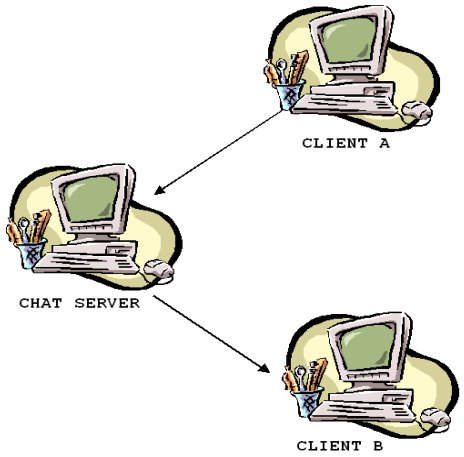
Πτυχιακή Εργασία

Σχεδίαση και δημιουργία ενός λογισμικού πακέτου σε JAVA (Server-Client) για διαδικτυακή συνομιλία (Chat).



Σπουδαστές: Μισεδάκης Δημήτρης - Λυράκης Λευτέρης

Επιβλέπων: Πετράκης Νικόλαος

Χανιά

2002

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Πριν μερικά χρόνια, στο χώρο της πληροφορικής, οι γλώσσες προγραμματισμού προορίζονταν κυρίως για την δημιουργία εφαρμογών, διευκολύνοντας τον προγραμματιστή στην σχεδίαση και την κατασκευή των προγραμμάτων. Όμως η εμφάνιση των δικτύων καθώς και η ραγδαία ανάπτυξη - εξάπλωση του Διαδικτύου (Internet) έκανε επιτακτική την ανάγκη για την δημιουργία και την καθιέρωση μιας γλώσσας που να εξυπηρετεί τον προγραμματιστή, τόσο στην δημιουργία κλασικών εφαρμογών, όσο και στην δημιουργία εφαρμογών δικτύου. Το κενό αυτό ήρθε να καλύψει η JAVA η οποία σήμερα έχει επικρατήσει στο χώρο αυτό. Όταν η JAVA πρωτοπαρουσιάστηκε έφερε επανάσταση στη δημιουργία ιστοσελίδων με την χρήση JAVA Applet και άλλων δυνατοτήτων που προσέφερε [1].

Η πτυχιακή μας θέλαμε να έχει σχέση με τον προγραμματισμό και κυρίως με εφαρμογές Διαδικτύου, για αυτό το λόγο επιλέξαμε την JAVA σαν την καταλληλότερη γλώσσα προγραμματισμού για την υλοποίηση της. Πολλές ήταν οι πιθανές εφαρμογές που θα μπορούσαμε να διαλέξουμε, αλλά επιλέξαμε να ασχοληθούμε με μια που σχετίζεται με μια από τις πιο διαδεδομένες δραστηριότητες στο χώρο των δικτύων που είναι η διαδικτυακή συνομιλία (chat). Έτσι για θέμα της πτυχιακής μας αποφασίσαμε να σχεδιάσουμε ένα πρόγραμμα που να υλοποιεί ένα τέτοιο περιβάλλον.

Ευελπιστούμε στα παρακάτω κεφάλαια να αναπτύξουμε όσο καλύτερα μπορούμε το θέμα της πτυχιακής μας και παράλληλα να περάσουμε στους αναγνώστες κάποιες βασικές γνώσεις για την JAVA. Στο δεύτερο κεφάλαιο παραθέτουμε ένα σύντομο οδηγό για την JAVA, εξηγώντας τις βασικές ιδέες για την δομή της γλώσσας (τάξεις, μέθοδοι κτλ) και περιγράφοντας εντολές που χρησιμοποιήσαμε στην δημιουργία της πτυχιακής μας. Στο τρίτο κεφάλαιο περιγράφουμε περιληπτικά τα πακέτα των προγραμμάτων που χρησιμοποιήσαμε και επεξηγούμε αναλυτικότερα τις λειτουργίες του που θεωρούμε χρήσιμες. Στο επόμενο κεφάλαιο, δηλαδή στο τέταρτο, αναλύουμε τον κώδικα του προγράμματός μας εξετάζοντας κάθε τάξη χωριστά. Στο πέμπτο κεφάλαιο αναφερόμαστε στο τρόπο εγκατάστασης του πακέτου (Client – Server) και στην χρήση του. Τέλος στο έκτο και τελευταίο κεφαλαίο διατυπώνουμε τα συμπεράσματα μας και αναφέρουμε πιθανές επεκτάσεις (βελτιώσεις) που μπορεί μελλοντικά να προστεθούν στον πρόγραμμα. Στο παράρτημα υπάρχει ένας οπτικός δίσκος (CD) στον οποίο βρίσκονται τα προγράμματα που χρησιμοποιήσαμε για την δημιουργία της πτυχιακής μας καθώς και ο κώδικας του προγράμματός μας (MILY Chat).

Θα θέλαμε εδώ να ευχαριστήσουμε τον επιβλέποντα, κύριο Πετράκη Νικόλαο για την πολύτιμη βοήθεια που μας προσέφερε για την πραγματοποίηση αυτής της πτυχιακής, καθώς και τον συμφοιτητή μας Μαθιουδάκη Χάρη για την υλικοτεχνική υποδομή που μας παρείχε.

2. ΣΥΝΤΟΜΟΣ ΟΔΗΓΟΣ ΤΗΣ JAVA

Ακολουθεί μια σύντομη αναφορά στην ιστορία της JAVA καθώς και οι βασικές δομές – εντολές της γλώσσας.

2.1. ΙΣΤΟΡΙΑ ΤΗΣ JAVA

Η JAVA αναπτύχθηκε από την SUN Microsystems το 1991 με σκοπό να χρησιμοποιηθεί για τον έλεγχο των έξυπνων συσκευών του μέλλοντος, όπως interactive TV. Οι ερευνητές της ήθελαν οι συσκευές αυτές να έχουν την δυνατότητα να επικοινωνούν μεταξύ τους. Έτσι δημιούργησαν μια πρωτότυπη συσκευή που την ονόμασαν Star7. Η βασική ιδέα ήταν να σχεδιάσουν το λειτουργικό σύστημα της συσκευής αυτής χρησιμοποιώντας την C++. Όμως ο James Gosling βλέποντας της δυσκολίες που δημιουργούσε η C++ στην ανάπτυξη του λογισμικού, αποφάσισε να δημιουργήσει μια νέα γλώσσα, την JAVA. Παρ' όλο που η JAVA δεν σχεδιάστηκε αρχικά σαν μια αυτόνομη γλώσσα προγραμματισμού, τα πλεονεκτήματα που είχε και την έκαναν ιδανική για το Star7 αποδείχτηκαν εξίσου χρήσιμα και για εφαρμογές του Διαδικτύου που τότε είχε αρχίσει να αναπτύσσεται. Η πρώτη επίσημη έκδοση της γλώσσας ήταν τον Αύγουστο του 1995. Από τότε ακολούθησαν πολλές νέες εκδόσεις που την βελτίωναν συνεχώς ώστε να γίνει σήμερα η κατεξοχήν γλώσσα του Διαδικτύου.

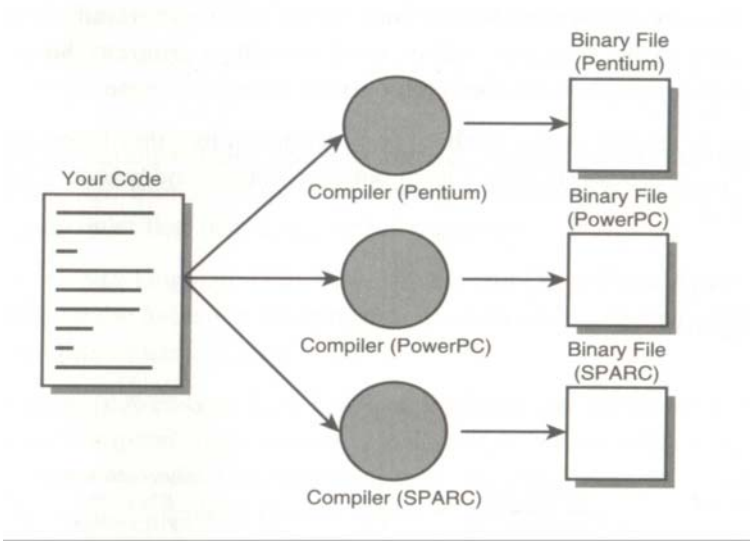
2.2. ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΗΣ JAVA

Οι λόγοι που η JAVA επικράτησε στο Διαδίκτυο είναι:

i) Το μέγεθος των προγραμμάτων της είναι μικρό, με αποτέλεσμα οι ιστοσελίδες να φορτώνονται πιο γρήγορα.

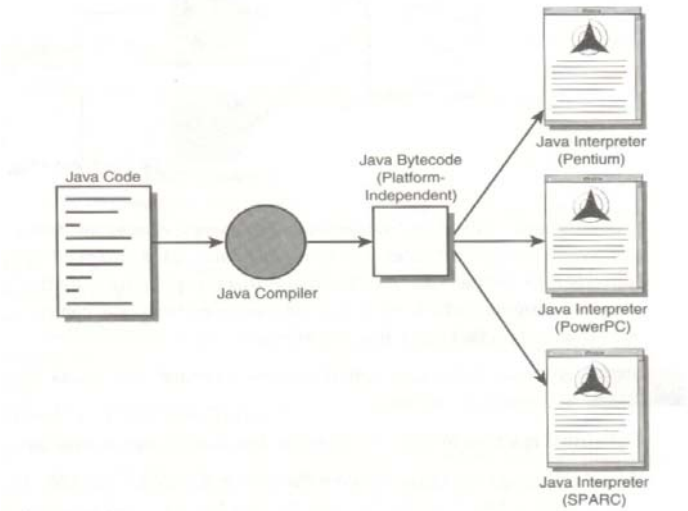
ii) Η JAVA προσφέρει υψηλό επίπεδο προστασίας.

iii) Το μεγαλύτερο πλεονέκτημα της είναι η συμβατότητα που έχει με όλα τα λειτουργικά συστήματα (Windows, Unix, Macintosh).
Όταν μεταγλωττίζουμε (compilation) ένα πρόγραμμα στη C ή στις περισσότερες άλλες γλώσσες, ο compiler μεταφράζει τον πηγαίο κώδικα σε κώδικα μηχανής. Αυτός ο κώδικας είναι αποκλειστικός για τον τύπο του επεξεργαστή που έγινε η μεταγλώττιση, όπως φαίνεται στο Σχήμα 2.1. Εάν θέλουμε να χρησιμοποιήσουμε το ίδιο πρόγραμμα σε άλλη πλατφόρμα, πρέπει να μεταφέρουμε τον πηγαίο κώδικα στη νέα πλατφόρμα και να ξανακάνουμε μεταγλώττιση. Σε πολλές περιπτώσεις, αλλαγές στον πηγαίο κώδικα απαιτούνται πριν γίνει η μεταγλώττιση στο καινούργιο κώδικα.



Σχήμα 2.1. Ο παραδοσιακών τρόπος μεταγλώττιση των προγραμμάτων [1]

Τα προγράμματα στην JAVA πετυχαίνουν αυτή την ανεξαρτησία με τη χρήση της εικονικής μηχανής (Virtual Machine). Η εικονική μηχανή μετατρέπει τα μεταγλωττισμένα JAVA προγράμματα σε εντολές τις οποίες το λειτουργικό σύστημα μπορεί να χειριστεί, όπως φαίνεται στο Σχήμα 2.2. Ο κώδικας που δημιουργείται μπορεί να τρέξει σε οποιοδήποτε λειτουργικό σύστημα, αρκεί να υπάρχει εγκατεστημένος ο διερμηνευτής JAVA (JAVA Interpreter).



Σχήμα 2.2 Ο τρόπος μεταγλώττισης προγραμμάτων της JAVA για την δημιουργία κώδικα που τρέχει σε πολλές πλατφόρμες [1]

iv) Ένα τελευταίο και εξίσου σημαντικό πλεονέκτημα της JAVA είναι ότι είναι αντικειμενοστρεφής γλώσσα προγραμματισμού. Σε αντίθεση με παλαιότερες γλώσσες που χειριζόταν τα προγράμματα σαν μια λίστα από εντολές που υπαγόρευαν στον επεξεργαστή τι να κάνει. Στις αντικειμενοστρεφείς γλώσσες, ο τρόπος που βλέπουμε τα προγράμματα είναι σαν ένα σύνολο από «αντικείμενα» που συνεργάζονται με προκαθορισμένους τρόπους για να επιτύχουν τον σκοπό τους.

2.3. ΤΡΟΠΟΣ ΣΥΝΤΑΞΗΣ ΚΑΙ ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ

2.3.1. ΤΑΞΕΙΣ ΚΑΙ ΜΕΘΟΔΟΙ

Όπως προαναφέραμε η JAVA είναι αντικειμενοστρεφής γλώσσα προγραμματισμού. Αλλά όταν γράφουμε ένα πρόγραμμα στη JAVA δεν ορίζουμε ανεξάρτητα «αντικείμενα». Αντίθετα ορίζουμε τάξεις (Class) αντικειμένων. Για να γίνει καλύτερα κατανοητό αυτό θα αναφέρουμε ένα παράδειγμα. Έστω ότι επιθυμούμε να δημιουργήσουμε μια τάξη για ένα κουμπί που θα εκτελεί μια εντολή (Command Button). Όταν η τάξη αυτή ολοκληρωθεί θα πρέπει να ορίζει τα εξής χαρακτηριστικά για το Command Button.

- Ένα κείμενο που θα επεξηγεί το σκοπό του Command Button.
- Το μέγεθος του Command Button.
- Στοιχεία της εμφάνισης του, όπως αν έχει τρισδιάστατη σκιά.

Η τάξη αυτή επίσης θα πρέπει να ορίζει πως θα συμπεριφέρεται το Command Button.

- Αν το κουμπί απαιτεί μονό ή διπλό πάτημα για να χρησιμοποιηθεί.
- Αν θα πρέπει να αγνοεί τα πατήματα του ποντικιού εντελώς.
- Τι θα κάνει όταν πατηθεί το κουμπί.

Αφού έχουμε ορίσει αυτή την τάξη, μπορούμε να δημιουργήσουμε όσα αντικείμενα αυτής επιθυμούμε, αλλάζοντας απλώς κάποια από τα χαρακτηριστικά της. Τα χαρακτηριστικά (συμπεριφορά) κάθε τάξης ορίζονται από τις μεθόδους. Οι μέθοδοι είναι ένα τμήμα κώδικα σε μια τάξη που χρησιμοποιούντε για να πετύχουν συγκεκριμένους σκοπούς, όπως οι functions χρησιμοποιούνται στις άλλες γλώσσες.

2.3.2. ΜΕΤΑΒΛΗΤΕΣ

Στην JAVA οι μεταβλητές χωρίζονται σε δυο γενικές κατηγορίες. Στις μεταβλητές στιγμιότυπου (instance variables) οι οποίες ορίζουν χαρακτηριστικά ενός συγκεκριμένου αντικειμένου και μόνο αυτού, και στις μεταβλητές τάξεις (class variables) οι οποίες ορίζουν ένα χαρακτηριστικό μιας ολόκληρης τάξης.
Παρακάτω φαίνεται η γενική σύνταξη ενός προγράμματος σε JAVA με τα στοιχεία που έχουμε αναφέρει.

```
class name {  // έτσι ορίζουμε μια τάξη με όνομα name
String var1;  // δηλώνουμε μία μεταβλητή στιγμιότυπου
static string var2; // δηλώνουμε μία μεταβλητή τάξης
void method1() { // δηλώνουμε την μέθοδο
….  // κώδικας μεθόδου
}
public static void main(string arg[ ]) { // κύρια μέθοδος
…. // ακολουθεί ο κώδικας
}
```

Εκτός από το όνομα της η δήλωση μιας μεταβλητής πρέπει να καθορίζει και τον τύπο των δεδομένων που θα φιλοξενεί. Ο τύπος μπορεί να είναι οποιοσδήποτε από τα ακόλουθα:

- Ένας από τους βασικούς τύπους δεδομένων (int, byte, etc.).
- Το όνομα μιας τάξης.
- Ένας πίνακας (Array).

2.3.3. ΕΝΤΟΛΕΣ

Παρακάτω ακολουθεί μια σύντομη αναφορά σε κάποιες βασικές εντολές στην JAVA που χρησιμοποιήσαμε για την δημιουργία του προγράμματος.

2.3.3.1. ΔΗΜΙΟΥΡΓΙΑ ΝΕΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ

Όταν γράφουμε ένα πρόγραμμα στην JAVA, ορίζουμε ένα σύνολο από τάξεις. Τις τάξεις αυτές τις χρησιμοποιούμε για να δημιουργήσουμε τα αντικείμενα τους με τα οποία δουλεύουμε γι' αυτό εξάλλου η JAVA είναι αντικειμενοστρεφής γλώσσα. Σε αυτό το κεφάλαιο θα δείξουμε πώς δημιουργούμε νέα αντικείμενα τάξεων.

Για την δημιουργία ενός νέου αντικειμένου, χρησιμοποιούμε την εντολή new, ακολουθούμενα από το όνομα της τάξης της οποίας θέλουμε να είναι το αντικείμενο όπως φαίνεται στο παρακάτω παράδειγμα.

```
Random randInfo = new Random();
```

Οι παρενθέσεις μπορεί να είναι κενές, όπως στο παραπάνω παράδειγμα, ή να περιέχουν παραμέτρους οι οποίες εισάγονται στο αντικείμενο και καθορίζουν κάποια χαρακτηριστικά του. Αν δημιουργήσουμε δύο νέα αντικείμενα με τις ίδιες παραμέτρους, τότε αυτά τα αντικείμενα ναι μεν έχουν τα ίδια χαρακτηριστικά, αλλά το καθένα υπάρχει ανεξάρτητα του άλλου. Αλλάζοντας τις παραμέτρους, δημιουργούμε δύο νέα αντικείμενα της ίδιας τάξης, αλλά με διαφορετικά χαρακτηριστικά.

2.3.3.2. ΤΕΛΕΣΤΕΣ

Η JAVA αναγνωρίζει τους συνηθισμένους αριθμητικούς και λογικούς τελεστές. Παρακάτω φαίνεται μια εντολή η οποία δηλώνει και αρχικοποιεί την μεταβλητή x με την τιμή 3. Βλέπουμε ότι η εντολή οριοθετείται με το ελληνικό ερωτηματικό (semicolon) και δηλώνουμε ότι η μεταβλητή x μπορεί να αποθηκεύσει ακέραιους (integers).

```
int x = 3;
```

Ο πίνακας που ακολουθεί περιέχει τους σημαντικότερους αριθμητικούς και λογικούς τελεστές.

., [ ], ( )	Η τελεία (dot) χρησιμοποιείται για να καλέσουμε μεθόδους και μεταβλητές μέσα από αντικείμενα και τάξεις. Οι αγκύλες (brackets) χρησιμοποιούντε για να ορίσουμε πίνακες. Οι παρενθέσεις (parenthesis) έχουν ευρεία χρήση από μαθηματικές εκφράσεις έως ορισμούς μεθόδων.
++, --, !	Ο τελεστής (++) αυξάνει την μεταβλητή κατά ένα. Ο τελεστής (--) μειώνει την μεταβλητή κατά ένα. Ο τελεστής (!) είναι το λογικό ΟΧΙ (NOT).
New (type) expression	Ο τελεστής new χρησιμοποιείται για να δημιουργούμε νέα αντικείμενα των τάξεων.
*, /, %	Πολλαπλασιασμός, διαίρεση, υπόλοιπο ακέραιας διαίρεσης.
+, -	Πρόσθεση, αφαίρεση.
<<, >>	Αριστερή και δεξιά ολίσθηση
== , !=	Έλεγχος ισότητας και ανισότητας.
&&	Λογικό ΚΑΙ (AND)
^	Λογικό Αποκλειστικό Η (XOR)
||	Λογικό Η (OR)
? :	Συντομογραφία για το (if … then … else)

Οι τελεστές του παραπάνω πίνακα είναι τοποθετημένοι με βάση την προτεραιότητά τους [1].

2.3.3.3. ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΚΑΙ ΕΠΑΝΑΛΗΨΗΣ

Η ΕΝΤΟΛΗ ΕΛΕΓΧΟΥ if

Ένα από τα σημαντικά στοιχεία του προγραμματισμού είναι η δυνατότητα του προγράμματος να αποφασίζει τι θα κάνει.
Η εντολή if χρησιμοποιεί μια έκφραση boolean για να αποφασίσει αν μια ομάδα εντολών πρέπει να εκτελεστεί ή όχι.
Η εντολή if συντάσσεται ως εξής:

```
if (συνθήκη)
… // εντολές
else
… // εντολές
```

Εάν ισχύει η συνθήκη του if τότε θα εκτελεστούν οι εντολές που το ακολουθούν, διαφορετικά θα εκτελεστούν οι εντολές που βρίσκονται κάτω από το else. Για την καλύτερη κατανόηση της εντολής, ακολουθεί ένα παράδειγμα.

```
if (a == 10)
      System.out.println("a = 10");
else
      System.out.println("a ≠ 10");
```

Το παραπάνω πρόγραμμα ελέγχει αν η τιμή της μεταβλητής είναι ίση με δέκα και ανάλογα με το αποτέλεσμα εμφανίζει ένα μήνυμα.

Η ΕΝΤΟΛΗ ΕΠΑΝΑΛΗΨΗΣ for

Η for επαναλαμβάνει τις εντολές που την ακολουθούν όσο ικανοποιείται η συνθήκη.
Η εντολή for συντάσσεται ως εξής:

```
for (αρχικοποίηση μεταβλητής ; Έλεγχος συνθήκης; μεταβολή) {
… // εντολές
}
```

Οι εντολές στη for εκτελούνται όσο η συνθήκη που ελέγχουμε είναι αληθής. Ακολουθεί ένα παράδειγμα.

```
 for (int i = 0; i < firstNames.length; i++)
      System.out.println(firstNames[i] + " " + lastNames[i]);
```

Στον πίνακα (array) fisrtNames, είναι αποθηκευμένη μια λίστα με ονόματα, ενώ στο lastName μια λίστα με επίθετα. Με την εντολή for που χρησιμοποιούμε, τυπώνουμε τα περιεχόμενα, το ένα δίπλα στο άλλο, για κάθε θέση i των arrays, όσο το array firstNames περιέχει περισσότερα στοιχεία.

Η ΕΝΤΟΛΗ ΕΠΑΝΑΛΗΨΗΣ while

Όπως οι επαναλήψεις με τη for, οι επαναλήψεις while επιτρέπουν σε ένα τμήμα κώδικα JAVA να εκτελείται επαναλαμβανόμενα μέχρι μια συνθήκη να ικανοποιηθεί.

Το while χρησιμοποιείται για να επαναληφθούν ορισμένες εντολές όσο η συνθήκη ικανοποιείται.
Η εντολή while συντάσσεται ως εξής:

```
while (συνθήκη) {
… // εντολές
}
```

Όσο η συνθήκη είναι αληθής θα εκτελούνται οι εντολές μέσα στη while. Ακολουθεί ένα παράδειγμα.

```
int x=1,i=1;
while (i < 10)
      x = x * i++;
```

Το παράδειγμα είναι πολύ απλό και απλώς το γράψαμε για να δείξουμε ότι οι αγκύλες ({}) δεν είναι απαραίτητες στην περίπτωση που μετά την συνθήκη ακολουθεί μία μόνο εντολή.

2.3.3.4. ΠΙΝΑΚΕΣ (ARRAYS)

Οι πίνακες είναι ένας τρόπος να αποθηκεύσουμε μια ομάδα από αντικείμενα που είναι του ίδιου τύπου, για παράδειγμα ακέραιοι. Κάθε αντικείμενο από τη ομάδα τοποθετείται σε δικιά του θέση, η οποία αριθμείται, έτσι ώστε να μπορείς να προσπελάσεις τη πληροφορία εύκολα. Οι πίνακες μπορούν να περιέχουν κάθε τύπο πληροφορίας η οποία αποθηκεύεται σε μια μεταβλητή.
Ο πίνακας ορίζεται ως εξής:

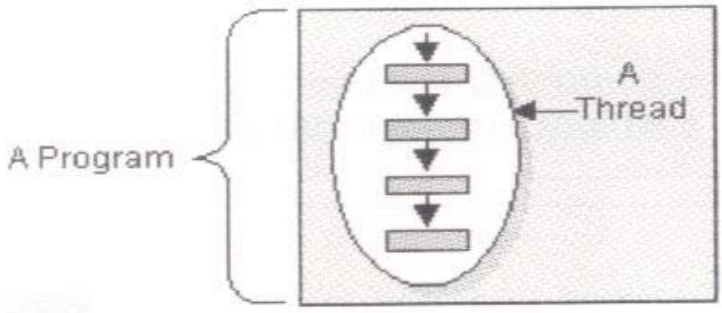
```
(τύπος) όνομα[ # ];
```

Το όνομα είναι το όνομα του πίνακα ενώ ο τύπος ορίζει το είδος των πληροφοριών που αποθηκεύονται στον πίνακα. Ο αριθμός μέσα στις αγκύλες καθορίζει το μέγεθος του πίνακα.

2.4. ΝΗΜΑΤΑ (THREADS)

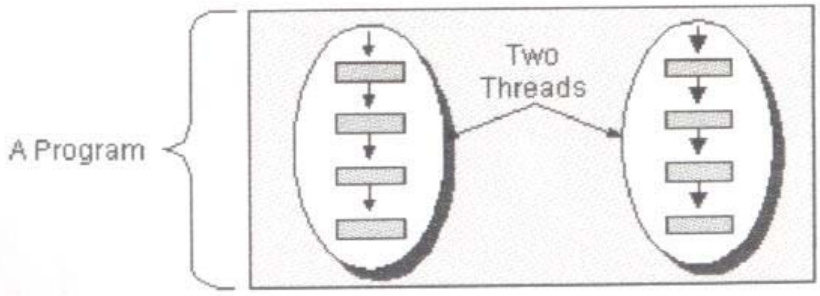
Από τα πρώτα προγράμματα που μαθαίνουμε όταν ασχοληθούμε με μια γλώσσα προγραμματισμού, είναι να υπολογίζουμε μια λίστα από πρώτους αριθμούς ή να τοποθετούμε σε αλφαβητική σειρά μια λίστα από ονόματα. Αυτά τα προγράμματα ονομάζονται ακολουθιακά (sequential): Το καθένα έχει μια αρχή, μια σειρά εκτέλεσης του προγράμματος, και ένα τέλος. Σε κάθε χρονική στιγμή κατά την διάρκεια που το πρόγραμμα τρέχει υπάρχει μια συγκεκριμένη ροή εκτέλεσης [4].

Για την δημιουργία όμως της πτυχιακής μας ήταν απαραίτητο να χρησιμοποιήσουμε νήματα (threads). Τα νήματα είναι παρόμοια με τα sequential προγράμματα που περιγράψαμε προηγουμένως αφού και αυτά έχουν μια αρχή μια σειρά εκτέλεσης προγράμματος και ένα τέλος. Αλλά, ένα thread δεν αποτελεί αυτόνομο πρόγραμμα, αφού δεν μπορεί να εκτελεστεί από μόνο του παρά μόνο, εκτελείται ως τμήμα ενός προγράμματος όπως φαίνεται στο σχήμα 2.3.



Σχήμα 2.3. Μονό νήμα (Single Thread) [4]

Το πλεονέκτημα όμως που προσφέρουν τα νήματα είναι ότι μπορούμε να χρησιμοποιήσουμε πολλαπλά νήματα σε ένα μόνο πρόγραμμα, τα οποία τρέχουν ταυτόχρονα, όπως φαίνεται στο σχήμα 2.4.



Σχήμα 2.4. Διπλό – πολλαπλό νήμα (Multi Thread) [4]

2.5. APPLETS – APPLICATION

Όπως θα εξηγήσουμε παρακάτω υπάρχουν δυο τρόποι για να γράψουμε ένα πρόγραμμα στην JAVA.

Ένας τρόπος είναι να το υλοποιήσουμε ως application, δηλαδή σαν μια εφαρμογή η οποία εκτελείται αυτόνομα με την χρήση του JAVA Interpreter. Η λογική δημιουργίας applications σε JAVA δεν διαφέρει από αυτή κάθε άλλης γλώσσας προγραμματισμού. Αυτό συνεπάγεται ότι ως χρήστες ενός προγράμματος JAVA, γραμμένου ως application, είμαστε υποχρεωμένοι να κατεβάσουμε το πρόγραμμα για να το εκτελέσουμε αφού δεν γίνεται να το προσπελάσουμε μέσω ενός Web Browser (Internet Explorer, Netscape Navigator, Opera etc.).

Σε αντίθεση, με την χρήση Applet, έχουμε την δυνατότητα να γράφουμε προγράμματα τα οποία αποτελούν μέρος μιας ιστοσελίδας. Έτσι δεν είμαστε υποχρεωμένοι να κατεβάσουμε το πρόγραμμα στο σκληρό δίσκο, αφού μπορούμε να το εκτελέσουμε από τον διακομιστή στον οποίο βρίσκεται.

3. ΠΡΟΓΡΑΜΜΑΤΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΣΑΜΕ

Σε αυτό το κεφάλαιο θα αναφερθούμε περιληπτικά στα προγράμματα που χρησιμοποιήσαμε για την πραγματοποίηση της πτυχιακής μας.

3.1. J – CREATOR

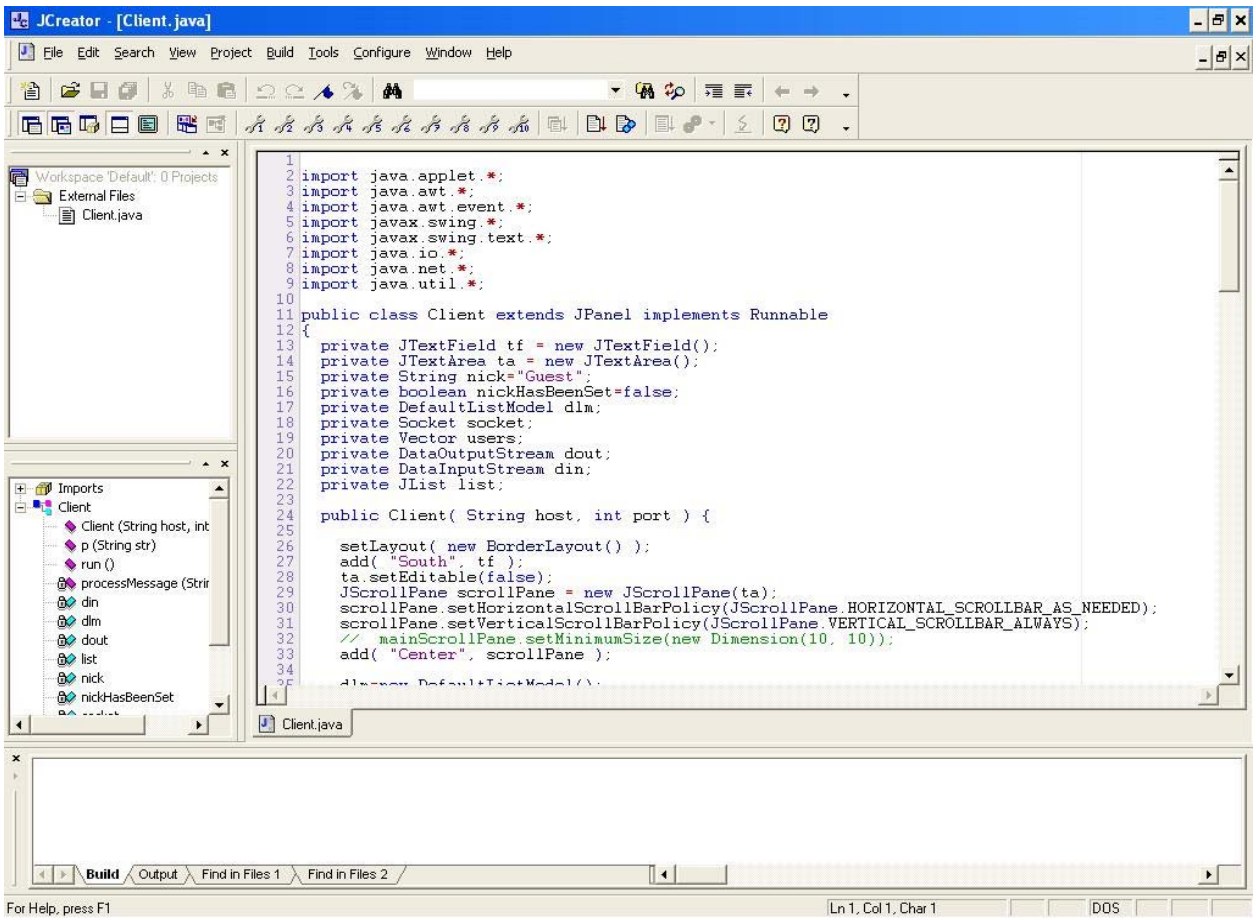
Τον πηγαίο κώδικα (Source Code) στην JAVA μπορούμε να τον γράψουμε και σε ένα απλό κειμενογράφο. Όταν αρχίσαμε να ασχολούμαστε με την πτυχιακή μας χρησιμοποιούσαμε το σημειωματάριο (Notepad) των Windows για να γράφουμε τον κώδικα. Η μεταγλώττιση των προγραμμάτων την κάναμε με την χρήση του μεταγλωττιστή (compiler), που διατίθεται στο πακέτο της JAVA από την SUN, μέσω περιβάλλοντος MS-DOS.

Όμως υπάρχουν πάρα πολλά προγράμματα τα οποία είναι σχεδιασμένα να διευκολύνουν την σύνταξη προγραμμάτων στην JAVA με τα εργαλεία (Utilities) που περιέχουν.

Εμείς επιλέξαμε να χρησιμοποιήσουμε ένα σχετικά απλό πρόγραμμα της Xinox Software, το J-Creator, το οποίο διατίθεται δωρεάν στο Διαδίκτυο στην διεύθυνση http://www.j-creator.com.

Στην διεύθυνση αυτή υπάρχουν δύο εκδόσεις του προγράμματος. Η Shareware έκδοση η οποία διατίθεται για τριάντα μέρες προς δοκιμή και η οποία έχει περισσότερες δυνατότητες από την Freeware έκδοση, την οποία και χρησιμοποιήσαμε. Η εγκατάσταση του προγράμματος είναι απλή και πραγματοποιείται τρέχοντας το εκτελέσιμο αρχείο Setup.exe και ακολουθώντας τις οδηγίες του προγράμματος εγκατάστασης. Το μόνο που πρέπει να προσέξουμε είναι όταν μας ζητηθεί, να παρέχουμε το σωστό φάκελο στον οποίο έχουμε εγκαταστήσει την JAVA.

Αφού εγκαταστήσουμε το πρόγραμμα και το τρέξουμε για πρώτη φορά η εικόνα που θα αντικρίσουμε είναι η εξής (Εικόνα 3.1.):



Εικόνα 3.1. Περιβάλλον του J - Creator

Στην κορυφή του περιβάλλοντος υπάρχει το μενού επιλογών του προγράμματος.
Το περιβάλλον εργασίας όπως βλέπουμε περιλαμβάνει:

- Ένα παράθυρο στο οποίο φαίνονται τα αρχεία τα οποία έχουμε ανοίξει (File View)
- Ένα παράθυρο στο οποίο τυπώνονται μηνύματα από τα εργαλεία του προγράμματος (Output View)
- Ένα παράθυρο στο οποίο μπορούμε να δούμε τις τάξεις και τις μεθόδους του προγράμματος στο οποίο δουλεύουμε (Class View)
- Τέλος το μεγαλύτερο μέρος της οθόνης καταλαμβάνεται από το παράθυρο εργασίας στο οποίο επεξεργαζόμαστε το πρόγραμμα.

Εκτός από τα παραπάνω κυρίως παράθυρα υπάρχουν ακόμα και οι γραμμές εργαλείων (Tool Bars), οι οποίες περιέχουν διάφορα εργαλεία και που μπορούμε να τις ενεργοποιήσουμε ή να τις απενεργοποιήσουμε όποτε θέλουμε.

Τα παραπάνω παράθυρα καθώς και τις γραμμές εργαλείων μπορούμε να τα τοποθετήσουμε όπως θέλουμε στο χώρο εργασίας, έτσι ώστε να μας διευκολύνουν.

Στις επόμενες σελίδες θα δούμε την χρήση αυτών των εργαλείων αναλυτικότερα.

FILE VIEW

Το παράθυρο αρχείων (File View) μας επιτρέπει να επιλέξουμε κάποιο από τα αρχεία του προγράμματος μας. Έχει την δυνατότητα να εμφανίζει το πρόγραμμα μας και τις τάξεις του, με μορφή «δέντρου». Το πρόγραμμα μας μπορεί να αποτελείται από πολλές τάξεις και οι τάξεις αυτές από άλλες υποτάξεις (Subclasses). Όταν αρχίζουμε να γράφουμε ένα πρόγραμμα στο J-Creator, δημιουργείται ένας φάκελος με όλες τις σχετικές με το πρόγραμμα μας πληροφορίες. Κάθε τάξη αυτού του εικονικού δέντρου έχει το δικό της μενού που περιέχει τις μεθόδους που χρησιμοποιεί. Οι μέθοδοι και οι τάξεις μπορούν να αφαιρεθούν από το εικονικό δέντρο πατώντας το πλήκτρο Delete, αλλά το ίδιο το πρόγραμμα (Project) μπορεί να αφαιρεθεί μόνο από το μενού αρχείων (File Menu).

OUTPUT VIEW

Στο παράθυρο (Output View) τυπώνονται πληροφορίες για τις ενέργειες τις οποίες πραγματοποιούμε. Έτσι για παράδειγμα όταν κάνουμε μεταγλώττιση εμφανίζεται ένα μήνυμα που μας πληροφορεί αν η μεταγλώττιση ήταν επιτυχές ή όχι. Σε περίπτωση που κατά την διάρκεια της μεταγλώττισης υπήρξε κάποιο σφάλμα, θα εμφανιστεί ένα σχετικό μήνυμα στο Output View που θα μας ενημερώνει για το είδος του σφάλματος καθώς και σε ποια γραμμή του κώδικα έγινε. Έχουμε την δυνατότητα με διπλό πάτημα του αριστερού κουμπιού του ποντικιού να μεταφερθούμε στην γραμμή αυτή του κώδικα.

CLASS VIEW

Το παράθυρο (Class View) μας επιτρέπει να πλοηγηθούμε στις τάξεις και τις μεθόδους του προγράμματός μας. Περιέχει μια λίστα στην οποία εμφανίζονται αναλυτικά, κάθε τάξη και μέθοδος, του προγράμματος μας, καθώς και οι μεταβλητές που χρησιμοποιούμε. Σημαντικό πλεονέκτημα είναι ότι καθώς γράφουμε τον κώδικα η λίστα αυτή ανανεώνεται ώστε να μην χρειάζεται να πατάμε το κουμπί ανανέωσης. Οι μέθοδοι στο παράθυρο αυτό εμφανίζονται με ροζ χρώμα, ενώ οι μεταβλητές με κυανό. Μπορούμε ακόμα με διπλό πάτημα του αριστερού πλήκτρου του ποντικιού, να μεταφερθούμε στο σημείο του κώδικα που δηλώνεται η μεταβλητή αυτή.

ΠΑΡΑΘΥΡΟ ΕΡΓΑΣΙΑΣ

Το παράθυρο εργασίας είναι στην ουσία ένας κειμενογράφος, ο οποίος όμως παρέχει ορισμένες διευκολύνσεις στον προγραμματιστή. Όπως ότι αριθμεί αυτόματα τις γραμμές του κώδικα ώστε να είναι εύκολος ο εντοπισμός λαθών. Επίσης για να γίνεται το πρόγραμμα πιο κατανοητό αναγνωρίζει και τυπώνει από μόνο του τις μεθόδους και τις μεταβλητές με διαφορετικό χρώμα. Στην Shareware έκδοση του προγράμματος, την οποία δεν διαθέτουμε, όταν χρησιμοποιούμε μια τάξη της βιβλιοθήκης της JAVA, το πρόγραμμα μας εμφανίζει σε μια λίστα τις δυνατές μεθόδους, της τάξης αυτής, που μπορούμε να επιλέξουμε.

ΜΕΝΟΥ ΕΠΙΛΟΓΩΝ

Το μενού επιλογών διαθέτει τις εξής επιλογές:

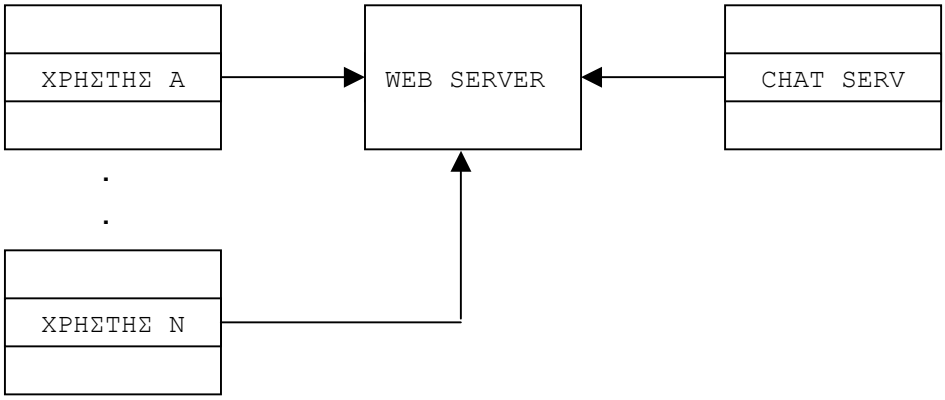
- Το μενού FILE. Αυτό το μενού μας δίνει την δυνατότητα να δημιουργήσουμε και να διαχειριστούμε τα προγράμματα μας. Έτσι σε αυτό το μενού μπορούμε να ανοίξουμε καινούργια αρχεία, να αποθηκεύσουμε τον κώδικα που έχουμε γράψει σε υπάρχοντα ή να φορτώσουμε προγράμματα τα οποία ήδη έχουμε αρχίσει να κατασκευάζουμε. Τέλος, είναι το μενού από το οποίο μπορούμε να τυπώσουμε τα προγράμματα μας, όπως και να τερματίσουμε το J-Creator.
- Το μενού EDIT. Σε αυτό το μενού περιέχονται όλα τα εργαλεία τα οποία χρησιμοποιούνται για την εύκολη διαμορφώση του κώδικα του προγράμματος. Υπάρχει η επιλογή διαγραφή (Delete), αναίρεση (Undo), αποκοπή (Cut), αντιγραφή (Copy), επικόλληση (Paste) και η επιλογή (Read Only) που μας εμποδίζει να επέμβουμε στον κώδικα. Στις προχωρημένες (Andvance) επιλογές αυτού του μενού υπάρχουν τα εργαλεία που μας επιτρέπουν να επιλέγουμε κομμάτια του κώδικα, σύμφωνα με τις παραμέτρους που τους δίνουμε, καθώς και άλλα εργαλεία χρήσιμα που όμως δεν κρίνουμε απαραίτητο να αναφέρουμε.
- Το μενού SEARCH. Εδώ υπάρχουν όλα τα απαραίτητα εργαλεία για να βρούμε ένα συγκεκριμένο τμήμα του κώδικα μέσα στο πρόγραμμα μας. Επίσης μπορούμε να αντικαταστήσουμε το τμήμα αυτό με ένα άλλο κομμάτι κώδικα αυτόματα όσες φορές επαναλαμβάνεται αυτό μέσα στο πρόγραμμα μας.
- Το μενού VIEW. Είναι το μενού που μας επιτρέπει να μορφοποιήσουμε το περιβάλλον εργασίας, έτσι ώστε να μας εξυπηρετεί. Μπορούμε να επιλέξουμε ποια από τα Tool Bars θέλουμε να εμφανίζονται, αν θα εμφανίζεται η αρίθμηση γραμμών του κώδικα, η ακόμα και αν θέλουμε το παράθυρο εργασίας να εμφανίζεται σε πλήρη οθόνη.

- Το μενού BUILD. Από αυτό το μενού μπορούμε να μεταγλωττίσουμε τα προγράμματα που έχουμε δημιουργήσει. Επίσης μπορούμε και να τα τρέξουμε μέσω της επιλογής Execute. Εκτός από την προαναφερθείσα εντολή, έχουμε την δυνατότητα να εκτελέσουμε ένα πρόγραμμα ανά βήμα-βήμα (Step).
- Το μενού TOOLS. Το μενού αυτό περιέχει εργαλεία τα οποία δεν χρησιμοποιήσαμε για την δημιουργία της πτυχιακής μας και δεν γνωρίζουμε πολλά για αυτά, όπως η επιλογή File Spliter που μας επιτρέπει να διασπάσουμε το πρόγραμμα σε περισσότερα αρχεία.
- Το μενού PROJECT. Όπως έχουμε προαναφέρει, ένα πρόγραμμα Project μπορεί να αποτελείται από πολλές τάξεις. Από αυτό το μενού μπορούμε να προσθέσουμε ή να αφαιρέσουμε τάξεις στο πρόγραμμα μας.
- Το μενού CONFIGURE. Αυτό το μενού έχει δύο επιλογές. Η επιλογή Options μας επιτρέπει να αλλάξουμε χαρακτηριστικά που σχετίζονται με την λειτουργία του J-Creator, όπως για παράδειγμα αν θα αποθηκεύει την εργασία μας κατά την έξοδο του προγράμματος. Η επιλογή Customize, επιτρέπει να γίνουν αλλαγές σε πολλά από τα χαρακτηριστικά του J-Creator όπως για παράδειγμα να εισάγουμε παραμέτρους στο προς εκτέλεση πρόγραμμα.
- Το μενού WINDOW. Περιέχει όλα τα απαραίτητα εργαλεία για να μορφοποιήσουμε τα παράθυρα στα οποία εργαζόμαστε κατά τέτοιο τρόπο ώστε να μας διευκολύνουν. Υπάρχουν επιλογές όπως να ανοίξουμε νέα παράθυρα, να κλείσουμε υπάρχοντα ή να στοιχίσουμε τα παράθυρα σε κάθετη ή οριζόντια διάταξη.
- Το μενού HELP. Όπως είναι προφανές, εδώ βρίσκονται όλα τα απαραίτητα αρχεία βοήθειας (Help Files), χρήσιμες διευθύνσεις στο Διαδίκτυο που αφορούν το πρόγραμμα καθώς και μια μηχανή αναζήτησης για να μπορούμε γρήγορα να εντοπίσουμε το πρόβλημα που μας αφορά. Ακόμα μπορούμε να δούμε την έκδοση του προγράμματος και να ελέγξουμε αν υπάρχει κάποια πιο πρόσφατη.

Κάτω από το μενού επιλογών υπάρχει η γραμμή εργαλείων (Tool Bars). Αποτελείται από μια σειρά κουμπιών τα οποία εκτελούν ενέργειες που θα μπορούσαν να γίνουν από το μενού επιλογών, αλλά για την διευκόλυνση του χρήστη, υπάρχουν εδώ. Το μενού αυτό, μπορούμε να το διαμορφώσουμε όπως επιθυμούμε ώστε να περιέχει τις ενέργειες που εκτελούμε συχνότερα. Υπάρχει ακόμα η δυνατότητα να εμφανίζουμε ή να κρύβουμε το μενού αυτό.

3.2. APACHE HTTP SERVER

Από τις πρώτες αποφάσεις που πήραμε ήταν ότι το πρόγραμμα που θα δημιουργούσαμε θα ήταν Applet. Όπως ήδη έχουμε προαναφέρει αυτό συνεπάγεται πως με την χρήση ενός Web Browser ο κάθε χρήστης θα μπορούσε να χρησιμοποιήσει το πρόγραμμα χωρίς να είναι υποχρεωμένος να το έχει στον σκληρό του δίσκο. Όμως αυτή η επιλογή μας ανάγκασε να εγκαταστήσουμε έναν δικτυακό διακομιστή (Web Server), ο οποίος περιέχει την ιστοσελίδα με το Applet. Αυτό φαίνεται και στο σχήμα 3.1.



Σχήμα 3.1. Χρήση δικτυακού διακομιστή (Web Server)

Το πρόγραμμα που χρησιμοποιήσαμε είναι το Apache HTTP Server το οποίο είναι διαθέσιμο στο Διαδίκτυο στη σελίδα http://httpd.apache.org. Το πρόγραμμα αυτό λειτουργεί σε περιβάλλον MS-DOS αλλά είναι συμβατό και με τα Windows. Για να το εγκαταστήσουμε τρέχουμε το αρχείο εγκατάστασης και ακολουθούμε τις οδηγίες που μας δίνονται. Δηλώνουμε ποιο επιθυμούμε να είναι το όνομα του διακομιστή δικτύου, καθώς και το όνομα του διαχειριστή (Administrator). Μετά την εγκατάσταση, για να ρυθμίσουμε τις ιδιότητες του διακομιστή (server), όπως σε ποια πόρτα ¨ακούει¨, ανοίγουμε το αρχείο κειμένου με τις ρυθμίσεις (configuration file) (httpd.conf). Αυτό το αρχείο φαίνεται στην εικόνα 3.2.

```
# Port: The port to which the standalone server listens.  Certain firewall
# products must be configured before Apache can listen to a specific port.
# Other running httpd servers will also interfere with this port.  Disable
# all firewall, security, and other services if you encounter problems.
# To help diagnose problems use the Windows NT command NETSTAT -a
#
Port 80

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin @

#
# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
# 127.0.0.1 is the TCP/IP local loop-back address, often named localhost. Your
# machine always knows itself by this address. If you use Apache strictly for
# local testing and development, you may use 127.0.0.1 as the server name.
#
ServerName localhost
```

Εικόνα 3.2. Αρχείο διαμόρφωσης του Apache

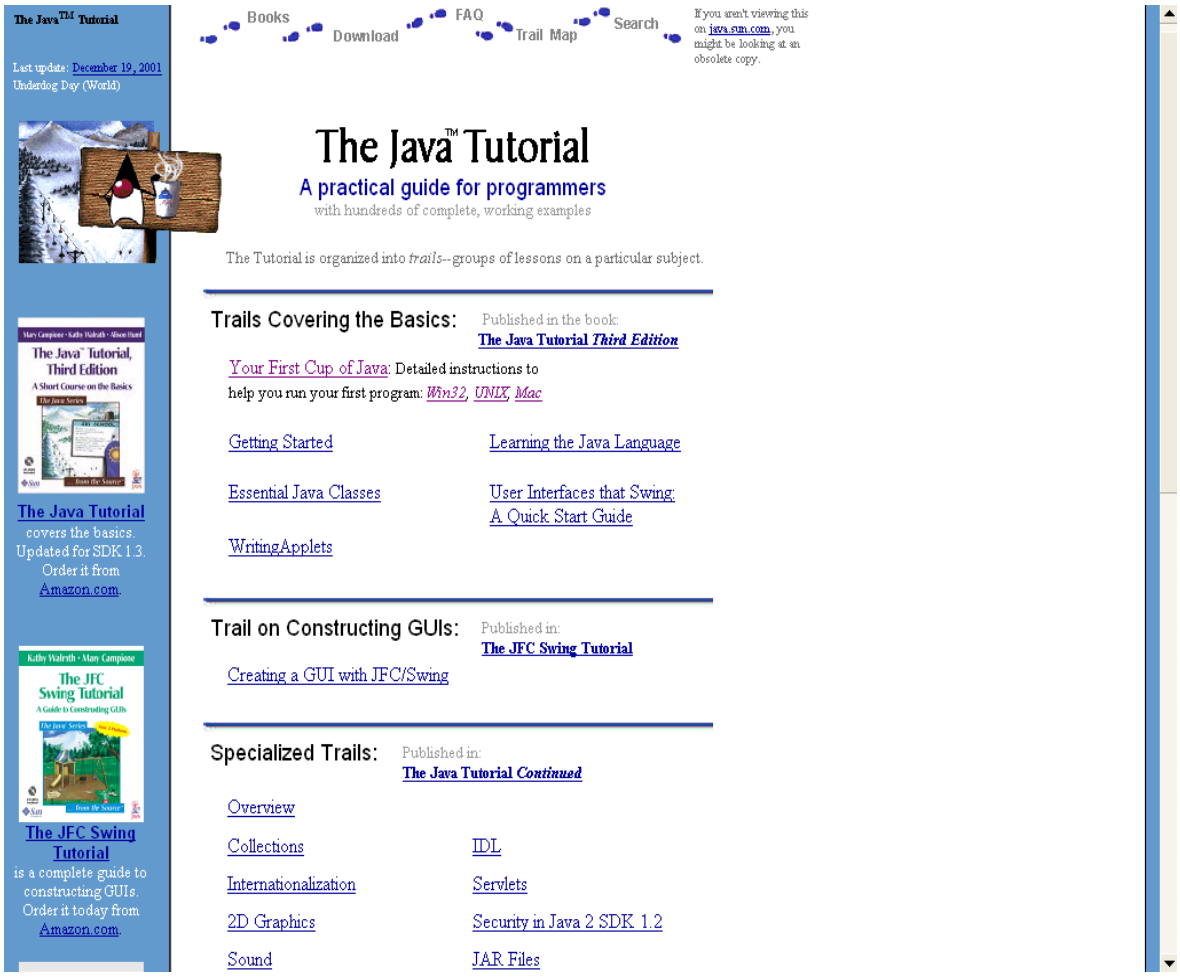
Όπως βλέπουμε το αρχείο διαμόρφωσης είναι ένα αρχείο κειμένου στο οποίο μπορούμε να καθορίσουμε τα χαρακτηριστικά του διακομιστή δικτύου. Βασικά χαρακτηρίστηκα που μπορούμε να ορίσουμε είναι η θύρα στην οποία θα ¨ακούει¨ ο διακομιστής καθώς και το όνομα αυτού (IP address).

Το applet το οποίο έχουμε κατασκευάσει το φορτώνουμε σε μια σελίδα (index.html) την οποία τοποθετούμε στο φάκελο htdocs του Apache. Έτσι πλέον όσοι συνδέονται με το διακομιστή φορτώνουν τη ιστοσελίδα αυτή και επομένως τρέχουν το applet.

3.3. ΠΗΓΕΣ ΠΛΗΡΟΦΟΡΙΩΝ

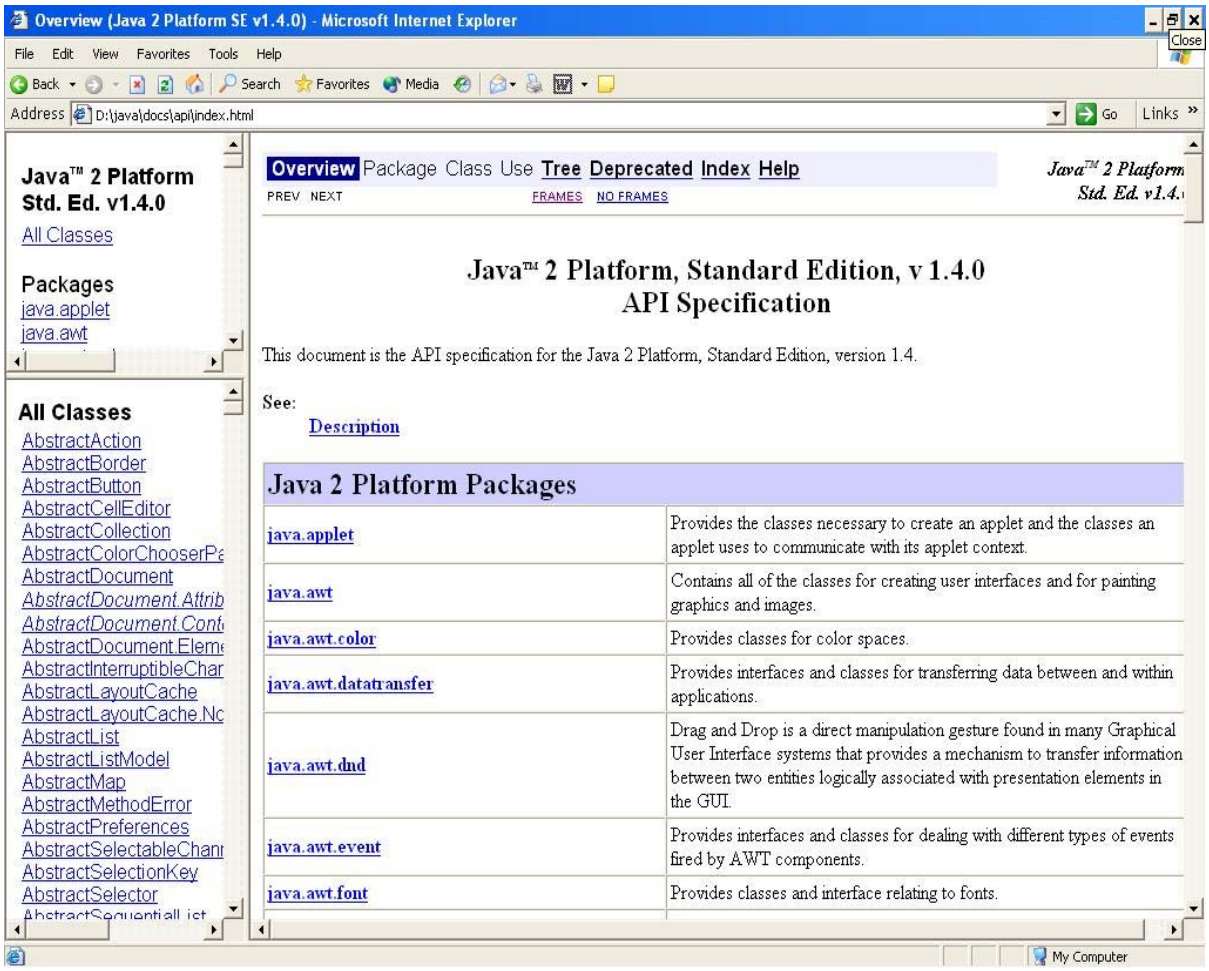
Για την JAVA στο διαδίκτυο (Internet) υπάρχει ένας απεριόριστος πλούτος πηγών. Εδώ θα αναφερθούμε μόνο στις πληροφορίες που μπορέσαμε να συλλέξουμε από την επίσημη ιστοσελίδα της SUN (εκδότρια εταιρία της JAVA). Στην ιστοσελίδα http://java.sun.com μπορούμε να βρούμε δύο πολύ χρήσιμα αρχεία σε μορφή html, το tutorial και τα documentetion files (docs). Παρ' όλο που τα παραπάνω αρχεία δεν είναι ακριβώς προγράμματα τα αναφέρουμε εδώ γιατί ήταν πολύτιμα στην προσπάθειά μας για την πραγματοποίηση αυτής της πτυχιακής.

- Το tutorial μας παρείχε τις βασικές γνώσεις γύρω από τη JAVA και περιέχει πολλά παραδείγματα κώδικα αλλά και εκτελέσιμων αρχείων που μας πρόσφεραν ιδέες για να ξεπεράσουμε διάφορα προβλήματα. Τα «μαθήματα» ξεκινάνε από βασικές γνώσεις (εντολές – σύνταξη) που οφείλει να γνωρίζει ο νέος χρήστης της JAVA και καταλήγουν σε εξειδικευμένα θέματα για έμπειρους προγραμματιστές.



Εικόνα 3.3. Τα tutorial της JAVA στο Διαδίκτυο

- Πιο σημαντικό από το tutorial ήταν το αρχείο των documentation files (docs). Σε αυτό το αρχείο περιέχονται όλες οι τάξεις καθώς και οι μέθοδοι - εντολές των βιβλιοθηκών της JAVA. Όποτε χρειαζόμασταν διευκρινήσεις ανατρέχαμε σε αυτά τα αρχεία. Για να τα δούμε τρέχουμε το αρχείο index.html και διαλέγουμε την επιλογή API & Language. Έπειτα επιλέγουμε το Java 2 Platform API Specification. Η ιστοσελίδα που ανοίγει φαίνεται στην εικόνα 3.4.



Εικόνα 3.4. Τα documentation files

Στο αριστερό κάτω παράθυρο βρίσκονται όλες οι τάξεις (classes) της JAVA ενώ στο δεξί φαίνονται για την επιλεγμένη τάξη όλες οι μέθοδοί της. Είναι δυνατόν, χρησιμοποιώντας το ευρετήριο λέξεων που έχει ο Internet Explorer, να εντοπίσουμε από τη λίστα την επιθυμητή τάξη.

4. ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ (SOURCE CODE)

Όπως έχουμε αναφέρει προηγουμένως, ένα πρόγραμμα στη JAVA μπορεί να αποτελείται από μία ή περισσότερες τάξεις που συνεργάζονται μεταξύ τους. Το πρόγραμμα το οποίο έχουμε υλοποιήσει αποτελείται από επτά τάξεις. Την λειτουργία αυτών των τάξεων θα αναλύσουμε σε αυτό το κεφάλαιο.

4.1. ΙΣΤΟΣΕΛΙΔΑ MLCLIENT

Στο τρίτο κεφάλαιο εξηγήσαμε ότι τα Applets δεν είναι αυτόνομα προγράμματα, αλλά απαιτείται η ύπαρξη ενός browser για την λειτουργία τους. Η πτυχιακή μας θέλαμε εξ' αρχής να είναι Applet έτσι ώστε να είναι φιλική προς τον χρήστη. Το κομμάτι του κώδικα που ακολουθεί είναι γραμμένο σε κώδικα ιστοσελίδας (Hyper Text Markup Language), ο οποίος φορτώνει την τάξη ClientApplet.class σε ένα παράθυρο browser με διαστάσεις 300 X 300. Επίσης εισάγουμε δύο παραμέτρους, την παράμετρο host με τιμή localhost και την παράμετρο port με τιμή 5000. Η πρώτη παράμετρος περιέχει ουσιαστικά την ip διεύθυνση του υπολογιστή που τρέχει τον διακομιστή (Server). Η δεύτερη παράμετρος περιέχει τη θύρα στην οποία ο διακομιστής «ακούει» αιτήσεις για σύνδεση από τους πελάτες (Client).

```
<html>
<head>
<title>MILY CHAT</title>
</head>
<body bgcolor="#000000" text="#000000">
<applet code="ClientApplet.class" width="300" height="300">
<PARAM NAME="host" VALUE="localhost">
<PARAM NAME="port" VALUE="5000">
</applet>
</body>
</html>
```

4.2. Τάξη ClientApplet

Η τάξη αυτή φορτώνεται όπως είδαμε από την ιστοσελίδα. Στις πρώτες έξι γραμμές του κώδικα εισάγουμε κάποιες βιβλιοθήκες της JAVA απαραίτητες για την λειτουργία του προγράμματος. Έπειτα δηλώνουμε το όνομα της τάξης μας (ClientApplet) η οποία είναι επέκταση της τάξης Japplet. Αυτό σημαίνει ότι «κληρονομεί» ιδιότητες και χαρακτηριστικά (μεθόδους) από την τάξη Japplet που ανήκει στις βιβλιοθήκες της JAVA. Η τάξη αυτή περιέχει μία μόνο μέθοδο, την init(), η οποία είναι τύπου void, δηλαδή δεν επιστρέφει καμία τιμή. Η μέθοδος αυτή αποθηκεύει τις παραμέτρους που παίρνει από το αρχείο client.html σε δύο μεταβλητές (host, port). Αυτό γίνεται στο τμήμα του κώδικα που ακολουθεί:

```
String host = getParameter( "host" );
int port = Integer.parseInt( getParameter( "port" ) );
this.getContentPane().add( "Center", new MLClient( host, port ) );
```

Η μέθοδος getParameter που χρησιμοποιούμε επιστρέφει αντικείμενα τύπου string. Έτσι με την μέθοδο parseInt( ) της τάξης Integer μετατρέπουμε το string που εισάγουμε από την παράμετρο port σε integer. Τέλος με τη μέθοδο getContentPane( ) παίρνουμε το ContentPane αυτού του αντικειμένου της ClientApplet (δηλώνεται με το this) και τοποθετούμε στο κέντρο αυτού ένα νέο αντικείμενο της τάξης client το οποίο δέχεται ως παραμέτρους τις μεταβλητές host και port. Παρακάτω ακολουθεί όλος ο κώδικας της τάξης ClientApplet.

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
import javax.swing.text.*;

public class ClientApplet extends JApplet
{
 public void init() {
  String host = getParameter( "host" );
  int port = Integer.parseInt( getParameter( "port" ) );
  this.getContentPane().add( "Center", new Client( host, port ) );
  }
}
```

4.3. ΤΑΞΗ MLClient

Η τάξη αυτή είναι η κύρια τάξη για την δημιουργία και λειτουργία του client. Με αυτή τη τάξη καθορίζουμε τον τρόπο επικοινωνίας του client με τον server, καθώς και το γραφικό περιβάλλον του client.

Στο τμήμα του κώδικα που ακολουθεί εισάγουμε τις βιβλιοθήκες της JAVA που χρησιμοποιούμε περαιτέρω στο πρόγραμμα. Οι βιβλιοθήκες αυτές σχετίζονται με την δημιουργία του γραφικού περιβάλλοντος (swing - awt) και με το δικτυακό τμήμα του προγράμματος (net – io).

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.net.*;
import java.util.*;
```

Στη συνέχεια ορίζουμε την τάξη μας (client), ως δημόσια (public), και δηλώνουμε ότι είναι επέκταση της τάξης Jpanel. Αυτό σημαίνει ότι κληρονομεί χαρακτηριστικά και ιδιότητες της τάξης αυτής. Επίσης με την εντολή (implements runnable) δηλώνουμε ότι η τάξη μας υλοποιεί το interface runnable, το οποίο παρέχει τη δυνατότητα στα αντικείμενα της τάξης client να χρησιμοποιούν νήματα (threads), υλοποιώντας την run μέθοδο του interface.

Στις επόμενες γραμμές του κώδικα δηλώνουμε τις μεταβλητές στιγμιότυπου (instance variables) που χρησιμοποιούμε στο πρόγραμμα. Τις δηλώνουμε όλες ως ιδιωτικές (private) επειδή τις χρησιμοποιούμε μόνο για την τάξη αυτή. Οι μεταβλητές που ορίζουμε είναι:

- Ένα JTextField που το ονομάζουμε tf
- Ένα JTextArea που το ονομάζουμε ta
- Μια μεταβλητή τύπου String που την ονομάζουμε nick και αποθηκεύουμε σε αυτήν την τιμή “Guest”
- Μια μεταβλητή τύπου Boolean που την ονομάζουμε nickHasBeenSet και θέτουμε σαν αρχική τιμή “false”
- Μια μεταβλητή που την ονομάζουμε dlm και είναι τύπου DefaultListModel, δηλαδή παρέχει μεθόδους για την διαχείριση μιας λίστας από αντικείμενα
- Μια μεταβλητή τύπου Socket που την ονομάζουμε socket
- Μια μεταβλητή τύπου διανύσματος (Vector) που την ονομάζουμε users
- Μια μεταβλητή τύπου DataOutputStream που την ονομάζουμε dout
- Μια μεταβλητή τύπου DataInputStream που την ονομάζουμε din
- Μια μεταβλητή τύπου JList που την ονομάζουμε list και με την οποία υλοποιούμε το γραφικό περιβάλλον της λίστας με τους χρήστες

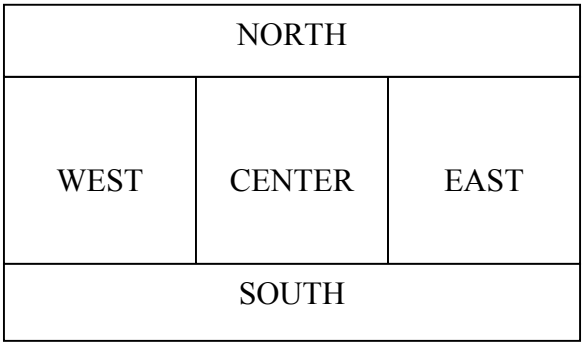
```
public class MLClient extends JPanel implements Runnable
{
  private JTextField tf = new JTextField();
  private JTextArea ta = new JTextArea();
  private String nick="Guest";
  private boolean nickHasBeenSet=false;
  private DefaultListModel dlm;
  private Socket socket;
  private Vector users;
  private DataOutputStream dout;
  private DataInputStream din;
  private JList list;
```

CONSTRUCTOR ΤΗΣ ΤΑΞΗΣ MLClient

Κατασκευαστές (constructors) ονομάζονται οι ειδικές μέθοδοι που χρησιμοποιούμε για την δημιουργία και την αρχικοποίηση νέων αντικειμένων μιας τάξης. Οι κατασκευαστές αρχικοποιούν το νέο αντικείμενο και τις μεταβλητές του, δημιουργούν άλλα αντικείμενα που το αντικείμενο μας χρειάζεται, και εκτελούν γενικά κάθε άλλη εργασία που χρειάζεται το αντικείμενο για την αρχικοποίηση του. Όπως είπαμε ο κατασκευαστής είναι

και αυτός μια μέθοδος, συνεπώς δηλώνεται και αυτός όπως κάθε μέθοδος, με τη μόνη διαφορά ότι έχει ως όνομα, το όνομα της τάξης και ότι δεν επιστρέφει καμία τιμή (return). Αντίθετα με άλλες μεθόδους δεν μπορούμε να καλέσουμε τον κατασκευαστή άμεσα. Αλλά τον καλεί η JAVA αυτόματα όταν δημιουργούμε ένα νέο αντικείμενο της τάξης του κατασκευαστή [4].

Όπως βλέπουμε ο κατασκευαστής που δημιουργήσαμε για τον client δέχεται δύο παραμέτρους. Μια παράμετρο τύπου string (host) που αντιπροσωπεύει την διεύθυνση (IP) του διακομιστή (server) και μια παράμετρο τύπου ακέραιου (int) που αντιπροσωπεύει την θύρα (port) που «ακούει» ο διακομιστής (server). Κατά την δημιουργία της τάξης client, ορίσαμε ότι επεκτείνει την τάξη της JAVA, Jpanel. Η τάξη Jpanel είναι τύπου container «κιβώτιο», δηλαδή είναι μια τάξη η οποία μπορεί να περιέχει άλλες τάξεις, της βιβλιοθήκης swing. Με την εντολή setLayout διαμορφώνουμε τον τρόπο με τον οποίο τοποθετούμε τα αντικείμενα που εισάγουμε μέσα στο container ως BorderLayout. Αυτό συνεπάγεται ότι χωρίζουμε το αντικείμενο του client που δημιουργούμε σε πέντε τμήματα, ένα κεντρικό (center) και τα υπόλοιπα τέσσερα γύρω από αυτό. Τα τμήματα αυτά τα ονομάζουμε σύμφωνα με τα σημεία του ορίζοντα όπως φαίνεται στο σχήμα 4.1.



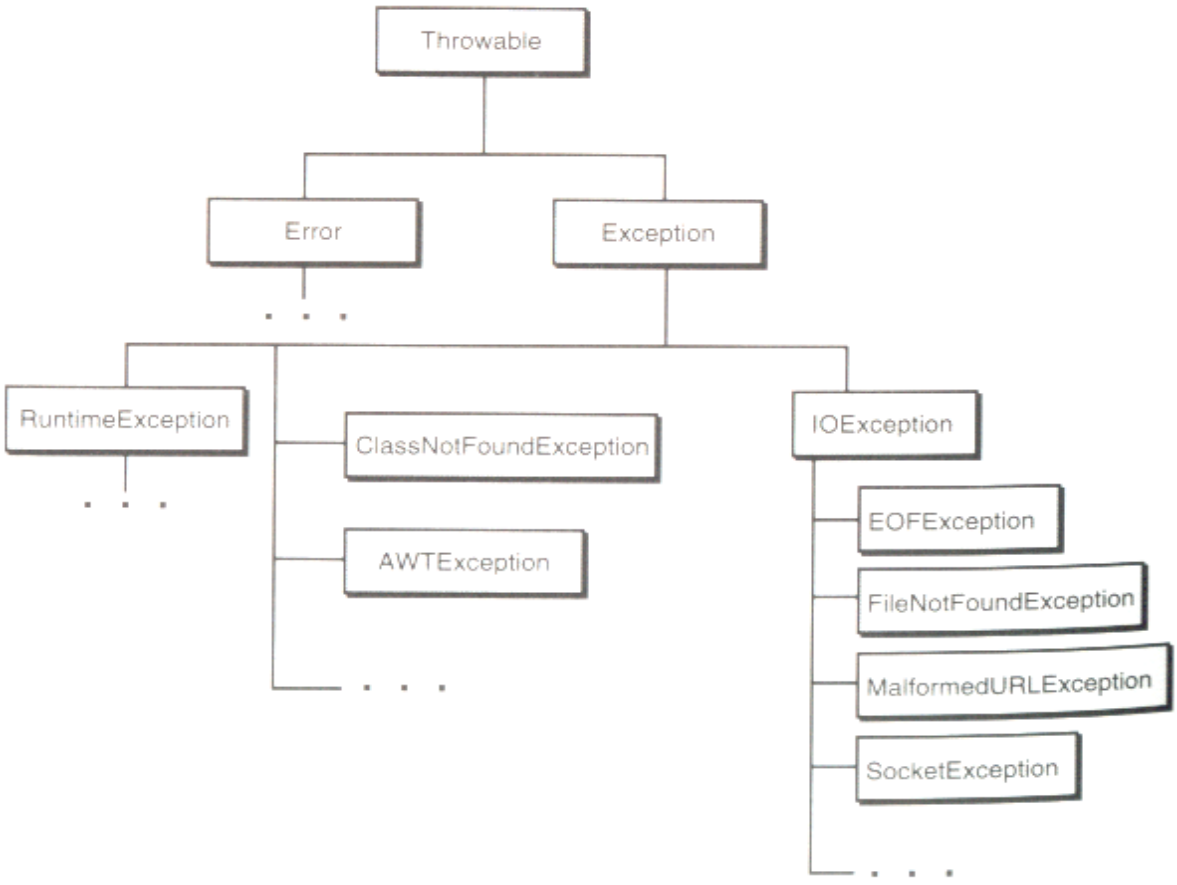
Σχήμα 4.1. Διάταξη BorderLayout [4]

Στο νότιο (South) τμήμα τοποθετούμε με την εντολή add το πεδίο κειμένου (text field) και θέτουμε ότι ο χρήστης δεν μπορεί να επέμβει στην περιοχή κειμένου (text area). Έπειτα τοποθετούμε την περιοχή κειμένου μέσα σε ένα container τύπου JScrollPane το οποίο διαθέτει οριζόντιες και κατακόρυφες μπάρες ολίσθησης και ορίζουμε πότε και αν θα εμφανίζονται αυτές. Το ScrollPane αυτό το τοποθετούμε στο κεντρικό τμήμα του client. Στο ανατολικό τμήμα του client τοποθετούμε ένα νέο JScrollPane στο οποίο ως όρισμα έχουμε θέσει μια λίστα (Jlist) η οποία περιέχει τη μεταβλητή dlm που είναι τύπου DefaultListModel δηλαδή υλοποιεί ένα διάνυσμα (vector) στο οποίο τοποθετούμε την τιμή All Users.

Με αυτό το κομμάτι κώδικα ολοκληρώσαμε το γραφικό τμήμα του client. Εμείς όμως ως χρήστες δεν έχουμε την δυνατότητα να αλληλεπιδράσουμε με το γραφικό αυτό περιβάλλον. Αυτή τη δυνατότητα μας την παρέχει η μέθοδος addActionListener με την οποία τοποθετούμε ένα ActionListener interface στο JTextField (tf), με συνέπεια ότι κάθε φορά που πατάμε το πλήκτρο enter η μέθοδος actionPerformed να εκτελείται. Η οποία στην περίπτωση μας καλεί την μέθοδο ProcessMessage με όρισμα το κείμενο που έχουμε γράψει στο JTextField (tf). Πριν προχωρήσουμε παρακάτω πρέπει να αναφερθούμε αναλυτικότερα σε δυο βασικές έννοιες της JAVA:

ΕΛΕΓΧΟΣ ΛΑΘΩΝ

Οι προγραμματιστές, σε κάθε γλώσσα προγραμματισμού προσπαθούν να γράψουν προγράμματα χωρίς λάθη (bugs). Αυτό όμως στην πραγματικότητα είναι ουτοπικό, λάθη συμβαίνουν στα προγράμματα είτε επειδή ο προγραμματιστής δεν πρόβλεψε κάθε πιθανή κατάσταση, είτε επειδή συμβαίνουν καταστάσεις έξω από τον έλεγχο του προγράμματος, όπως αλλοιωμένα δεδομένα από τον χρήστη. Τα λάθη στην JAVA είναι όλα αντικείμενα τάξεων που κληρωνόμουν ιδιότητες από την τάξη throwable. Ένα αντικείμενο της τάξης αυτής δημιουργείται όταν ένα λάθος συμβαίνει [1]. Η ιεραρχία της τάξης throwable και των υποτάξεων της (Subclasses) φαίνεται στο σχήμα 4.2.



Σχήμα 4.2. Ιεραρχία της τάξης Throwable

Όπως βλέπουμε τα λάθη που μπορεί να εμφανιστούν στη JAVA χωρίζονται σε δυο κύριες κατηγορίες, στα σφάλματα (Errors) και στις εξαιρέσεις (Exceptions). Στη κατηγορία Error ανήκουν τα λάθη που οφείλονται σε δυσλειτουργίες της εικονικής μηχανής της JAVA. Τα λάθη αυτά είναι σπάνια και δεν θα μας απασχολήσουν. Στην κατηγορία Exception ανήκουν τα λάθη τα οποία αναφέραμε παραπάνω.

Ο έλεγχος ενός τμήματος κώδικα στη JAVA για πιθανά λάθη γίνεται με τις εντολές try και catch όπως φαίνεται στο ακόλουθο παράδειγμα.

```
try {

      socket = new Socket( host, port );

        din = new DataInputStream( socket.getInputStream() );
      dout = new DataOutputStream( socket.getOutputStream() );

      new Thread( this ).start();
    } catch( IOException ie ) {          p( ie.toString() ); }
```

Το τμήμα του κώδικα που βρίσκεται ανάμεσα στις εντολές try και catch το ελέγχουμε για πιθανό λάθος τύπου IOException όπως δηλώνουμε με την εντολή catch. Αν δεν υπάρξει τέτοιο λάθος το πρόγραμμα συνεχίζει να εκτελείται κανονικά διαφορετικά τυπώνουμε το λάθος που εμφανίσθηκε.

ΥΠΟΔΟΧΕΣ (SOCKETS)

Για δικτυακές εφαρμογές, δυο από τις πιο χρήσιμες τάξεις που προσφέρει η JAVA είναι η Socket και η ServerSocket. Οι τάξεις αυτές μας προσφέρουν τις απαραίτητες μεθόδους για την επικοινωνία ηλεκτρονικών υπολογιστών μέσω δικτύου. Η τάξη Socket δέχεται δυο παραμέτρους, την IP διεύθυνση του υπολογιστή με τον οποίο γίνεται η σύνδεση, καθώς και την θύρα επικοινωνίας. Ενώ η τάξη ServerSocket δέχεται ως παράμετρο, μόνο την θύρα στην οποία «ακούει» για πιθανές αιτήσεις προς σύνδεση. Αυτό θα το δούμε αναλυτικότερα όταν θα εξηγήσουμε την λειτουργία του προγράμματός μας. Από’ κει και πέρα η επικοινωνία γίνεται με την χρήση των τάξεων DataInputStream και DataOutputStream[1],[4].

Παρακάτω ακολουθεί το τμήμα κώδικα που έχουμε αναλύσει ως τώρα. Μέσα στο τμήμα try-catch ελέγχουμε για IOExceptions. Στο κομμάτι αυτό του κώδικα δημιουργούμε ένα αντικείμενο της τάξης Socket με παραμέτρους την IP διεύθυνση του Server καθώς και την θύρα επικοινωνίας. Επίσης δηλώνουμε δυο μεταβλητές (din - dout) για τα Streams (Input – Output) που θα χρησιμοποιήσουμε. Τέλος με την μέθοδο start «τρέχουμε» την μέθοδο run ( ) του προγράμματος, δηλαδή ουσιαστικά ξεκινούμε ένα νήμα (thread), το οποίο διαχειρίζεται τα μηνύματα που δέχεται ο Client από τον Server.

```
public MLClient( String host, int port ) {

      setLayout( new BorderLayout() );
      add( "South", tf );
      ta.setEditable(false);

      JScrollPane scrollPane = new JScrollPane(ta);

      scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_S
CROLLBAR_AS_NEEDED);

      scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROL
LBAR_ALWAYS);

      add( "Center", scrollPane );

      dlm=new DefaultListModel();
      dlm.addElement("All Users");

      list=new JList(dlm);
      JScrollPane listScrollPane = new JScrollPane(list);

      listScrollPane
.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEED
ED);
      listScrollPane
.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

      list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);


      add( "East", listScrollPane );

    tf.addActionListener( new ActionListener() {
      public void actionPerformed( ActionEvent e ) {
        processMessage( e.getActionCommand() );
      }
    } );

    try {

      socket = new Socket( host, port );

        din = new DataInputStream( socket.getInputStream() );
      dout = new DataOutputStream( socket.getOutputStream() );

      new Thread( this ).start();
    } catch( IOException ie ) {           p( ie.toString() ); }
  }
```

ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ – ΔΗΜΙΟΥΡΓΙΑ ΥΠΟΔΟΧΗΣ (SOCKET)

Όμως πριν να δούμε το τμήμα αυτό θα εξετάσουμε τα μηνύματα που στέλνει ο Client στον Server. Δηλαδή θα δούμε την μέθοδο processMessage που αναφέραμε παραπάνω.

Η ΜΕΘΟΔΟΣ processMessage ΤΟΥ MLClient

Όπως είδαμε όταν γράψουμε κάτι στο TextField (tf) και πατήσουμε το πλήκτρο enter, τότε με το interface ActionListener καλούμε τη μέθοδο ProcessMessage που έχει ως παράμετρο το κείμενο (String) που είχαμε πληκτρολογήσει στο JTextField. Η processMessage διαμορφώνει το μήνυμα που θα στείλει ο Client στο Server.

Ορίζουμε ότι η μέθοδος processMessage είναι ιδιωτική (private) και τύπου (void), δηλαδή δεν επιστρέφει τίποτα. Όλο τον κώδικα της μεθόδου, τον ελέγχουμε με τη χρήση των εντολών try-catch, για IOExceptions. Εάν το κείμενο (Message) που γράψαμε στο JTextField ξεκινάει με το σύμβολο κάγκελο (#), δηλαδή όταν θέλουμε να ορίσουμε το «ψευδώνυμο» μας (nick), τότε υπάρχουν δύο περιπτώσεις. Εάν το διάφορο της μεταβλητής NickHasBeenSet είναι αληθές (True), δηλαδή δεν υπάρχει άλλος χρήστης με αυτό το όνομα, τότε αποθηκεύουμε το μήνυμα στη μεταβλητή message, αφού αφαιρέσουμε το κάγκελο. Στη συνέχεια στέλνουμε το μήνυμα, « "<nick> "+message », στη DataOutputStream (dout), με τη μέθοδο WriteUTF. Αλλιώς τυπώνουμε στο JtextArea ένα μήνυμα που μας ειδοποιεί ότι δεν μπορούμε να αλλάξουμε το όνομα μας (nick), επειδή κάποιος άλλος χρήστης το χρησιμοποιεί.

Διαφορετικά εάν το μήνυμα ξεκινάει με το «παπάκι» (@), δηλαδή όταν το μήνυμα απευθύνεται σε ένα συγκεκριμένο χρήστη, το όνομα του οποίου βρίσκεται ακριβώς μετά το παπάκι, αρχικοποιούμε τις μεταβλητές toUser – msg, θέτοντας τις τιμές τους ίσες με το κενό. Στη συνέχεια με τη μέθοδο StringTokenizer χωρίζουμε το Message σε δυο τμήματα. Το πρώτο τμήμα, που είναι το όνομα του χρήστη στον οποίο στέλνουμε το μήνυμα, το αποθηκεύουμε στη μεταβλητή toUser, ενώ το δεύτερο, που είναι το μήνυμα αυτό, το αποθηκεύουμε στη μεταβλητή msg. Στη συνέχεια στέλνουμε το μήνυμα, « "<MSG "+toUser+" "+nick+">"+msg », στη DataOutputStream (dout), με τη μέθοδο WriteUTF. Η μεταβλητή nick περιέχει το όνομα του χρήστη που στέλνει το μήνυμα.

Για να στείλουμε μήνυμα σε ένα χρήστη, αντί να χρησιμοποιήσουμε τη μέθοδο με το παπάκι, μπορούμε ακόμα να επιλέξουμε τον χρήστη μέσα από τη λίστα που εμφανίζεται στο δεξιό μέρος του client. Εάν από την λίστα έχουμε επιλέξει το AllUsers, τότε στέλνουμε στη DataOutputStream με τη μέθοδο WriteUTF το μήνυμα « "<toAll "+nick+">"+message ». Διαφορετικά αν κάποιος χρήστης είναι επιλεγμένος στέλνουμε το μήνυμα « "<MSG "+toUser+" "+nick+">"+message ». Αν τίποτα από τα παραπάνω δεν συμβαίνει στέλνουμε το μήνυμα « "<toAll "+nick+">"+message ». Στο τέλος θέτουμε τη τιμή του JTextField ίση με το κενό, ώστε ο χρήστης να μπορεί να γράψει ένα νέο μήνυμα. Για να γίνουν τα παραπάνω πιο κατανοητά, ας υποθέσουμε ότι ο χρήστης Κώστας επιθυμεί να στείλει στον Γιάννη το μήνυμα «Γεια σου Γιάννη». Το μήνυμα που σταλεί από τον MLClient είναι το εξής: "<MSG "+toUser+" "+nick+">"+message όπου στην μεταβλητή toUser είναι αποθηκευμένο το όνομα του Γιάννη, στην μεταβλητή nick το όνομα του Κώστα, ενώ στην message το μήνυμα Γεια σου Γιάννη.

Παρακάτω ακολουθεί το τμήμα κώδικα που αναλύσαμε πιο πάνω.

```
  private void processMessage( String message ) {
    try {
            //Send To Server
      if (message.trim().startsWith("#")) {
            if (!nickHasBeenSet) {
                  message=message.substring(1);
                  dout.writeUTF("<nick> "+message );

            }
            else {
                  ta.append("\nYou cannot change your nickname\n");

            }
      }
      else if (message.trim().startsWith("@")) {
                  String toUser="",msg="";
                  message=message.substring(1);
            StringTokenizer strtok=new StringTokenizer(message);
                  if (strtok.hasMoreTokens()) {
                        toUser=strtok.nextToken();
                        while (strtok.hasMoreTokens())
msg+=strtok.nextToken()+" ";
                        dout.writeUTF( "<MSG "+toUser+" "+nick+">"+msg
);
                  }
      }
      else if (list.getSelectedValue()!=null){
      String toUser=(String)list.getSelectedValue();

            if (toUser.equals("All Users")){
                dout.writeUTF( "<toAll "+nick+">"+message );
            }
            else {
                  dout.writeUTF( "<MSG "+toUser+" "+nick+">"+message
);
            }
      }
      else dout.writeUTF( "<toAll "+nick+">"+message );

      tf.setText("");
    } catch( IOException ie ) {
      p( ie.toString() );
      p(ie.getMessage());
    }
  }
```

ΜΗΝΥΜΑΤΑ ΠΡΟΣ ΔΙΑΚΟΜΙΣΤΗ (SERVER)

ΜΕΘΟΔΟΣ run ΝΗΜΑ (THREAD) ΤΗΣ ΤΑΞΗΣ MLClient

Όπως και στην processMessage έτσι και εδώ ελέγχουμε με τις εντολές try και catch για IOExceptions. Το κομμάτι αυτό του κώδικα, διαβάζει τα μηνύματα που δέχεται ο Client από τον Server. Επειδή αυτή είναι μια διαδικασία που γίνεται συνεχώς και παράλληλα με άλλες εργασίες του Client, το τμήμα αυτό είναι υλοποιημένο σαν νήμα (Thread).

Όλος ο κώδικας είναι γραμμένος μέσα σε ένα ατέρμονα βρόχο (Loop) με τη χρήση της εντολής while. Με την μέθοδο readUTF, ο Client δέχεται τα μηνύματα από τον Server και τα αποθηκεύει στη μεταβλητή str. Για την υλοποίηση αυτού του τμήματος κώδικα χρησιμοποιούμε μεθόδους τις οποίες έχουμε υλοποιήσει στη τάξη Message. Την τάξη αυτή την αναλύουμε αφού έχουμε τελειώσει την αναφορά μας στον Client. Σας προτείνουμε, πριν διαβάσετε τα παρακάτω και για την καλύτερη κατανόηση τους, να δείτε την τάξη αυτή. Γενικά μπορούμε να πούμε ότι με τις μεθόδους που περιέχονται στη τάξη Message, μπορούμε να διαμορφώσουμε μηνύματα τύπου String.

Όπως είπαμε τα μηνύματα που στέλνει ο Server στον Client τα αποθηκεύουμε στην μεταβλητή str. Δημιουργούμε ένα νέο αντικείμενο της τάξης Message, και το ονομάζουμε message, και έχει σαν όρισμα την μεταβλητή str, δηλαδή το μήνυμα που μας έστειλε ο Server. Εάν η μέθοδος name( ) του αντικειμένου message που δημιουργήσαμε επιστρέφει την τιμή OK τότε θέτουμε την μεταβλητή nickHasBeenSet ως αληθής και αποθηκεύουμε στην μεταβλητή nick την τιμή που επιστρέφει η μέθοδος value( ) της Message που στην συγκεκριμένη περίπτωση είναι το όνομα του χρήστη που προστέθηκε. Διαφορετικά αν η τιμή που επιστρέφει η μέθοδος value είναι ίση με « USER_EXISTS » τότε τυπώνουμε ένα μήνυμα που ειδοποιεί τον χρήστη ότι κάποιος άλλος έχει ήδη επιλέξει αυτό το όνομα. Εάν η name ( ) επιστρέφει την τιμή « "NEW_USER" » και το nick υπάρχει τότε δεν κάνει τίποτα, αλλιώς το προσθέτουμε στην λίστα με την μέθοδο addElement( ). Αν η τιμή που επιστρέφει η μέθοδος value ( ) είναι ίση με « "REMOVE_USER" » τότε αφαιρούμε τον χρήστη από την λίστα. Εάν η τιμή που επιστρέφει η μέθοδος value ( ) είναι ίση με « "USERS" », που αυτό συμβαίνει όταν πρωτοσυνδέεται ο Client στον Server, τότε ο Client παίρνει την λίστα με τους ήδη συνδεμένους χρήστες. Αυτό γίνεται με την χρήση της τάξης StringTokenizer. Διαφορετικά, σε οποιαδήποτε άλλη περίπτωση τυπώνουμε το όνομα του χρήστη και το μήνυμα που έστειλε.

Τέλος έχουμε δημιουργήσει μια μέθοδο που την ονομάσαμε p ( ) η οποία τυπώνει τα μηνύματα που δέχεται σαν παράμετρο στο JTextArea. Αυτό το κάναμε γιατί είναι κάτι που χρησιμοποιούσαμε συχνά και με την χρήση της μεθόδου αυτής γλιτώνουμε αρκετές γραμμές κώδικα.

Παρακάτω ακολουθεί το τμήμα του κώδικα που περιγράψαμε αναλυτικά πιο πάνω.

```
public void run() {
    try {

      while (true) {
            //Read From Server
        String str = din.readUTF();
            Message message=new Message(str);
            if (message.msgType().equals("OK")) {
                  p("User "+message.value(1) +" has been added");
                  nickHasBeenSet=true;
                  nick=message.value(1);
            }
            else if (message.msgType().equals("USER_EXISTS")) {
                  p("User "+message.value(1) +" exists already");
            }else if (message.msgType().equals("NEW_USER")) {
                  if (nickHasBeenSet &&
nick.equals(message.value(1))){
                  }
                  else {
                        p("New User ("+message.value(1) +") has
joinned the chat");
                        dlm.addElement(message.value(1));
                  }
            }else if (message.msgType().equals("REMOVE_USER")) {
                        dlm.removeElement(message.value(1));
            }else if (message.msgType().equals("USERS")) {
                        StringTokenizer strtok=new
StringTokenizer(message.body());
                        while (strtok.hasMoreTokens()) {
                              dlm.addElement(strtok.nextToken());
                        }
            }else ta.append(message.value(1)+": "+message.body()+"\n"
);
      }
    } catch( IOException ie ) {
      p( ie.toString() );
    }
  }

      public void p(String str) {
            ta.append(str+"\n");
      }

}
```

ΜΗΝΥΜΑΤΑ ΠΟΥ ΔΕΧΕΤΑΙ ΑΠΟ ΤΟΝ ΔΙΑΚΟΜΙΣΤΗ (SERVER)

4.4. ΤΑΞΗ Message

Σε αυτήν την τάξη αναφερθήκαμε όταν εξετάσαμε την λειτουργία του Client. Η τάξη αυτή παρέχει μεθόδους που σκοπός τους είναι η επεξεργασία μηνυμάτων. Ο Client με τον Server επικοινωνούν με κωδικοποιημένα μηνύματα που μπορούν να περιέχουν μια επικεφαλίδα (Header), το όνομα του παραλήπτη, το όνομα του αποστολέα και φυσικά το ίδιο το μήνυμα. Η τάξη Message δέχεται ως παράμετρο το μήνυμα αυτό, και μας δίνει την δυνατότητα χρησιμοποιώντας τις μεθόδους της να επεξεργαστούμε τα μηνύματα αυτά.

Όπως είδαμε και στις προηγούμενες τάξεις που εξετάσαμε, αρχικά εισάγουμε τις βιβλιοθήκες της JAVA, των οποίων τις μεθόδους σκοπεύουμε να χρησιμοποιήσουμε. Την τάξη Message την δηλώσαμε ως δημόσια (public) επειδή τις μεθόδους της τις χρησιμοποιούμε και σε άλλα αντικείμενα τάξεων, σαν του Client όπως είδαμε. Έπειτα δημιουργούμε ένα πίνακα. Τον πίνακα αυτό τον ονομάζουμε values και δηλώνουμε ότι είναι πέντε θέσεων, και δέχεται δεδομένα κειμένου (string). Επίσης δηλώνουμε μια μεταβλητή τύπου String, που την ονομάζουμε body και θα περιέχει το μήνυμα του αποστολέα. Τα δύο παραπάνω στοιχεία που ορίσαμε, τα θέσαμε ως ιδιωτικά (private), γιατί χρησιμοποιούντε αποκλειστικά από αντικείμενα της τάξης αυτής.

Στη συνέχεια ακολουθεί ο Constructor, όπου διασπάμε το μήνυμα (string), που δέχεται ο Constructor σαν παράμετρο. Αρχικά αποθηκεύουμε την θέση του συμβόλου (<) ως ακέραιο στη μεταβλητή start και αντίστοιχα τη θέση του συμβόλου (>) στη μεταβλητή end. Αυτό το πετυχαίνουμε χρησιμοποιώντας τη μέθοδο της τάξης String indexOf. Τη διαδικασία αυτή την ακολουθούμε για να διαχωρίσουμε την επικεφαλίδα αναγνώρισης που έχουμε τοποθετήσει σε κάθε μήνυμα και η οποία βρίσκεται ανάμεσα στα δύο παραπάνω σύμβολα, από το κυρίως σώμα του μηνύματος. Για να ελέγξουμε αν το μήνυμα που εξετάζουμε είναι σωστό ή έχει αλλοιωθεί κατά την μεταφορά μέσω του δικτύου, χρησιμοποιούμε την συνθήκη if που ελέγχει αν η θέση του συμβόλου (>) είναι μεγαλύτερη από την θέση του συμβόλου (<), και αν η θέση του συμβόλου (<) είναι μεγαλύτερη από το μείον ένα. Αν οι δύο αυτές συνθήκες, είναι αληθείς, θεωρούμε ότι το μήνυμα δεν έχει αλλοιωθεί. Έπειτα αποθηκεύουμε στη μεταβλητή header την επικεφαλίδα αναγνώρισης του μηνύματος, χρησιμοποιώντας την μέθοδο indexOf, της τάξης String. Με την ίδια διαδικασία, αποθηκεύουμε το καθ' αυτό μήνυμα, στη μεταβλητή body. Με αυτό τον τρόπο επιτύχαμε να χωρίσουμε την επικεφαλίδα (header) από το σώμα του μηνύματος. Πρέπει όμως να διασπάσουμε και τις πληροφορίες που περιέχονται στην επικεφαλίδα (header). Για να γίνει αυτό ορίζουμε ένα νέο αντικείμενο της τάξης StringTokenizer, στο οποίο δίνουμε ως παράμετρο την header. Η τάξη αυτή διασπάει τη επικεφαλίδα (header) στις λέξεις από τις οποίες αποτελείται, τις οποίες με την χρήση ενός while loop τις αποθηκεύουμε χωριστά στον πίνακα values. Αν η συνθήκη if που ελέγξαμε δεν είναι αληθής, τότε αποθηκεύουμε τα μηνύματα "error" και "Malformed Header", στις δυο πρώτες θέσεις του πίνακα και το κενό string στις υπόλοιπες τρεις.

Εκτός από τον Constructor, η τάξη αυτή περιέχει τρεις μεθόδους:

- Η μέθοδος name ( ) η οποία επιστρέφει την πρώτη θέση του πίνακα value, δηλαδή το κωδικό όνομα που χρησιμοποιούμε για να διαχωρίσουμε τα διαφορετικά μηνύματα μεταξύ τους. Τέτοια για παράδειγμα είναι (NEW_USER, REMOVE_USER κτλ).
- Η μέθοδος body ( ) η οποία επιστρέφει την μεταβλητή body που περιέχει το κυρίως μήνυμα.
- Η μέθοδος value ( ) η οποία επιστρέφει τα περιεχόμενα του πίνακα από τις θέσεις ένα έως τέσσερα, ανάλογα με την παράμετρο που του δίνουμε. Αν αυτή η παράμετρος είναι μεγαλύτερη από το τέσσερα, τότε επιστρέφει κενό.

```
import java.io.*;
import java.util.*;

public class Message {
      private String[] values= new String[5];
      private String body;
      public Message(String msg) {
            int start=msg.indexOf("<");
            int end=msg.indexOf(">");
            if ((end>start)&&(start>-1)) {
                  String header = msg.substring(start+1,end);
                  body=msg.substring(end+1);
                  StringTokenizer tok=new StringTokenizer(header);
                  int i=0;
                  while (tok.hasMoreTokens()) {
                        values[i]=tok.nextToken().trim();
                        i++;
                  }
                  for (;i<5;i++){
                        values[i]="";
                  }
            }
            else {
                  values[0]="error";
                  values[1]="Malformed Header";
                  values[2]=values[3]=values[4]="";
            }
      }
      public String name() {
            return(values[0]);
      }
      public String body() {
            return(body);
      }
      public String value(int i) {
            if ((i>0)&&(i<5)) {
                  return (values[i]);
            }
            else {
                  return ("");
            }
      }
}
```

ΚΩΔΙΚΑΣ ΤΗΣ ΤΑΞΗΣ Message

4.5. ΤΑΞΗ GUIServer

Μέχρι τώρα εξηγήσαμε την λειτουργία των τάξεων που αφορούν τον πελάτη (Client) καθώς και της βοηθητικής τάξης Message που την χρησιμοποιούμε για την επεξεργασία των μηνυμάτων που ανταλλάζει ο Client με τον Server. Σε αυτό το κεφάλαιο θα ασχοληθούμε με την λειτουργία των τάξεων που αφορούν τον διακομιστή (Server) και πιο συγκεκριμένα την τάξη GUIServer. Επειδή η τάξη αυτή είναι η μεγαλύτερη που γράψαμε, θα την αναλύσουμε χωρίζοντας την σε μικρότερα τμήματα.

ΟΡΙΣΜΟΣ ΤΑΞΗΣ – ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Όπως σε όλες τις προηγούμενες τάξεις που έχουμε εξετάσει ως τώρα, αρχικά εισάγουμε τις βιβλιοθήκες της JAVA που χρησιμοποιούμε. Η τάξη GUIServer ορίζουμε ότι είναι υποτάξη της τάξης JFrame και ότι υλοποιεί το interface runnable. Στο interface runnable έχουμε αναφερθεί προηγουμένως, ενώ τα αντικείμενα της τάξης JFrame είναι παράθυρα με έτοιμες κάποιες βασικές λειτουργίες τους, όπως η μεγιστοποίηση ή το κλείσιμο. Τέλος σε αυτό το κομμάτι του κώδικα, δηλώνουμε τις μεταβλητές που χρησιμοποιούμε.

- Μια μεταβλητή τύπου JFrame που την ονομάζουμε frame.
- Μια μεταβλητή τύπου ServerSocket που την ονομάζουμε ss.
- Ένα πίνακα τύπου Hashtable που τον ονομάζουμε outputstreams
- Ένα JTextArea που το ονομάζουμε ta.
- Μια μεταβλητή τύπου ακεραίου που την ονομάζουμε port και αποθηκεύει την θύρα επικοινωνίας.
- Μια μεταβλητή τύπου ακεραίου που την ονομάζουμε count
- Μια μεταβλητή τύπου JDialog που την ονομάζουμε dialog.
- Ένα πίνακα τύπου Hashtable που τον ονομάζουμε users.

```
import java.util.*;
import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;

public class GUIServer extends JFrame implements Runnable {

      private static JFrame frame;
      private ServerSocket ss;
      private Hashtable outputStreams = new Hashtable();
      private JTextArea ta=new JTextArea();
      private static int port;
      private static int count;
      private static JDialog dialog;
      private static Hashtable users=new Hashtable();
```

CONSTRUCTOR ΤΗΣ ΤΑΞΗΣ GUIServer

Ο constructor είναι υπεύθυνος για την δημιουργία του γραφικού περιβάλλοντος του Server. Στην JAVA και πιο συγκεκριμένα στη βιβλιοθήκη Swing, υπάρχουν τρεις δυνατές μορφές στην εμφάνιση των στοιχείων (π.χ. κουμπιά) του γραφικού περιβάλλοντος. Εμείς επιλέξαμε το λεγόμενο «μεταλλικό προφίλ» (Metal Look And Feel). Με την εντολή super(¨Server¨) αναφερόμαστε στον Constructor της JFrame, και ονομάζουμε το παράθυρο μας Server.

Με την μέθοδο setsize, ορίζουμε ότι το παράθυρο μας έχει μέγεθος (400 x 500) εικονοστοιχεία (pixels). Έπειτα δημιουργούμε ένα νέο αντικείμενο της τάξης Jpanel που το ονομάζουμε contentPane και θέτουμε ότι έχει διαμόρφωση τύπου BorderLayout. Με παρόμοιο τρόπο όπως και στον Constructor του Client καθορίζουμε το πλαίσιο του contentPane και το θέτουμε ως το contentPane αυτής της σελίδας. Ακόμα καθορίζουμε τα χαρακτηριστικά του TextArea, όπως για παράδειγμα αν μπορούμε ή όχι να γράψουμε σε αυτό, και το εισάγουμε ως όρισμα σε ένα JscrollPane. Αυτό το JscrollPane το τοποθετούμε στο κέντρο του contentPane. Τέλος, καλούμε την μέθοδο run για το αντικείμενο της τάξης που δημιουργήσαμε.

```
public GUIServer () throws IOException {

            super("Server");
            try {

UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
          } catch (Exception e) {
            System.err.println("There was an error during the loading
            of Look and Feel:\n"+e);
            System.exit(0);
        }
            setSize(new Dimension(500,400));
            JPanel contentPane = new JPanel(new BorderLayout());
      contentPane.setBorder(BorderFactory.createEmptyBorder(2,2,2,2));
            setContentPane(contentPane);
            ta.setFont(new Font("Courier",Font.PLAIN,12));
            ta.setLineWrap(true);
            ta.setWrapStyleWord(true);
            ta.setEditable(false);
            JScrollPane taScrollPane = new JScrollPane(ta);
      taScrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCR
OLLBAR_ALWAYS);
            taScrollPane.setMinimumSize(new Dimension(10, 10));

            taScrollPane.setBorder(BorderFactory.createCompoundBorder(
BorderFactory.createEmptyBorder(5,5,5,5),taScrollPane.getBorder()));


            contentPane.add(taScrollPane,BorderLayout.CENTER);

            new Thread(this).start();
}
```

ΚΩΔΙΚΑΣ CONSTRUCTOR ΤΗΣ ΤΑΞΗΣ GUIServer

ΜΕΘΟΔΟΣ run ΤΗΣ ΤΑΞΗΣ GUIServer

Στο κομμάτι του κώδικα που θα εξετάσουμε, υλοποιούμε ένα από τα πιο σημαντικά τμήματα του Server. Η λογική στην οποία βασίζεται ο Server είναι ότι συνεχώς πρέπει να «ακούει» σε μια θύρα για πιθανές αιτήσεις από Clients και να δημιουργεί ξεχωριστές συνδέσεις με αυτούς, οι οποίες διαχειρίζονται από την τάξη ServerThread. Για αυτόν ακριβώς τον λόγο επειδή απαιτείται αυτή η διαδικασία να γίνεται συνεχώς παράλληλα με άλλες λειτουργίες του Server, το κομμάτι αυτό είναι υλοποιημένο ως νήμα (thread). Αρχικά δημιουργούμε ένα νέο αντικείμενο της τάξης ServerSocket που δέχεται ως παράμετρο την θύρα και το ονομάζουμε ss. Έπειτα, χάρη σε ένα ατέρμονα βρόχο (loop), που υλοποιείται με την εντολή while και με την χρήση της μεθόδου accept ( ), δημιουργούμε μια υποδοχή (socket) επικοινωνίας για κάθε Client καθώς και ένα αντικείμενο DataOutputStream, που το ονομάζουμε dout. Τα δυο αυτά αντικείμενα τα αποθηκεύουμε στον Hashtable, OutputStreams, και έπειτα δημιουργούμε ένα νέο αντικείμενο της τάξης ServerThread, με παραμέτρους το αντικείμενο του GUIServer (this) που δημιουργήσαμε, καθώς και το Socket επικοινωνίας. Αν η θύρα στην οποία τρέχουμε τον Server είναι κατειλημμένη από κάποιο άλλο πρόγραμμα, τότε εμφανίζεται λάθος IOException, οπότε όπως φαίνεται στον κώδικα που ακολουθεί την εντολή catch, αυξάνουμε την τιμή της θύρας κατά ένα και ξανατρέχουμε το νήμα.

```
public void run(){
        try {
                ss = new ServerSocket( port );
                p("port: "+port);
                p( "Listening on "+ss );

                   while (true) {

                    Socket s = ss.accept();
                    p( "Connection from "+s );
                   DataOutputStream dout = new DataOutputStream(
                   s.getOutputStream() );
                   outputStreams.put( s, dout );
                   new ServerThread( this, s );
                  }
        }
        catch (IOException ex){
                p(ex.getMessage());
                port++;
                p("Trying at port "+port);
                new Thread(this).start();

        }
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ run ΤΗΣ ΤΑΞΗΣ GUIServer

ΑΠΑΡΙΘΜΗΣΗ ΣΤΟΙΧΕΙΩΝ ΤΟΥ HASHTABLE OUTPUTSTREAMS

Η παρακάτω μέθοδος getOutputStreams ( ), είναι αντικείμενο τύπου Enumeration, και επιστρέφει τα στοιχεία του HashTable OutputStreams απαριθμημένα με την μέθοδο elements.

```
Enumeration getOutputStreams() {
    return outputStreams.elements();
  }
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ ΑΠΑΡΙΘΜΗΣΗΣ getOutputStreams

ΜΕΘΟΔΟΣ getUsersMessage

Αρχικά δημιουργούμε ένα StringBuffer που το ονομάζουμε str και έχει ως περιεχόμενο το String <USERS>. Η τάξη StringBuffer μοιάζει με την String, με μόνη διαφορά ότι τα περιεχόμενα του StringBuffer μπορούν να μεταβληθούν. Η for που ακολουθεί απαριθμεί τα κλειδιά του HashTable users και την απαρίθμηση αυτή την αποθηκεύει στην μεταβλητή enum. Όσο η enum έχει στοιχεία, τα προσθέτουμε, αφού τα μετατρέψουμε σε String στο StringBuffer str. Τέλος με την μέθοδο toString μετατρέπουμε το str σε String και το επιστρέφουμε όταν κάποιος καλεί την μέθοδο getUsersMessage.

```
public String getUsers() {
      StringBuffer str=new StringBuffer("<USERS>");

      for (Enumeration enum=users.keys();enum.hasMoreElements();){
            str.append(" ").append((String)enum.nextElement());
      }
      return(str.toString());
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ getUsers

ΜΕΘΟΔΟΣ sendToAll

Η μέθοδος αυτή δεν επιστρέφει τίποτα, αφού την ορίσαμε ως κενή (void), αλλά δέχεται ως παράμετρο ένα String. Με την μέθοδο sendToAll, ένα μήνυμα στέλνεται σε όλους τους χρήστες που είναι συνδεμένοι στον Server. Η λογική που χρησιμοποιήσαμε είναι παρόμοια με της μεθόδου getUserMessage. Απαριθμούμε τα περιεχόμενα του πίνακα OutputStreams και με την χρήση της εντολής for, στέλνουμε το String της παραμέτρου σε όλους τους χρήστες, μέσω της μεθόδου WriteUTF.

```
void sendToAll( String message ) {
    synchronized( outputStreams ) {
      for (Enumeration e = getOutputStreams(); e.hasMoreElements(); )
{
        DataOutputStream dout = (DataOutputStream)e.nextElement();
        try {
          dout.writeUTF( message );
        } catch( IOException ie ) { System.out.println( ie ); }
      }
    }
  }
```

ΚΩΔΙΚΑΣ ΤΗΣ ΜΕΘΟΔΟΥ sendToAll

ΜΕΘΟΔΟΣ sendToUser

Τη μέθοδο αυτή την δημιουργήσαμε ώστε να μπορούμε να στείλουμε κάποιο μήνυμα σε ένα συγκεκριμένο χρήστη. Δέχεται τρεις παραμέτρους τύπου String. Η παράμετρος toUser αντιπροσωπεύει τον παραλήπτη του μηνύματος, η fromUser τον αποστολέα και η msg το μήνυμα. Με την εντολή if, ελέγχουμε αν το HashTable users περιέχει τον παραλήπτη του μηνύματος (toUser). Αν όντως υπάρχει αυτός ο χρήστης, τότε χρησιμοποιούμε τη μέθοδο sendMessage για να του αποστείλουμε το μήνυμα, διαμορφώνοντας το header όπως φαίνεται παρακάτω στον κώδικα.

```
void sendToUser(String toUser,String fromUser,String msg){
      if (users.containsKey(toUser)) {
      ((ServerThread)users.get(toUser)).sendMessage("<toAll
"+fromUser+"> "+msg);
      }
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ sendToUser

ΜΕΘΟΔΟΣ removeConnection

Η μέθοδος αυτή χρησιμοποιείται όταν αποσυνδέεται ένας χρήστης για οποιοδήποτε λόγο. Αρχικά αφαιρούμε το κλειδί Socket αυτού του αντικειμένου καθώς και το dout (DataOutputStream) από το HashTable OutputStreams, έπειτα κλείνουμε το socket και καλούμε την μέθοδο removeUser, η οποία όπως θα δούμε παρακάτω αφαιρεί τον user από το HashTable users.

```
void removeConnection( Socket s, String user ) {

  synchronized( outputStreams ) {

    p( "Removing connection to "+s );

    outputStreams.remove( s );

    try {
      s.close();
    } catch( IOException ie ) {
      p( "Error closing "+s );
      ie.printStackTrace();
    }
  }

    removeUser(user);

}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ removeConnection

ΜΕΘΟΔΟΣ removeConnection

Η λειτουργία αυτής της μεθόδου είναι ίδια με την προηγούμενη. Η μόνη διαφορά είναι ότι παίρνει μόνο μια παράμετρο (s) και αφαιρεί την σύνδεση ενός πελάτη (Client), ο οποίος είχε συνδεθεί ως φιλοξενούμενος (guest), δηλαδή δεν είχε δημιουργήσει ένα όνομα χρήστη (user name). Η μέθοδος αυτή αφαιρεί από το HashTable outputStreams το socket επικοινωνίας και έπειτα το τερματίζει.

```
void removeConnection( Socket s ) {

  synchronized( outputStreams ) {

    p( "Removing connection to "+s );

    outputStreams.remove( s );

    try {
      s.close();
    } catch( IOException ie ) {
      p( "Error closing "+s );
      ie.printStackTrace();
    }
  }
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ removeConnection

ΜΕΘΟΔΟΣ addUser

Τη μέθοδο αυτή την χρησιμοποιούμε για να προσθέσουμε τους νέους χρήστες στο HashTable users. Δέχεται δύο παραμέτρους, ένα String που προσδιορίζει τον χρήστη που προσθέτουμε και χρησιμοποιείται ως κλειδί (key) στο HashTable users και ένα αντικείμενο της τάξης ServerThread που τοποθετείται ως τιμή (value) στο HashTable users.

Αρχικά ελέγχουμε με την εντολή if, αν το όνομα του χρήστη που θέλουμε να προσθέσουμε, υπάρχει ήδη στο HashTable. Αν υπάρχει τότε η μέθοδος επιστρέφει την τιμή ένα. Διαφορετικά τοποθετεί το αντικείμενο ServerThread στο HashTable users και αντιστοιχεί ως κλειδί σε αυτό το string user. Αντίθετα από πριν, η τιμή που επιστρέφει η μέθοδος, είναι το μηδέν.

```
synchronized public int addUser(String user,ServerThread s) {
      if (users.containsKey(user))
            return (1);
      else {
            users.put(user,s);
            return (0);
      }
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ addUser

ΜΕΘΟΔΟΣ removeUser

Σε αυτή τη μέθοδο είχαμε αναφερθεί όταν εξετάζαμε τον κώδικα της μεθόδου removeConnection. Η μέθοδος αυτή αφαιρεί τον χρήστη που αποσυνδέθηκε από το HashTable users. Δέχεται ως παράμετρο ένα String που αντιπροσωπεύει το όνομα του χρήστη που αποσυνδέθηκε. Εάν το όνομα αυτό υπάρχει ως κλειδί στο HashTable users, τότε αφαιρείται και με την μέθοδο sendToAll ειδοποιούνται οι υπόλοιποι χρήστες. Επίσης στο TextArea του Server, εμφανίζεται ένα μήνυμα που ενημερώνει ποιος χρήστης αποσυνδέθηκε.

```
synchronized public int removeUser(String user) {
      if (users.containsKey(user))  {
            users.remove(user);
            sendToAll("<REMOVE_USER "+user+">");
            p("User "+user+" removed");
      }
      return (0);
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ removeUser

Η ΚΥΡΙΩΣ ΜΕΘΟΔΟΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ main

Κάθε πρόγραμμα στην JAVA περιέχει μια κυρίως μέθοδο (main), η οποία τρέχει όταν δημιουργούμε ένα νέο αντικείμενο της τάξης στην οποία η μέθοδος βρίσκεται. Στην κυρίως μέθοδο (main) που γράψαμε, αρχικά καλούμε την μέθοδο showDialog, η οποία όπως θα δούμε παρακάτω εμφανίζει ένα παράθυρο διαλόγου (dialog), στο οποίο εισάγουμε την θύρα την οποία επιθυμούμε να ακούει ο διακομιστής (server). Οι υπόλοιπες εντολές εκτελούνται αφού κλείσει το παράθυρο αυτό και αν έχει πατηθεί το πλήκτρο OK. Με την εντολή new, δημιουργούμε ένα νέο αντικείμενο της τάξης GUIServer. Έπειτα προσθέτουμε ένα WindowListener στο αντικείμενο που δημιουργήσαμε και τέλος με τη μέθοδο setVisible κάνουμε το παράθυρο (frame) ορατό, αφού όπως είδαμε η τάξη GUIServer που εξετάζουμε κληρονομεί από την τάξη JFrame.

```
public static void main(String[] args) throws IOException {
      showDialog();
      if (count==1){
      frame = new GUIServer();
      frame.addWindowListener(new AppCloser());
      frame.setVisible(true);
}
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ main

ΜΕΘΟΔΟΣ showDialog

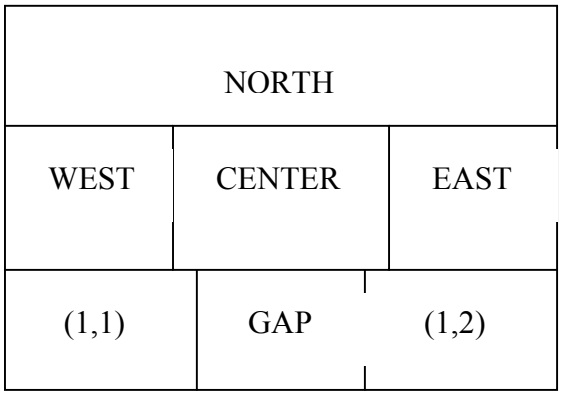
Όπως προαναφέραμε, αυτή η μέθοδος εμφανίζει ένα παράθυρο, στο οποίο δηλώνουμε σε ποια θύρα επιθυμούμε να ακούει ο Server. Αρχικά δημιουργούμε ένα νέο αντικείμενο της τάξης showDialog το οποίο το αντιστοιχούμε στο αντικείμενο frame της τάξης GUIServer που δημιουργήσαμε. Το παράθυρο αυτό το ονομάζουμε port και δηλώνουμε ότι αν δεν τερματιστεί η λειτουργία αυτού του παραθύρου, το frame είναι ανενεργό. Τα παραπάνω τα πετυχαίνουμε εισάγοντας τις κατάλληλες παραμέτρους στο νέο αντικείμενο dialog που δημιουργήσαμε όπως φαίνεται στον κώδικα.

Έτσι μέχρι τώρα έχουμε καθορίσει κάποιες γενικές ιδιότητες που έχει το παράθυρο διαλόγου. Μένει να διαμορφώσουμε το περιεχόμενο του. Όπως έχουμε δει, κατά την δημιουργία του γραφικού περιβάλλοντος, τόσο του Client όσο και του Server, αρχικά δημιουργούμε το contentPane στο οποίο τοποθετούμε όλα τα υπόλοιπα στοιχεία. Το contentPane που χρησιμοποιούμε το ονομάζουμε panel και είναι αντικείμενο της τάξης JPanel. Θέτουμε ότι η διαμόρφωσή του είναι τύπου BorderLayout και ότι έχει περιθώριο διαστάσεων (10 x 10 x 10 x 10). Έπειτα υλοποιούμε ένα νέο αντικείμενο της τάξης DigitsTextField και το προσθέτουμε στο κέντρο του panel. Στο νότιο τμήμα του panel προσθέτουμε ένα νέο αντικείμενο της τάξης JPanel, που όμως αυτή τη φορά το διαμορφώνουμε κατά GridLayout (πλέγμα), το οποίο

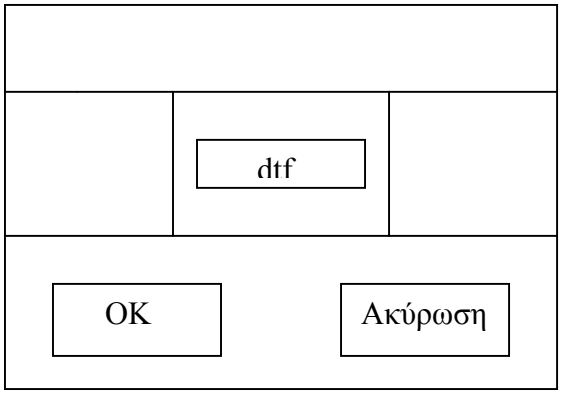
διαθέτει μία γραμμή και δύο στήλες. Το κενό ανάμεσα στις στήλες ορίζουμε ότι είναι είκοσι εικονοστοιχεία (pixels). Αυτό το νέο JPanel το ονομάζουμε buttonPanel. Στις δύο θέσεις του buttonPanel, τοποθετούμε δύο αντικείμενα της τάξης JButton. Το πρώτο κουμπί το ονομάζουμε «OK» και αναθέτουμε σε αυτό ένα ActionListener, του οποίου η μέθοδος ActionPerformed ενεργοποιείται όταν ο χρήστης πατήσει το κουμπί. Η μέθοδος αυτή χρησιμοποιεί την μέθοδο getValue του αντικειμένου dtf, ώστε να αποθηκεύσει στην μεταβλητή port την τιμή που ο χρήστης εισάγει στο αντικείμενο dtf, που όπως θα δούμε είναι ένα JTextfield. Έπειτα κλείνουμε το αντικείμενο dialog με την μέθοδο dispose, δηλαδή κλείνει το παράθυρο που άνοιξε. Το δεύτερο αντικείμενο της τάξης JButton που δημιουργήσαμε, το ονομάζουμε «Ακύρωση» και η λειτουργία του είναι να κλείνει το παράθυρο όποτε ενεργοποιηθεί.

Τέλος προσθέτουμε ένα WindowListener στο αντικείμενο dialog, του οποίου η λειτουργία είναι όμοια με αυτή που περιγράψαμε στον WindowListener της main μεθόδου. Οι τρεις τελευταίες γραμμές του κώδικα καθορίζουν την εμφάνιση που θα έχει το παράθυρο που δημιουργήσαμε. Η μέθοδος pack( ) διαμορφώνει το μέγεθος του παράθυρου, ανάλογα με τα περιεχόμενα αυτού. Ενώ η μέθοδος setVisible το κάνει ορατό.

Η διαδικασία που ακολουθήσαμε για την δημιουργία του γραφικού περιβάλλοντος γίνεται πιο κατανοητή αν δούμε τα σχήματα 4.1. 4.2. Βλέπουμε ότι κεντρικό τμήμα του borderLayout, έχουμε τοποθετήσει το dtf, ενώ στο νότιο έχουμε τοποθετήσει το JPanel, buttonPanel και σε αυτό τα δύο κουμπιά σε διάταξη gridLayout.



Σχήμα 4.1.Διάταξη χώρου του παραθύρου πριν την τοποθέτηση αντικειμένων



Σχήμα 4.2.Τελική μορφή του παραθύρου μετά την τοποθέτηση αντικειμένων

```
public static void showDialog() {

      dialog=new JDialog(frame,"Port",true);

      JPanel panel = new JPanel();
      panel.setLayout(new BorderLayout());

      panel.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
      dialog.setContentPane(panel);


      final DigitsTextField dtf=new DigitsTextField(5);
      panel.add(dtf,BorderLayout.CENTER);



      GridLayout grid = new GridLayout(1,2);
      grid.setHgap(20);
      JPanel buttonPanel=new JPanel(grid);

buttonPanel.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));

      JButton button1=new JButton("OK");
      button1.addActionListener(new ActionListener(){
                  public void actionPerformed(ActionEvent e) {
                        port=dtf.getValue();
                        System.out.println(port);
                        dialog.setVisible(false);
                        dialog.dispose();
                        count=1;
                  }
            });
      buttonPanel.add(button1);


      JButton button3=new JButton("Ακύρωση");
      button3.addActionListener(new ActionListener(){
                  public void actionPerformed(ActionEvent e) {
                        dialog.setVisible(false);
            dialog.dispose();
                        }
            });

      buttonPanel.add(button3);

      panel.add(buttonPanel,BorderLayout.SOUTH);

      dialog.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
            dialog.setVisible(false);
            dialog.dispose();
      }
    });

      dialog.pack();
      dialog.setLocationRelativeTo(frame);
      dialog.setVisible(true);
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ showDialog ΤΗΣ ΤΑΞΗΣ GUIServer

4.6. ΤΑΞΗ ServerThread

Είδαμε ότι ο Server, όταν δεχθεί μια αίτηση σύνδεσης από ένα Client, κατασκευάζει ένα Socket επικοινωνίας και δημιουργεί ένα νέο αντικείμενο της τάξης ServerThread. Στο κεφάλαιο αυτό θα επεξηγήσουμε την λειτουργία αυτής της τάξης. Γενικά μπορούμε να αναφέρουμε ότι για κάθε νέα σύνδεση Client, δημιουργείται ένα ξεχωριστό αντικείμενο της τάξης ServerThread, που αναλαμβάνει την επικοινωνία ανάμεσα στον Client και στον Server.

ΟΡΙΣΜΟΣ ΤΑΞΗΣ – ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Όπως σε όλες τις προηγούμενες τάξεις που έχουμε εξετάσει έως τώρα, αρχικά εισάγουμε τις βιβλιοθήκες της JAVA που χρησιμοποιούμε. Η τάξη ServerThread ορίζουμε ότι είναι επέκταση της τάξης Thread. Έπειτα δηλώνουμε τις μεταβλητές στιγμιότυπου που χρησιμοποιούμε περαιτέρω και οι οποίες είναι:

- Μια μεταβλητή που την ονομάζουμε server και αποθηκεύει αντικείμενα της τάξης GUIServer
- Μια μεταβλητή τύπου Socket και την ονομάζουμε socket
- Μια μεταβλητή τύπου DataOutputStream που την ονομάζουμε dout
- Μια μεταβλητή τύπου boolean που την ονομάζουμε nickHasBeenSet
- Μια μεταβλητή τύπου String που την ονομάζουμε user

Ο constructor της τάξης δέχεται ως παραμέτρους ένα αντικείμενο της τάξης GUIServer και ένα αντικείμενο της τάξης Socket. Τα αντικείμενα αυτά τα αποθηκεύουμε στις μεταβλητές server και socket που δηλώσαμε προηγουμένως. Τέλος στη μεταβλητή dout αποθηκεύουμε ένα αντικείμενο της τάξης DataOutputStream για το socket επικοινωνίας που πήραμε και με τη μέθοδο start ( ), καλούμε την run ( ) μέθοδο στην οποία αναφερόμαστε παρακάτω.

```
import java.io.*;
import java.net.*;

public class ServerThread extends Thread
{
      private GUIServer server;
      private Socket socket;
      private DataOutputStream dout;
      private boolean nickHasBeenSet=false;
      private String user;

  public ServerThread( GUIServer server, Socket socket ) {
    this.server = server;
    this.socket = socket;

      try {
            dout = new DataOutputStream( socket.getOutputStream() );
    } catch( IOException ie ) {
      System.out.println( ie );
    }
    start();
  }
```

ΜΕΘΟΔΟΣ sendMessage

Τη μέθοδο αυτή την χρησιμοποιούμε για να στείλουμε μηνύματα από το αντικείμενο του ServerThread στον Client, του οποίου διαχειρίζεται την επικοινωνία. Δηλώνουμε κατά τον ορισμό της μεθόδου, ότι είναι συγχρονισμένη (synchronized). Αυτό το κάνουμε για να αποφύγουμε τυχών εμπλοκές (conflict) που μπορούν να δημιουργηθούν από άλλα αντικείμενα που χρησιμοποιούν αυτή τη μέθοδο. Η παράμετρος της μεθόδου με την μέθοδο writeUTF αποστέλλεται στον Client.

```
public synchronized void sendMessage(String message) {
   try {
      dout.writeUTF( message );
   } catch( IOException ie ) {
    server.p( ie.toString() );
   }
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ sendMessage

ΜΕΘΟΔΟΣ run ΤΗΣ ΤΑΞΗΣ ServerThread

Η μέθοδος run ( ) είναι η κύρια μέθοδος της τάξης ServerThread. Σε αυτή τη μέθοδο γίνεται η επεξεργασία των μηνυμάτων που δέχεται ο Server από τον Client και το αντίστροφο. Αρχικά αποθηκεύουμε στην μεταβλητή din ένα αντικείμενο DataInputStream για το Socket επικοινωνίας. Έπειτα με την μέθοδο getUsersMessage ( ) παίρνουμε ένα String που περιέχει του χρήστες που είναι ήδη συνδεμένοι. Την «λίστα» αυτή με την μέθοδο sendMessage που περιγράψαμε προηγουμένως, την στέλνουμε στον Client. Με ένα ατέρμονα βρόχο (while (true)), διαβάζουμε συνεχώς, πιθανά μηνύματα που στέλνει ο Client και τα αποθηκεύουμε στην μεταβλητή str.

Το επόμενο τμήμα του κώδικα επεξεργάζεται αυτά τα μηνύματα, με την χρήση της τάξης message. Σε περίπτωση που η μέθοδος name ( ) επιστρέψει το string ¨nick¨, τότε προσθέτουμε με τη χρήση της μεθόδου addUser ( ) τον χρήστη στο HashTable users και με τις μεθόδους sendMessage ( ) και sendToAll ( ), ειδοποιούμε τους υπόλοιπους χρήστες ότι ένας νέος χρήστης συνδέθηκε. Αν όμως κάποιος άλλος χρήστης έχει ήδη επιλέξει αυτό το όνομα τότε ειδοποιείται ο Client ότι δεν γίνεται να επιλέξει αυτό το όνομα.

Αν το string που επιστρέφει η name είναι το ¨msg¨, που σημαίνει ότι το μήνυμα που ακολουθεί απευθύνεται σε ένα συγκεκριμένο χρήστη, τότε χρησιμοποιούμε την μέθοδο sendToUser ( ) για να στείλουμε το μήνυμα στο χρήστη αυτό. Οι παράμετροι που εισάγουμε στην sendToUser είναι το περιεχόμενο της δεύτερης και τρίτης θέσης του array value και το String, δηλαδή το μήνυμα του αποστολέα, που επιστρέφει η μέθοδος body ( ). Στην δεύτερη θέση του πίνακα values είναι αποθηκευμένο το όνομα του παραλήπτη, ενώ στην τρίτη το όνομα του αποστολέα. Εάν δεν ισχύουν τα παραπάνω, τότε χρησιμοποιούμε την μέθοδο sendToAll ( ) και το μήνυμα του αποστολέα το λαμβάνουν όλοι οι χρήστες. Τέλος στην περίπτωση που ο

ατέρμονα βρόχος διακοπεί επειδή χάθηκε η επικοινωνία με τον Client, τότε ανάλογα με το αν ο χρήστης είχε nick ή όχι, χρησιμοποιούμε μια από τις δύο removeConnections για να διαγράψουμε όλα τα στοιχεία επικοινωνίας του χρήστη.

```
  public void run() {
    try {
            DataInputStream din = new DataInputStream(
socket.getInputStream() );

            sendMessage(server.getUsersMessage());
            server.p(server.getUsersMessage());
            while (true) {

              String str = din.readUTF();
                  server.p("MSG: "+str);
                  Message message=new Message(str);
              if (message.name().trim().toLowerCase().equals("nick")) {
                  int reply=server.addUser(message.body().trim(),this);
                  if (reply==0) {
                        //The nickname has been accepted
                        user=message.body().trim();
                        nickHasBeenSet=true;
                        server.p("New User: "+message.body().trim());
                        sendMessage("<OK "+message.body().trim()+">");
                        server.sendToAll( "<NEW_USER
"+message.body().trim()+"> " );

                  }
                  else if (reply==1) {
                        //The nickname exists already
                        server.p("User "+message.body().trim()+" already
exists.");
                              sendMessage("<USER_EXISTS
"+message.body().trim()+">");
                        }
            }
              else if (message.name().trim().equals("MSG")) {
                  //Message to a particular user

      server.sendToUser(message.value(1),message.value(2),message.body()
);
      }
              else {
                  server.sendToAll( "<toAll "+message.value(1)+">
"+message.body() );
                  }
            }
            } catch( EOFException ie ) {
                        server.p("EOFException");
            } catch( IOException ie ) {
            } finally {

                  if (nickHasBeenSet) server.removeConnection(
socket,user );
                  else server.removeConnection( socket );
            }
      }
}
```

ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ run ΤΗΣ ΤΑΞΗΣ ServerThread

ΤΑΞΗ DigitsTextField

Όταν εξετάζαμε την τάξη GUIServer, δημιουργήσαμε ένα αντικείμενο dialog απ' όπου οποίο ο χρήστης μπορούσε να εισάγει την θύρα στην οποία «ακούει» ο Server. Για να γίνει αυτό έπρεπε να τοποθετήσουμε ένα JTextField στο οποίο θα εισαγόταν ο αριθμός. Αν όμως τοποθετούσαμε ένα απλό αντικείμενο TextField θα ήταν δυνατόν να εισαχθούν όλα τα αλφαριθμητικά σύμβολα. Για αυτό το λόγο, δημιουργήσαμε την τάξη DigitsTextField, η οποία κληρονομεί τις μεθόδους της JTextField, αλλά γίνεται ο χρήστης να γράψει σε αυτή, μόνο αριθμούς από το πέντε έως το εξήντα πέντε χιλιάδες. Όλα τα αντικείμενα κειμένου υλοποιούν το interface Document. Με τις μεθόδους αυτού του interface, μεταξύ των άλλων, ελέγχουν και τα στοιχεία που μπορούν να εισαχθούν στο αντικείμενο. Η τάξη JTextField με την μέθοδο getDefaultModel «φορτώνει» ένα αντικείμενο της τάξης PlainDocument το οποίο ελέγχει το κείμενο το οποίο εισάγεται στο JTextField. Αυτή λοιπόν την τάξη έπρεπε να την αλλάξουμε ώστε να αφήνουμε να εισάγονται στο JTextField μόνο αριθμοί. Αν το κάναμε όμως αυτό θα εισάγαμε αυτούς τους περιορισμούς και στις υπόλοιπες περιοχές κειμένου που χρησιμοποιήσαμε. Έτσι αναγκαστήκαμε να επέμβουμε στον κώδικα της createDefaultModel (override) και αντί να θέτει ως Document για το JTextField το PlainDocument να θέτει ένα αντικείμενο της τάξης DigitsDocument η οποία είναι μια τάξη που δημιουργήσαμε εμείς. Η DigitsDocument είναι υποτάξη (subclass) της PlainDocument, αλλά έχουμε υπερσκελίσει (override) την μέθοδο της insertString ( ), έτσι ώστε να επιτρέπεται να εισαχθούν μόνο αριθμοί [3],[4]. Κάναμε αυτή την εισαγωγή για να γίνει αντιληπτός ο σκοπός ύπαρξης της τάξης.

Αρχικά δηλώνουμε ότι η DigitsTextField είναι υποτάξη της JTextField. Ο Constructor της τάξης δέχεται ως παράμετρο ένα ακέραιο (integer) και με την εντολή super, καλεί τον αντίστοιχο Constructor της JTextField ο οποίος δημιουργεί ένα JTextField με όσες στήλες δείχνει ο ακέραιος. Όπως είπαμε υπεσκελίσαμε (override) την createDefaultModel και έτσι πλέον δημιουργεί ένα αντικείμενο της τάξης DigitsDocument, το οποίο δέχεται ως παράμετρο, τις στήλες του JTextField. Η μόνη μέθοδος του DigitsDocument που αλλάζουμε σε σχέση με το PlainDocument, είναι η insertString έτσι ώστε να επιτρέπει να εισαχθούν μόνο αριθμοί. Αυτό το πετυχαίνουμε με την χρήση της εντολής if, αν ένα από τα παρακάτω συμβαίνει τότε δεν επιτρέπει να τυπωθεί τίποτα στο DigitsTextField.

- Αν το μήκος του κειμένου που θέλει να τυπώσει ο χρήστης είναι μεγαλύτερο από τον αριθμό των στηλών μείον τις στήλες που ήδη έχουν τυπωθεί
- Αν το πρώτο στοιχείο είναι γράμμα
- Αν αυτό που θα τυπώσουμε είναι μεγαλύτερο από τον αριθμό των στηλών
- Αν τα στοιχεία που είναι ήδη τυπωμένα είναι μεγαλύτερα από τον αριθμό των στηλών, μείον μια θέση

Η μέθοδος getValue ( ) χρησιμοποιείται για να επιστρέψει την τιμή της θύρας που εισάγει ο χρήστης.

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;

public class DigitsTextField extends JTextField {

      public DigitsTextField(int cols) {
        super(cols);

    }

    protected Document createDefaultModel() {
          return new DigitsDocument(getColumns());
    }

    class DigitsDocument extends PlainDocument {

        private int cols;

        public DigitsDocument(int cols) {
            this.cols=cols;
        }

        public void insertString(int offs, String str, AttributeSet a)
throws BadLocationException {
              if ((str==null)||(str.length()>cols-
super.getLength())||(!Character.isDigit(str.charAt(0)))||(offs>cols-
1)||(super.getLength()>cols-1)) return;
              super.insertString(offs, str, a);
          }
    }

      public int getValue() {
            if (isEmpty()) return(0);
            else return(Integer.valueOf(getText()).intValue());
      }

      public boolean isEmpty() {
            return(getText().trim().equals(""));
      }

      protected void finalize(){
            System.out.println("Final dtf");
      }

}
```

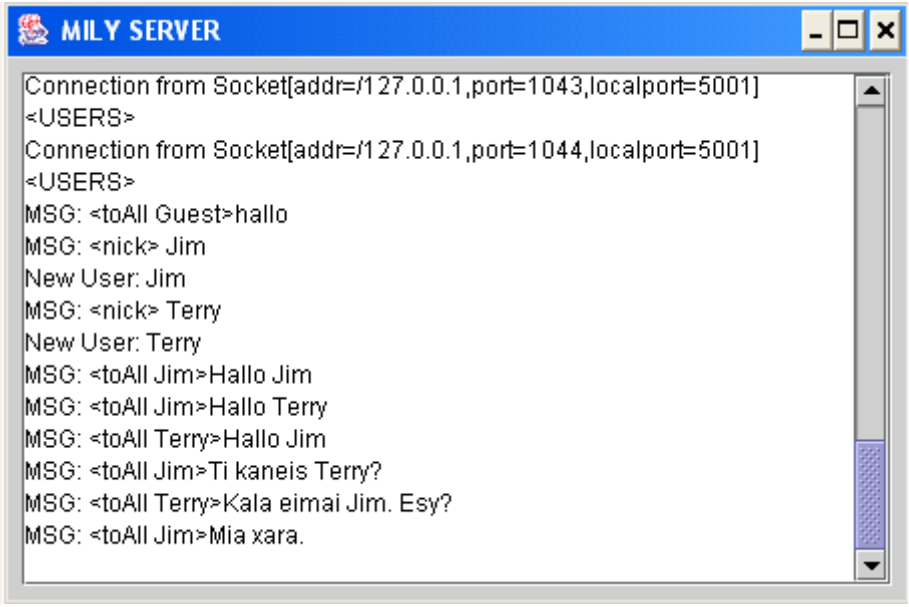
ΚΩΔΙΚΑΣ ΤΑΞΗΣ DigitsTextField

5. ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΚΑΙ ΧΡΗΣΗ

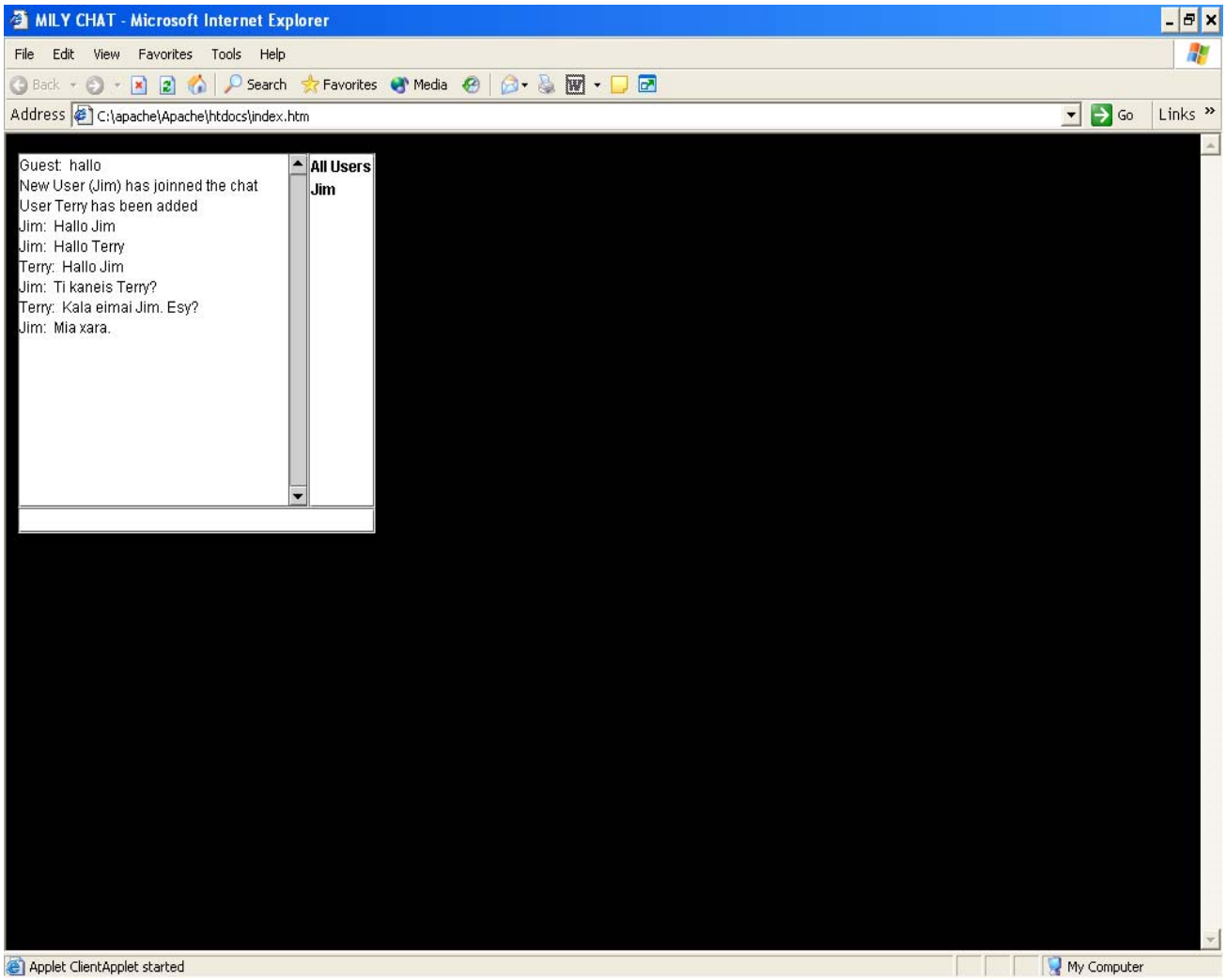
Σε αυτό το κεφάλαιο αναφερόμαστε στον τρόπο εγκατάστασης και χρήσης του πακέτου MILY Chat. Στον οπτικό δίσκο (cd) που συνοδεύει την πτυχιακή μας υπάρχει ένα εκτελέσιμο αρχείο (exe) στο φάκελο MILY, που ονομάζεται install.exe. Το αρχείο αυτό εγκαθιστά τα αρχεία του πακέτου MILY Chat στον φάκελο C:\Program Files\Apache Group\Apache\htdocs. Στην λειτουργία του Apache web server, καθώς και στις ρυθμίσεις που χρειάζεται να γίνουν, έχουμε αναφερθεί στο τρίτο κεφάλαιο. Έπειτα ανοίγουμε το Apache web server και στην συνέχεια από το J-Creator, «τρέχουμε» την τάξη ΄΄GUIServer΄΄ και δίνουμε μια τιμή στη θύρα επικοινωνίας. Πρέπει η τιμή αυτή να είναι ίδια με την τιμή της θύρας που έχουμε ορίσει στον Client. Όταν γίνουν όλα τα παραπάνω το MILY Chat είναι έτοιμο προς χρήση. Οι χρήστες μπορούν να συνδεθούν απλώς ανοίγοντας την ιστοσελίδα με το Applet.

Το πρόγραμμα είναι πάρα πολύ απλό στην χρήση του. Ότι γράφουμε στο TextField που εμφανίζεται στο κάτω μέρος της οθόνης, με το πάτημα του πλήκτρου enter στέλνεται σε όλους τους χρήστες και τυπώνεται στην TextArea, τόσο του αποστολέα όσο και του παραλήπτη. Για να στείλουμε ένα μήνυμα σε κάποιο συγκεκριμένο χρήστη, γράφουμε το σύμβολο «παπάκι» (@) μπροστά από το όνομα του παραλήπτη και πατάμε το enter. Επίσης ένας άλλος τρόπος είναι να επιλέξουμε τον παραλήπτη από την λίστα με τους χρήστες, να γράψουμε το μήνυμα που θέλουμε να του στείλουμε και να πατήσουμε το enter. Τέλος αν θέλουμε να συνδεθούμε ως χρήστες με κάποιο όνομα (nick), αντί guest, μπροστά από το όνομα που επιθυμούμε να έχουμε, γράφουμε το σύμβολο «κάγκελο» (#) και πατάμε enter.

Στις εικόνες 6.1. και 6.2. που ακολουθούν βλέπουμε το περιβάλλον του διακομιστή (Server) και πελάτη (Client). Στον διακομιστή φαίνονται οι πιθανές συνδέσεις χρηστών που έχουν γίνει και πληροφορίες γύρω από αυτές. Στο περιβάλλον του πελάτη, στο κεντρικό παράθυρο υπάρχουν τα μηνύματα, αριστερά υπάρχει η λίστα με τους χρήστες, ενώ στο κάτω τμήμα του παραθύρου υπάρχει το TextField στο οποίο ο χρήστης εισάγει τα μηνύματα του.



Εικόνα 6.1. Περιβάλλον διακομιστή (Server)



Εικόνα 6.2. Περιβάλλον πελάτη (Client)

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Όταν ξεκινήσαμε την ενασχόλησή μας με τον προγραμματισμό στην JAVA, σκοπός μας ήταν να αποκτήσουμε κάποιες βασικές γνώσεις γύρω από αυτήν, χωρίς όμως να γνωρίζουμε αν θα καταφέρναμε να ολοκληρώσουμε τον στόχο μας, την δημιουργία δηλαδή της παρούσας πτυχιακής. Σχεδόν ένα χρόνο μετά οι προσπάθειές μας ανταμείφθηκαν, μπορούμε να πούμε ότι γνωρίζουμε αρκετά καλά την βασική δομή της γλώσσας και έχουμε εξοικειωθεί με την χρήση της.

Όταν αρχίσαμε δεν γνωρίζαμε τίποτα γύρω από την JAVA, εκτός ότι είναι μια ευρέως διαδεδομένη γλώσσα προγραμματισμού που εξελίσσεται συνεχώς και χρησιμοποιείται πολύ στις εφαρμογές Διαδικτύου. Το πρώτο πρόβλημα που αντιμετωπίσαμε, ήταν η έλλειψη βιβλιογραφίας στα ελληνικά. Τα βιβλία που κυκλοφορούσαν ήταν πολύ λίγα και κανένα δεν μας ικανοποιούσε. Έτσι αναγκαστικά στραφήκαμε στην ξενόγλωσση βιβλιογραφία, όπου υπάρχουν πλούσιες πηγές πληροφοριών. Το βιβλίο που επιλέξαμε για να μάθουμε τις βασικές γνώσεις γύρω από την γλώσσα είναι το [1]. Για αρκετό καιρό διαβάζαμε αυτό το βιβλίο και μάθαμε πως δομείται η γλώσσα και πως συντάσσεται. Οι γνώσεις όμως που μας πρόσφερε δεν ήταν αρκετές, αλλά μας έδωσαν την ώθηση να κοιτάξουμε παραπέρα και να βρούμε πληροφορίες στο Διαδίκτυο (Internet).

Από την επίσημη ιστοσελίδα της SUN, κατεβάσαμε το tutorial και τα documentation αρχεία. Σε αυτά ανατρέχαμε όποτε δεν καταλαβαίναμε κάτι ή θέλαμε να δούμε την χρήση και τις μεθόδους μιας τάξης, δηλαδή συνεχώς, ειδικά στις αρχές. Στο Διαδίκτυο επίσης υπάρχουν πάρα πολλές διευθύνσεις που βοηθάνε τον προγραμματιστή από τον πιο αρχάριο έως και τον πιο προχωρημένο. Ψάχνοντας σε αυτές βρήκαμε πολλά προγράμματα που υλοποιούσαν chat server, άλλα πολύπλοκα και άλλα πιο απλά, και που μας έδωσαν τις ιδέες για την δημιουργία του δικού μας πακέτου.

Το πρόγραμμα που σας παρουσιάσαμε, σχετικά με τα επαγγελματικά προγράμματα που κυκλοφορούν στην αγορά είναι πολύ απλό, αλλά χρειάσθηκε πολύ κόπος και χρόνος για την υλοποίησή του. Στο μέλλον γίνεται να επεκταθεί έτσι ώστε να περιλαμβάνει περισσότερες λειτουργίες όπως να ανοίγει χωριστά παράθυρα επικοινωνίας για κάθε χρήστη, ή να δίνει την δυνατότητα στον server να κρατάει στατιστικά δεδομένα σε μια βάση δεδομένων (data base).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Laura Lemay, Rogers Cadenhead, “Teach Yourself JAVA in 21 days”, SAMS, Indianapolis, 1999

[2] http://java.sun.com, επίσημη ιστοσελίδα της SUN στο Διαδίκτυο

[3] http://java.sun.com/docs/index.html, τα documentetions files της JAVA

[4] http://java.sun.com/docs/books/tutorial/index.html, το tutorial