις ολισθητές των συχνοτήτων του σήματος εισόδου και ο δεύτερος για τον ολισθητή του σήματος δειγματοληψίας.

Η αντιστοίχιση που κάνουν είναι αυτή που φαίνεται στο επόμενο πίνακα:

Πίνακας 6.3 Hash table για τις ετικέτες των ολισθητών

κλειδιά (ακέραιες τιμές)	**τιμές (ετικέτα -string)**
10	΄΄1΄΄
20	΄΄2΄΄
30	΄΄3΄΄
40	΄΄4΄΄
50	΄΄5΄΄
60	΄΄6΄΄
70	΄΄7΄΄
80	΄΄8΄΄

Στον κώδικα που ακολουθεί φαίνεται ο τρόπος που έγινε η αντιστοίχιση.

```
        //Create the label table for the sliders
        Hashtable sld1LabelTable = new Hashtable();
        sld1LabelTable.put( new Integer( 10 ), new JLabel("1") );
        sld1LabelTable.put( new Integer( 20 ), new JLabel("2") );
        sld1LabelTable.put( new Integer( 30 ), new JLabel("3") );
        sld1LabelTable.put( new Integer( 40 ), new JLabel("4") );

        Hashtable sld2LabelTable = new Hashtable();
        sld2LabelTable.put( new Integer( 10 ), new JLabel("1") );
        sld2LabelTable.put( new Integer( 20 ), new JLabel("2") );
        sld2LabelTable.put( new Integer( 30 ), new JLabel("3") );
        sld2LabelTable.put( new Integer( 40 ), new JLabel("4") );
        sld2LabelTable.put( new Integer( 50 ), new JLabel("5") );
        sld2LabelTable.put( new Integer( 60 ), new JLabel("6") );
        sld2LabelTable.put( new Integer( 70 ), new JLabel("7") );
        sld2LabelTable.put( new Integer( 80 ), new JLabel("8") );

        //Turn on labels at major tick marks.
        sliderSine1.setMajorTickSpacing(10);
        sliderSine1.setMinorTickSpacing(1);
        sliderSine1.setPaintTicks(true);
        sliderSine1.setLabelTable( sld1LabelTable );
        sliderSine1.setPaintLabels(true);

        sliderSine2.setMajorTickSpacing(10);
        sliderSine2.setMinorTickSpacing(1);
        sliderSine2.setPaintTicks(true);
        sliderSine2.setLabelTable( sld1LabelTable );
        sliderSine2.setPaintLabels(true);

        sliderSine3.setMajorTickSpacing(10);
        sliderSine3.setMinorTickSpacing(1);
        sliderSine3.setPaintTicks(true);
        sliderSine3.setLabelTable( sld1LabelTable );
        sliderSine3.setPaintLabels(true);

        sliderSampler.setMajorTickSpacing(10);
        sliderSampler.setMinorTickSpacing(2);
```

```
        sliderSampler.setPaintTicks(true);
        sliderSampler.setLabelTable( sld2LabelTable );
        sliderSampler.setPaintLabels(true);
```

Για κάθε ένα ολισθητή στη συνέχεια ορίστηκαν κάποιες παράμετροι:

Πίνακας 6.4 Παράμετροι των ολισθητών

παράμετρος	**παράμετρος**	**τιμή**
`sliderName.setMajorTickSpacing(X);`	απόσταση μεταξύ των μεγάλων δεικτών	10
`sliderName.setMinorTickSpacing(Y);`	απόσταση μεταξύ των μικρών δεικτών	1 (input) 2 (sampler)
`sliderName.setPaintTicks(true);`	εμφάνιση των δεικτών	true
`sliderName.setLabelTable(tableName);`	hash table για τις ετικέτες	sld1LabelTable sld2LabelTable
`sliderName.setPaintLabels(true);`	ενεργοποίηση των ετικετών	true

6.4.7. Αρχικοποίηση των μεταβλητών

Όταν η εφαρμογή ξεκινά θα πρέπει κάποιες μεταβλητές να πάρουν συγκεκριμένες τιμές και να γίνουν και κάποιες ενέργειες. Ο κώδικας ο οποίος φροντίζει γι΄αυτά ακολουθεί:

```
//slider sine2 and slider sine3 are disabled at startup
        sliderSine2.setEnabled(false);
        sliderSine3.setEnabled(false);
        chkSine3.setEnabled(false);

// populate text boxes at start up
        sine1Text.setText(Double.toString(freqs[0]));
        sine2Text.setText(Double.toString(freqs[1]));
        sine3Text.setText(Double.toString(freqs[2]));
        samplerText.setText(Double.toString(samplingFreq));
        calculateAliasingFreqs();
```

```
        outSine1Text.setText(Double.toString(aliasingFreqs[0]));
        outSine2Text.setText(Double.toString(aliasingFreqs[1]));
        outSine3Text.setText(Double.toString(aliasingFreqs[2]));

// at start up we have only one frequency
      inputSignal.setNorm(1);
      sampledSignal.setNorm(1);
      outputSignal.setNorm(1);
      inputSignalSpectrum.setNorm(1);
      sampledSignalSpectrum.setNorm(1);
      outputSignalSpectrum.setNorm(1);

// paint everything at start up
        sampledSignal.setAliasingFreqs(aliasingFreqs);
        outputSignal.setAliasingFreqs(aliasingFreqs);
        inputSignal.repaint();
        sampledSignal.repaint();
        outputSignal.repaint();
        inputSignalSpectrum.repaint();
        sampledSignalSpectrum.repaint();
        outputSignalSpectrum.repaint();
```

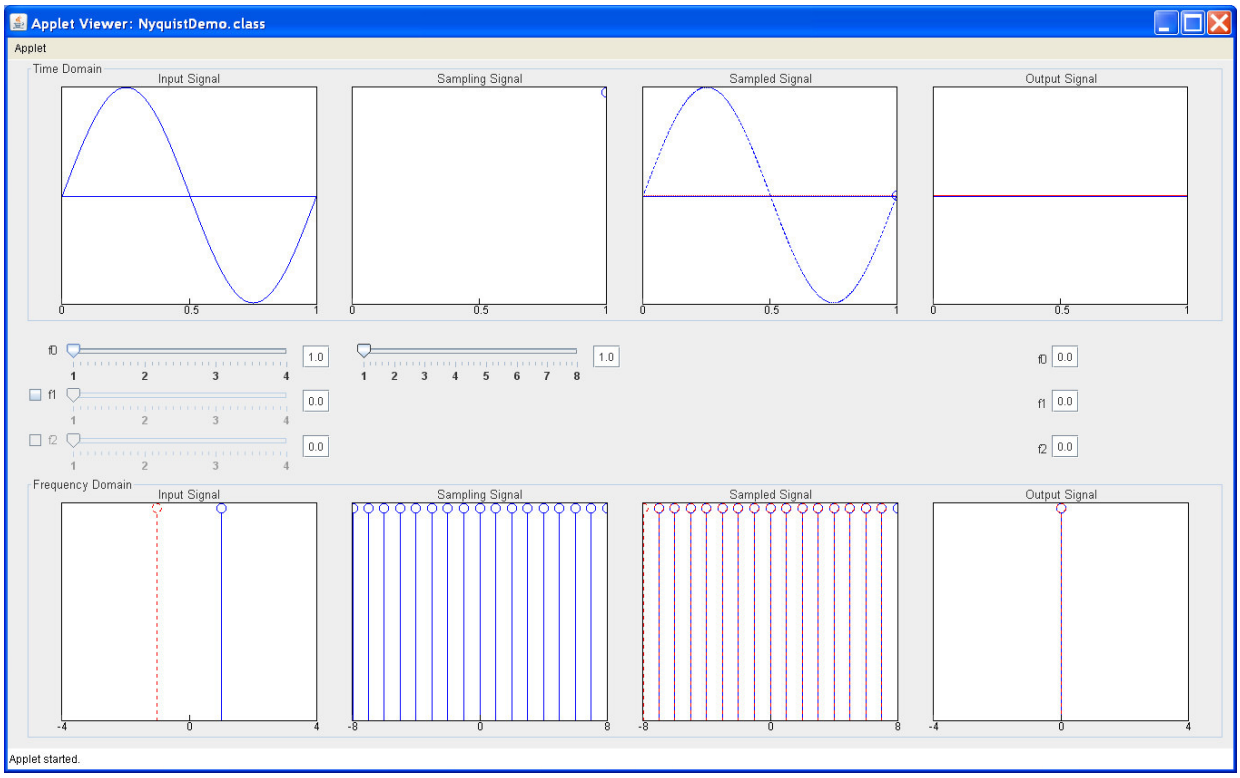
Αρχικά απενεργοποιούνται οι δυο ολισθητές για τις συχνότητες f1 και f2 του σήματος εισόδου (αυτό περιέχει μόνο μια συχνότητα, την f0). Επίσης απενεργοποιείται και το check box του τρίτου ολισθητή (της συχνότητας f2). Αυτό ενεργοποιείται μόνο αν έχει ενεργοποιηθεί ο ολισθητής 2.

Στη συνέχεια και με την εντολή `textBoxName.setText(text)` τοποθετούνται οι αρχικές τιμές των συχνοτήτων στα αντίστοιχα text boxes. Για να γραφούν οι σωστές συχνότητες εξόδου, γίνεται πρώτα ο υπολογισμός τους με τη συνάρτηση `calculateAliasingFreqs()`.

Καθώς, κατά την εκκίνηση της μικροεφαρμογής, το σήμα εισόδου αποτελείται από ένα και μόνο ημίτονο, θα πρέπει να ενημερωθούν τα αντικείμενα inputSignal, sampledSignal και outputSignal στο πεδίο του χρόνου καθώς και τα αντίστοιχα σήματα στο πεδίο των συχνοτήτων (τα inputSignalSpectrum, sampledSignalSpectrum και outputSignalSpectrum δηλαδή). Αυτό γίνεται μέσω της μεθόδου `setNorm(int)` που έχουν τα αντικείμενα αυτά.

Τέλος, ενημερώνονται οι τιμές του πίνακα `aliasingFreqs` για τα αντικείμενα sampledSignal και outputSignal και ξανασχεδιάζονται όσες γραφικές παραστάσεις είναι απαραίτητο.

Το παράθυρο της μικροεφαρμογής που προκύπτει με την εκτέλεση του πιο πάνω κώδικα (συνολικά) φαίνεται στην εικόνα 6.8



Εικόνα 6.8 Η διεπαφή της μικροεφαρμογής

6.5. Τα στοιχεία ελέγχου

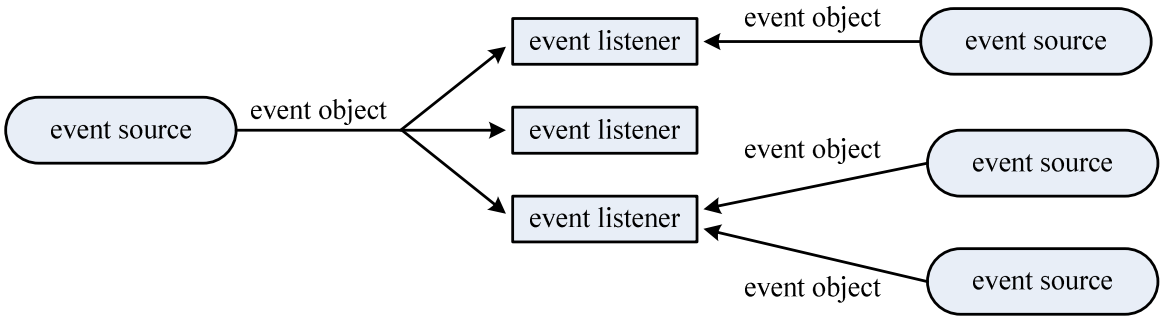
Η μικροεφαρμογή, όπως παρουσιάστηκε έως τώρα, έχει τη μορφή που επιλέχθηκε εξ' αρχής αλλά και η οποία δοκιμάστηκε στην εξομοίωση με το Matlab. Πέρα όμως από τη μορφή, πρέπει να προστεθεί και η λειτουργικότητα. Πρέπει λοιπόν να είναι σε θέση ο χρήστης να μεταβάλλει το σήμα εισόδου αλλά και την συχνότητα δειγματοληψίας ώστε να μπορέσει να «ανακαλύψει» το θεώρημα του Nyquist. Οι μεταβολές αυτές γίνονται με την μετακίνηση των ολισθητών και την επιλογή ή την αποεπιλογή των 2 check boxes. Τα 6 στοιχεία αυτά αποτελούν τα αντικείμενα ελέγχου της μικροεφαρμογής από τον χρήστη. Στην παράγραφο αυτή θα αναλυθεί πως η μεταβολή ενός από τα στοιχεία ελέγχου θα επηρεάζει την απόκριση της μικροεφαρμογής.

6.5.1. Τα γεγονότα (events) και οι ακροατές (listeners) στη Java

Ένα event (στο κείμενο θα χρησιμοποιείται αυτός ο όρος αν και ο όρος γεγονός είναι επίσης δόκιμος) δημιουργείται όταν κάτι αλλάζει μέσα σε ένα γραφικό περιβάλλον διεπαφής. Για παράδειγμα αν ο χρήστης πατήσει ένα κουμπί, επιλέξει ένα check box, γράψει κάτι σε ένα text field, κουνήσει το ποντίκι κ.α., τότε θα ενεργοποιηθεί ένα event.

Το αντικείμενο που ενεργοποίησε το event ονομάζεται και event source. Το event source θα αποστείλει ένα άλλο αντικείμενο, ένα event object το οποίο θα περιέχει πληροφορίες σχετικά με το event, όπως για παράδειγμα ένα αναγνωριστικό για το αντικείμενο που δημιούργησε το event. Αυτό το event object αναγνωρίζεται και επεξεργάζεται από ένα event listener το οποίο είναι ένα αντικείμενο συνδεδεμένο με το event source. Διαφορετικά είδη από γραφικά αντικείμενα έχουν και διαφορετικούς τύπους από event listeners. Για παράδειγμα ένας ολισθητής έχει ένα ChangeListener() (μεταξύ άλλων) αλλά όχι ένα ItemListener(). Αντιθέτως ένα check box έχει και ChangeListener() και ItemListener().

Οποιοσδήποτε αριθμός από αντικείμενα event listeners μπορούν να «ακροάζονται» για οποιοδήποτε τύπο event από οποιοδήποτε αριθμό από event sources. Για παράδειγμα μπορούμε να έχουμε σε ένα πρόγραμμα ένα event listener ανά event source. Ή μπορούμε να έχουμε έναν και μόνο event listener για όλα τα events από όλα τα sources. Μπορούμε να έχουμε ακόμα και περισσότερους listeners από έναν για ένα και μόνο event από ένα και μόνο συγκεκριμένο source.



Εικόνα 6.9 Σχέσεις μεταξύ event source και event listener

Τα events και τα event listeners βοηθάνε στο να προγραμματίζεται η εκτέλεση κάποιων ενεργειών όταν συμβεί κάτι σε κάποιο αντικείμενο του γραφικού περιβάλλοντος διεπαφής

χρήστη. Για παράδειγμα το τι θα συμβεί αν πατήσει ο χρήστης ένα κουμπί ή αν μετακινήσει έναν ολισθητή.

6.5.2. Οι eventListeners για τους ολισθητές και τα check boxes του applet

Υπάρχουν αρκετοί τρόποι να χρησιμοποιηθεί ένας event listener. Για κάθε αντικείμενο της κλάσης Swing υπάρχουν αρκετοί τέτοιοι event listeners. Στα διαθέσιμα **application programming interface** (**API**) στο διαδίκτυο μπορεί κανείς να βρει πώς θα υλοποιήσει τον listener που τον ενδιαφέρει. Έτσι για τους ολισθητές χρησιμοποιήθηκαν ChangeEvents και υλοποιήθηκαν ChangeListeners ενώ για τα check boxes χρησιμοποιήθηκαν ItemEvents και υλοποιήθηκαν ItemListeners.

```
// listen for changes to the sliders or the check boxes
        sliderSine1.addChangeListener(new ChangeListener() {  //
            @Override
            public void stateChanged(ChangeEvent ce) {
                freqs[0] = (double)sliderSine1.getValue()/10.0;
                sine1Text.setText(Double.toString(freqs[0]));
                calculateAliasingFreqs();
                outSine1Text.setText(Double.toString(aliasingFreqs[0]));
                sampledSignal.setAliasingFreqs(aliasingFreqs);
                outputSignal.setAliasingFreqs(aliasingFreqs);
                inputSignal.repaint();
                sampledSignal.repaint();
                outputSignal.repaint();
                inputSignalSpectrum.repaint();
                sampledSignalSpectrum.repaint();
                outputSignalSpectrum.repaint();
            }
        });
        sliderSine2.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent ce) {
                freqs[1] = (double)sliderSine2.getValue()/10.0;
                sine2Text.setText(Double.toString(freqs[1]));
                calculateAliasingFreqs();
                outSine2Text.setText(Double.toString(aliasingFreqs[1]));
                sampledSignal.setAliasingFreqs(aliasingFreqs);
```

```
            outputSignal.setAliasingFreqs(aliasingFreqs);
            inputSignal.repaint();
            sampledSignal.repaint();
            outputSignal.repaint();
            inputSignalSpectrum.repaint();
            sampledSignalSpectrum.repaint();
            outputSignalSpectrum.repaint();
        }
    });
    sliderSine3.addChangeListener(new ChangeListener() {
        @Override
        public void stateChanged(ChangeEvent ce) {
            freqs[2] = (double)sliderSine3.getValue()/10.0;
            sine3Text.setText(Double.toString(freqs[2]));
            calculateAliasingFreqs();
            outSine3Text.setText(Double.toString(aliasingFreqs[2]));
            sampledSignal.setAliasingFreqs(aliasingFreqs);
            outputSignal.setAliasingFreqs(aliasingFreqs);
            inputSignal.repaint();
            sampledSignal.repaint();
            outputSignal.repaint();
            inputSignalSpectrum.repaint();
            sampledSignalSpectrum.repaint();
            outputSignalSpectrum.repaint();
        }
    });

    sliderSampler.addChangeListener(new ChangeListener() {
        @Override
        public void stateChanged(ChangeEvent ce) {
            samplingFreq = (double)sliderSampler.getValue()/10.0;
            samplerText.setText(Double.toString(samplingFreq));
            calculateAliasingFreqs();
            outSine1Text.setText(Double.toString(aliasingFreqs[0]));
            outSine2Text.setText(Double.toString(aliasingFreqs[1]));
            outSine3Text.setText(Double.toString(aliasingFreqs[2]));
            samplingSignal.setSamplingFreq(samplingFreq);
            sampledSignal.setSamplingFreq(samplingFreq);
            sampledSignal.setAliasingFreqs(aliasingFreqs);
```

```
            outputSignal.setAliasingFreqs(aliasingFreqs);
            samplingSignal.repaint();
            sampledSignal.repaint();
            outputSignal.repaint();
            samplingSignalSpectrum.setSamplingFreq(samplingFreq);
            samplingSignalSpectrum.repaint();
            sampledSignalSpectrum.setSamplingFreq(samplingFreq);
            sampledSignalSpectrum.repaint();
            outputSignalSpectrum.repaint();
        }
    });

    chkSine2.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chkSine2.isSelected()) {
              sliderSine2.setEnabled(true);
              chkSine3.setEnabled(true);
              freqs[1] = sliderSine2.getValue()/10.0;
              inputSignal.setNorm(2);
              sampledSignal.setNorm(2);
              outputSignal.setNorm(2);
              calculateAliasingFreqs();
              inputSignalSpectrum.setNorm(2);
              sampledSignalSpectrum.setNorm(2);
              outputSignalSpectrum.setNorm(2);
            } else {
              sliderSine2.setEnabled(false);
              chkSine3.setSelected(false);
              chkSine3.setEnabled(false);
              freqs[1] = 0.0;
              inputSignal.setNorm(1);
              sampledSignal.setNorm(1);
              outputSignal.setNorm(1);
              aliasingFreqs[1] = 0.0;
              inputSignalSpectrum.setNorm(1);
              sampledSignalSpectrum.setNorm(1);
              outputSignalSpectrum.setNorm(1);
```

```
            }
            sine2Text.setText(Double.toString(freqs[1]));
            outSine2Text.setText(Double.toString(aliasingFreqs[1]));
            inputSignal.repaint();
            sampledSignal.repaint();
            outputSignal.repaint();
            inputSignalSpectrum.repaint();
            sampledSignalSpectrum.repaint();
            outputSignalSpectrum.repaint();
        }
    });
    chkSine3.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chkSine3.isSelected()) {
              sliderSine3.setEnabled(true);
              freqs[2] = sliderSine3.getValue()/10.0;
              calculateAliasingFreqs();
              inputSignal.setNorm(3);
              sampledSignal.setNorm(3);
              outputSignal.setNorm(3);
              inputSignalSpectrum.setNorm(3);
              sampledSignalSpectrum.setNorm(3);
              outputSignalSpectrum.setNorm(3);
            } else {
              sliderSine3.setEnabled(false);
              freqs[2] = 0.0;
              aliasingFreqs[2] = 0.0;
              inputSignal.setNorm(2);
              sampledSignal.setNorm(2);
              outputSignal.setNorm(2);
              inputSignalSpectrum.setNorm(2);
              sampledSignalSpectrum.setNorm(2);
              outputSignalSpectrum.setNorm(2);
            }
            sine3Text.setText(Double.toString(freqs[2]));
            outSine3Text.setText(Double.toString(aliasingFreqs[2]));
            inputSignal.repaint();
```

```
                sampledSignal.repaint();
                outputSignal.repaint();
                inputSignalSpectrum.repaint();
                sampledSignalSpectrum.repaint();
                outputSignalSpectrum.repaint();
            }
        });

    }
```

Η λειτουργία κάθε ενός από τους τρεις ChangeListeners() για τους τρεις ολισθητές του σήματος εισόδου είναι παρόμοια και γι΄αυτό θα περιγραφεί μόνο για τον πρώτο. Αρχικά ορίζεται ένας νέος ChangeListener() ο οποίος συνδέεται με το αντικείμενο sliderSine1. Η μόνη μέθοδος που περιέχει ο ChangeListener() είναι η `stateChanged( ChangeEvent e)` η οποία και γίνεται override. Σε αυτή ορίζεται το τι θα γίνει εάν αλλάξει η κατάσταση του ολισθητή, αν δηλαδή μετακινηθεί ή αν γίνει enabled ή disabled (με τα δυο τελευταία να ισχύουν μόνο για τους ολισθητές sliderSine2 και sliderSine3).

Το πρώτο το οποίο θα πρέπει να γίνει είναι να διαβαστεί η νέα τιμή του ολισθητή και να αποθηκευτεί στην αντίστοιχη θέση του πίνακα `freqs[]`. Μετά ενημερώνεται με την καινούργια τιμή το text box δίπλα στον ολισθητή. Κατόπιν υπολογίζονται εκ νέου οι αναδιπλωμένες συχνότητες και ενημερώνεται και το αντίστοιχο text box. Ακολουθεί η ενημέρωση των τιμών του πίνακα `aliasingFreqs[]` για τα αντικείμενα sampledSignal και outputSignal και ξανασχεδιάζονται όσες γραφικές παραστάσεις είναι απαραίτητο, δηλαδή το σήμα εισόδου, το δειγματοληπτημένο σήμα και το σήμα εξόδου, τόσο στο πεδίο του χρόνου όσο και στο πεδίο των συχνοτήτων.

Η λειτουργία του ChangeListener() για τον ολισθητή που καθορίζει τη συχνότητα δειγματοληψίας, μοιάζει με αυτή που περιγράφηκε μόλις. Αρχικά ορίστηκε έναν νέο ChangeListener() τον οποίο συνδέθηκε με το αντικείμενο sliderSsmpler. Στη μέθοδο `stateChanged( ChangeEvent e)` καθορίζεται ότι αν αλλάξει η τιμή του ολισθητή, θα ενημερωθεί πρώτα η τιμή της μεταβλητής `samplingFreq` και το αντίστοιχο text box. Μετά θα υπολογιστούν οι νέες αναδιπλωμένες συχνότητες με την ταυτόχρονη ενημέρωση των text boxes στο σήμα εξόδου. Ακολουθεί η ενημέρωση της τιμής της μεταβλητής `samplingFreq` για τα αντικείμενα samplingSignal και sampledSignal όπως και για τα αντικείμενα

samplingSignalSpectrum και sampledSignalSpectrum. Επίσης ενημερώνονται οι τιμές του πίνακα `aliasingFreqs[]` για τα αντικείμενα sampledSignal και outputSignal, όπως και στην περίπτωση του sliderSine1. Τα γραφήματα που πρέπει να ξανασχεδιαστούν είναι όλα εκτός από το σήμα εισόδου (στο πεδίο του χρόνου και στο πεδίο των συχνοτήτων).

Από τα δυο check boxes, θα περιγραφεί η λειτουργία του δεύτερου μόνο. Αρχικά ορίζεται έναν νέο ItemListener() τον οποίο συνδέεται με το αντικείμενο chkSine3. Η μέθοδος η οποία γίνεται override τώρα είναι η `itemStateChanged( ChangeEvent e)`. Σε αυτή διακρίνονται δυο περιπτώσεις, του τι γίνεται εάν το check box είναι πλέον επιλεγμένο και τι εάν έχει αποεπιλεγεί. Στην μεν πρώτη περίπτωση θα πρέπει να ενεργοποιηθεί αρχικά ο sliderSine3 και η τιμή που έχει να αποθηκευτεί στην αντίστοιχη θέση του πίνακα `freqs[]`. Στη συνέχεια υπολογίζονται και πάλι οι αναδιπλωμένες συχνότητες. Η επιλογή του chkSine3 σημαίνει ότι το σήμα εισόδου αποτελείται πλέον από 3 ημίτονα. Επειδή θέλουμε το πλάτος όλων των σημάτων να είναι κανονικοποιημένο στη μονάδα, θα πρέπει να ενημερωθεί η τιμή της μεταβλητής `norm` στα αντικείμενα inputSignal, sampledSignal και outputSignal καθώς και στα αντίστοιχα αντικείμενα στο πεδίο των συχνοτήτων.

Στη περίπτωση που το check box ήταν επιλεγμένο αλλά όχι πια, τότε ο ολισθητής sliderSine3 απενεργοποιείται και η αντίστοιχη συχνότητα τίθεται στο 0. Το ίδιο συμβαίνει και με την αναδιπλωμένη συχνότητα ενώ ενημερώνονται τα σχετικά αντικείμενα ότι η τιμή της `norm` θα πρέπει να αλλάξει σε 2.

Πάντως και στις δυο περιπτώσεις θα πρέπει να ενημερωθούν τα κατάλληλα text boxes και να ξανασχεδιαστεί τα σήμα εισόδου, το δειγματοληπτημένο και το σήμα εξόδου όπως και τα φάσματά τους.

6.5.3. Ο τρόπος υπολογισμού των αναδιπλωμένων συχνοτήτων

Σε πολλά σημεία του κώδικα που παρουσιάστηκε μέχρι τώρα γίνεται κλήση της μεθόδου `calculateAliasingFreqs()` η οποία υπολογίζει (όπως μαρτυρεί και το όνομά της) τις αναδιπλωμένες συχνότητες στο σήμα εξόδου. Ο υπολογισμός αυτός γίνεται με απλό μαθηματικό τρόπο. Όπως έχει αναφερθεί και στο κεφάλαιο 3, αν δεν ικανοποιείται η συνθήκη $f_s > 2f_{max}$, τότε εμφανίζονται αναδιπλωμένες συχνότητες που στην πραγματικότητα δεν υπάρχουν στο αρχικό σήμα. Στην περίπτωση που στην είσοδο υπάρχει ένα σήμα που προκύπτει από το άθροισμα απλών ημιτονικών σημάτων, τότε οι αναδιπλωμένες συχνότητες

προκύπτουν από την διαφορά των συχνοτήτων του αρχικού σήματος από την συχνότητα δειγματοληψίας ($f_0 - f_s$, $f_1 - f_s$ και $f_2 - f_s$,). Για να είναι σίγουρο ότι η πράξη της αφαίρεσης δεν θα δώσει μεγάλο αριθμό από δεκαδικά ψηφία (λόγω ακρίβειας της μηχανής), έχει υλοποιηθεί και μια συνάρτηση στρογγυλοποίησης σε καθορισμένο αριθμό δεκαδικών ψηφίων, η `round(double value, int places)`.

```
/**
 * Function for calculating aliasing frequencies
 *
 * @param     void: acts on public variable freqs[]
 * @return    void: result is stored in public variable aliasingFreqs[]
 */
void calculateAliasingFreqs(){
  if (freqs[0] >= samplingFreq/2.0) {
        aliasingFreqs[0] = round(freqs[0] - samplingFreq,2);
        //aliasingFreqs[0] = round(freqs[0]%samplingFreq,2);
  }
  else {
        aliasingFreqs[0] = freqs[0];
  }

  if (sliderSine2.isEnabled() && (freqs[1] >= samplingFreq/2.0)) {
        aliasingFreqs[1] = round(freqs[1] - samplingFreq,2);
  }
  else if (sliderSine2.isEnabled() && (freqs[1] < samplingFreq/2.0)) {
              aliasingFreqs[1] = freqs[1];
        }
  else if (!sliderSine2.isEnabled()) {
        aliasingFreqs[1] = 0.0;
  }

  if (sliderSine3.isEnabled() && (freqs[2] >= samplingFreq/2.0)) {
        aliasingFreqs[2] = round(freqs[2] - samplingFreq,2);
  }
  else if (sliderSine3.isEnabled() && (freqs[2] < samplingFreq/2.0)) {
        aliasingFreqs[2] = freqs[2];
  }
```

```
      else if (!sliderSine3.isEnabled()) {
            aliasingFreqs[2] = 0.0;
      }
}

    /**
     * Function for rounding aliasing frequencies to N decimal places
     *
     * @param     value (double):   value which needs to be rounded to specific
     *                              number of decimal places
     * @param     places (int):     number of decimal places
     * @return    double :    result is a double value even though internally
     *                        the calculations are done using BigDecimal
     */
    public static double round(double value, int places) {
        if (places < 0) throw new IllegalArgumentException();

        BigDecimal bd = new BigDecimal(value);
        bd = bd.setScale(places, RoundingMode.HALF_UP);
        return bd.doubleValue();
    }
```

Αν η συνθήκη $f_s > 2f_{max}$ ικανοποιείται τότε δεν έχουμε την εμφάνιση αναδιπλωμένων συχνοτήτων. Για καλύτερη λειτουργία της μικροεφαρμογής όμως θεωρούμε ότι υπάρχουν αναδιπλωμένες συχνότητες οι οποίες όμως έχουν την ίδια τιμή με τις κανονικές.

6.6. Τα γραφήματα

Στην παρουσίαση της μικροεφαρμογής μέχρι τώρα έχει γίνει μια επιφανειακή αναφορά στα γραφήματα που υλοποιήθηκαν. Ουσιαστικά έχει γίνει αναφορά μόνο στην δημιουργία των αντικειμένων στα οποία θα σχεδιαστούν τα γραφήματα και τη τοποθέτησή τους με την μέθοδο add(). Στην παράγραφο αυτή θα παρουσιαστεί ο τρόπος με τον οποίο υλοποιούνται οι διάφορες γραφικές παραστάσεις. Στον κώδικα έχουν δημιουργηθεί για το σκοπό αυτό 3 κλάσεις. Η κλάση DrawingPanel είναι αυτή η οποία αναλαμβάνει τη σχεδίαση του γραφήματος στην κατάλληλη περιοχή της μικροεφαρμογής. Τα γραφήματα αυτά μπορεί να απεικονίζουν σήματα συνεχή στο χρόνο ή διακριτά (στο χρόνο ή τη συχνότητα). Οι κλάσεις ContinuousSignal και StemSignal υπάρχουν για να γίνεται καλύτερα αυτός ο διαχωρισμός.

6.6.1. Η κλάση DrawingPanel

Για τη σχεδίαση γραμμών και γεωμετρικών σχημάτων στη Java χρησιμοποιούνται συνήθως αντικείμενα δυο κλάσεων, ή της κλάσης Canvas ή της κλάσης JPanel (ή Panel). Όμως ποιο από τα δυο θα πρέπει να χρησιμοποιηθεί στην μικροεφαρμογή που αναπτύσσεται εδώ; Η μεν κλάση Canvas ανήκει στο πακέτο AWT ενώ η JPanel στο Swing. Πέρα από το γεγονός ότι γίνεται προσπάθεια η μικροεφαρμογή να υλοποιηθεί μόνο με αντικείμενα Swing, υπάρχει και άλλος ένας λόγος να χρησιμοποιηθεί η κλάση JPanel. Αυτός είναι ότι η κλάση αυτή είναι πιο «ελαφριά» από την Canvas. Επειδή στην εφαρμογή θα υπάρχουν 16 τέτοια αντικείμενα (8 για τα γραφήματα και 8 για τα πλαίσια και τους άξονες) είναι καλύτερο να χρησιμοποιηθεί η κλάση JPanel καθώς έτσι η επανασχεδίαση των αντικειμένων να γίνεται πιο γρήγορα και χωρίς να παρουσιάζεται τρεμόπαιγμα (flickering) όταν θα γίνεται ανανέωση των γραφημάτων. Μια δοκιμαστική έκδοση που υλοποιήθηκε έδειξε ότι ακόμα και με 4 αντικείμενα Canvas το flickering γινόταν αισθητό όταν η εφαρμογή έτρεχε μέσα από έναν φυλλομετρητή.

Ακολουθεί η αναλυτική παρουσίαση του πως υλοποιείται η κλάση DrawingPanel. Η κλάση αυτή θα ανήκει στο πακέτο draw που έχει δημιουργηθεί για να περιλαμβάνει όλα τα σχεδιαστικά στοιχεία της μικροεφαρμογής. Από τις δηλώσεις import για την κλάση αυτή που φαίνονται πιο κάτω μπορεί να διαπιστωθεί ότι οι περισσότερες είναι από το πακέτο java.awt. Αυτό δεν έρχεται σε αντίθεση με την απαίτηση να χρησιμοποιούνται αντικείμενα του Swing

και όχι του AWT. Το Swing ουσιαστικά επεκτείνει το AWT και έτσι μπορεί κανείς να χρησιμοποιήσει και κλάσεις του τελευταίου. Ίσως για το λόγο αυτό πολλές κλάσεις του AWT (όπως η Color ή η Graphics) δεν έχουν αντικατασταθεί με νέες από το πακέτο Swing.

```
package draw;

import javax.swing.JPanel;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Stroke;
import java.util.Arrays;

import draw.ContinuousSignal;
import draw.StemSignal;
```

Η κλάση λοιπόν DrawingPanel θα πρέπει να κληρονομεί και να επεκτείνει τη Jpanel, όπως φαίνεται και στο κομμάτι κώδικα που ακολουθεί.

```
public class DrawingPanel extends JPanel {

	private double freqs[] = {1.0, 0.0, 0.0};
	private double aliasingFreqs[] = {0.0, 0.0, 0.0};
	private double samplingFreq = 1.0;
	private int norm;
	private int stemCircleRadius = 10;
	private int plotAreaInset = 10;
	private String type;
	private Color lineColor = Color.blue;

	public DrawingPanel() {
		setSize(300, 280);
		setBackground(Color.white);
	}

	public DrawingPanel(double[] f){
		this.freqs = f;
```

```
        setSize(300, 280);
        setBackground(Color.white);
        type = "singleContinuous";
    }

    public DrawingPanel(double[] f, Color c){
        this.freqs = f;
        this.lineColor = c;
        setSize(300, 280);
        setBackground(Color.white);

        type = "singleContinuous";
    }

    public DrawingPanel(double fs){
        this.samplingFreq = fs;
        setSize(300, 280);
        setBackground(Color.white);
        type = "singleStem";
    }

    public DrawingPanel(double[] f, double fs){
        this.freqs = f;
        this.samplingFreq = fs;
        setSize(300, 280);
        setBackground(Color.white);
        type = "multiplot";
    }

    public DrawingPanel(double[] f, double fs, String type){
        this.freqs = f;
        this.samplingFreq = fs;
        setSize(300, 280);
        setBackground(Color.white);
        this.type = type;
    }

    public DrawingPanel(double[] f, String type){
```

```
            this.freqs = f;
            setSize(300, 280);
            setBackground(Color.white);
            this.type = type;
      }

      public DrawingPanel(double fs, String type){
            this.samplingFreq = fs;
            setSize(300, 280);
            setBackground(Color.white);
            this.type = type;
      }
```

Οι μεταβλητές τις οποίες έχουν όλα τα αντικείμενα της κλάσης DrawingPanel και ο σκοπός της κάθε μιας φαίνονται στον πίνακα 6.3.

Πίνακας 6.5 Μεταβλητές της κλάσης DrawingPanel

Όνομα μεταβλητής	Τύπος	Σκοπός
freqs	double[3]	Αποθηκεύονται οι τιμές των συχνοτήτων του σήματος εισόδου
aliasingFreqs	double[3]	Αποθηκεύονται οι τιμές των αναδιπλωμένων συχνοτήτων
samplingFreq	double	Για να κρατά την τιμή της συχνότητας δειγματοληψίας
norm	int	Ο αριθμός των ημιτόνων στο σήμα εισόδου (μπορεί να είναι 1 ή 2 ή 3). Για την κανονικοποίηση των σημάτων.
stemCircleRadius	int	Η ακτίνα του κύκλου που σχεδιάζεται στο άκρο κάθε δείγματος.
plotAreaInset	int	Η απόσταση μεταξύ των ορίων του JPanel και της περιοχής όπου σχεδιάζεται το γράφημα.
type	String	Ο τύπος του γραφήματος. Η μεταβλητή αυτή χρησιμοποιείται για να γίνει διάκριση του πως θα σχεδιαστεί το κάθε γράφημα.
lineColor	Color	Το χρώμα που θα χρησιμοποιηθεί για τη σχεδίαση του γραφήματος. Η προεπιλογή είναι μπλέ.

Οι constructors της κλάσης DrawingPanel, ορίζουν τρία βασικά στοιχεία για κάθε αντικείμενο, το μέγεθος, το χρώμα και τον τύπο. Ανάλογα με τα ορίσματα που δέχεται κάθε constructor, ανάλογος είναι και ο τύπος του γραφήματος. Με βάση των τύπο αυτόν επιλέγονται τα σήματα και οι διαδικασίες για τη σχεδίασή τους. Πριν όμως παρουσιαστεί πώς γίνεται αυτό, πρέπει να αναφερθούν οι setters και οι getters της κλάσης.

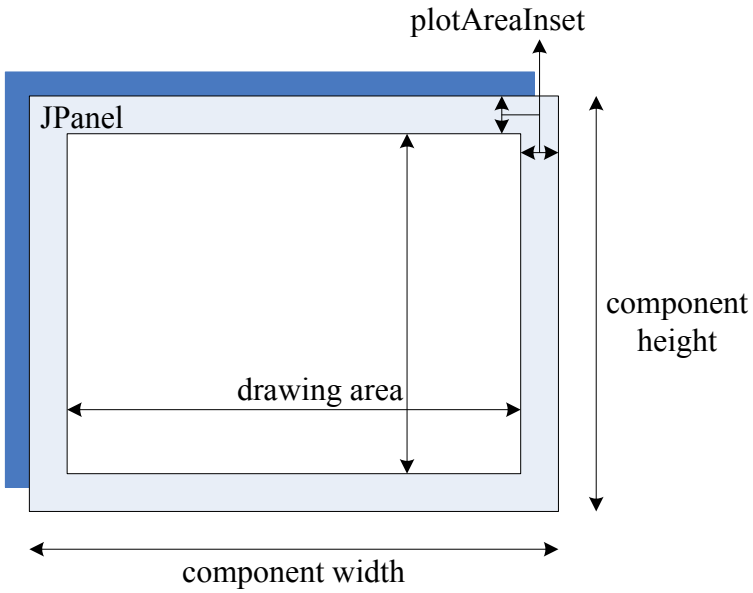
```
/**
 * Setters
 */
  public void setSamplingFreq(double Fs){
        this.samplingFreq = Fs;
  }

  public void setAliasingFreqs(double[] fa){
        this.aliasingFreqs = fa;
  }

  public void setNorm(int n){
        this.norm = n;
  }

/**
 * Getters
 */
  public int getMaxX(){
        return this.getWidth()-2*plotAreaInset;
  }
```

Οι setters χρησιμοποιούνται για να δοθούν τιμές στις μεταβλητές samplingFreq, aliasingFreqs και norm. Ο μοναδικός getter επιστρέφει το πλάτος της ωφέλιμης περιοχής του γραφήματος. Μια λεπτομέρεια που αξίζει να σημειωθεί είναι ότι δεν υπάρχει setter για την μεταβλητή freqs. Ο λόγος είναι ότι η μεταβλητή αυτή είναι πίνακας και άρα για την java είναι ένα αντικείμενο. Άρα οποιαδήποτε αλλαγή στην τιμή του αντικειμένου στον κυρίως κώδικα που παρουσιάστηκε στην παράγραφο 6.5 (όπου υπάρχει ένας πίνακας με το ίδιο όνομα) επηρεάζει την τιμή του πίνακα στο αντικείμενο της κλάσης DrawinPanel. Φυσικά κάτι τέτοιο ισχύει και για τον πίνακα aliasingFreqs αλλά απλώς δεν χρησιμοποιήθηκε αυτή η

προσέγγιση αλλά επιλέχθηκε να υλοποιηθεί ο setter ώστε να είναι πιο εύκολος στην αντίληψη ο κώδικας και να επισημανθεί και αυτό το χαρακτηριστικό.

6.6.2. Η μέθοδος σχεδίασης paintComponent(g) της κλάσης DrawingPanel

Για να γίνει η σχεδίαση των γραφημάτων αλλά και οποιωνδήποτε άλλων σχημάτων θα πρέπει να χρησιμοποιηθεί η μέθοδος `paintComponent(Graphics g)` η οποία θα γίνει override. Αυτή η μέθοδος δέχεται σαν είσοδο ένα αντικείμενο της κλάσης Graphics. Αρχικά υπολογίζεται το μέγεθος της περιοχής όπου θα γίνει η σχεδίαση. Αυτή δεν είναι όλη η περιοχή του JPanel καθώς είναι επιθυμητό να υπάρχει χώρος για τους άξονες και τις ετικέτες. Έτσι η περιοχή σχεδίασης είναι μικρότερη κατά `2*plotAreaInset` pixels σε κάθε διάσταση απ' ότι το JPanel.



Εικόνα 6.10 Η περιοχή σχεδίασης μέσα στο JPanel

Στο τμήμα του κώδικα που ακολουθεί φαίνονται επίσης και οι διαφορετικές περιπτώσεις οι οποίες λαμβάνονται υπόψη για τη σχεδίαση των γραφημάτων. Έτσι υπάρχουν οι ακόλουθες περιπτώσεις ανάλογα με την τιμή της μεταβλητής type:

- όταν type = singleContinuous πρόκειται να σχεδιαστεί μια μόνο γραφική παράσταση με συνεχείς τιμές στο πεδίο του χρόνου. Αυτή η περίπτωση περιλαμβάνει το γράφημα του σήματος εισόδου και αυτό του σήματος εξόδου. Η διαφοροποίηση είναι ότι αυτό

του σήματος εξόδου σχεδιάζεται με κόκκινο χρώμα. Σε κάθε περίπτωση ορίζεται ένα νέο αντικείμενο της κλάσης ContinuousSignal για δεδομένες συχνότητες εισόδου, μέγεθος και παράγοντα κανονικοποίησης και στη συνέχεια σχεδιάζεται με τη μέθοδο plotGraph.

- όταν type = singleStem πρόκειται να σχεδιαστεί μια μόνο γραφική παράσταση με διακριτές τιμές στο πεδίο του χρόνου. Αυτή η περίπτωση περιλαμβάνει το γράφημα του σήματος δειγματοληψίας. Έτσι ορίζεται ένα νέο αντικείμενο της κλάσης StemSignal για δεδομένο μέγεθος, απόσταση μεταξύ των δειγμάτων και ύψος όλων των stems και στη συνέχεια σχεδιάζεται με τη μέθοδο plotStems.

- όταν type = multiPlot πρόκειται να σχεδιαστεί ένας συνδυασμός συναρτήσεων. Πρόκειται για την περίπτωση του δειγματοληπτημένου σήματος που περιλαμβάνει πέρα από το ίδιο το δειγματοληπτημένο σήμα, την καμπύλη του σήματος εισόδου (με μπλε χρώμα και διακεκομμένες γραμμές) και την καμπύλη του σήματος εξόδου (με κόκκινο χρώμα και διακεκομμένες γραμμές. Για το λόγο αυτό ορίζονται δυο νέα αντικείμενα της κλάσης ContinuousSignal (ένα για το σήμα εισόδου και ένα για το σήμα εξόδου) τα οποία σχεδιάζονται με τη μέθοδο plotGraph και ένα νέο αντικείμενο της κλάσης StemSignal για το δειγματοληπτημένο σήμα το οποίο στη συνέχεια σχεδιάζεται με τη μέθοδο plotStems.

- όταν type = spectrum πρόκειται να σχεδιαστεί το φάσμα του σήματος εισόδου ή αυτό του σήματος εξόδου. Ορίζονται δυο πίνακες για την αποθήκευση των τετμημένων των φασματικών γραμμών. Ο ένας έχει τις τετμημένες των πραγματικών (θετικών) συχνοτήτων ενώ ο δεύτερος των αρνητικών συχνοτήτων. Στη συνέχεια υπολογίζονται οι τετμημένες και τέλος, με την μέθοδο plotStems, σχεδιάζονται οι φασματικές γραμμές. Οι θετικές συχνότητες σχεδιάζονται με μπλε συνεχόμενη γραμμή και οι αρνητικές με κόκκινη διακεκομμένη.

- όταν type = diracCombSpectrum πρόκειται να σχεδιαστεί το φάσμα του σήματος δειγματοληψίας. Όπως έχει ήδη αναφερθεί και στο κεφάλαιο 3, το φάσμα αυτού του σήματος αποτελείται από μια άπειρη σειρά από φασματικές γραμμές με απόσταση μεταξύ τους ίση με την συχνότητα δειγματοληψίας. Οι τετμημένες των φασματικών γραμμών υπολογίζονται από τη μέθοδο evaluateDiracCombSpectrum ενώ όλες οι φασματικές γραμμές σχεδιάζονται με χρώμα μπλε.

- όταν type = sampledSignalSpectrum πρόκειται να σχεδιαστεί το φάσμα του δειγματοληπτημένου σήματος. Αυτό προκύπτει από την συνέλιξη του φάσματος του σήματος πληροφορίας με το σήμα δειγματοληψίας. Εδώ χρησιμοποιείται η μέθοδος convolution() για να εξομοιωθεί η συνέλιξη μεταξύ του φάσματος του σήματος δειγματοληψίας και του φάσματος του σήματος πληροφορίας. Για το τελευταίο χρησιμοποιείται τη μέθοδο copyOfRange του τύπο Arrays ώστε να μείνουν μόνο οι συχνότητες του πίνακα freqs οι οποίες πραγματικά συμμετέχουν στο φάσμα του σήματος εισόδου. Αν δεν γινόταν αυτό τότε το σήμα εισόδου θα παρουσιάζονταν να έχει πάντα 3 συχνότητες ακόμα και αν κάποια ή κάποιες από αυτές θα ήταν ίσες με το μηδέν.

```
      // override paintComponent
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);

        int plotAreaHeight = this.getHeight()-2*plotAreaInset;     //Evaluate
plot area height
        int plotAreaWidth = this.getWidth()-2*plotAreaInset;       //Evaluate
plot area width

        // Calculate the middle of y-axis
        int middleY = (int) plotAreaHeight/2;

        if (type == "singleContinuous") {
            // Set the line color and draw a horizontal axis
            g.setColor(Color.blue);
            g.drawLine(0, plotAreaInset+middleY, plotAreaInset+plotAreaWidth-1,
plotAreaInset+middleY);
            g.setColor(lineColor);

            int[] size = {plotAreaWidth, plotAreaHeight};
            ContinuousSignal singleSignal = new ContinuousSignal(freqs, size,
norm);
            plotGraph(singleSignal.getXcoord(),singleSignal.getYcoord(),      g,
"line");
          }
```

```
          if (type == "singleStem") {
            g.setColor(Color.blue);
            StemSignal samplingSignal = new StemSignal(plotAreaWidth,
(int)(plotAreaWidth/samplingFreq), 100);
            plotStems(samplingSignal.getXcoord(), g);
          }

          if (type == "multiplot") {
            // Set the line color and draw a horizontal axis
            g.setColor(Color.blue);
            g.drawLine(plotAreaInset, plotAreaInset+middleY,
plotAreaWidth+plotAreaInset-1, plotAreaInset+middleY);

            int[] size = {plotAreaWidth, plotAreaHeight};
            // add input signal
            ContinuousSignal inputSignal = new ContinuousSignal(freqs, size,
norm);
            plotGraph(inputSignal.getXcoord(), inputSignal.getYcoord(), g,
"dash");
            //add output signal
            g.setColor(Color.red);
            ContinuousSignal outputSignal = new ContinuousSignal(aliasingFreqs,
size, norm);
            plotGraph(outputSignal.getXcoord(), outputSignal.getYcoord(), g,
"dash");
            // add sampling signal x input signal
            g.setColor(Color.blue);
            StemSignal sampledSignal = new StemSignal(plotAreaWidth,
(double)(plotAreaWidth/samplingFreq), outputSignal.getYcoord());
            plotStems(sampledSignal.getCoords(), g);
          }

          if (type == "spectrum") {
            g.setColor(Color.blue);
            //arrays for frequencies stems and mirror stems
            int[] stems = new int[norm];
            int[] stemsImg = new int[norm];

            for (int i = 0; i < norm; i=i+1) {
              stems[i] =(int) (plotAreaWidth/2 + freqs[i]*plotAreaWidth/8);
```

```
      stemsImg[i] =(int) (plotAreaWidth/2 - freqs[i]*plotAreaWidth/8);
    }
    //StemSignal inputSignalSpectrum = new StemSignal(stems, 100);
    //plotStems(inputSignalSpectrum.getXcoord(), g);
    plotStems(stems, g);
    g.setColor(Color.red);
    plotStems(stemsImg, g, "dash");
  }

  if (type == "diracCombSpectrum") {
    g.setColor(Color.blue);
    int[] stems = new int[(int) (8/samplingFreq)*2+1];

    double[] xs = evaluateDiracCombSpectrum(samplingFreq);
    for (int i = 0; i < xs.length; i=i+1){
      stems[i] =(int) (plotAreaWidth/2 + xs[i]*plotAreaWidth/16);
    }
    //StemSignal inputSignalSpectrum = new StemSignal(stems, 100);
    //plotStems(inputSignalSpectrum.getXcoord(), g);
    plotStems(stems, g);
  }

  if (type == "sampledSignalSpectrum") {
    g.setColor(Color.blue);
    int[] stems = new int[(int) (8/samplingFreq)*2*norm+1];
    int[] stemsImg = new int[(int) (8/samplingFreq)*2*norm+1];

    double[] newFreqsArray = Arrays.copyOfRange(freqs, 0, norm);
    double[] xs = evaluateDiracCombSpectrum(samplingFreq);
    double[] conv = convolution(xs, newFreqsArray);

    for (int i = 0; i < stems.length; i=i+1){
      stems[i] =(int) (plotAreaWidth/2 + conv[i]*plotAreaWidth/16);
    }
    for (int i = 0; i < stemsImg.length; i=i+1){
      stemsImg[i] =(int) (plotAreaWidth/2 - conv[i]*plotAreaWidth/16);
    }
```

```
            //StemSignal inputSignalSpectrum = new StemSignal(stems, 100);
            //plotStems(inputSignalSpectrum.getXcoord(), g);
            plotStems(stems, g);
            g.setColor(Color.red);
            plotStems(stemsImg, g, "dash");
          }
    }
```

6.6.3. Οι μέθοδοι σχεδίασης συνεχών και διακριτών σημάτων

Για να σχεδιαστεί το γράφημα μιας συνάρτησης με συνεχείς τιμές τόσο στον οριζόντιο, όσο και στον κατακόρυφο άξονα (συνεχής συνάρτηση) υλοποιήθηκε η μέθοδος plotGraph.

```
/**
 * Method for plotting a graph by drawing lines between 2 subsequent points
 *
 * @param   x (int [])  :     array holding the values of the x-coordinate
 * @param   y (int [])  :     array holding the values of the y-coordinate
 * @param   graph (Graphics): the graphics component which will be (re)drawn
 * @param   linetype(String): if this String is set to "dash", a dashed line
 *                            will be drawn. In any other case the line will
 *                            be continuous.
 * @return  void        :     result is the drawn line
 */
void plotGraph(int[] x, int[] y, Graphics graph, String linetype){

        if (linetype == "dash") {
          Graphics2D g2d = (Graphics2D) graph.create(); // copy graph instance
          Stroke dashed = new BasicStroke(1, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND, 0, new float[]{4, 4}, 0);
          g2d.setStroke(dashed);

          int cnt = 1;
          boolean toggle = true;
          for (int i = 1; i < x.length; i=i+1) {
              cnt--;
              if (cnt == 0){
                    toggle = !toggle;
                    cnt = 1;
              }
```

```
                if (toggle){
                      int x1 =plotAreaInset+x[i-1];
                      int y1 =plotAreaInset+y[i-1];
                      int x2 =plotAreaInset+x[i];
                      int y2 =plotAreaInset+y[i];
                      g2d.drawLine(x1,y1,x2,y2);
                }
            }
          g2d.dispose(); //get rid of the copy

          } else  {
          for (int i = 1; i < x.length; i++) {
                int x1 = plotAreaInset+x[i-1];
                int y1 = plotAreaInset+y[i-1];
                int x2 = plotAreaInset+x[i];
                int y2 = plotAreaInset+y[i];
                graph.drawLine(x1,y1,x2,y2);
          }
         }
}
```

Η λειτουργία της μεθόδου είναι απλή. Δέχεται σαν εισόδους τις συντεταγμένες (x,y) όλων των σημείων του γραφήματος (σε διαφορετικούς πίνακες τις τετμημένες και τις τεταγμένες) και στη συνέχεια σχεδιάζει μια ευθεία γραμμή μεταξύ κάθε διαδοχικού ζεύγους τιμών (x_i,y_i). Οι άλλες δυο παράμετροι της μεθόδου είναι το αντικείμενο της κλάσης Graphics στο οποίο η μέθοδος επιδρά και ένα string το οποίο καθορίζει αν η γραμμή που θα σχεδιαστεί θα είναι διακεκομμένη (dashed) ή όχι.

Αν η τιμή της παραμέτρου linetype είναι ίση με dash, τότε θα σχεδιαστεί διακεκομμένη γραμμή. Δυστυχώς τα αντικείμενα της κλάσης Graphics δεν έχουν τη δυνατότητα σχεδίασης διακεκομμένης γραμμής από μόνα τους. Δημιουργώντας όμως ένα νέο αντικείμενο της κλάσης Graphics2D και αντιγράφοντας σε αυτό το αντικείμενο graph της κλάσης Graphics μπορούμε να χρησιμοποιήσουμε την κλάση Stroke η οποία επιτρέπει την σχεδίαση γραφικών αντικειμένων με διακεκομμένο περίγραμμα. Και πάλι όμως, αν η απόσταση μεταξύ δυο διαδοχικών σημείων της καμπύλης είναι μικρή, τότε η γραμμή που τα ενώνει δεν εμφανίζεται διακεκομμένη καθώς απαιτείται ένα συγκεκριμένο ελάχιστο μήκος γραμμής για να είναι

εμφανές το διακεκομμένο σχέδιο. Έτσι προστέθηκε ένα τμήμα κώδικα το οποίο υλοποιεί έναν απαριθμητή προς τα κάτω και μια Boolean μεταβλητή ώστε η σχεδίαση των γραμμών να μην γίνεται ανάμεσα σε κάθε διαδοχικό ζεύγος (x_i,y_i) αλλά να προσπερνιόνται τόσα ζεύγη όσα και η αρχική τιμή του απαριθμητή πριν ξανασχεδιαστεί μια γραμμή για άλλα τόσα ζεύγη. Η αρχική τιμή αυτή επιλέχθηκε τελικά να είναι το 1 και να σχεδιάζονται οι γραμμές ανά δεύτερο διαδοχικό ζεύγος (x_i,y_i) καθώς το οπτικό αποτέλεσμα ήταν το πιο ικανοποιητικό.

Μετά τη σχεδίαση της καμπύλης είναι καλό να ελευθερώνονται όσοι πόροι του συστήματος χρησιμοποιούνται από το αντίγραφο g2d του γραφικού αντικειμένου graph.

Η σχεδίαση τώρα των διακριτών σημάτων γίνεται με τη χρήση stems, δηλαδή ευθύγραμμων τμημάτων με ένα κύκλο στο άκρο τους. Η μέθοδος που σχεδιάζει τα διακριτά σήματα είναι η plotStems η οποία όμως έχει γίνει overload και υπάρχουν 2 εκδόσεις της. Στην πρώτη περίπτωση σχεδιάζεται ένα διακριτό σήμα με συγκεκριμένο πλάτος, δηλαδή με όλα τα ευθύγραμμα τμήματα να έχουν το ίσο μήκος. Σαν παράμετροι στη μέθοδο δίνονται οι τιμές των τετμημένων για τα stems και το αντικείμενο της κλάσης Graphics στο οποίο η μέθοδος επιδρά. Το ύψος των ευθύγραμμων τμημάτων υπολογίζεται να είναι τόσο ώστε στο γράφημα, οι κύκλοι στην άκρη τους να εφάπτονται του πάνω τμήματος της περιοχής σχεδίασης.

```
/**
 * Method for plotting a stem plot (spikes with circles at the end)
 *
 * @param   stems (int [])    :     array holding the values of the x-
coordinate for the stems
 * @param   graph (Graphics): the graphics component which will be (re)drawn
 * @return  void        :     result is the drawn stems
 */
void plotStems(int[] stems, Graphics graph) {


      int maxY = this.getHeight()-2*plotAreaInset;
      for (int i = 0; i < stems.length; i++) {
            graph.drawLine(plotAreaInset+stems[i],       plotAreaInset+maxY,
plotAreaInset+stems[i], plotAreaInset+stemCircleRadius);
            graph.drawOval(plotAreaInset+stems[i]-stemCircleRadius/2,
plotAreaInset, stemCircleRadius, stemCircleRadius);
    }
}
```

Μια βελτιωμένη εκδοχή της πιο πάνω μεθόδου περιλαμβάνει μια ακόμα παράμετρο τύπου String με όνομα linetype. Αν η τιμή της μεταβλητής linetype είναι ίση με ¨dash¨, τότε η σχεδίαση του ευθύγραμμου τμήματος όσο και του κύκλου στο άκρο του γίνεται με διακεκομμένη γραμμή. Η λογική είναι ίδια με αυτή της μεθόδου plotGraph.

```
/**
 * Method for plotting a stem plot (spikes with circles at the end)
 *
 * @param   stems (int [])    :     array holding the values of the x-
coordinate for the stems
 * @param   graph (Graphics): the graphics component which will be (re)drawn
 * @param   linetype(String): if this String is set to "dash", a dashed line
 *                                          will be drawn. In any other
case the line will
 *                                          be continuous.
 * @return  void        :     result is the drawn line
 */
void plotStems(int[] stems, Graphics graph, String linetype) {

      int maxY = this.getHeight()-2*plotAreaInset;

        if (linetype == "dash") {
            Graphics2D g2d = (Graphics2D) graph.create(); // copy graph
instance
              Stroke dashed = new BasicStroke(1, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND, 0, new float[]{4, 4}, 0);
              g2d.setStroke(dashed);

              for (int i = 0; i < stems.length; i++) {
                    g2d.drawLine(plotAreaInset+stems[i], plotAreaInset+maxY,
plotAreaInset+stems[i], plotAreaInset+stemCircleRadius);
                    g2d.drawOval(plotAreaInset+stems[i]-stemCircleRadius/2,
plotAreaInset, stemCircleRadius, stemCircleRadius);
              }
              g2d.dispose(); //get rid of the copy
        } else {
```

```
                for (int i = 0; i < stems.length; i++) {
                        graph.drawLine(plotAreaInset+stems[i],
plotAreaInset+maxY, plotAreaInset+stems[i], plotAreaInset+stemCircleRadius);
                        graph.drawOval(plotAreaInset+stems[i]-
stemCircleRadius/2, plotAreaInset, stemCircleRadius, stemCircleRadius);
                }
        }
}
```

Η μέθοδος plotStems, όπως παρουσιάστηκε μπορεί να χρησιμοποιηθεί για τη σχεδίαση του σήματος δειγματοληψίας αλλά και όλων των γραφικών παραστάσεων στο πεδίο των συχνοτήτων. Δεν μπορεί όμως να χρησιμοποιηθεί για την σχεδίαση του δειγματοληπτημένου σήματος καθώς εκεί δεν έχουν όλα τα δείγματα το ίδιο πλάτος. Έτσι επιβάλλεται να υπάρχει μια δεύτερη εκδοχή της μεθόδου η οποία να μπορεί να σχεδιάσει δείγματα μεταβλητού πλάτους. Το πρότυπο της μεθόδου αυτής είναι το :

```
void plotStems(int[][] stemPoints, Graphics graph)
```

Η παράμετρος `int[][] stemPoints` περιλαμβάνει τόσο τις τετμημένες των δειγμάτων (στην πρώτη διάσταση του πίνακα), όσο και το μήκος του κάθε κατακόρυφου τμήματος (στην δεύτερη διάσταση του πίνακα).

```
/**
 * Method for plotting a stem plot (spikes with circles at the end) consisting
of
 * stems of variable height
 *
 * @param   stems (int [][]): array holding the values of the x-coordinate for
 *                            the stems (first dimension) and also the height
 *                            of each stem (second dimension)
 * @param   graph (Graphics): the graphics component which will be (re)drawn
 * @return  void            : result is the drawn line
 */
void plotStems(int[][] stemPoints, Graphics graph) {

      int maxY = this.getHeight()-2*plotAreaInset;
      int halfY = maxY/2;
      for (int i = 0; i < stemPoints[0].length; i++) {
            graph.drawLine(plotAreaInset+stemPoints[0][i],
```

```
plotAreaInset+halfY,                                plotAreaInset+stemPoints[0][i],
plotAreaInset+stemPoints[1][i]+(Integer.signum(halfY-
stemPoints[1][i]))*stemCircleRadius/2);

            graph.drawOval(plotAreaInset+stemPoints[0][i]-stemCircleRadius/2,
plotAreaInset+stemPoints[1][i]-stemCircleRadius/2,          stemCircleRadius,
stemCircleRadius);


    }
}
```

6.6.4. Λοιπές μέθοδοι της κλάσης DrawingPanel

Η κλάση DrawingPanel έχει άλλες δυο custom μεθόδους. Μια για τον υπολογισμό του φάσματος του σήματος δειγματοληψίας και μια για την εξομοίωση της συνέλιξης.

Η πρώτη έχει την εξής δήλωση

```
double[] evaluateDiracCombSpectrum (double fs)
```

Δέχεται σαν είσοδο την συχνότητα δειγματοληψίας και επιστρέφει έναν πίνακα που περιέχει τις τετμημένες των φασματικών γραμμών. Ο πίνακας αυτός έχει μέγεθος ίσο με τη μέγιστη συχνότητα που μπορεί να έχει το σήμα δειγματοληψίας (το 8) διαιρεμένο με τη συχνότητα δειγματοληψίας και πολλαπλασιασμένο επί 2 ώστε να περιληφθούν και οι φασματικές γραμμές με αρνητικό πρόσημο. Για να περιληφθεί όμως και η φασματική γραμμή στη συχνότητα 0 θα πρέπει να προστεθεί και μια ακόμα θέση στον πίνακα:

```
double[] xs = new double[(int) (8/samplingFreq)*2+1];
```

```
/**
 * Method for evaluating the spectrum of a Dirac Comb Series (sampling
function)
 *
 * @param   fs (double) :     the frequency of the Dirac Comb Series
 *                            (sampling frequency)
 * @return  double [] :result is an array containing the x-coordinates of the
 *                      spectrum's stems
 */
double[] evaluateDiracCombSpectrum (double fs) {


      double[] xs = new double[(int) (8/samplingFreq)*2+1];
```

```
    for (int i = 0; i < xs.length/2; i=i+1) {
      xs[i] =-samplingFreq*(i+1);
    }
    for (int i = xs.length/2+1; i < xs.length; i=i+1) {
      xs[i] =samplingFreq*(i-xs.length/2);
    }
    xs[xs.length/2] = 0;
    return xs;

}
```

Τέλος, υπάρχει και η μέθοδος που εξομοιώνει την πράξη της συνέλιξης. Δέχεται σαν εισόδους το φάσμα του σήματος δειγματοληψίας και το φάσμα του σήματος πληροφορίας. Επιστρέφει έναν πίνακα που έχει μήκος ίσο με το γινόμενο των μηκών των άλλων δυο πινάκων.

```
/**
 * Method for emulating the convolution function between 2 arrays
 * stems of variable height
 *
 * @param   d (double[]): array holding the values of the x-coordinate for
 *                        the first array (sampling signal spectrum)
 * @param   f (double[]): array holding the values of the x-coordinate for
 *                        the second array (input signal spectrum)
 * @return  double []   : result is an array containing the x-coordinates of
 *                        the result of the convolution
 */
double[] convolution (double[] d, double[] f) {

      double[] conv = new double[f.length*d.length];

    for (int i = 0; i < d.length; i=i+1) {
      for (int j = 0; j < f.length; j=j+1) {
            conv[f.length*i+j] =d[i]+f[j];
      }
    }
    return conv;
}
```

6.6.5. Η κλάση ContinuousSignal

Τα σήματα συνεχούς χρόνου που χρησιμοποιούνται στην μικροεφαρμογή είναι αντικείμενα της κλάσης ContinuousSignal. Η κλάση αυτή έχει τρεις ιδιωτικές μεταβλητές, όλες τύπου int. Οι δυο είναι για τις συντεταγμένες x και y και η τρίτη, ο παράγοντας κανονικοποίησης.

```
package draw;

public class ContinuousSignal {

	private int xcoord[];
	private int ycoord[];
	private int norm;

	public ContinuousSignal(){

	}

	public ContinuousSignal(double[] periods, int[] size, int norm){
		this.norm = norm;
		xcoord = new int[size[0]];
		for (int i = 0; i < xcoord.length; i++) {
			xcoord[i] = i;
		}
		ycoord = getSignal(periods, size);
	}

	//getters
	public int[] getXcoord() {
		return xcoord;
	}

	public int[] getYcoord() {
		return ycoord;
	}

	public int[][] getCoords() {

		int[][] coords = new int[2][xcoord.length];
```

```
            for (int i = 0; i<xcoord.length; i++) {
                  coords [0][i] = xcoord[i];
                  coords [1][i] = ycoord[i];
            }
            return coords;
      }


/**
 * Method for calculating the y-coordinates of the points of a normalized
 * signal composed form up to three frequencies
 *
 * @param   periods (double []):    array holding the signal's frequencies
 * @param   size (int [])     :     the size of the drawing area
 *                                  (width x heigth) in pixels
 * @return  int[]       :     result is the y-coordinates of the signal
 */
int[] getSignal(double periods[], int[] size) {


      ycoord = new int[size[0]];
      for (int i = 0; i < xcoord.length; i++) {
            ycoord[i] = getNormalizedSine(i, size[1]/2-1, size[0]-1, periods,
norm);
      }
    return ycoord;
}

/**
 * Method for calculating the y-coordinate of one point of a normalized
 * signal composed form up to three frequencies, given it's x-coordinate
 *
 * @param   x (int)     :     the x-coordinate
 * @param   halfY (int) :     half the size of the y axis
 * @param   maxX (int)  :     the size of the x axis
 * @param   periods (double []):    array holding the signal's frequencies
 * @param   norm (int)  :     number of frequencies in the signal. Max = 3.
 * @return  int[]       :     result is the y-coordinate of the point
 */
 int getNormalizedSine(int x, int halfY, int maxX, double periods[], int norm)
{
      double piDouble = 2 * Math.PI;
    double factor1 = (double)periods[0] * piDouble / (double)maxX ;
    double factor2 = (double)periods[1] * piDouble / (double)maxX ;
```

```
    double factor3 = (double)periods[2] * piDouble / (double)maxX ;
    return (int) ((double)halfY - (Math.sin(x * factor1) + Math.sin(x * factor2) + Math.sin(x * factor3) )* (double)halfY/(double)norm);
}

}
```

Η κλάση περιλαμβάνει επίσης και 2 μεθόδους (πέρα από τους getters). Η πρώτη είναι η

```
int[] getSignal(double periods[], int[] size)
```

η οποία υπολογίζει τις τεταγμένες των σημείων που απαρτίζουν την καμπύλη. Η μέθοδος αυτή καλεί την δεύτερη μέθοδο

```
int getNormalizedSine(int x, int halfY, int maxX, double periods[], int norm)
```

η οποία επιστρέφει την τιμή της τεταγμένης ενός σημείου, δεδομένης της τετμημένης του.

6.6.6. Η κλάση StemSignal

Τα διακριτά σήματα που χρησιμοποιούνται στην μικροεφαρμογή είναι αντικείμενα της κλάσης StemSignal. Η κλάση αυτή έχει τρεις ιδιωτικές μεταβλητές, δυο τύπου int και μια τύπου double. Οι δυο μεταβλητές τύπου int περιέχουν τις συντεταγμένες x και y του διακριτού σήματος. Η τρίτη μεταβλητή κρατά και αυτή την τετμημένη x αλλά σε μορφή double. Ο λόγος που γίνεται αυτό είναι ότι αν οι υπολογισμοί γίνουν με την τετμημένη x σαν ακέραια μεταβλητή, υπάρχει μια απώλεια ακρίβειας. Έτσι όλοι οι υπολογισμοί γίνονται με την τετμημένη σαν πραγματικό αριθμό και μετά γίνεται casting σε ακέραια τιμή.

```
package draw;

public class StemSignal {

      private double xcoord[];
      private int xcoordInt[];
      private int ycoord[];

      public StemSignal(){
```

```
}

public StemSignal(int[] points){

      xcoordInt = new int[points.length];
      xcoordInt = points;
      ycoord = new int[points.length];
        for (int i = 0; i < xcoordInt.length; i++) {
              ycoord[i] = 1;
        }
}

public StemSignal(int[] points, int value){

      xcoordInt = new int[points.length];
      xcoordInt = points;
      ycoord = new int[points.length];
        for (int i = 0; i < xcoordInt.length; i++) {
              this.ycoord[i] = value;
        }
}

public StemSignal(int[] points, int[] contour){

      xcoordInt = points;
    ycoord = getStems(points, contour);
}

public StemSignal(int xSize, double step){

      xcoord = new double[(int) (xSize/step)];
      xcoordInt = new int[(int) (xSize/step)];
      ycoord = new int[(int) (xSize/step)];
      xcoord[0] = (step-1);
      xcoordInt[0] = (int) xcoord[0];
      ycoord[0] = 1;
      for (int i=1; i<xcoord.length; i++){
            xcoord[i] = xcoord[i-1]+step;
```

```
                xcoordInt[i] = (int)xcoord[i];
                ycoord[i] = 1;
        }
    }

    public StemSignal(int xSize, double step, int value){

        xcoord = new double[(int) (xSize/step)];
        xcoordInt = new int[(int) (xSize/step)];
        ycoord = new int[(int) (xSize/step)];
        xcoord[0] = (step-1);
        xcoordInt[0] = (int) xcoord[0];
        ycoord[0] = value;
        for (int i=1; i<xcoord.length; i++){
                xcoord[i] = xcoord[i-1]+step;
                xcoordInt[i] = (int)xcoord[i];
                ycoord[i] = value;
        }
    }

    public StemSignal(int xSize, double step, int[] contour){

        xcoord = new double[(int) (xSize/step)];
        xcoordInt = new int[(int) (xSize/step)];
        ycoord = new int[(int) (xSize/step)];
        xcoord[0] = (step-1);
        xcoordInt[0] = (int) xcoord[0];
        for (int i=1; i<xcoord.length; i++){
                xcoord[i] = xcoord[i-1]+step;
                xcoordInt[i] = (int)xcoord[i];
        }
        ycoord = getStems(xcoordInt, contour);
    }

    // getters
    public int[] getXcoord() {
        return xcoordInt;
    }
```

```
	public int[] getYcoord() {
		return ycoord;
	}

	public int[][] getCoords() {

		int[][] stemPlotPoints = new int[2][xcoord.length];
		for (int i = 0; i<xcoord.length; i++) {
			stemPlotPoints[0][i] = xcoordInt[i];
			stemPlotPoints[1][i] = ycoord[i];
		}
		return stemPlotPoints;
	}

/**
 * Method for getting the height of the stems (y-coords) of a stem plot
 * that is enveloped by a continuous signal
 *
 * @param   points (int[])     :     the x-coordinates
 * @param   contour (int[])    :     the y-coordinates of the continuous
signal
 * @return  int[]        :     result is the y-coordinates of the stem
plot
 */
private int[] getStems(int[] points, int[] contour) {

	int[] stems = new int[points.length];
    for (int i = 0; i < stems.length; i++) {
       stems[i] = contour[points[i]];
	}
    return stems ;
}

}
```

Βλέποντας κανείς τον πιο πάνω κώδικα της κλάσης, παρατηρεί ότι υπάρχουν αρκετοί constructors για τα αντικείμενα της. Ο λόγος είναι ότι στους constructors έχει ενσωματωθεί η λογική του υπολογισμού των τεταγμένων y του διακριτού σήματος ανάλογα με τις παραμέτρους που έχει ο κάθε constructor.

Έτσι, πέρα από τον κενό constructor έχουμε και τους εξής:

- `StemSignal(int[] points)`: Απλός constructor όπου δίνονται οι τετμημένες των διακριτών σημείων ενώ οι τεταγμένες τίθενται στην τιμή 1.
- `StemSignal(int[] points, int value)`: όπως ο προηγούμενος, απλώς οι τεταγμένες των σημείων τίθενται στην ακέραια τιμή `value`.
- `StemSignal(int[] points, int[] contour)`: Σε αυτόν τον constructor δίνονται οι τετμημένες των σημείων και οι τεταγμένες μιας συνεχούς συνάρτησης ώστε να υπολογιστούν οι τεταγμένες του διακριτού σήματος με τη βοήθεια της μεθόδου `getStems(points, contour)`.
- `StemSignal(int xSize, double step)`: Σε αυτόν τον constructor δίνονται το πλάτος του γραφήματος (μήκος άξονα x) και το βήμα ανά πόσα σημεία θα λαμβάνεται νέα διακριτή τιμή (η περίοδος δειγματοληψίας δηλαδή αλλά σε pixels). Επειδή απαιτούνται πράξεις, οι υπολογισμοί γίνονται αρχικά με την τετμημένη σαν πραγματικό αριθμό και μετά γίνεται casting σε ακέραια τιμή. Οι τεταγμένες των σημείων τίθενται στην τιμή 1.
- `StemSignal(int xSize, double step, int value)`: όπως ο προηγούμενος, απλώς οι τεταγμένες των σημείων τίθενται στην ακέραια τιμή `value`.
- `public StemSignal(int xSize, double step, int[] contour)`: όπως οι δυο προηγούμενοι constructors με τη διαφορά ότι οι τεταγμένες προσδιορίζονται από μια συνεχόμενη καμπύλη της οποίας οι τεταγμένες βρίσκονται στον πίνακα contour.

Στην κλάση StemSignal επίσης υπάρχει και μια μέθοδος (πέρα από τους getters) η οποία δέχεται σαν είσοδο τις τετμημένες των δειγμάτων και τις τεταγμένες ενός συνεχούς σήματος και επιστρέφει τις τεταγμένες των δειγμάτων οι οποίες ακολουθούν το πλάτος του συνεχούς σήματος.

```
private int[] getStems(int[] points, int[] contour)
```

Η συνάρτηση αυτή είναι ιδιωτική και χρησιμοποιείται μόνο από τους constructors της κλάσης.

6.7. Οι άξονες και τα πλαίσια

Όσα έχουν υλοποιηθεί μέχρι αυτό το σημείο αποτελούν το ποιο ουσιώδες τμήμα της μικροεφαρμογής η οποία μπορεί να τρέξει ακόμα και αν δεν υλοποιηθούν οι άξονες, αρκεί να διαγραφούν από τον μέχρι τώρα κώδικα οι συγκεκριμένες αναφορές. Η ύπαρξη όμως των αξόνων βοηθούν όχι μόνο στην εμφάνιση της μικροεφαρμογής αλλά και την καλύτερη κατανόηση από το χρήστη. Στην παράγραφο αυτή θα παρουσιαστεί ο τρόπος με τον οποίο υλοποιούνται τα πλαίσια γύρω από τις γραφικές παραστάσεις και οι άξονες. Η κλάση GlassPanel είναι αυτή στα αντικείμενα της οποίας σχεδιάζονται τα πλαίσια και οι άξονες. Οι κλάσεις MyBorder και Axes υλοποιούν αυτά τα δυο στοιχεία.

6.7.1. Η κλάση GlassPanel

Κατά την υλοποίηση του applet παρουσιάστηκε ένα πρόβλημα που αφορούσε τους άξονες των γραφημάτων. Το πρόβλημα ήταν ότι αν οι άξονες σχεδιάζονταν στο ίδιο JPanel με τα γραφήματα, θα έπρεπε να ξανασχεδιάζονται κάθε φορά μαζί με το γράφημα παρόλο που δεν θα άλλαζαν, πράγμα που θα οδηγούσε σε μεγαλύτερη κατανάλωση υπολογιστικών πόρων. Το ίδιο συμβαίνει και με τα πλαίσια γύρω από τα γραφήματα. Ο διαρκής επανασχεδιασμός τους ενώ δεν είναι απαραίτητο μπορεί να οδηγούσε σε τρεμόπαιγμα όταν εκτελούνταν η μικροεφαρμογή. Τότε προέκυψε η ιδέα να υπάρξει ένα δεύτερο JPanel πάνω από τα γραφήματα, στο οποίο θα σχεδιάζονταν μόνο μια φορά τόσο τα πλαίσια όσο και οι άξονες και το οποίο θα ήταν διαφανές ώστε να φαίνεται το γράφημα από το υποκείμενο JPanel. Η δημιουργία μιας τέτοιας κλάσης κρίθηκε πιο απλή διαδικασία από το να χρησιμοποιηθεί το glass pane του applet (δες παράγραφο 6.4.1). Ο κώδικάς της ακολουθεί:

```
package draw;

import javax.swing.JPanel;
import java.awt.Graphics;
import draw.MyBorder;
import draw.Axes;

public class GlassPanel extends JPanel {
```

```
      private int plotAreaInset = 10;
      boolean symmetricalAxes = false;
      int xMax = 1;

      public GlassPanel() {

            this.setOpaque(false);
            this.setVisible( true );
      }

// setters
      public void setPlotAreaInset(int inset) {
            this.plotAreaInset = inset;
      }

      public void setAxesSymmetry(boolean symmetry) {
            this.symmetricalAxes = symmetry;
      }

      public void setXAxesMaxValue(int xMax) {
            this.xMax = xMax;
      }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);

   MyBorder border= new MyBorder(plotAreaInset, getRootPane().getBackground());
        border.addBorder(this, g);
        int[] size = new int[2];
        size[0] = this.getWidth();
        size[1] = this.getHeight();
        Axes axes = new Axes(size, symmetricalAxes);
        axes.setXMaxValue(xMax);
        axes.addHorizontalAxes(this, g);
        //axes.addVerticalAxes(this, g);
          }
}
```

Ο κώδικας είναι πολύ απλός. Μετά την εισαγωγή των απαραίτητων κλάσεων ακολουθεί η υλοποίηση της κλάσης GlassPanel. Η μεταβλητή `plotAreaInset` είναι το περιθώριο μεταξύ της περιοχής του γραφήματος και του ορίου του GlassPanel. Η Boolean μεταβλητή `symmetricalAxes` καθορίζει αν οι άξονες θα είναι συμμετρικοί με το μηδέν να βρίσκεται στο μέσο του άξονα, ή όχι, με το μηδέν να βρίσκεται στο αριστερότερο σημείο του άξονα x. Τέλος η ακέραια μεταβλητή `xMax` είναι η μέγιστη τιμή του άξονα x.

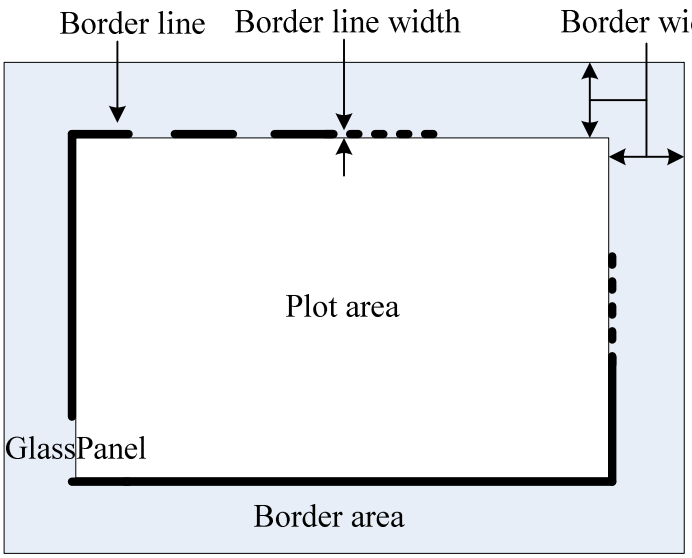
Ο constructor της κλάσης ορίζει ότι τα αντικείμενά της θα είναι διάφανή (`this.setOpaque(false)`) και ορατά (`this.setVisible(true)`).

Η κλάση έχει επίσης τους απαραίτητους setters ενώ, όπως και στην κλάση DrawwingPanel, η σχεδίαση γίνεται κάνοντας override τη μέθοδο `paintComponent(Graphics g)`.

Στη μέθοδο αυτή ορίζεται αρχικά ένα νέο αντικείμενο της κλάσης MyBorder με περιθώριο ίσο με `plotAreaInset` και χρώμα υπόβαθρου, το χρώμα του root pane. Το αντικείμενο αυτό μετά προστίθεται στο αντικείμενο της κλάσης GlassPanel. Κατόπιν ορίζεται και ένα αντικείμενο της κλάσης Axes για συγκεκριμένο μέγεθος και με συγκεκριμένη συμμετρία. Για το αντικείμενο αυτό τίθεται η μέγιστη τιμή του άξονα x ο οποίος προστίθεται και αυτός στο αντικείμενο της κλάσης GlassPanel.

6.7.2. Η κλάση MyBorder

Τα αντικείμενα της κλάσης MyBorder έχουν τα χαρακτηριστικά που θεωρήθηκε ότι πρέπει να έχει το περιθώριο γύρω από τη περιοχή της σχεδίασης.



Εικόνα 6.11 Τα χαρακτηριστικά του περιθωρίου

Έτσι η κλάση αυτή έχει τέσσερις ιδιωτικές μεταβλητές οι οποίες φαίνονται στον πίνακα 6.4.

Πίνακας 6.6 Οι μεταβλητές της κλάσης MyBorder

Όνομα μεταβλητής	Τύπος	Σκοπός
borderWidth	int	Το εύρος του περιθωρίου γύρω από την περιοχή σχεδίασης. Μετράται σε pixel.
borderColor	Color	Το χρώμα που θα χρησιμοποιηθεί στο περιθώριο. Η προεπιλογή είναι ανοικτό γκρι.
borderLine	boolean	Όταν η μεταβλητή αυτή έχει τιμή true, τότε μέσα στο περιθώριο σχεδιάζεται το περίγραμμα της περιοχής σχεδίασης
borderLineWidth	int	Το πλάτος του περιγράμματος σε pixels..

```
package draw;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.geom.Area;
import java.awt.geom.Rectangle2D;
```

```
public class MyBorder {

	private int borderWidth;
	private Color borderColor;
	private boolean borderLine;
	private int borderLineWidth;

	public MyBorder() {

		borderWidth = 15;
		borderColor = Color.lightGray;
		borderLine = true;
		borderLineWidth = 1;
	}

	public MyBorder(int width, Color color){

		this.borderWidth = width;
		this.borderColor = color;
		borderLine = true;
		borderLineWidth = 1;
	}

	public MyBorder(int width){

		this.borderWidth = width;
		this.borderColor = Color.lightGray;
		borderLine = true;
		borderLineWidth = 1;
	}

	public MyBorder(int width, Color color, boolean line, int lineWidth){

		this.borderWidth = width;
		this.borderColor = color;
		this.borderLine = line;
		this.borderLineWidth = lineWidth;
```

```
	}


/**
 * Method for adding a border to a GlassPanel object
 *
 * @param   panel (GlassPanel): GlassPanel to which the border will be added
 * @param   graph (Graphics)  : Graphics object for drawing the border
 * @return  void        :     result is the drawn border
 */
public void addBorder(GlassPanel panel, Graphics graph)
	{
		int height = panel.getHeight();
		int width = panel.getWidth();


		// create copy of Graphics object
		Graphics g2c = graph.create();
		// evaluate border area
		Rectangle2D rectangleNotToDrawIn = new Rectangle2D.Double(borderWidth, borderWidth, width-2*borderWidth-1, height-2*borderWidth-1);
		Area outside = calculateRectOutside(width, height, rectangleNotToDrawIn);
		//fill border area with selected color
		g2c.setClip(outside);
		g2c.setColor(borderColor);
		g2c.fillRect(0, 0, width, height);
		//draw a border line inside the border area
		g2c.setColor(Color.black);
		g2c.drawRect(borderWidth-borderLineWidth, borderWidth-borderLineWidth, width-2*borderWidth, height-2*borderWidth);
		// get rid of the copy
		g2c.dispose();
	}


/**
 * Method for getting the area outside a rectangular
 *
 * @param   width (int)       :     width of panel (GlassPanel)
 * @param   height (int)      :     height of panel (GlassPanel)
 * @param   r (Rectangle2D)   :     Rectangular not to draw in. It will be on
 *                                  top of plotted graph
 * @return  Area        :     area outside inner rectangular and inside panel's
 *                            borders
```

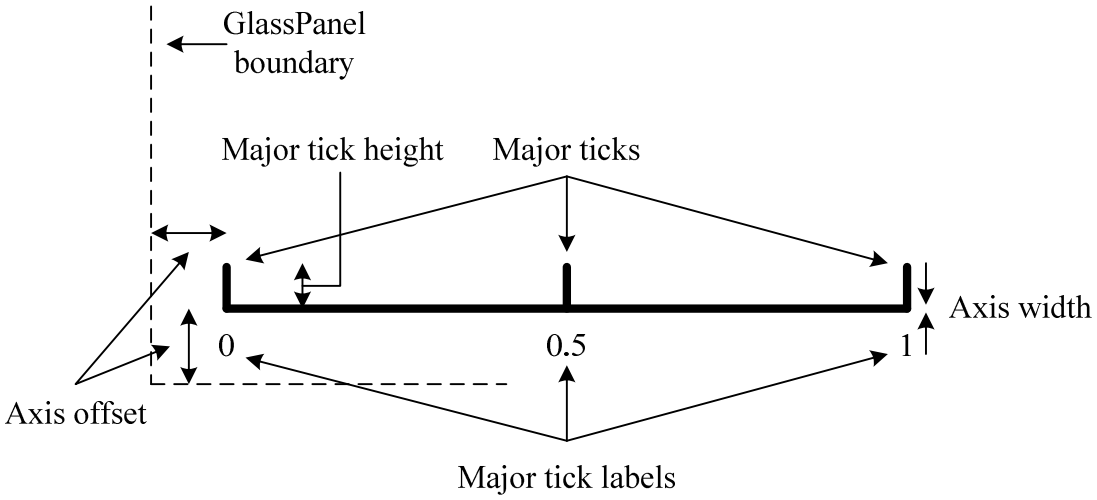
```
 */
      private Area calculateRectOutside(int width, int height, Rectangle2D r)
      {
            Area outside = new Area(new Rectangle2D.Double(0, 0, width,
height));
            outside.subtract(new Area(r));
            return outside;
  }

}
```

Για να είναι δυνατόν να σχεδιαστεί το περιθώριο γύρω από την περιοχή σχεδίασης θα πρέπει να μπορεί να επιλεγεί η περιοχή έξω από ένα παραλληλόγραμμο. Η μέθοδος `Area calculateRectOutside(int width, int height, Rectangle2D r)` κάνει ακριβώς αυτή τη δουλειά. Δέχεται σαν παραμέτρους το πλάτος και το ύψος της συνολικής περιοχής (τα οποία θα είναι ίσα με το πλάτος και το ύψος του αντικειμένου της κλάσης GlassPanel) και την περιοχή r η οποία θα πρέπει να είναι μικρότερη από το μέγεθος του JPanel. Η μέθοδος ορίζει μια περιοχή με μέγεθος ίσο με της συνολικής περιοχής και στη συνέχεια αφαιρεί από αυτή μια περιοχή ίδια με την r. Το αποτέλεσμα είναι το περιθώριο που φαίνεται και στην εικόνα 6.11.

Η μέθοδος `addBorder(GlassPanel panel, Graphics graph)` από την άλλη χρησιμοποιείται για να προστίθεται το περιθώριο που έχει σχεδιαστεί σε ένα αντικείμενο της κλάσης GlassPanel. Η μέθοδος αυτή δημιουργεί αρχικά ένα αντίγραφο του αντικειμένου σχεδίασης Graphics. Κατόπιν ορίζει ένα ορθογώνιο έξω από το οποίο θα σχεδιαστεί το περιθώριο και με τη μέθοδο `Area outside = calculateRectOutside(width, height, rectangleNotToDrawIn);` υπολογίζει την περιοχή του περιθωρίου. Με τη μέθοδο `setClip(outside)` ορίζεται ότι μόνο στην περιοχή του περιθωρίου θα γίνεται σχεδίαση. Το επόμενο βήμα είναι να επιλεγεί το χρώμα και να γεμίσει με αυτό όλο το ορθογώνιο. Λόγω όμως της `setClip(outside)` τελικά γεμίζει με το χρώμα μόνο το περιθώριο. Κατόπιν σχεδιάζεται το εσωτερικό περίγραμμα του περιθωρίου και τέλος ελευθερώνονται όσοι πόροι του συστήματος είχαν δεσμευτεί από το αντίγραφο του αντικειμένου graph.

6.7.3. Η κλάση Axes

Τα αντικείμενα της κλάσης Axes είναι οι άξονες οι οποίοι σχεδιάζονται πάνω στα αντικείμενα της κλάσης GlassPanel. Η αρχική προσέγγιση είχε τόσο οριζόντιους όσο και κατακόρυφους άξονες. Η χρησιμότητα των κατακόρυφων αξόνων ελέγχεται καθώς όλα τα σήματα είναι κανονικοποιημένα και έχουν πλάτος 1. Εξάλλου η κατανόηση του θεωρήματος του Nyquist δεν απαιτεί την αναγραφή του πλάτους του κάθε σήματος. Αν δεν χρησιμοποιηθούν οι κατακόρυφοι άξονες τότε μπορεί να περιοριστεί και το εύρος του περιθωρίου γύρω από κάθε περιοχή σχεδίασης. Έτσι εδώ δεν θα παρουσιαστεί η υλοποίηση των κατακόρυφων αξόνων αν και δεν διαφέρει σημαντικά από την υλοποίηση των οριζόντιων αξόνων. Τα κυριότερα χαρακτηριστικά του κάθε αντικειμένου της κλάσης Axes φαίνονται στο επόμενο σχήμα.



Εικόνα 6.12 Τα χαρακτηριστικά του οριζόντιου άξονα

Έτσι η κλάση αυτή έχει τις εξής ιδιωτικές μεταβλητές

Πίνακας 6.5 Οι μεταβλητές της κλάσης Axes

Όνομα μεταβλητής	Τύπος	Σκοπός
axisWidth	int	Το πλάτος της γραμμής του άξονα. Μετράται σε pixel.
axisColor	Color	Το χρώμα που θα χρησιμοποιηθεί για τον άξονα. Η

		προεπιλογή είναι μπλε.
horizontalMajorTicks	int[]	Οι τετμημένες των σημείων όπου θα σχεδιαστούν τα ticks.
majorTickHeight	int	Το ύψος των ticks σε pixels.
axisOffset	int	Πόσο απέχει ο άξονας από τα όρια του αντικειμένου του GlassPanel
xMaxValue	int	Η μέγιστη τιμή του οριζόντιου άξονα
symmetricalXAxis	boolean	Όταν η τιμή της μεταβλητής είναι true, ο οριζόντιος άξονας έχει το 0 στο κέντρο αλλιώς το 0 θα είναι στην αριστερή άκρη

```
package draw;

import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;

public class Axes {

	private int axisWidth;
	private Color axisColor;
	private int [] horizontalMajorTicks;
	private int majorTickHeight;
	private int axisOffset;
	private int xMaxValue = 1;
	private boolean symmetricalXAxis;

	public Axes() {
		axisWidth = 1;
		axisColor = Color.blue;
		horizontalMajorTicks = getHorizontalMajorTicks(3, 200);
		majorTickHeight = 5;
```

```
            axisOffset = 10;
            symmetricalXAxis = false;
      }

      public Axes(int[] size) {
            axisWidth = 1;
            axisColor = Color.blue;
            horizontalMajorTicks = getHorizontalMajorTicks(3, size[0]);
            majorTickHeight = 5;
            axisOffset = 10;
            symmetricalXAxis = false;
      }

      public Axes(int[] size, boolean symmetricalX) {
            axisWidth = 1;
            axisColor = Color.blue;
            horizontalMajorTicks = getHorizontalMajorTicks(3, size[0]);
            majorTickHeight = 5;
            axisOffset = 10;
            symmetricalXAxis = symmetricalX;
      }

      public Axes(int[] size, int[] numOfTicks) {
            axisWidth = 1;
            axisColor = Color.blue;
            horizontalMajorTicks     =     getHorizontalMajorTicks(numOfTicks[0],
size[0]);
            majorTickHeight = 5;
            axisOffset = 10;
            symmetricalXAxis = false;
      }

      public Axes(int[] size, int[] numOfTicks, int offset) {
            axisWidth = 1;
            axisColor = Color.blue;
            horizontalMajorTicks     =     getHorizontalMajorTicks(numOfTicks[0],
size[0]);
            majorTickHeight = 5;
            axisOffset = offset;
```

```
            symmetricalXAxis = false;
      }

      //setters
      public void setXMaxValue(int x) {
            xMaxValue = x;
      }

      public void setSymmetricalXAxis(boolean symmetry) {
            symmetricalXAxis = symmetry;
      }

      public void setSymmetricalYAxis(boolean symmetry) {
            symmetricalYAxis = symmetry;
      }

/**
 * Method for getting the coordinates for the ticks
 *
 * @param   numOfTicks (int): number of ticks for horizontal axis
 * @param   length (int)      :     length of x-axis
 * @return  int[]             :     int array containing coordinates of ticks
 */
private int[] getHorizontalMajorTicks(int numOfTicks, int length){

      int[] ticks = new int[numOfTicks];
      ticks[0] = -1;                        // axis is inside border area
      for (int i = 1; i<numOfTicks; i++){
            ticks[i] = i*(length-1)/(numOfTicks-1);
      }
      return ticks;
}

/**
 * Method for setting the labels of the horizontal axis
 *
 * @param   maxValue (int)          :     x-axis max value
 * @param   numOfTicks (int)  :     number of ticks for horizontal axis
```

```
 * @param   symmetrical (boolean):  true if axis is to be draw from -maxValue
 *                                  to maxValue, false if axis is to be draw
 *                                  from 0 to maxValue
 * @return  String[]                :     string array containing labels
 */
private String[] setMajorTickLabelsHor(int maxValue, int numOfTicks, boolean
symmetrical){

      double tmp;
      String[] tickLabels = new String[numOfTicks];

      if (symmetrical) {
            for (int i = 0; i<numOfTicks/2; i++) {
                  tmp = -maxValue + i*maxValue/(double)((double)numOfTicks/2.0-
1.0);
                  if (tmp == Math.rint(tmp)){
                        tickLabels[i] =Integer.toString((int)tmp);
                  } else {
                        tickLabels[i] =Double.toString(tmp);
                  }
            }
            for (int i = (int)((double)numOfTicks/2)+1; i<numOfTicks; i++) {
                  tmp                                 =                                 (i-
(double)numOfTicks/2)*maxValue/(double)((double)numOfTicks/2-1);
                  if (tmp == Math.rint(tmp)){
                        tickLabels[i] =Integer.toString((int)tmp);
                  } else {
                        tickLabels[i] =Double.toString(tmp);
                  }
            }
            tickLabels[numOfTicks/2] ="0";
      } else {
            for (int i = 0; i<numOfTicks; i++) {
                  tmp = i*maxValue/(double)(numOfTicks-1);
                  if (tmp == Math.rint(tmp)){
                        tickLabels[i] =Integer.toString((int)tmp);
                  } else {
                        tickLabels[i] =Double.toString(tmp);
                  }
```

```
            }
        }
        return tickLabels;
}
/**
 * Method for adding a horizontal axis to a GlassPanel object
 *
 * @param panel (GlassPanel): GlassPanel object to which the axis will be added
 * @param   graph (Graphics)  :     Graphics object for drawing the axis
 * @return  void        :     result is the drawn axis
 */
public void addHorizontalAxes(GlassPanel panel, Graphics graph)
{
    int height = panel.getHeight()-2*axisOffset;
    int width = panel.getWidth()-2*axisOffset;


    horizontalMajorTicks = getHorizontalMajorTicks(horizontalMajorTicks.length,
width);
    String[] majorTicksLabels = new String[horizontalMajorTicks.length];
    majorTicksLabels              =              setMajorTickLabelsHor(xMaxValue,
horizontalMajorTicks.length, symmetricalXAxis);


    for (int i = 0; i < horizontalMajorTicks.length; i++) {
      graph.drawLine(axisOffset + horizontalMajorTicks[i], height+axisOffset-1,
axisOffset + horizontalMajorTicks[i], height+axisOffset-2-majorTickHeight);
      Font font = new Font("TimesRoman", Font.PLAIN, 8);
      FontMetrics metrics = graph.getFontMetrics(font);
      int fontHeight = metrics.getHeight();
      graph.setColor(Color.black);
      if (fontHeight > axisOffset) {
            graph.drawString(majorTicksLabels[i],                  axisOffset-1+
horizontalMajorTicks[i]     -     (metrics.stringWidth(majorTicksLabels[i])+0)/2,
height + axisOffset + 12);
      } else {
            graph.drawString(majorTicksLabels[i],                  axisOffset-1+
horizontalMajorTicks[i]     -     (metrics.stringWidth(majorTicksLabels[i])+0)/2,
height + axisOffset + fontHeight);
            }
    }
}
}
```

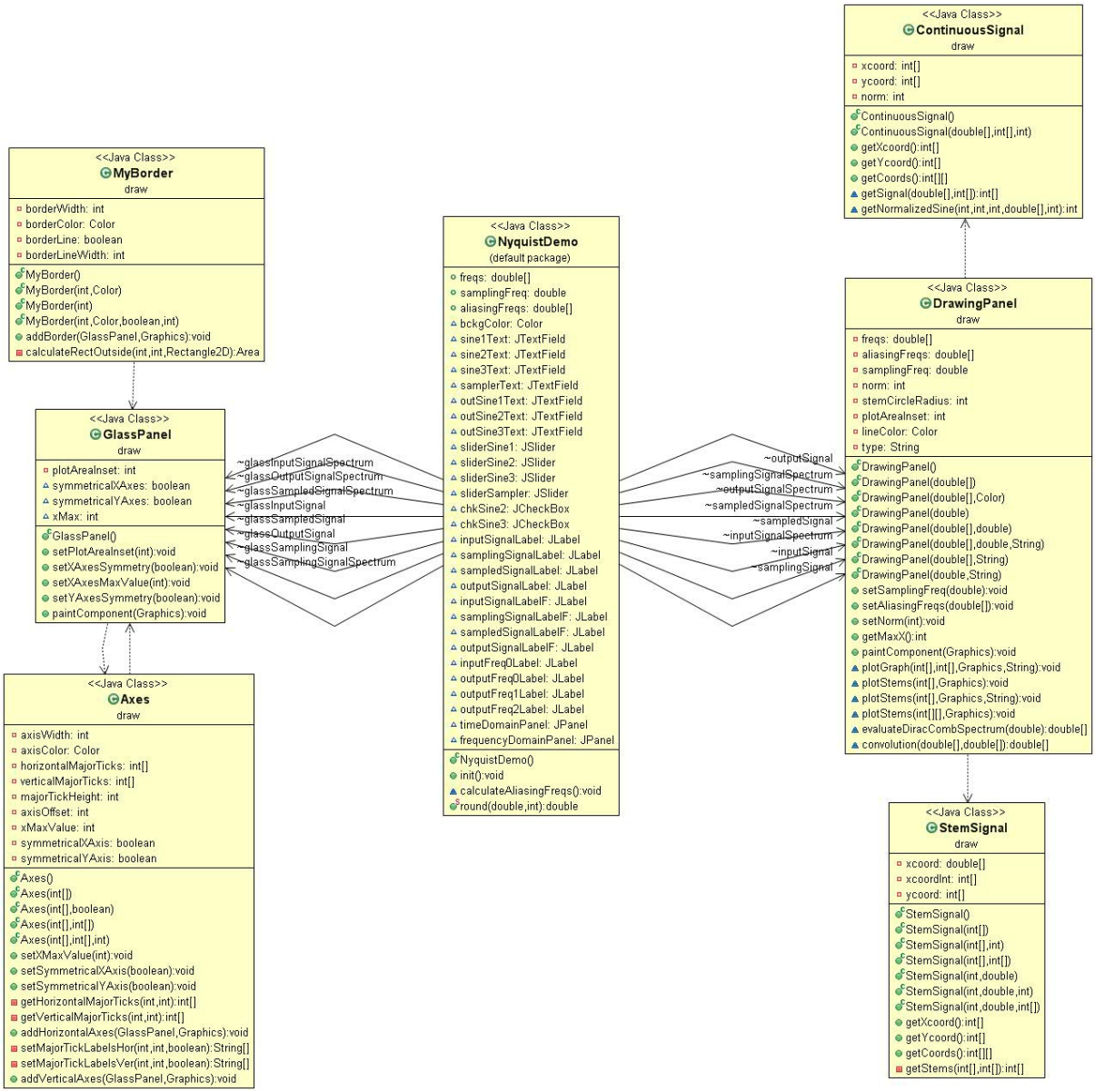
Η μέθοδος `getHorizontalMajorTicks(int numOfTicks, int length)` χρησιμοποιείται για τον υπολογισμό των τετμημένων όπου θα σχεδιαστούν τα ticks, οι κάθετες δηλαδή γραμμές κατά μήκους του άξονα. Σημαντικό σημείο που πρέπει να προσεχθεί είναι ότι ο άξονας σχεδιάζεται στο περιθώριο του γραφήματος και άρα το πρώτο σημείο δεν βρίσκεται στη θέση 0 αλλά στο -1.

Πέρα όμως από τα ticks θα πρέπει να σχεδιαστούν (κυριολεκτικά) και τα labels για κάθε ένα tick. Τα labels υπολογίζονται με τη μέθοδο `setMajorTickLabelsHor(int maxValue, int numOfTicks, boolean symmetrical)`. Η μέθοδος αυτή διακρίνει 2 περιπτώσεις. Αν ο άξονας είναι συμμετρικός τότε στα άκρα θα πρέπει να γραφούν οι τιμές – `xMaxValue` και `xMaxValue`, ειδάλλως οι τιμές θα είναι 0 και `xMaxValue`. Επειδή οι τιμές που υπολογίζονται για να χρησιμοποιηθούν σαν labels, είναι τύπου double συμβαίνει το εξής, αν η τιμή είναι 0.5 φυσικά και πρέπει να γραφεί 0.5, αν όμως είναι 1.0 τότε είναι καλύτερο να γραφεί σαν 1. Για να εντοπιστούν ποιες τιμές είναι ακέραιοι αριθμοί και ποιες δεκαδικοί χρησιμοποιείται η συνάρτηση rint. Η rint(d) δέχεται σαν όρισμα ένα αριθμό double d και επιστρέφει τον double αριθμό ο οποίος είναι ο μαθηματικός ακέραιος ο πιο κοντινός στον d. Για παράδειγμα η rint(1.9) επιστρέφει 2.0 και η rint(3.0) επιστρέφει 3.0. Αυτή την ιδιότητα της rint χρησιμοποιεί η συνάρτηση `setMajorTickLabelsHor` για ελέγχει αν η τιμή ενός label είναι δεκαδικός αριθμός και οπότε το μετατρέπει κατευθείαν σε String ή είναι μαθηματικός ακέραιος και οπότε το κάνει πρώτα casting σε int και μετά το μετατρέπει σε String.

Αφού υπολογιστούν τα ticks και τα labels, μένει να προστεθεί ο άξονας στο αντικείμενο της κλάσης GlassPanel. Αυτό το αναλαμβάνει η συνάρτηση `addHorizontalAxes(GlassPanel panel, Graphics graph)` η οποία ουσιαστικά σχεδιάζει τον άξονα, τα ticks και τα labels. Εδώ το σημαντικό είναι ότι θα πρέπει τα labels να χωράνε μέσα στο περιθώριο που υπάρχει γύρω από την περιοχή του γραφήματος. Για το λόγο αυτό έχει επιλεγεί μέγεθος γραμματοσειράς το 8.

6.8 Το διάγραμμα κλάσεων

Στην επόμενη εικόνα φαίνεται το διάγραμμα κλάσεων της μικροεφαρμογής. Το διάγραμμα παράχθηκε από το ίδιο το Eclipse με τη χρήση του πρόσθετου ObjectAid®.



Εικόνα 6.13 Διάγραμμα κλάσεων

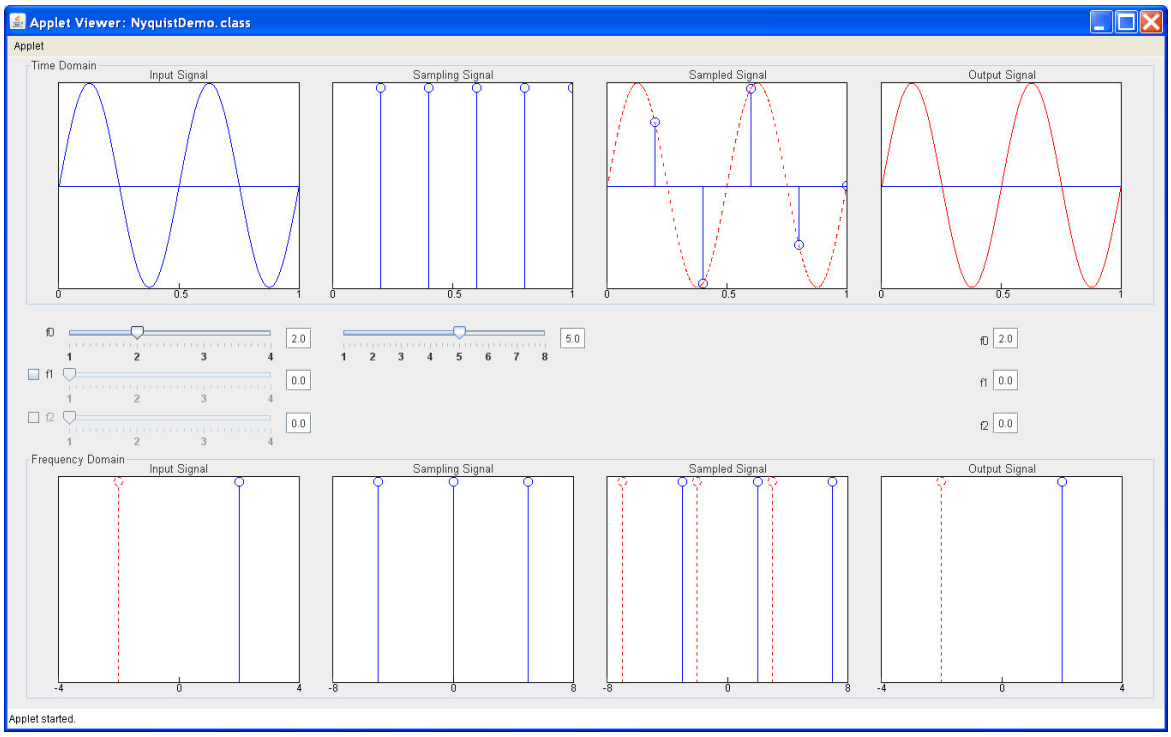
Στο κέντρο φαίνεται η βασική κλάση, η NyquistDemo. Σε αυτή υπάρχουν 8 στιγμιότυπα της κλάσης DrawingPanel (δεξιά) και 8 στιγμιότυπα της κλάσης GlassPanel (αριστερά). Η DrawingPanel έχει εξαρτήσεις με τις κλάσεις ContinuousSignal και StemSignal καθώς σε αυτή σχεδιάζονται ένα ή περισσότερα αντικείμενα από αυτές τις κλάσεις. Από την άλλη πλευρά, η κλάση GlassPanel σχετίζεται μόνο με τις κλάσεις Axes και MyBorder, καθώς σε κάθε GlassPanel σχεδιάζεται μόνο το περιθώριο και οι άξονες.

6.9 Η λειτουργία της μικροεφαρμογής

Στις προηγούμενες παραγράφους περιγράφηκε με αρκετή λεπτομέρεια ο τρόπος με τον οποίο υλοποιήθηκε η μικροεφαρμογή. Αναφέρθηκαν και αναλύθηκαν όλες οι κλάσεις που υλοποιήθηκαν και οι πιο σημαντικές μέθοδοι που αυτές περιέχου. Το αποτέλεσμα είναι αυτό που φαίνεται και στην εικόνα 6.5. Στην παράγραφο αυτή θα γίνει επίδειξη της λειτουργίας της μικροεφαρμογής μέσα από screen shots τα οποία θα δείχνουν τις σημαντικότερες λειτουργίες της μικροεφαρμογής.

6.7.4. Δειγματοληψία ημιτονικού σήματος συχνότητας f_0 με συχνότητα δειγματοληψίας $f_s>2\cdot f_0$

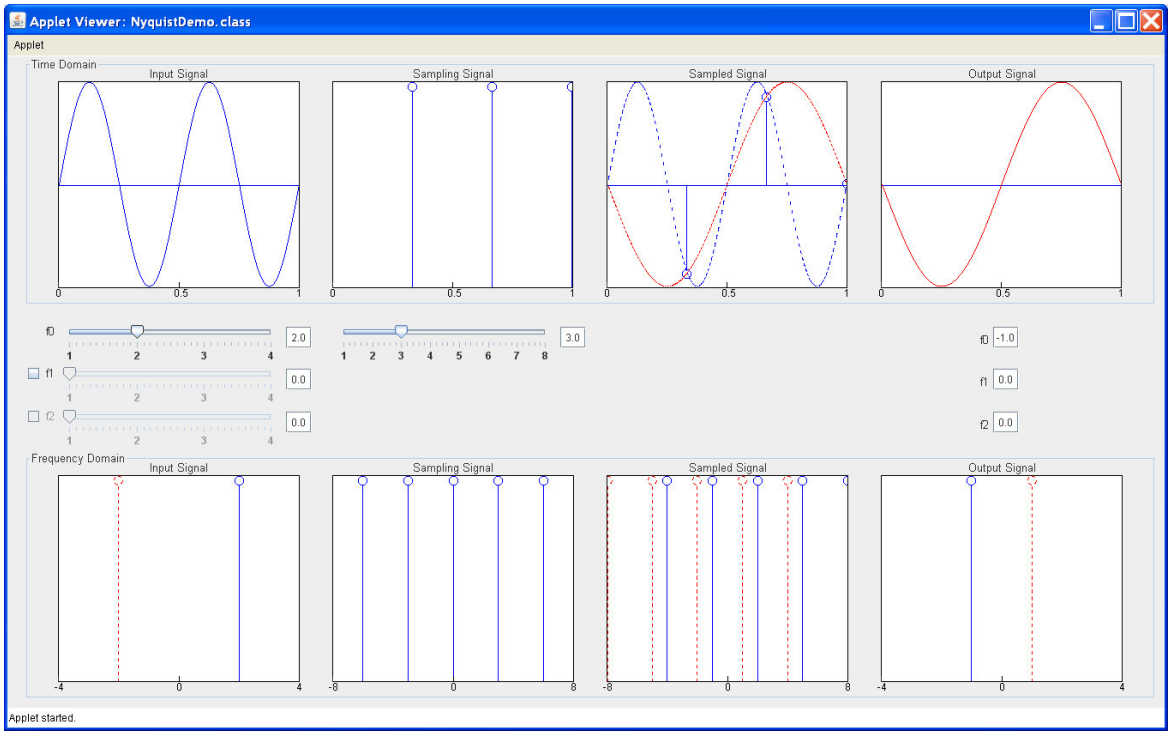
Θα ξεκινήσουμε από την απλή περίπτωση σωστής δειγματοληψίας ενός ημιτονικού σήματος. Στην επόμενη εικόνα φαίνεται καθαρά ότι το σήμα εισόδου έχει συχνότητα $f_0 = 2Hz$ και η συχνότητα δειγματοληψίας είναι $f_s=5Hz$. Η συνθήκη $f_s>2\cdot f_0$ του θεωρήματος του Nyquist ικανοποιείται και το αποτέλεσμα είναι το σήμα εξόδου να είναι ίδιο με το σήμα εισόδου.



Εικόνα 6.14 Δειγματοληψία ημιτονικού σήματος συχνότητας f_0 με συχνότητα δειγματοληψίας $f_s>2\cdot f_0$

6.7.5. Δειγματοληψία ημιτονικού σήματος συχνότητας f_0 με συχνότητα δειγματοληψίας $f_s<2\cdot f_0$

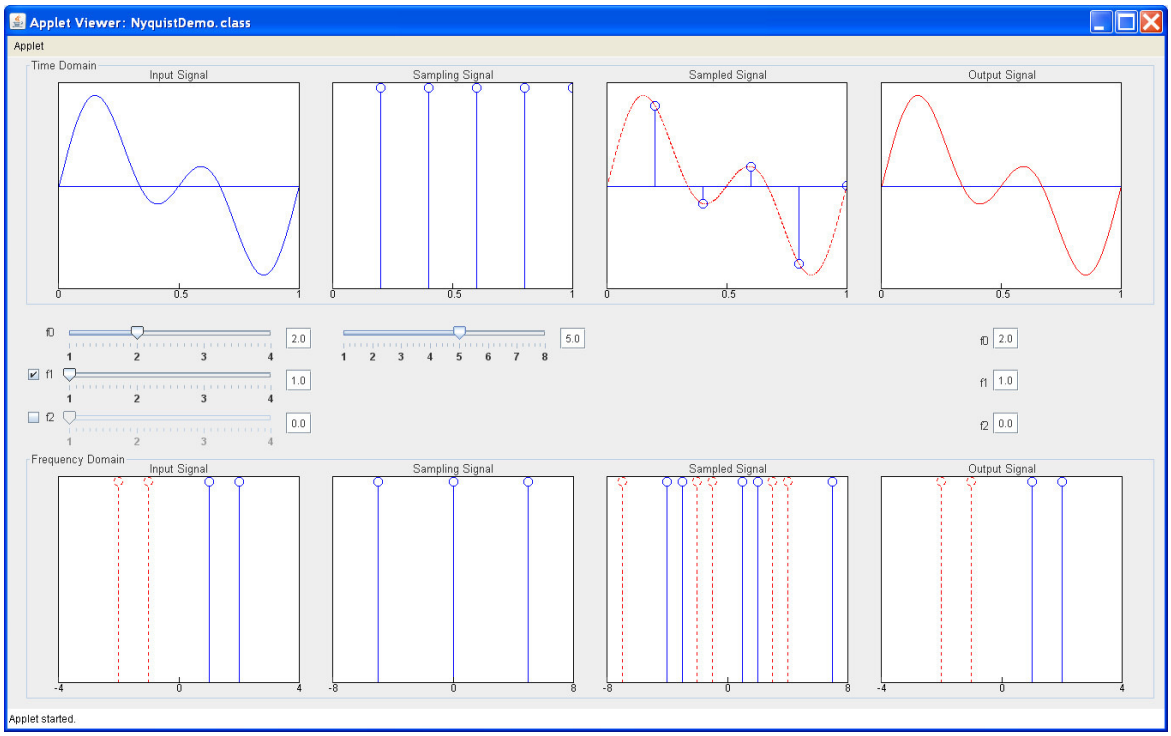
Η δεύτερη περίπτωση που θα παρουσιαστεί είναι αυτή της υποδειγματοληψίας ενός απλού ημιτονικού σήματος. Στην επόμενη εικόνα ένα σήμα εισόδου έχει συχνότητα $f_0 = 2Hz$ και η συχνότητα δειγματοληψίας είναι $f_s=3Hz$. Η συνθήκη $f_s>2\cdot f_0$ του θεωρήματος του Nyquist δεν ικανοποιείται. Κατά συνέπεια το σήμα εξόδου είναι διαφορετικό από το σήμα εισόδου. Το γεγονός αυτό φαίνεται και στο δειγματοληπτημένο σήμα όπου συνυπάρχουν το σήμα πληροφορίας και το σήμα εξόδου αλλά και από τις τιμές στα text boxes. Έτσι ενώ στην είσοδο εμφανίζεται f0 = 2.0 (η συχνότητα του ημιτόνου) στην έξοδο δίνει f0 = -1.0, η οποία είναι η αναδιπλωμένη συχνότητα. Και στο πεδίο των συχνοτήτων μπορεί κανείς να δει τη διαφορά καθώς στο φάσμα εξόδου περιλαμβάνονται φασματικές γραμμές που προέρχονται από τη συνέλιξη του φάσματος του σήματος εισόδου με τις φασματικές γραμμές του σήματος δειγματοληψίας που βρίσκονται στις θέσεις -3.3 και +3.3 (οι πρώτες αρμονικές).



Εικόνα 6.15 Δειγματοληψία ημιτονικού σήματος συχνότητας f_0 με συχνότητα δειγματοληψίας $f_s<2\cdot f_0$

6.7.6. Δειγματοληψία ημιτονικού σήματος 2 συχνοτήτων f_0, f_1 με συχνότητα δειγματοληψίας $f_s>2\cdot f_0$ και $f_s>2\cdot f_1$

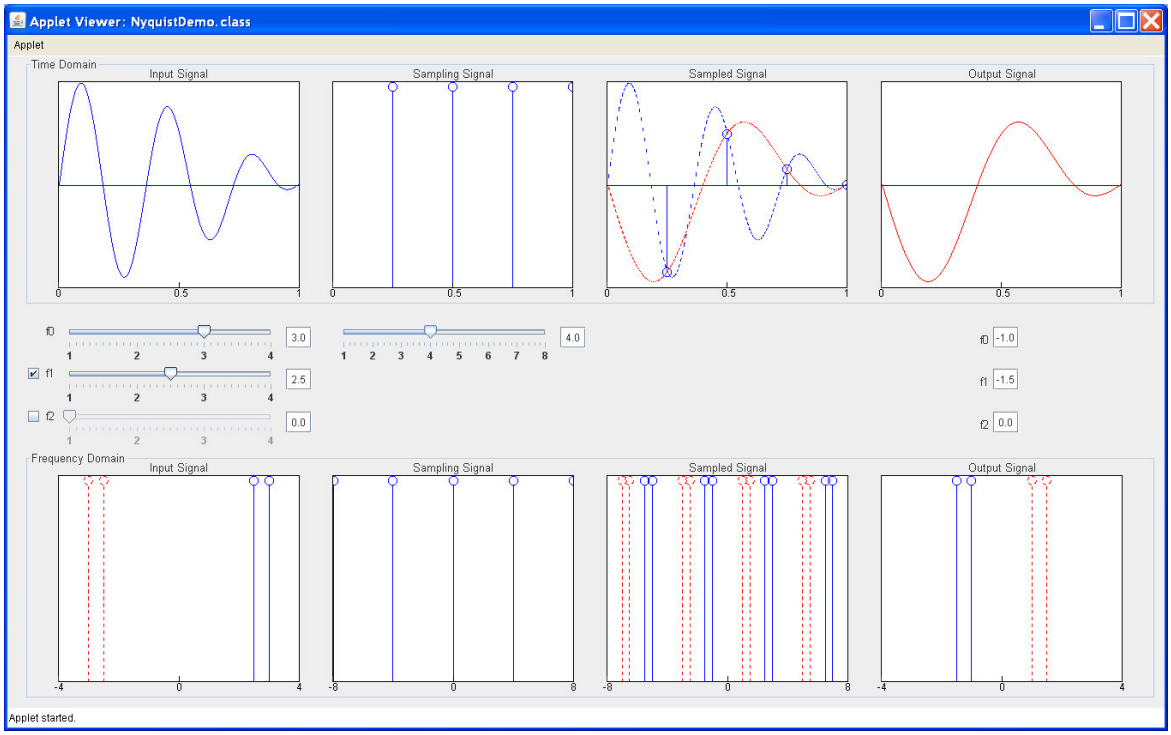
Εδώ, όπως και στην πρώτη περίπτωση (6.8.1) έχουμε σωστή δειγματοληψία και η έξοδος είναι ίδια με το σήμα εισόδου. Το φάσμα του σήματος εξόδου περιλαμβάνει τι ίδιες ακριβώς φασματικές γραμμές με αυτό του σήματος πληροφορίας.



Εικόνα 6.16 Δειγματοληψία ημιτονικού σήματος 2 συχνοτήτων f_0, f_1 με συχνότητα δειγματοληψίας $f_s>2\cdot f_0$ και $f_s>2\cdot f_1$

6.7.7. Δειγματοληψία ημιτονικού σήματος 2 συχνοτήτων f_0, f_1 με συχνότητα δειγματοληψίας $f_s<2\cdot f_0$ και $f_s<2\cdot f_1$

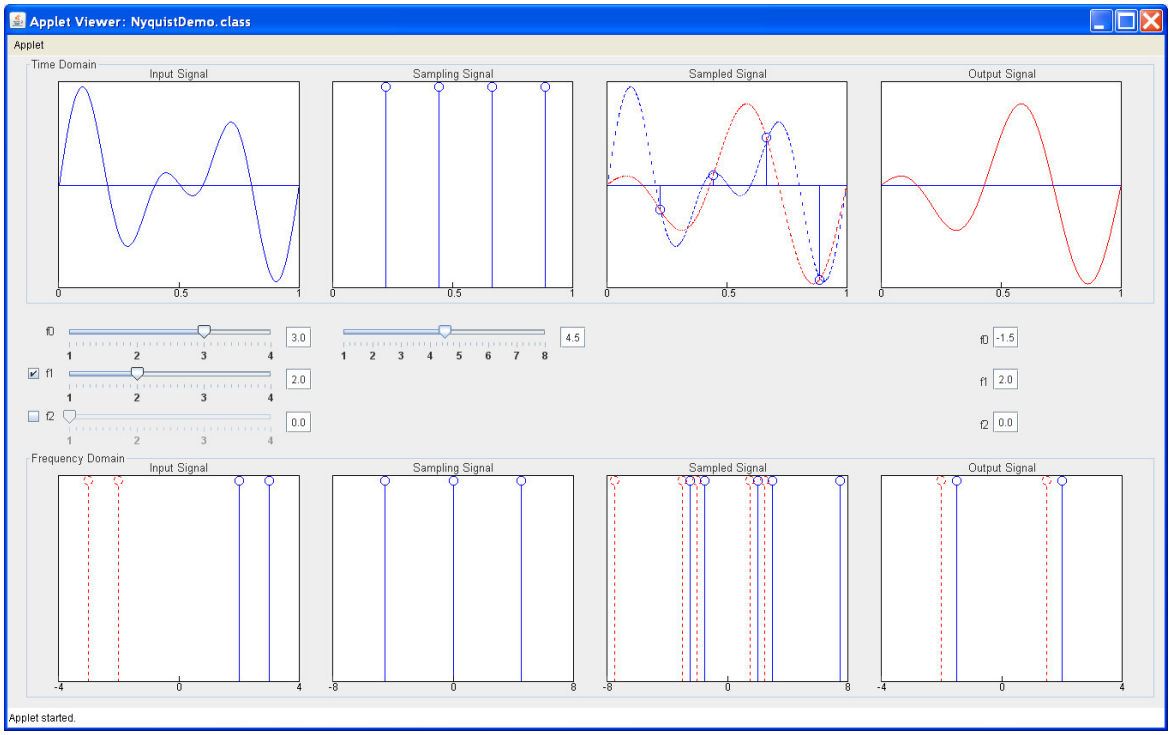
Εδώ, όπως και στην δεύτερη περίπτωση (6.8.2) δεν έχουμε σωστή δειγματοληψία καθώς η συχνότητα δειγματοληψίας είναι μικρότερη από το διπλάσιο και των δυο συχνοτήτων του σήματος πληροφορίας. Το αποτέλεσμα είναι να έχουμε 2 λανθασμένες συχνότητες στην έξοδο. Έτσι οι συχνότητες 3Hz και 2.5Hz του σήματος εισόδου, μεταφράζονται σε -1.0Hz και -1.5Hz στην έξοδο.



Εικόνα 6.17 Δειγματοληψία ημιτονικού σήματος 2 συχνοτήτων f_0, f_1 με συχνότητα δειγματοληψίας $f_s<2\cdot f_0$ και $f_s<2\cdot f_1$

6.7.8. Δειγματοληψία ημιτονικού σήματος 2 συχνοτήτων f_0, f_1 με συχνότητα δειγματοληψίας $f_s<2\cdot f_0$ και $f_s>2\cdot f_1$

Τέλος, θα παρουσιαστεί και η περίπτωση όπου το σήμα πληροφορίας περιλαμβάνει 2 συχνότητες $f_0 = 3Hz$ και $f_1 = 2Hz$ ενώ η συχνότητα δειγματοληψίας είναι $f_s = 4.5Hz$. Σε αυτή την περίπτωση η συχνότητα f_1 δειγματοληπτείται σωστά αλλά η f_0 όχι με αποτέλεσμα να έχουμε λανθασμένο σήμα στην έξοδο. Οι συχνότητες που περιλαμβάνει το σήμα εξόδου είναι $f_0 = -1.5Hz$ και $f_1 = 2Hz$.



Εικόνα 6.18 Δειγματοληψία ημιτονικού σήματος 2 συχνοτήτων f_0, f_1 με συχνότητα δειγματοληψίας $f_s<2\cdot f_0$ και $f_s>2\cdot f_1$

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα εργασία ξεκίνησε με την προοπτική η μικροεφαρμογή η οποία θα κατασκευάζονταν να μπορούσε να χρησιμοποιηθεί στην πράξη το Σχ. Έτος 2014-2015. Δυστυχώς εξωγενείς παράγοντες δεν επέτρεψαν την ολοκλήρωσή της εντός του χρονικού πλαισίου που θα επέτρεπε τη χρήση της στην διδασκαλία της συγκεκριμένης έννοιας (συχνότητα δειγματοληψίας). Παρόλα αυτά μπορεί να χρησιμοποιηθεί σε επόμενα σχολικά έτη. Η απουσία όμως δεδομένων από τη χρήση της, δεν επιτρέπει την αξιολόγησή της από παιδαγωγικής πλευράς, όσον αφορά την βελτίωση της κατανόησης της συγκεκριμένης έννοιας από τους μαθητές.

Μπορεί όμως να αξιολογηθεί, σε ένα βαθμό, για την καταλληλότητα, την εκπαίδευση και τη διαθεσιμότητα (Kamthan 2009), όπως αυτές ορίστηκαν στην παράγραφο 2.3.

Καταλληλότητα: Η μικροεφαρμογή που αναπτύχθηκε στοχεύει σε μια απλή αλλά πολύ σημαντική έννοια της διδακτέας ύλης. Ο μαθητής μπορεί να απομνημονεύσει το θεώρημα της δειγματοληψίας αλλά πάντα είναι προτιμότερο να το κάνει κτήμα του. Η μικροεφαρμογή, μέσα από την αλληλεπίδραση με τον μαθητή, τον οδηγεί στο να ανακαλύψει ο ίδιος το θεώρημα βοηθώντας με αυτόν τον τρόπο την αφομοίωσή του.

Εκπαίδευση: Το περιβάλλον αλληλεπίδρασης της μικροεφαρμογής (η διεπαφή) είναι φιλικό και εύκολο στη χρήση. Ο χρήστης δεν θα συναντήσει κάποιο πρόβλημα στο χειρισμό της μικροεφαρμογής ακόμα και αν δοκιμάσει να την χρησιμοποιήσει χωρίς οδηγίες από τον καθηγητή. Επειδή όμως ο μαθητής μπορεί να μην διακρίνει αμέσως όλη την πληροφορία η οποία υπάρχει στα γραφήματα, είναι χρήσιμο να υπάρξει μια επίδειξη ή επεξήγηση πρώτα. Το δεύτερο σημείο σχετικά με την εκπαίδευση αφορά την δημιουργία της μικροεφαρμογής. Αυτός ο οποίος θα την αναλάβει πρέπει να έχει κάποιες βασικές γνώσεις προγραμματισμού και σχετική ευχέρεια στον χειρισμό του απαραίτητου λογισμικού. Η εξοικείωση με την ανάπτυξη μικροεφαρμογών αποτελούσε ένα από τους στόχους και της παρούσας εργασίας. Μέσα από όλη τη διαδικασία της σχεδίασης και της ανάπτυξης, όπως παρουσιάστηκε στην εργασία, οι σπουδαστές αποκόμισαν γνώσεις και δεξιότητες οι οποίες μπορούν να φανούν χρήσιμες και στην ανάπτυξη άλλων μικροεφαρμογών.

Διαθεσιμότητα: Ένα σημαντικό σημείο στην επιλογή της μικροεφαρμογής η οποία αναπτύχτηκε ήταν η διαθεσιμότητά της. Μετά από έρευνα στο διαδίκτυο βρέθηκαν

μικροεφαρμογές Java αλλά και άλλο υλικό (π.χ. animated gif) το οποίο να παρουσιάζει το αποτέλεσμα του θεωρήματος της δειγματοληψίας. Καμία όμως από αυτές τις μικροεφαρμογές δεν έδινε τη δυνατότητα στο χρήστη να πειραματιστεί εκ των προτέρων και να εξαγάγει μόνος του το θεώρημα. Λειτουργούσαν περισσότερο ως μέσο επίδειξης του ήδη διδαχθέντος θεωρήματος. Ο βαθμός δε αλληλεπίδρασης με το χρήστη περιοριζόταν στο ελάχιστο δυνατό. Η μικροεφαρμογή που αναπτύχθηκε στο πλαίσιο της εργασίας παρουσίαζε χαρακτηριστικά τα οποία θεωρήθηκαν σημαντικά και τα οποία δεν υπήρχαν σε όσες βρέθηκαν στο διαδίκτυο. Το μεγαλύτερο όμως μειονέκτημα της ανάπτυξης μιας μικροεφαρμογής από το μηδέν, ο κίνδυνος δηλαδή να μην είναι αυτή έτοιμη εγκαίρως ώστε να χρησιμοποιηθεί στην τάξη, παρουσιάστηκε κατά την υλοποίησή της. Όπως αναφέρθηκε και στην αρχή των συμπερασμάτων, τελικά η μικροεφαρμογή ολοκληρώθηκε με τη λήξη των μαθημάτων στα ΕΠΑ.Λ. και έτσι δεν ήταν δυνατόν να χρησιμοποιηθεί στην πράξη για το Σχ. Έτος 2014-2015.

Η χρήση των ΤΠΕ στην διδασκαλία δεν αποτελεί μια τάση αλλά μια αναγκαιότητα η οποία αναγνωρίζεται από όλους όσους εμπλέκονται στην εκπαιδευτική διαδικασία αλλά και το ίδιο το Υπουργείο Παιδείας. Οι εκπαιδευτικοί πρέπει να παρακινούνται ώστε να ενσωματώνουν στη διδασκαλία τη χρήση των ΤΠΕ. Ένα μέρος της τεχνολογίας αυτής αποτελούν και οι μικροεφαρμογές Java. Στην εκπαίδευση υπάρχουν πολλοί εκπαιδευτικοί με το απαραίτητο υπόβαθρο ώστε να υλοποιήσουν τέτοιες μικροεφαρμογές. Η παρούσα εργασία αποδεικνύει ότι με την προϋπόθεση της ύπαρξης διαθέσιμου χρόνου είναι εφικτή η υλοποίησή τους.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Γεωργίου Θ., Καππος Ι., Λαδιάς Α., Μικρόπουλος Α., Τζιμογιάννης Α., Χαλκιά Κ. (2008) *Πολυμέσα - Δίκτυα (Γ' Γενικού Λυκείου)*, Αθήνα Ο.Ε.Δ.Β. ανακτήθηκε από http://ebooks.edu.gr/modules/ebook/show.php/DSGL-C104/423/2835,10766/ (13/03/2015)

Digital Academy (2013) *Εισαγωγή στη γλώσσα προγραμματισμού Java. Σημειώσεις Σεμιναρίου*, Αθήνα, ανακτήθηκε από http://www.dga.gr/web/publications/files/Java.pdf (13/03/2015)

ΕΜΠ (2013) *Εισαγωγή στη Γλώσσα Προγραμματισμού Java. Σημειώσεις Εργαστηρίου Πολυμέσων*, Τομέας Επικοινωνιών, Ηλεκτρονικής & Συστημάτων Πληροφορικής, Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ, ΕΜΠ. Ανακτήθηκε από http://www.ebooks4greeks.gr/dowloads/Pliroforiki/Glosses.program./Java__Downloaded_from_eBooks4Greeks.gr.pdf (13/03/2015)

Γαβαλάς Δ. (χ.η.) *Αντικειμενοστραφής Προγραμματισμός I, 9η Διάλεξη, Μικροεφαρμογές(applets) – Γραφικά*, Τμήμα Πολιτισμικής Τεχνολογίας & Επικοινωνίας, Πανεπιστήμιο Αιγαίου, ανακτήθηκε από http://dgavalas.ct.aegean.gr/OO_I/slides/OO_I_09.pdf (13/03/2015)

Κάβουρας Ι., Ρουκουνάκη Α. (2003) *Κεφάλαιο 20, Μικροεφαρμογές, Προγραμματισμός με Java, 1η έκδοση,* Εκδόσεις Κλειδάριθμος, Αθήνα, ανακτήθηκε από http://www.java-programming.eu/files/ProgrammingWithJava-CR-Ch20.pdf (13/03/2015)

Κουϊμτζής Γ., Χερτούρας Κ. (2009) *Ανάπτυξη και Αξιολόγηση Εφαρμογής JavaFX στην Εκμάθηση του Αλγορίθμου Ταξινόμησης Φυσαλίδας*, 3η Πανελλήνια Διημερίδα Καθηγητών Πληροφορικής, Αλεξανδρούπολη, ανακτήθηκε από

http://users.sch.gr/alouvris/2009/ergasies/pdkap7.pdf (02/04/2015)

Κασούτσας Α., (2012) *Τα Mathlets στη Μαθηματική Εκπαίδευση*, Ελληνικό Ανοικτό Πανεπιστήμιο, Σχολή Θετικών Επιστημών και Τεχνολογίας, Open Education - The Journal for Open and Distance Education and Educational Technology Volume 8, Number 1, 2012 Section one. (02/04/2015)

Θεοφανέλλης Τ., Μπλέκας Μ., Γιαμαρέλου Μ. (2008) *Μικροεφαρμογές Java (Java applets) ως εργαλείο υποστήριξης των εκπαιδευτικών στην Φυσική και τα Μαθηματικά*, 1ο Πανελλήνιο Εκπαιδευτικό Συνέδριο Ημαθίας, Νάουσα, ανακτήθηκε από http://ekped.gr/praktika/fe/05_51k.swf (02/04/2015)

Χατζηλυγερούδης Ι., Μακρής Χ., Ρήγκου Μ., (2014), *Οντοκεντρικός Προγραμματισμός, Java-Γενικά Χαρακτηριστικά (διαφάνειες),* Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, Πανεπιστήμιο Πατρών, Πάτρα, ανακτήθηκε από http://aigroup.ceid.upatras.gr/undergrad/java/docs/jav2-gen.pdf (02/04/2015)

Παπαστεργίου Μ., Μπέκας Δ., Νάρη Ε., Δίντσης Θ., Ζαχαρακόπουλος Χ., Αθανασίου Γ., Καρακωνσταντής Γ. (2005) *Ένας Εκπαιδευτικός Δικτυακός Τόπος με Μικροεφαρμογές Java για την Διδασκαλία της Πληροφορικής*, 3ο Συνέδριο στη Σύρο – ΤΠΕ στην Εκπαίδευση, Σύρος, ανακτήθηκε από
http://www.epyna.eu/agialama/synedrio_syros_3/pliroforikon/papastergiou.pdf (02/04/2015)

Kamthan P. (2009) *Java Applets in Education* ανακτήθηκε από www.irt.org/articles/js151/ (10/04/2015)

UoC, *Δειγματοληψία και Ανακατασκευή Σημάτων, Διάλεξη 3*, Biomedical Imaging and Applied Optics Laboratory, University of Cyprus ανακτήθηκε από http://www.eng.ucy.ac.cy/cpitris/courses/ECE623/presentations/Lecture3.pdf (15/04/2015)

Αναστασόπουλος Β. (1998) *Ανάλυση και Επεξεργασία Ψηφιακών Σημάτων*, Εργαστήριο Ηλεκτρονικής, Τομέας Ηλεκτρονικής και Υπολογιστών, Τμήμα Φυσικής, Πανεπιστήμιο Πατρών, Πάτρα

Τραγανίτης Α. (2005), *Ψηφιακές Επικοινωνίες, Θεωρία Δειγματοληψίας, A/D και D/A μετατροπείς (παρουσίαση)*, Εργαστήριο Τηλεπικοινωνιών, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, ανακτήθηκε από www.csd.uoc.gr/~hy430/HY430-2004_5.ppt (15/04/2015)

Viswanathan P. (2014) *Comparing Java IDEs: Eclipse vs. NetBeans vs. IntelliJ*, ανακτήθηκε από http://mobiledevices.about.com/od/additionalresources/fl/Comparing-Java-IDEs-Eclipse-vs-NetBeans-vs-IntelliJ.htm (16/04/2015)

Hamilton C. (2014) *Engineering choices:Eclipse, NetBeans or IntelliJ: Which is the best Java IDE?* ανακτήθηκε από http://jaxenter.com/eclipse-netbeans-or-intellij-which-is-the-best-java-ide-107980.html (16/04/2015)

www.java2s.com, *Plotting the sine function: Use the drawPoint() method to plot points* ανακτήθηκε από http://www.java2s.com/Tutorial/Java/0300__SWT-2D-Graphics/PlottingthesinefunctionUsethedrawPointmethodtoplotpoints.htm (17/04/2015)

Java Applet Tutorial ανακτήθηκε από http://www.realapplets.com/tutorial/index.html (17/04/2015)

Chua Hock-Chuan (2013) *Java Programming Tutorial, Custom Graphics*, ανακτήθηκε από https://www3.ntu.edu.sg/home/ehchua/programming/java/J4b_CustomGraphics.html (17/04/2015)

Winchester J. (2003) *Graphics Context - Quick on the draw*, International Business Machines Corp.**,** ανακτήθηκε από http://www.eclipse.org/articles/Article-SWT-graphics/SWT_graphics.html#Canvas (17/04/2015)

Fowler A. (n.d.) Painting in AWT and Swing, *Good Painting Code Is the Key to App Performance*, ανακτήθηκε από http://www.oracle.com/technetwork/java/painting-140037.html (18/04/2015)

The Java™ Tutorials, *Advanced Topics in Java2D*, ανακτήθηκε από https://docs.oracle.com/javase/tutorial/2d/advanced/index.html (18/04/2015)

The Java™ Tutorials, *Java Applets*, ανακτήθηκε από https://docs.oracle.com/javase/tutorial/deployment/applet/index.html, (16/04/2015)

The Java™ Tutorials, *Using Swing Components*, ανακτήθηκε από https://docs.oracle.com/javase/tutorial/uiswing/components/index.html (16/04/2015)

Thompson A. (2011) *How to draw in jPanel? (swing/graphics Java)*, conversation at stackoverflow.com, ανακτήθηκε από http://stackoverflow.com/questions/6118737/how-to-draw-in-jpanel-swing-graphics-java (18/04/2015)

Mortensen P. (2014) *Drawing a simple line graph in Java*, conversation at stackoverflow.com, ανακτήθηκε από http://stackoverflow.com/questions/8693342/drawing-a-simple-line-graph-in-java (16/04/2015)

Wall T. (2006) *Decorating/Overpainting Swing Components*, The Rabbit Hole, ανακτήθηκε από http://rabbit-hole.blogspot.gr/2006/04/decoratingoverpainting-swing.html (16/04/2015)

Leahy P. (n.d) *Event*, ανακτήθηκε από http://java.about.com/od/e/g/Event.htm (16/04/2015)

Oracle and/or its affiliates, *Java™ Platform, Standard Edition 7 API Specification*, ανακτήθηκε από http://docs.oracle.com/javase/7/docs/api/ (16/04/2015)

Πνευματικά δικαιώματα

Αθανάσιος Παπαζαχαρίας,

Τζώτζου Ευανθία

2015