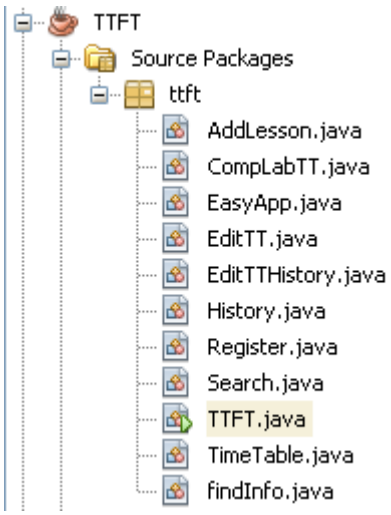
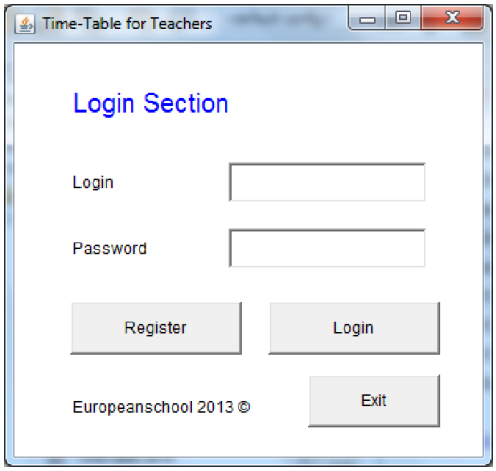
Criterion C: Development

This program is written on java. The program is orientated to imitate an interactive timetable that is easily accessible by the client and the input can be easily displayed when needed. All the inputted data will be saved in the timetable.txt file and the changes - History.txt file.

Program Functions



Main window

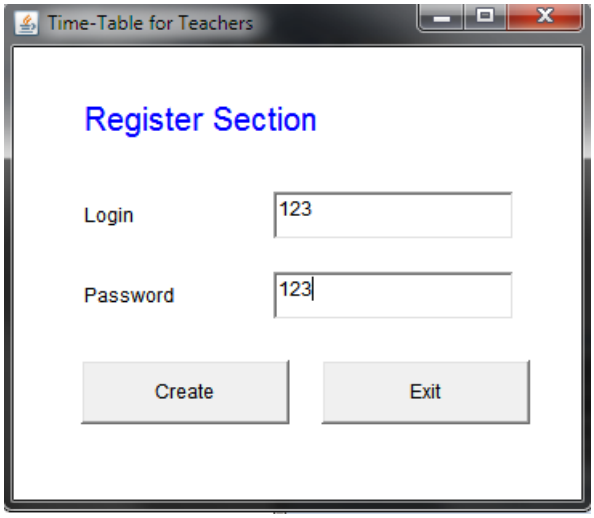


For the interface of the program, I have used Labels, TextFields and Buttons as shown below:

```
Label Title = addLabel("Login Section", 50, 50, 400, 50, this);
Label Login_ = addLabel("Login", 50, 110, 60, 50, this);
TextField T1 = addTextField("", 170, 120, 150, 30, this);
Label Password_ = addLabel("Password", 50, 160, 60, 50, this);
TextField T2 = addTextField("", 170, 170, 150, 30, this);
Button Register = addButton("Register", 50, 225, 130, 40, this);
Button Login = addButton("Login", 200, 225, 130, 40, this);
Button Exit = addButton("Exit", 230, 280, 100, 40, this);
Label CopyRight = addLabel("Europeanschool 2013 ©", 50, 280, 200, 50, this);
```

Register Function

To use the program, an account is needed, so the teacher must register his account. It also has a protection system, that won't allow a second person to register.

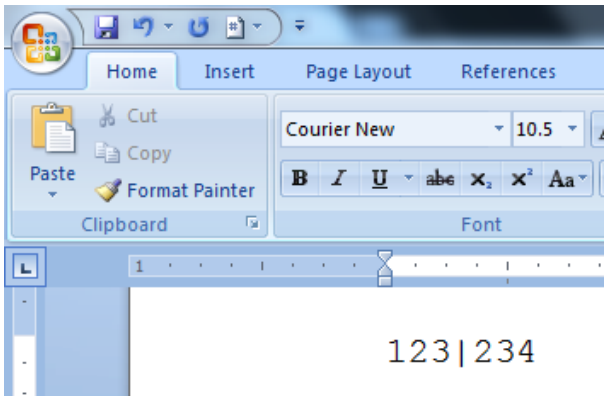


After pressing the create button all the info that was written in the TextFields will be saved in RandomAccessFile with the name main.txt, the file then is created after the code will be executed. The information from the TextFields will be written in one line in the newly created txt file; however the password will be modified: each of the character's ASCII code will be raised by one, that is used for the protection of the password from the unauthorized users.

```
void Create() {
    String login = T1.getText();
    String password = T2.getText();
    String passwordNew = "";
    try {
        RandomAccessFile main = new RandomAccessFile("main.txt", "rw");
        for (int i = 0; i < password.length(); i++) {
            int k = (int) password.charAt(i) + 1;
            passwordNew = passwordNew + (char) k;
        }
        main.writeBytes(login + "|" + passwordNew + "\n");

    } catch (IOException e) {
        e.getMessage();
    }
    dispose();
```

Thus the data in the file will look like this:



The first is account data and the second is the password.

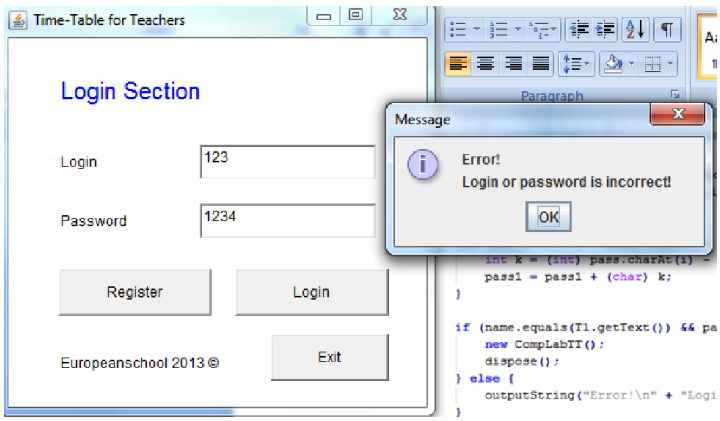
Login Function

```
void Login() {
    try {
        RandomAccessFile main = new RandomAccessFile("main.txt", "r"); //user name and password will be saved
        if (main.length() != 0) {
            String line = main.readLine();
            name = line.substring(0, line.indexOf("|"));
            String pass = line.substring(line.indexOf("|") + 1);

            String pass1 = "";
            for (int i = 0; i < pass.length(); i++) {
                int k = (int) pass.charAt(i) - 1;
                pass1 = pass1 + (char) k;
            }

            if (name.equals(T1.getText()) && pass1.equals(T2.getText())) {
                new CompLabTT();
                dispose();
            } else {
                outputString("Error!\n" + "Login or password is incorrect!");
            }
        } else {
            outputString("Please, register at first!");
        }
    } catch (IOException e) {
        e.getMessage();
    }
}
```

When the login button is pressed, the input data is checked in the main.txt file, if the data in the TextFields isn't matching data stored in the main.txt then an error will be displayed.

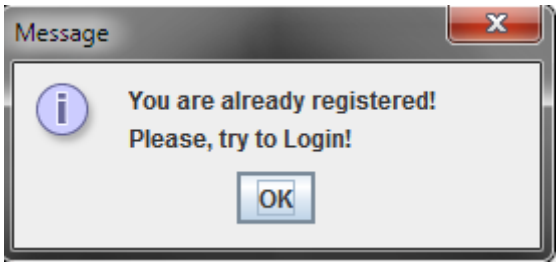


If the data is matching then the timetable window will be opened.

Register Check Function

```
    void RegisterCheck() {
        try {
            RandomAccessFile main = new RandomAccessFile("main.txt", "rw");
            if (main.length() == 0) {
                new Register();
            } else {
                outputString("You are already registered!\n" + "Please, try to Login!");
            }
        } catch (IOException e) {
            e.getMessage();
        }
    }
}
```

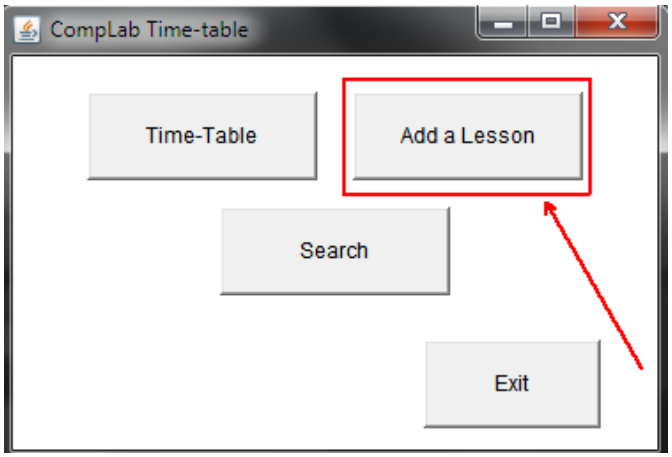
If a user has already registered and he has pressed the register button again, the program will check the main.txt files length, if the length doesn't equal to zero, the program will display a warning message.



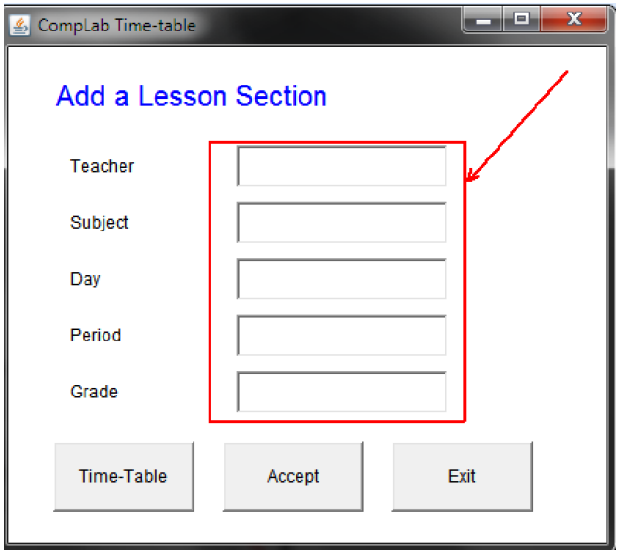
Input Data to the Timetable

There are many ways that able the user to input the data. One of them is a direct input through the data tables.

Add a Lesson Function



The button will take you to the Add a Lesson window:



In the window you will have to fill all the tables, if any of it is left out then the data won't be stored.

```
void Accept() {
    String teacher = Teacher.getText();
    String subject = Subject.getText();
    String day = Day.getText();
    String period = Period.getText();
    String grade = Grade.getText();
    if (teacher.equals("") || subject.equals("") || day.equals("") || period.equals("")
            || grade.equals("")) {
        outputString("Error!\n" + "fill in all fields!");

    } else {
```

If all of the data is written correctly, then the program will store all the input data in one line in the timetable.txt file.

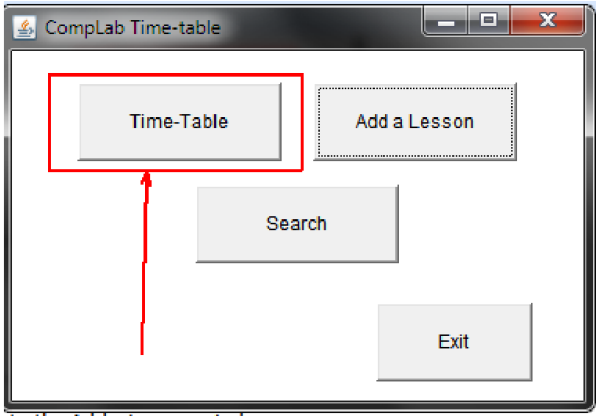
```
String record = teacher + "/" + subject + "/" + day + "/" + period + "/" + grade;
try {
    RandomAccessFile timetable = new RandomAccessFile("timetable.txt", "rw");
                                    //file to save all information as an records
    int length = (int) timetable.length();
    if (length == 0) {
        timetable.writeBytes(TTFT.name + "'s room TimeTable\n");
        timetable.writeBytes("\n");
    } else {
        timetable.seek(length);
    }
    System.out.println(record);
    timetable.writeBytes(record + "\n");
} catch (IOException e) {
    e.getMessage();
}
```

after accepting all data all textfields will be cleand:

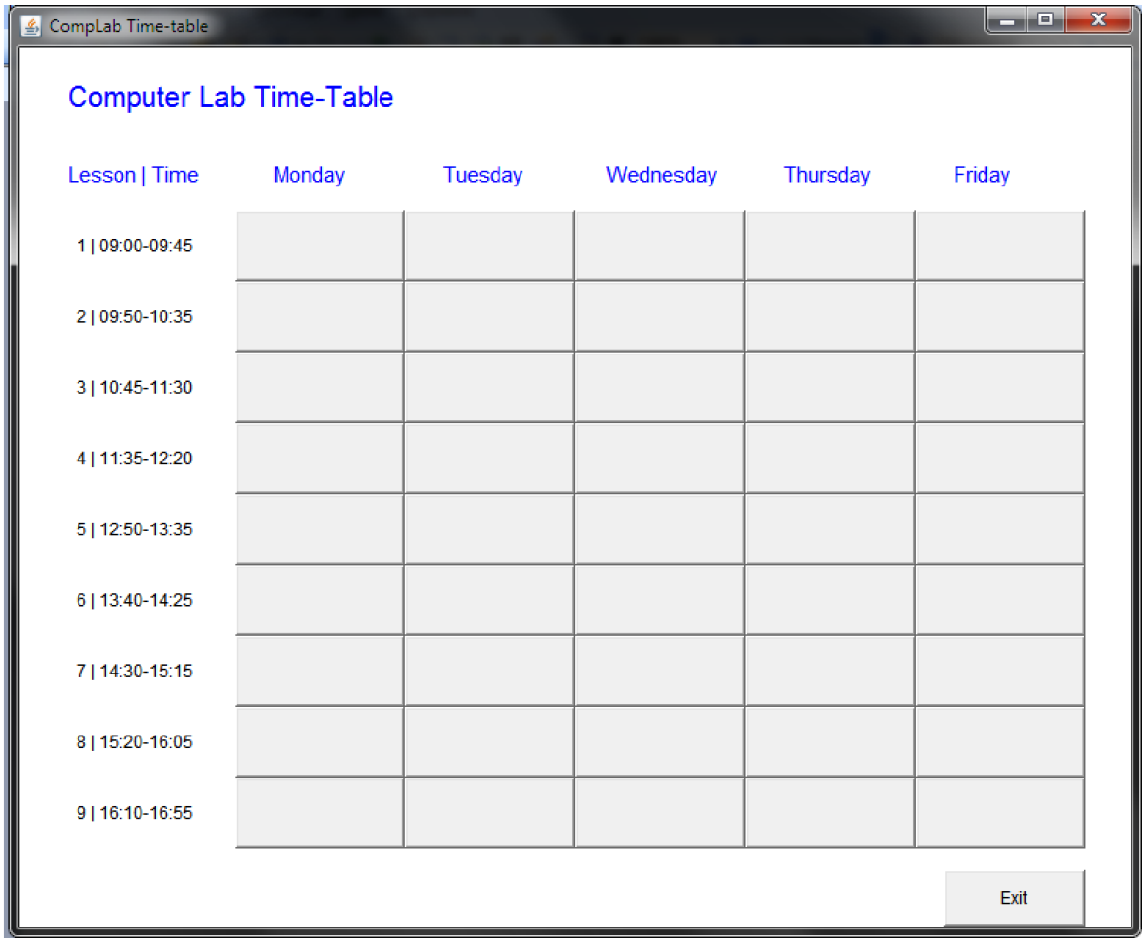
```
Teacher.setText("");
Subject.setText("");
Day.setText("");
Period.setText("");
Grade.setText("");
```

Time-Table Function

There is also another way, it will still involve the filling of the data, but the displacement can be chosen with a click, rather than by writing a day and a period.



After clicking the Timetable button, an appropriate window will be displayed.



```
static int i;  //Horizontal coordinate
static int j; //Vertical coordinate
static String[][] LessonInfo = new String[45][5];
                // lessons max number is 9 * 5 = 45
static String[][] check = new String[9][5];
```

The Timetable consists of 45 buttons. The information is stored in the **LessonInfo** array. Each row represents information from 5 fields: teacher's name, subject, grade, period and day.

Array **check** gives information if the lesson is occupied or not.

```
Lessons[a][b] = addButton(buttonTitle, 160 + b * 120, 145 + a * 50, 120, 50, this);
check[a][b] = "ready";
```

To create Buttons, first we need to create array:

```
static void createArray() {
    String line;
    //read information from file:
    //Teacher's name, subject, day, period, grade.
    try {
        count = 0;
        int index1;
        int index2;
        RandomAccessFile timetable = new RandomAccessFile("timetable.txt", "rw");
        timetable.readLine();  //  List title
        timetable.readLine();   //empty line
        // read lines from file and puts in the 2D array
        while (timetable.length() != timetable.getFilePointer()) {
            index1 = 0;
            line = timetable.readLine();
            index2 = line.indexOf("/");
            LessonInfo[count][0] = line.substring(index1, index2);
            index1 = index2;
            index2 = line.indexOf("/", index1 + 1);
            LessonInfo[count][1] = line.substring(index1 + 1, index2);
            index1 = index2;
            index2 = line.indexOf("/", index1 + 1);
            LessonInfo[count][2] = line.substring(index1 + 1, index2);
            index1 = index2;
            index2 = line.indexOf("/", index1 + 1);
            LessonInfo[count][3] = line.substring(index1 + 1, index2);
            index1 = index2;
            LessonInfo[count][4] = line.substring(index1 + 1);
            count++;
```

To draw time-table, the program will read data from timetable.txt file line by line. And all of the data will be written in the LessonInfo array as shown.

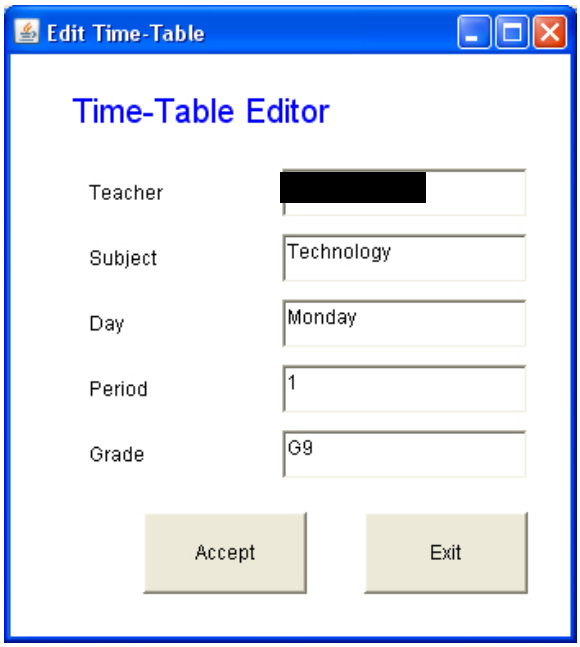
What to write on the button if the period is occupied:

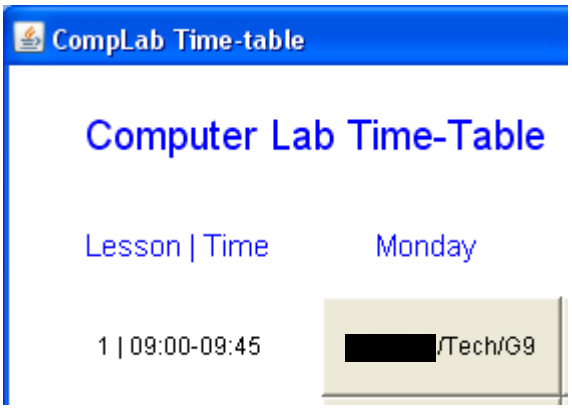
```
if (LessonInfo[z][1].length() > 4) {
    subject = LessonInfo[z][1].substring(0, 4);
} else {
    subject = LessonInfo[z][1];
}

teacherName = LessonInfo[z][0].substring(0, LessonInfo[z][0].indexOf(" ") + 2) + ".";
buttonTitle = teacherName + "/" + subject + "/" + LessonInfo[z][4];
```

subject title is max 4 letters;

if teachers name is: XX YY – XX Y. will be printed on the button;





To draw already occupied buttons we have code:

```
if (LessonInfo[z][2].equals("Monday")) {
    day = 0;
} else if (LessonInfo[z][2].equals("Tuesday")) {
    day = 1;
} else if (LessonInfo[z][2].equals("Wednesday")) {
    day = 2;
} else if (LessonInfo[z][2].equals("Thursday")) {
    day = 3;
} else {
    day = 4;
}
a = Integer.parseInt(LessonInfo[z][3]) - 1;
b = day;
Lessons[a][b] = addButton(buttonTitle, 160 + b * 120, 145 + a * 50, 120, 50, this);
check[a][b] = "ready";
```

```
//draw all the rest empty buttons
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 5; j++) {
        if (check[i][j] == null) {
            Lessons[i][j] = addButton(lessonName, 160 + j * 120, 145 + i * 50, 120, 50, this);
        }
    }
}
```

As all of the buttons are interactive, by clicking on them a timetable editor window will be displayed. So for example let's click Monday's 1st period.

When client clicks on button:

```
public void actions(Object source, String command) {
    if (source == Exit) {
        System.exit(0);
    }
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 5; j++) {
            if (source == Lessons[i][j]) {
                new EditTT();
                dispose();
            }
        }
    }
}
```

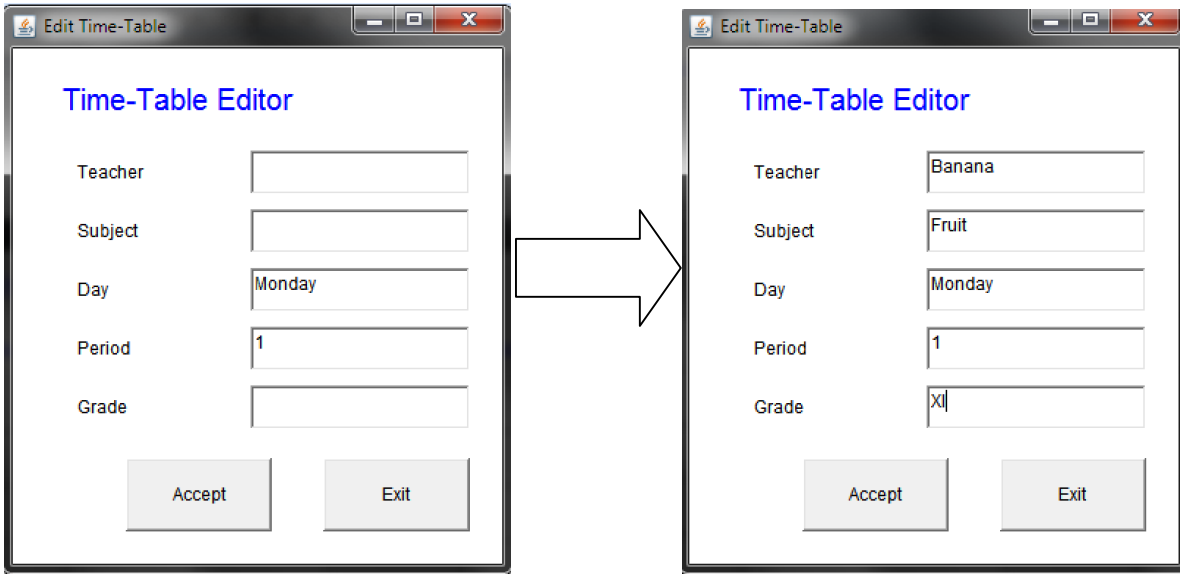
EditTT() class begins to work: first we need to take info about button:

```
void buttonInfo() {

    z = 0;
    String day;
    if (TimeTable.j == 0) {
        day = "Monday";
    } else if (TimeTable.j == 1) {
        day = "Tuesday";
    } else if (TimeTable.j == 2) {
        day = "Wednesday";
    } else if (TimeTable.j == 3) {
        day = "Thursday";
    } else {
        day = "Friday";
    }

    String period = "" + (TimeTable.i + 1);
    if (button != null) {
        // button is not empty
        while (z < TimeTable.count) {
            if (TimeTable.LessonInfo[z][2].equals(day) &&
                    TimeTable.LessonInfo[z][3].equals(period)) {
                Teacher.setText(TimeTable.LessonInfo[z][0]);
                Subject.setText(TimeTable.LessonInfo[z][1]);
                Day.setText(TimeTable.LessonInfo[z][2]);
                Period.setText(TimeTable.LessonInfo[z][3]);
                Grade.setText(TimeTable.LessonInfo[z][4]);
```

If the button isn't occupied, it means that there is no lesson on that period, however upon clicking on the button day and period will be assigned by the program's array function, yet all of the other fields should be filled by user.



All changes will be saved in History.txt file:

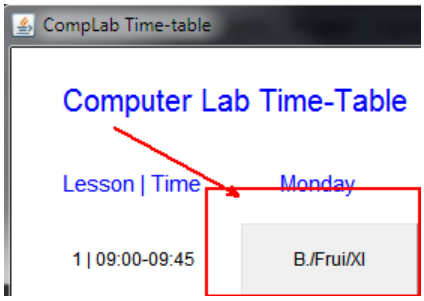
```
void HistoryOfActions() {

    try {
        RandomAccessFile OldInfo = new RandomAccessFile("History.txt", "rw");
        OldInfo.seek(OldInfo.length());
        OldInfo.writeBytes(TimeTable.LessonInfo[z][0] + "/" +
                TimeTable.LessonInfo[z][1] + "/" + TimeTable.LessonInfo[z][2] +
                "/" + TimeTable.LessonInfo[z][3] + "/" +
                TimeTable.LessonInfo[z][4] + "\n");

    } catch (IOException e) {
        e.getMessage();
    }
    }
     //if Button is empty - Accept() method works as input new lesson
    //if Button is already accepted - client should change information and then save
    // thus, change = delete + accept
```

Edit Function

If the button is occupied, then all of the data from that array will be displayed in an appropriate window field.



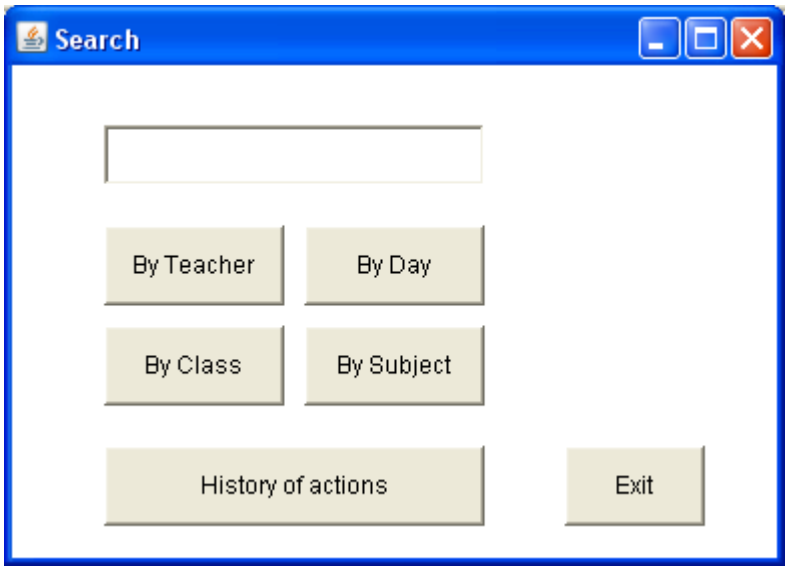
The data from the timetable can also be edited, if it's clicked by the user in the time-table.

If the filled data is clicked then the window with the inputted data will be given, it can be edited and the changes will be displayed in the timetable and the changed info will be stored in the history of actions.

```
void Edit() {
//delete the record and accept changies
try {
        RandomAccessFile newTT = new RandomAccessFile("timetable.txt", "rw");
                                    //delete line from file
        newTT.setLength(0);
        newTT.writeBytes(TTFT.name + "'s room TimeTable\n");
        newTT.writeBytes("\n");
        int i = 0;
        while (i < TimeTable.count) {
            if (i != z) {
                String record = TimeTable.LessonInfo[i][0] + "/" +
                        TimeTable.LessonInfo[i][1] + "/" +
                        TimeTable.LessonInfo[i][2] + "/" +
                        TimeTable.LessonInfo[i][3] + "/" +
                        TimeTable.LessonInfo[i][4];
                newTT.writeBytes(record + "\n");
            }
            i++;
        }

        TimeTable.check[TimeTable.i][TimeTable.j] = null;
```

Search button



Actions are:

```
text = SearchText.getText();

if (source == ByDay) {
    index = 2;
    new findInfo();
}
if (source == ByClass) {
    index = 4;
    new findInfo();
}
if (source == ByTeacher) {
    index = 0;
    new findInfo();
}
if (source == BySubject) {
    index = 1;
    new findInfo();
}
if (source == History) {
    new History();
}
if (source == Exit) {
    System.exit(0);
}
```

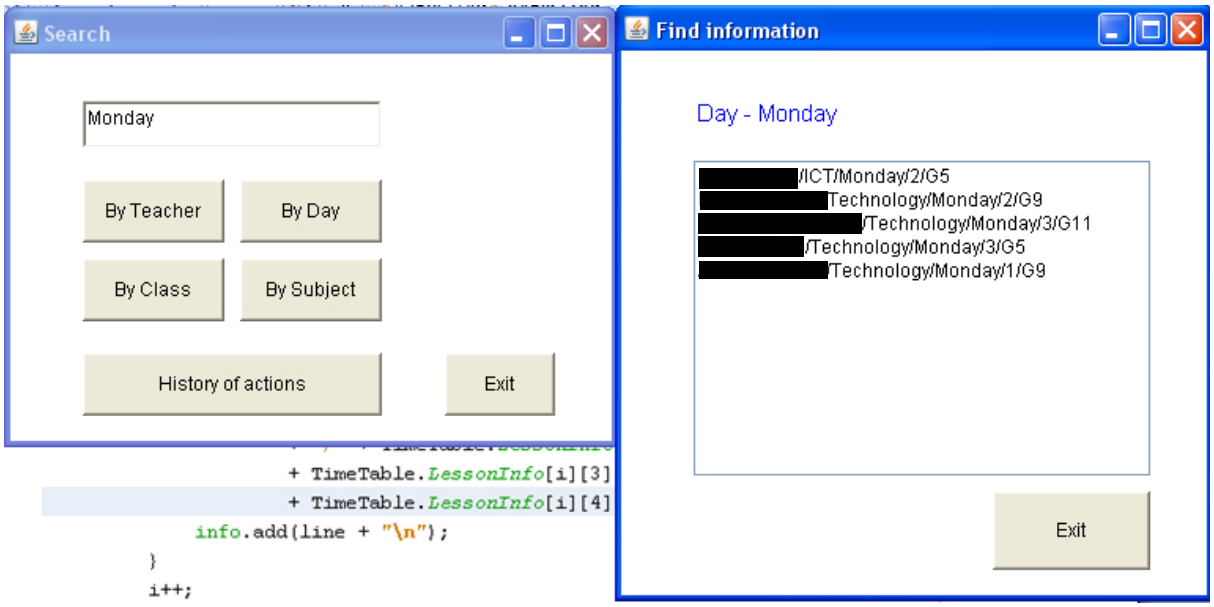
In **findInfo()** class firs constructor realizes what to search:

```
if (Search.index == 0) {
    Title.setText("Teacher - " + Search.text);
} else if (Search.index == 1) {
    Title.setText("Subject - " + Search.text);
} else if (Search.index == 2) {
    Title.setText("Day - " + Search.text);
} else {
    Title.setText("Grade - " + Search.text);
}

information();
```

And then in LessonInfo array (Timetable class) searches to find all needed lessons:

```
void information() {
    TimeTable.createArray();
    info.removeAll();
    int i = 0;
    int index = Search.index;
    String line;
    while (TimeTable.LessonInfo[i][0] != null) {
        if (TimeTable.LessonInfo[i][index].equals(Search.text)) {
            line = TimeTable.LessonInfo[i][0] + "/" + TimeTable.LessonInfo[i][1]
                    + "/" + TimeTable.LessonInfo[i][2] + "/"
                    + TimeTable.LessonInfo[i][3] + "/"
                    + TimeTable.LessonInfo[i][4];
            info.add(line + "\n");
        }
        i++;
    }
```



History class

In this class we have two methods: ShowHistory() and Edit()

```
void ShowHistory() {
    String text; // to read records from History.txt file
    HistoryA.removeAll();
    try {
        RandomAccessFile OldInfo = new RandomAccessFile("History.txt", "rw");
        while (OldInfo.getFilePointer() != OldInfo.length()) {
            text = OldInfo.readLine();
            HistoryA.add(text + "\n");
        }
```

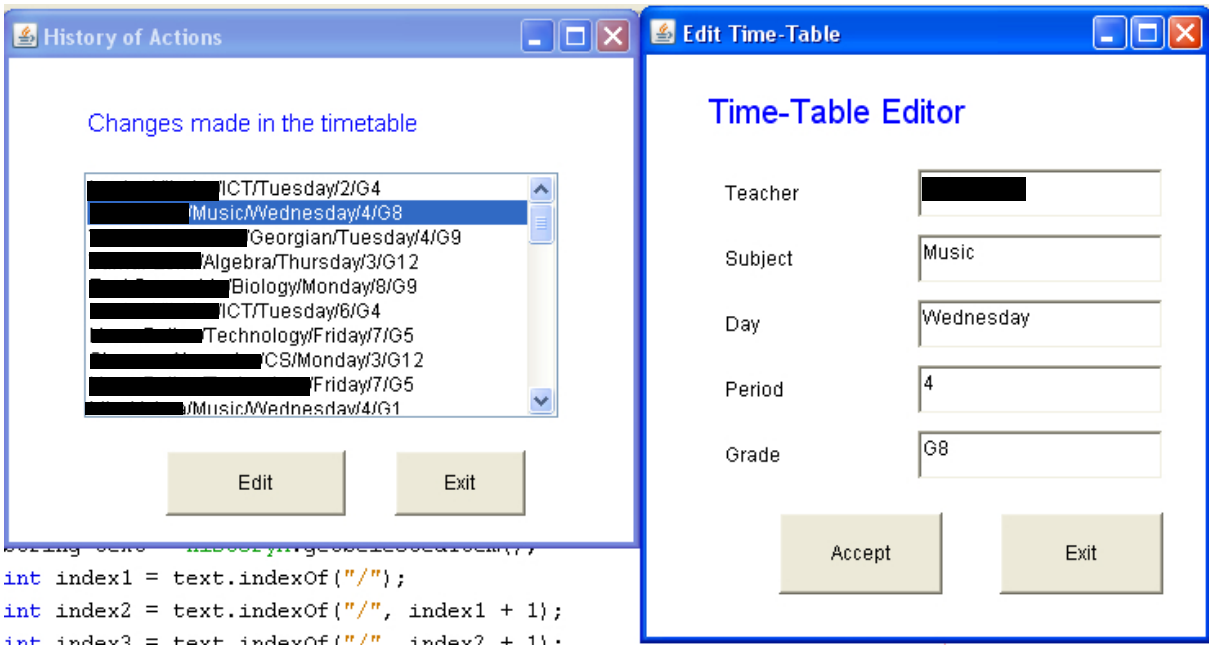
And Edit/recover information

```
void Edit() {
    String text = HistoryA.getSelectedItem();
    int index1 = text.indexOf("/");
    int index2 = text.indexOf("/", index1 + 1);
    int index3 = text.indexOf("/", index2 + 1);
    int index4 = text.indexOf("/", index3 + 1);

    name = text.substring(0, index1);
    subject = text.substring(index1 + 1, index2);
    day = text.substring(index2 + 1, index3);
    lesson = text.substring(index3 + 1, index4);
    grade = text.substring(index4 + 1);

    new EditTTHistory();
```

Reads selected line (each line represents one lesson that was changed)




```
int index1 = text.indexOf("/");
int index2 = text.indexOf("/", index1 + 1);
```

It's possible to Accept this lesson or firs change and then accept.

This changes will be represented in the timetable.txt file as well with EditTTHistory() class and Accept () method that works in the same way as AddLesson class Accept () method.

Words 820