Blood Pressure Monitoring System

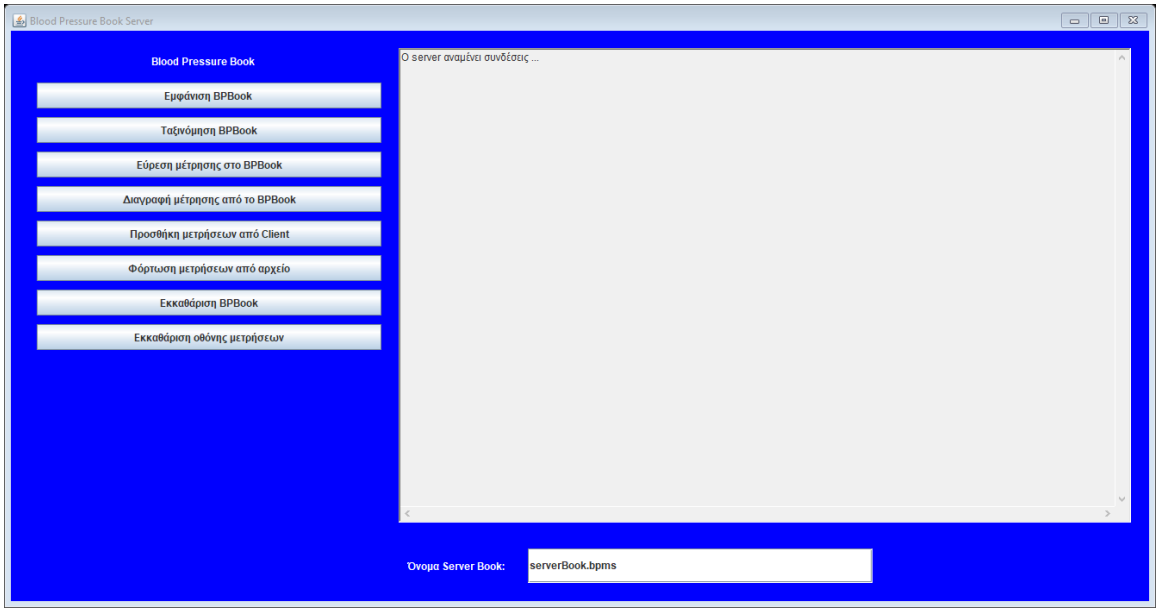
2ο παραδοτέο

Δασούλας Ιωάννης – 1053711

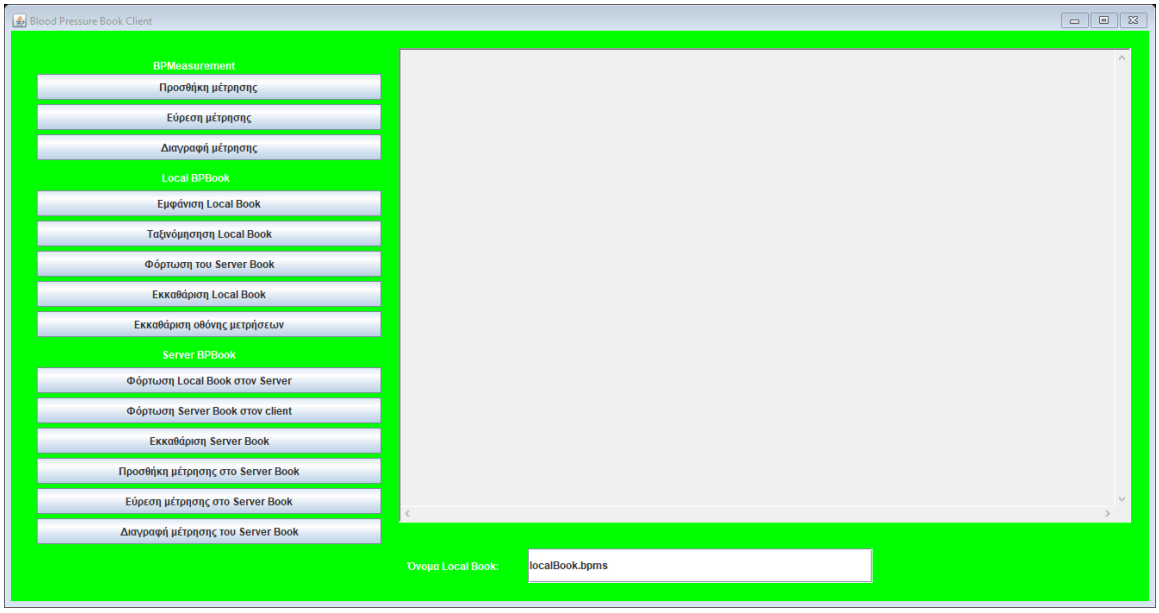
1. Περιγραφή λειτουργικότητας

Η εφαρμογή αποτελείται από 2 βασικές γραφικές διεπαφές, μία για τον server και μία για τον client. Ο server αποθηκεύει τις δικές του μετρήσεις σε έναν φάκελο (Server Book). Δίνεται η δυνατότητα να εμφανίζει και να ταξινομεί τις μετρήσεις, να βρίσκει και να διαγράφει κάποια μέτρηση, να προσθέτει μετρήσεις από τον client, να φορτώνει μετρήσεις από αρχείο και να καθαρίζει το αρχείο. Ο client μπορεί να διαχειρίζεται ένα τοπικό αρχείο μετρήσεων (Local Book) και το αρχείο του server, όποτε επιθυμεί. Στο τοπικό αρχείο, έχει τη δυνατότητα να προσθέτει, να εμφανίζει, να ταξινομεί, να βρίσκει και να διαγράφει μετρήσεις. Φυσικά, έχει και την δυνατότητα να φορτώσει το Local Book στον server ως Server Book ή να καθαρίζει το Local Book. Όσον αφορά το Server Book, μπορεί να στέλνει και να δέχεται μετρήσεις από αυτό, να ψάχνει ή να διαγράφει μετρήσεις και να το αποθηκεύει ως Server Book. Έχει δημιουργηθεί και η λειτουργία του Simulator, αλλά δεν έχει ολοκληρωθεί, για αυτό και δεν στάλθηκε σε αυτή την έκδοση.

Server:

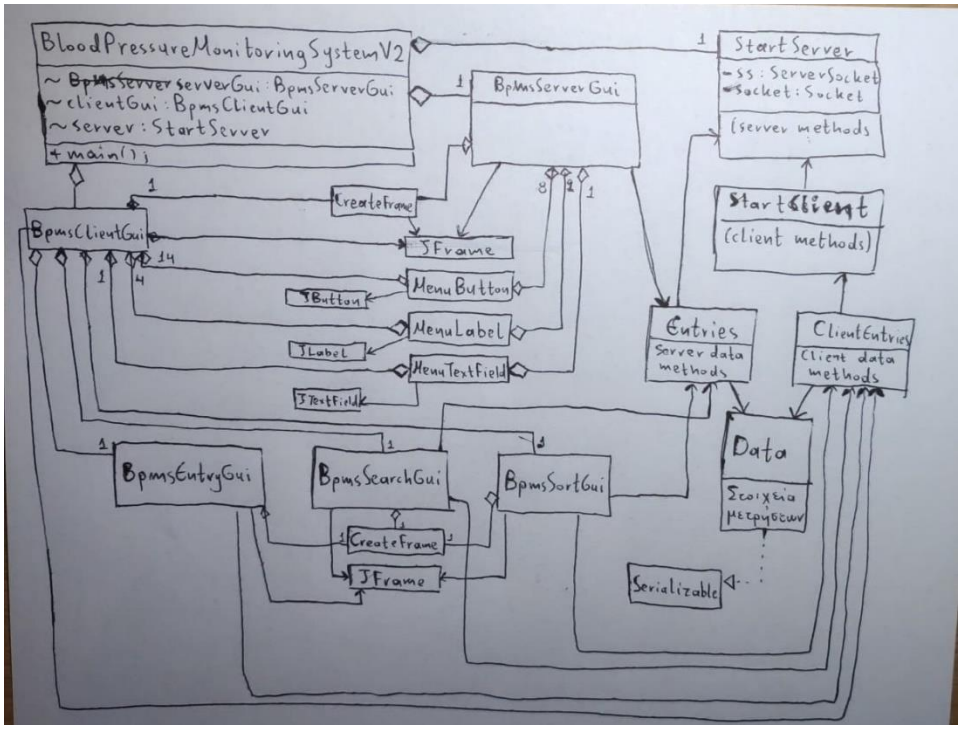


Client:



2. Class Diagram

Για συντομία και ευκρίνεια, δεν γράφτηκαν οι μέθοδοι και οι μεταβλητές των κλάσεων



3. Πρωτόκολλο επικοινωνίας Client – Server

Όταν γίνεται η εκκίνηση της εφαρμογής, δημιουργούνται τα οι 2 γραφικές διεπαφές και ξεκινά τη λειτουργία του ο server δημιουργώντας στιγμιότυπο της ServerSocket και μπαίνει σε λειτουργία αναμονής μέχρι να δημιουργηθεί κάποια σύνδεση με την συνάρτηση accept().

```
//Έναρξη λειτουργίας server
public static void runServer() throws IOException, ClassNotFoundException {
    //Δημιουργία socket
    ss = new ServerSocket(7777);

    BpmsServerGui.log.append("Ο server αναμένει συνδέσεις ...\n\n");

    socket = ss.accept(); //Αναμονή για σύνδεση...
```

Όταν μία σύνδεση βρεθεί, ο server δημιουργεί ένα στιγμιότυπο της InputStream για να λάβει το μήνυμα που του στέλνει ο client.

```
//Input stream για λήψη μηνύματος από client
InputStream messageStream = socket.getInputStream();
ObjectInputStream messageInputStream = new ObjectInputStream(messageStream);
String message = (String) messageInputStream.readObject();
```

Αποθηκεύεται το μήνυμα σε μεταβλητή τύπου String, κι έπειτα ανάλογα με το μήνυμα, επιτελεί ο server την ανάλογη λειτουργία. Τα μηνύματα που μπορεί να ληφθούν από τον client είναι 6:

- `GET_BOOK -> Ο client ζητάει να του σταλεί το Server Book`
- `SEND_BOOK -> Ο client ζητάει να στείλει το Server Book`
- `CLEAR_BOOK -> Ο client ζητάει να διαγραφεί το Server Book`
- `SEND_MEASUREMENT -> Ο client ζητάει να προσθέσει μέτρηση στο Server Book`
- `FIND_MEASUREMENT -> Ο client ζητάει να βρει μέτρηση στο Server Book`
- `DELETE_MEASUREMENT -> Ο client ζητάει να διαγραφεί μέτρηση από το Server Book`

```
//Λειτουργία ανά μήνυμα
if(message.equals("GET_BOOK")) {
    BpmsServerGui.log.append("Ελήφθη request: GET_BOOK από τον socket" + socket +"!\n\n");
    serverSendList();
}

else if(message.equals("SEND_BOOK")) {
    BpmsServerGui.log.append("Ελήφθη request: SEND_BOOK από τον socket" + socket +"!\n\n");
    serverGetList();
}

else if(message.equals("CLEAR_BOOK")) {
    BpmsServerGui.log.append("Ελήφθη request: CLEAR_BOOK από τον socket" + socket +"!\n\n");
    Entries.clearBook();
}

else if(message.equals("SEND_MEASUREMENT")) {
    BpmsServerGui.log.append("Ελήφθη request: SEND_MEASUREMENT από τον socket" + socket +"!\n\n");
    serverGetMeasurement();
}

else if(message.equals("FIND_MEASUREMENT")) {
    BpmsServerGui.log.append("Ελήφθη request: FIND_MEASUREMENT από τον socket" + socket +"!\n\n");
    serverGetName();
}
```

Η λειτουργία του client ξεκινάει με την επιλογή των ανάλογων κουμπιών. Ανάλογα με το μήνυμα ο client και ο server επιτελούν τις απαραίτητες λειτουργίες ανταλλάσσοντας δεδομένα με input και output streams. Όταν η λειτουργία που κλήθηκαν να κάνουν τελειώσει, κλείνουν τα sockets τους, τερματίζεται η μεταξύ τους σύνδεση και ο server επανεκκινείται αναμένοντας συνδέσεις.

```java
//Αποστολή τοπικών μετρήσεων στον server
public static void clientSendList()  throws IOException, ClassNotFoundException {
    //Δημιουργία σύνδεσης με server
    Socket socket = new Socket("localhost", 7777);
    BpmsClientGui.log.append("Ο client συνδέθηκε!\n\n");

    //Δημιουργία output stream για αποστολή μηνύματος
    OutputStream messageStream = socket.getOutputStream();

    ObjectOutputStream messageOutputStream = new ObjectOutputStream(messageStream);

    messageOutputStream.writeObject("SEND_BOOK");
    BpmsClientGui.log.append("Στάλθηκε request: SEND_BOOK στον server!\n");

    //Δημιουργία output stream για αποστολή δεδομένων

    OutputStream outputStream = socket.getOutputStream();

    ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);

    BpmsClientGui.log.append("Τα δεδομένα στάλθηκαν στον server\n\n");

    objectOutputStream.writeObject(ClientEntries.clientList);

    BpmsClientGui.log.append("Η σύνδεση τερματίστηκε.\n\n");

    //Κλείσιμο σύνδεσης
    socket.close();
}
```

4. Σχόλια/Παρατηρήσεις

Η άσκηση είναι ένα πολύ ενδιαφέρον παράδειγμα συνδυασμού λειτουργικότητας (προσθήκη, ταξινόμηση, εύρεση, διαγραφή, χειρισμό αρχείων), γραφικών διεπαφών και επικοινωνίας client – server. Παρόλα αυτά, ενώ τα δύο πρώτα κομμάτια είχαν υλοποιηθεί σε καλό επίπεδο από την monolithic, ακόμα, έκδοση, συνάντησα πρόβλημα στην κατανόηση του τρόπου με τον οποίον πρέπει να υλοποιηθεί η επικοινωνία server – client. Δυσκολεύτηκα, δηλαδή, να κατανοήσω το πρότυπο πρωτοκόλλου που ήταν προτεινόμενο στην άσκηση και δημιούργησα έναν δικό μου τρόπο επικοινωνίας, λιγότερο modular και re-usable, ίσως επειδή δεν είχα ασχοληθεί ξανά με τέτοια δομή επικοινωνίας. Θα μπορούσε ίσως, να δοθεί κάποια παραπάνω κατεύθυνση στο μάθημα ή στο εργαστήριο για την σωστότερη κατανόηση του τρόπου επικοινωνίας και την δομή του πρωτοκόλλου. Μετά την προθεσμία της παράδοσης, θα προσπαθήσω να υλοποιήσω το πρόβλημα με την δομή enum για τα μηνύματα που ήταν προτεινόμενη και την κλάση Request και θα ολοκληρωθεί και η Simulator λειτουργία.