Department of Management and Technology

Software Comprehension and Maintenance

Wireless IRC project

Implementation

Presentation’s Overview

- Demonstration
- Implementation steps
- Testing the implementation
- Communicating with the developper
- Integrating the changes into the program
- Trying to implement the timestamp
- Trying to remove the back command from the namelist

Step1:Adding the commands

First of all, I have created a new MyFavourites command to the main menu:

```
Canvas.addCommand (myfavouritesCommand);
```

Then I have added the commands save,delete, searchCommand to both the textbox that appears when selecting the message command and the myFavourites list:

```
textbox.addCommand(searchCommand);
textbox.addCommand(saveCommand);
 textbox.addCommand(deleteCommand);
exclusiveList.addCommand(searchCommand);
exclusiveList.addCommand(saveCommand);
exclusiveList.addCommand(deleteCommand);
```

<u>Step2:Creating a method for every command</u>:

```
else if (c == saveCommand) {

        saveData();

}

else if (c == deleteCommand) {

        deleteData();

}

if (c == searchCommand){
        searchData();

}
```

<u>Step3:Creating a store for the messages inside the public class Database</u>:

```
protected Vector messages= new Vector();

byte[] bm = getByteArray(messages);

recstore.addRecord(bm,0,bm.length);
```

Every time the device is closed, the method Database.save() is called and as a result the following method is executed:

```
recstore.setRecord(3,bm,0,bm.length);
```

Step4: Defining the content of the methods:

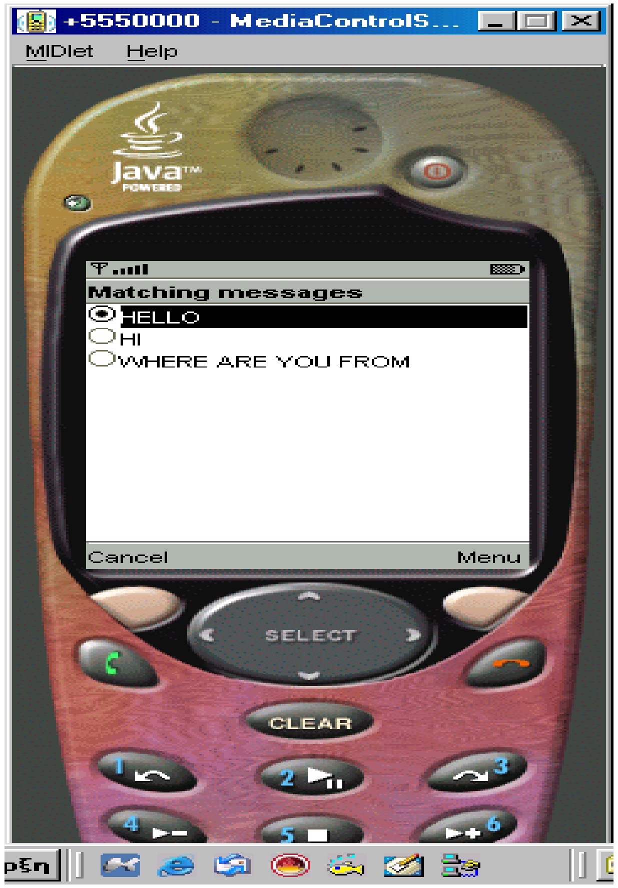
```
protected void saveData()  {
                    String str = textbox.getString();
                    db.messages.addElement(str);
                    db.save();
                    savedAlert.setString("Message saved");
                    display.setCurrent(savedAlert);
                    textbox = null;
                    display.setCurrent(textbox);
}
```

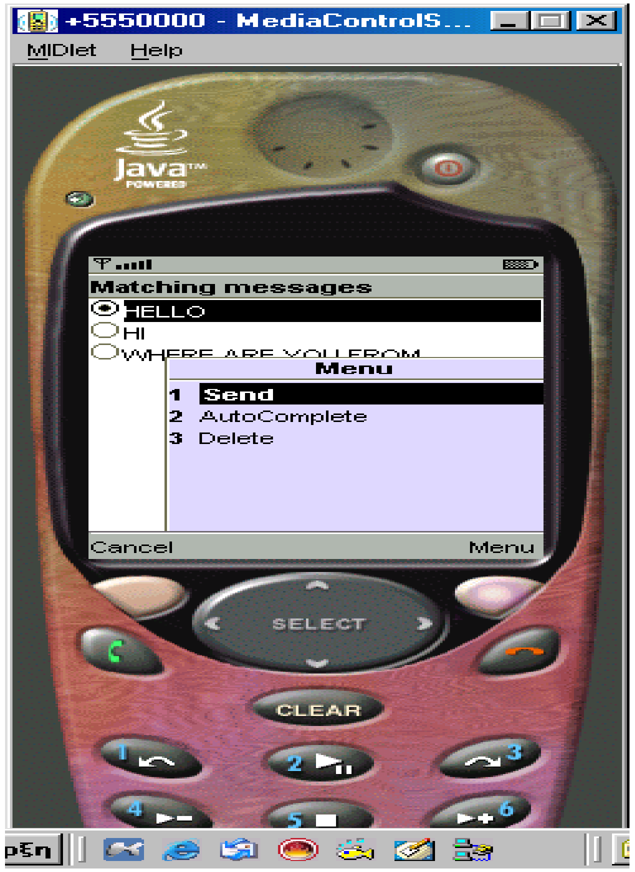
```
protected void deleteData()  {
        String str;
        if(listflag)
        str=exclusiveList.getString(exclusiveList.getSelectedIndex
());
        else    str = textbox.getString();
        for (int j=0;j<db.messages.size();j++){

        if(db.messages.elementAt(j).toString().startsWith(str))
          if(!listflag)
        textbox.setString(db.messages.elementAt(j).toString());
                                db.messages.removeElementAt(j);
                    db.save();
}       }
```

```
protected void searchData(){
        String[] stringArray = new String [10];
        String str = textbox.getString();
        exclusiveList = new List("Matching messages",
Choice.EXCLUSIVE);
        for (int j=0;j<db.messages.size();j++){
        if(db.messages.elementAt(j).toString().startsWith(str))

        exclusiveList.append(db.messages.elementAt(j).toString(
),null);}
      listflag=true;
      display.setCurrent(exclusiveList);
        }
```

```
 if(c == MyFavouritesCommand){


            exclusiveList = new List("Matching messages",
Choice.EXCLUSIVE);

            for (int j=0;j<db.messages.size();j++){


          exclusiveList.append(db.messages.elementAt(j).toString(),
null);

            }

            listflag=true;

            display.setCurrent(exclusiveList);

}
```

Testing the implementation:After implementing the changes I cleared the database file of the device.Then the Favourites Command worked





Communicating with the developper

My mail was the following**:**

Thank you very much for answering my mail.
Please find attached the WRLIrc file with the changes I have made.All my changes are the following:
ScreenOutput.java: lines 222-327
Database.java: lines 138,171-185,281
Utils.java: line 77

I have created a public Vector called messages.Its bytes are saved in the same byte[] in which the bytes of the array rs are saved.This way I did not have to create a new record (I have tried to do the least possible changes) but I had to create one more method (SplitVectorToString).

I would like to ask you to make to me any recommendations and suggestions about the implementation as this is the main subject of my assignement.
Yours sincerely,
Tzavala Polina

The developper’s answer was as follows:

I have now added my project to CVS, and added your user "polina" to the project. You are available to read and write from the cvs repository. I have also merged your code to my code. Was a little bit difficult because I had also made some changes to the code....

The autocomplete function looks OK, but I think it would be better if you make a list of all entries. And when you write some text, the list derease to these entries that match your text.

I made also some small changes to the code, especially in the database.java file. look at the repository for a update.

Other things to do:

- make a historylist, like mirc's arrow up, arrow down... (this is only saved in buffer)

✓ timestamps

- ignore list, could be comma separated

✓ make the codebase smaller. remove the textboxlistener class

- implent flood control

 nick list,(move back, next button to a command instead of an element)

Integrating the changes into the program :Looking for a CVS program

CVS is a program that stores all the versions of a file in a single file in a clever way that only stores the differences between versions.

TortoiseCVS is a front-end client to make using CVS easier and more intuitive. It allows developers to work with files controlled by CVS directly from Windows Explorer™.

A particular benefit is that TortoiseCVS has an SSH client built into it, without special setup. Anyone can easily commit code to SourceForge.net projects using TortoiseCVS.

Using Tortoise CVS

Indeed Tortoise is a user friendly program.All the actions needed to do a commit were the following:

Firsly, I made an account on SourceForge.net.The username I have chosen is my developper name. Then I filled in the following configuration form of The Tortoise

- Protocol: Secure Shell (:ext)
- Server: cvs.sourceforge.net
- Directory: /cvsroot/wirelessirc
- Username:polina

And finally I chose the commit command from the windows explorer.

Trying to implement the timestamp:

The timestamp is the time indication that appears before each message we receive

```
if(c==timestampCommand){
        canvas.removeCommand(timestampCommand);

        db.timestamp=!db.timestamp;

        db.save();

        if(db.timestamp==true)

        TimeStampString="TimeStampOn";

        else TimeStampString="TimeStampOff";

        timestampCommand=new Command
(TimeStampString,Command.SCREEN, 80);

        canvas.addCommand(timestampCommand);
```

```
public void writeInfo(String str) {
                String time="";
                if (db.timestamp) {
                        Calendar cal = Calendar.getInstance();
                        time = "[" +
cal.get(Calendar.HOUR_OF_DAY) + ":" +

(cal.get(Calendar.MINUTE)<10?"0":"") +
cal.get(Calendar.MINUTE) + "]";
                        }
                write(new Message("**** " + time+str,
Message.INFO));         }
```

Trying to remove the back command from the namelist and add it as a menu command

```
/**namelist.append(BACK, null);**/
    namelist.addCommand(backCommand);
if(c == backCommand){
                    if(display.getCurrent().equals(namelist))
                    listnames(c, s);
                    else if(display.getCurrent()==nameslist)
                    show();
                     else
                      display.setCurrent(namelist); }
```

But

- The commands of the Namelist don’t work correctly
- Flood control and
- Scrolling up and down of the status window

Have not yet been implemented

To be continued…