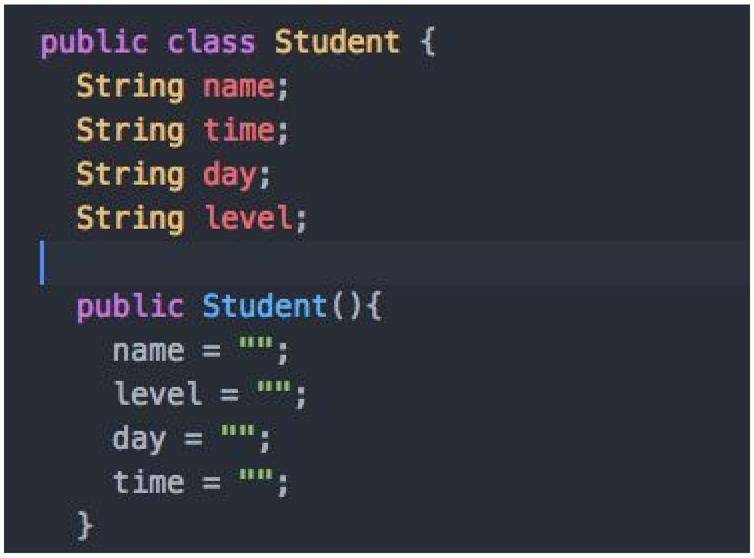
Student.java - where the object student is created

student.txt - where the information of all the students are stored

Manager.java - where all the methods are created, for example, the method for writing into a txt file

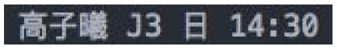
Page.java- windows created to have a better user experience



```java
public class Student {
  String name;
  String time;
  String day;
  String level;

  public Student(){
    name = "";
    level = "";
    day = "";
    time = "";
  }
```

Figure 1: Student.java

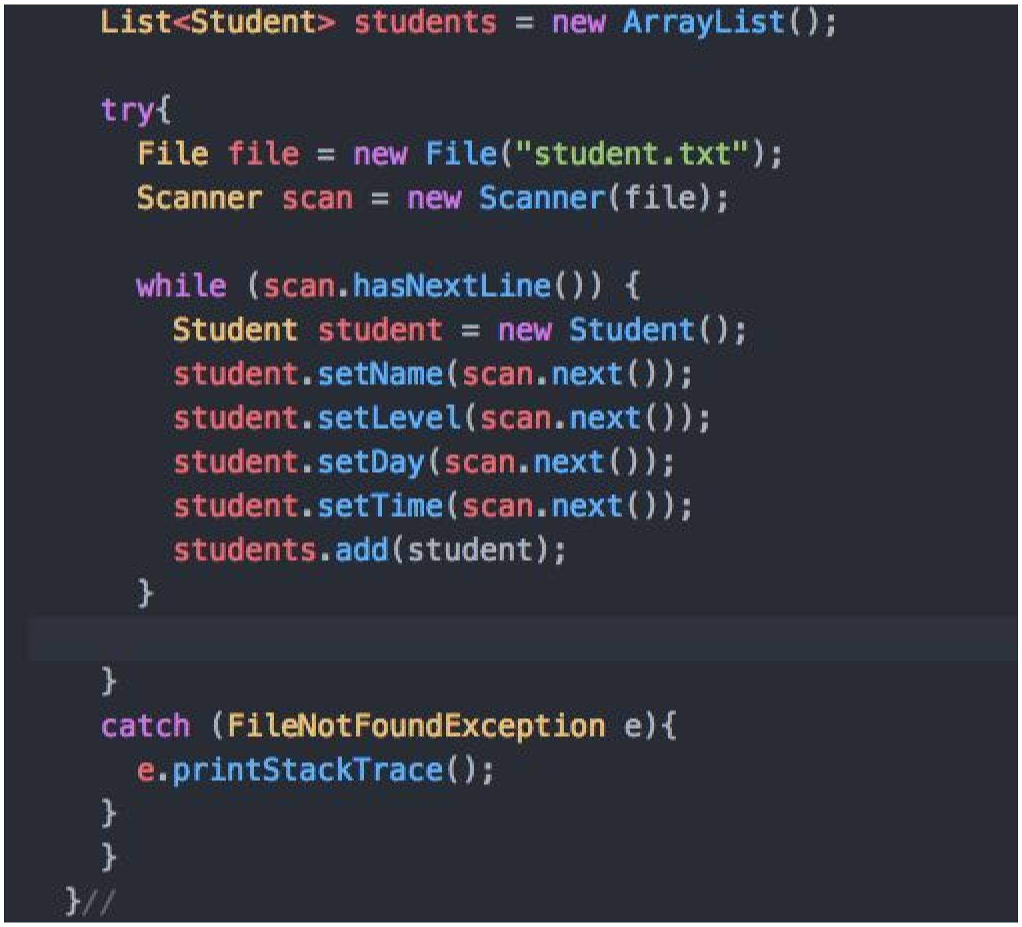
This is the setting of a student object, and those strings are correspondent to the information given in the txt file for each student



```
高子曦 J3 日 14:30
```

Figure 2: student.txt

This is a example of the format of student’s information provided by my customer in a txt file.



```
    List<Student> students = new ArrayList();

    try{
      File file = new File("student.txt");
      Scanner scan = new Scanner(file);

      while (scan.hasNextLine()) {
        Student student = new Student();
        student.setName(scan.next());
        student.setLevel(scan.next());
        student.setDay(scan.next());
        student.setTime(scan.next());
        students.add(student);
      }


    }
    catch (FileNotFoundException e){
      e.printStackTrace();
    }
    }
  }//
```

Figure 3: Manager.java

Manager.java is another java file that helps manage the students

This part of the code reads from the file, student.txt, and put the information into a List called students, which keep all the information about all the students.

The method I used for reading the txt file is the try and catch method. This way, I could be sure that the method would only stop only if there is no words left to read in the file.

As you can see, the order the information were added to the list are the same as the order in the txt file and how it is written in Student.java.

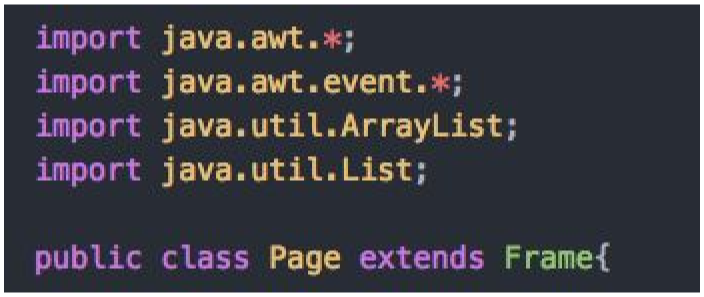

```java
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;

public class Page extends Frame{
```

Figure 4: Page.java

Page is the java file for my gui, which controls the output the users would see.

This page class inherited Frame, which means I could use methods from the Frame class.

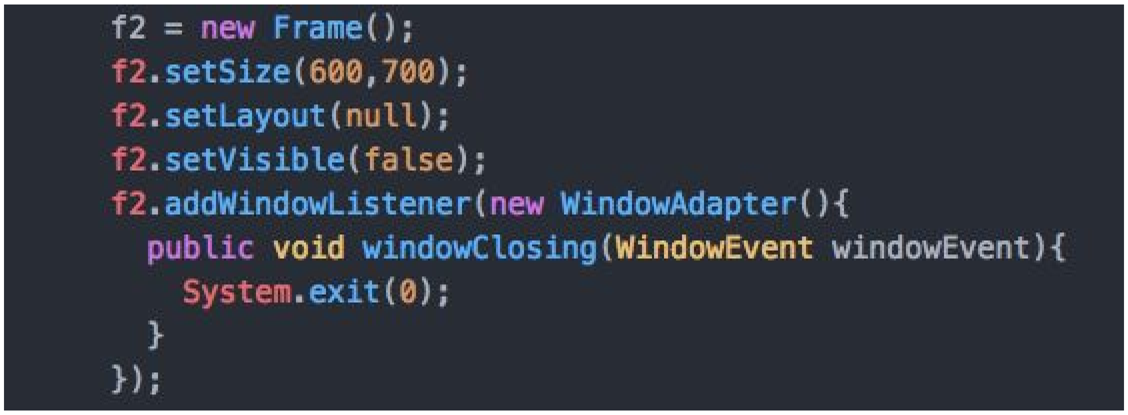
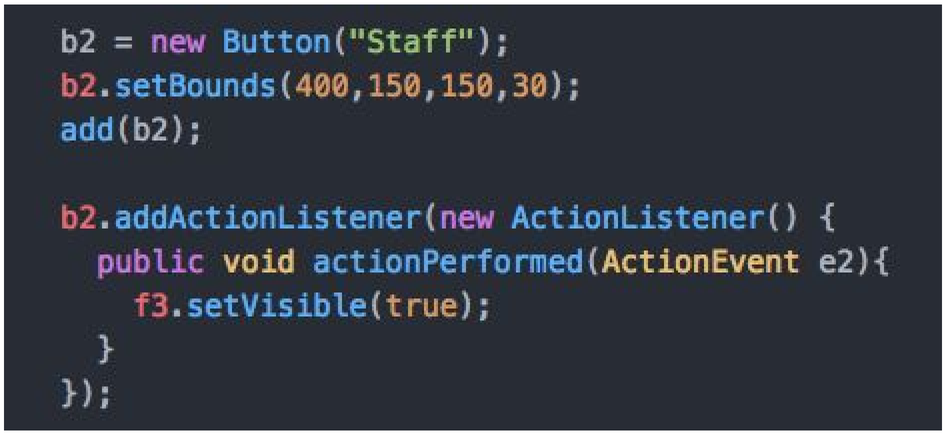

```java
    f2 = new Frame();
    f2.setSize(600,700);
    f2.setLayout(null);
    f2.setVisible(false);
    f2.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent windowEvent){
        System.exit(0);
      }
    });
```

Figure 5: Page.java

This is the set up for a frame created in Page.java. Except the first frame, I made all the other frames invisible to the users when the run the code. This is because I made each frame differently for different uses. For example, frame 2 is a page for the parents and customers, and frame 3 is a page for the staff working in the company. Thus, only frame 1 would show up first and guide people to make further decisions.



```
b2 = new Button("Staff");
b2.setBounds(400,150,150,30);
add(b2);

b2.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e2){
    f3.setVisible(true);
  }
});
```

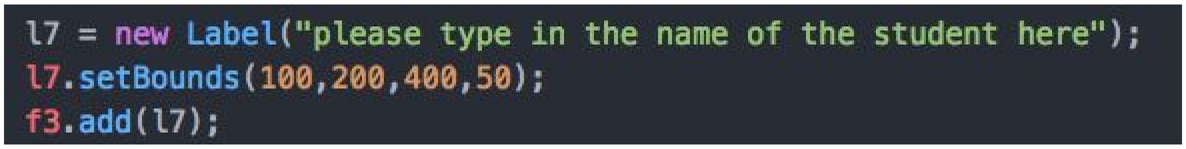
Figure 6: Page.java

b2 is a button created in frame 1. The action listener added to this button means that it would trigger a reaction if there is a action acted upon this button.

In this case, if the user click on the button, frame 3 would becomes visible and cover frame 1.

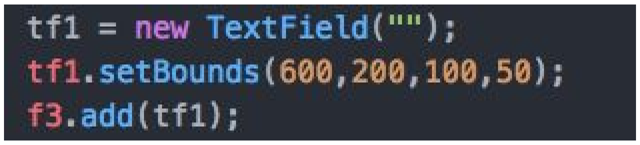
This could let me have multiple frames rather than having a big and messy one.

I created multiple frames also because it will look better and more like a program rather than just a tool being created.



```
l7 = new Label("please type in the name of the student here");
l7.setBounds(100,200,400,50);
f3.add(l7);
```

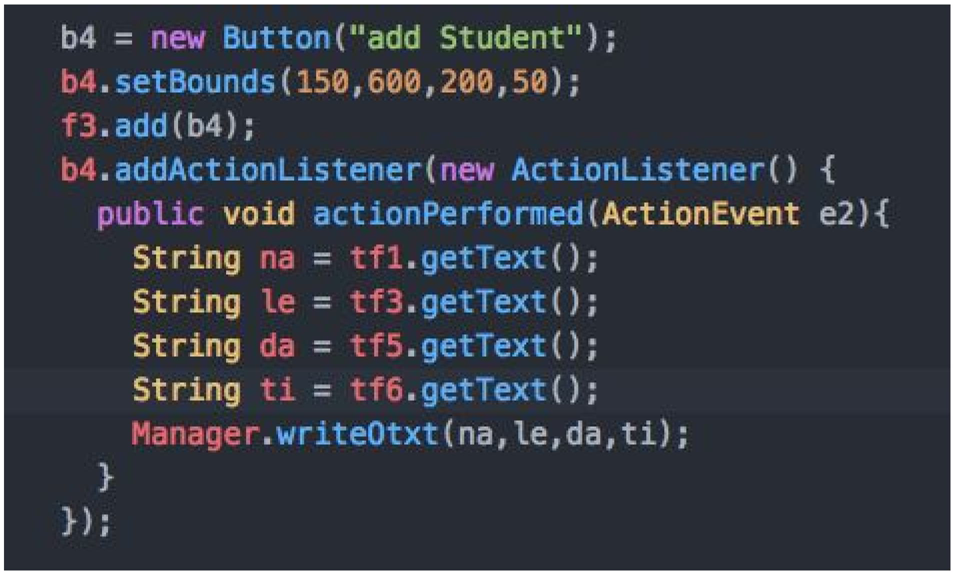
Figure 7: Page.java



```java
tf1 = new TextField("");
tf1.setBounds(600,200,100,50);
f3.add(tf1);
```

Figure 8: Page.java

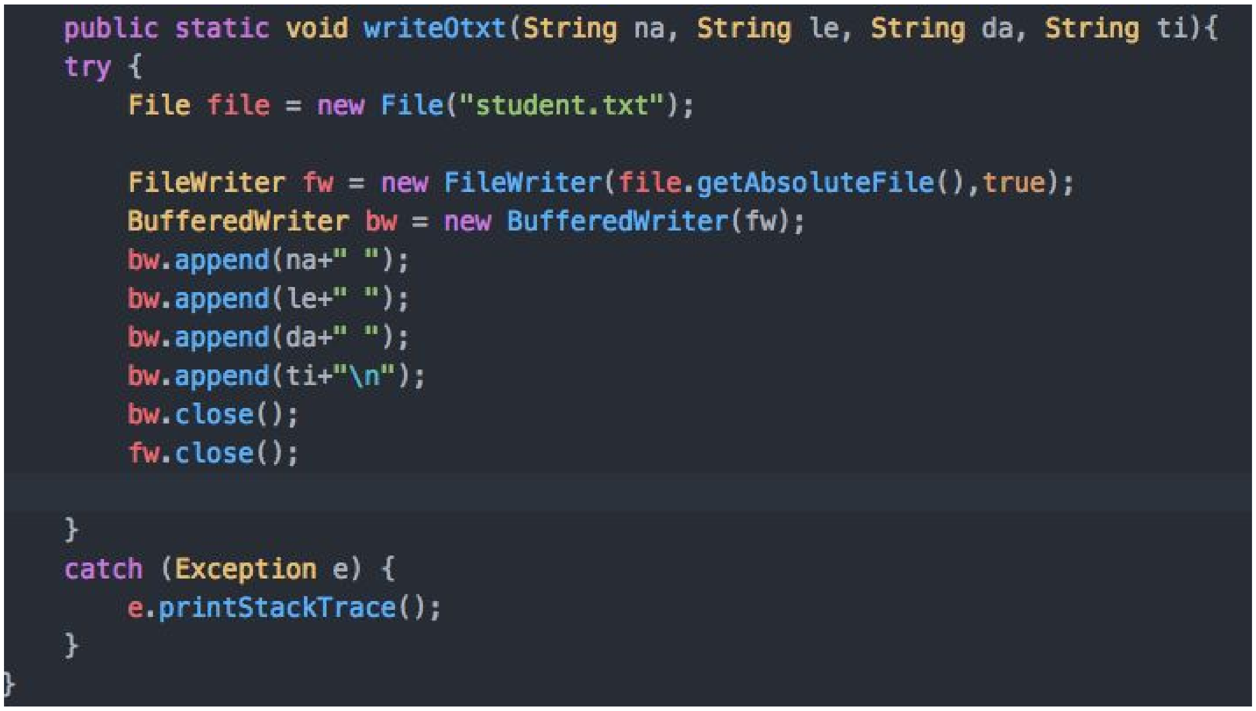
Figure 7 and 8 shows how I created the label that guide the users to put in the information and the text field which is where the user would put in the information.



```java
b4 = new Button("add Student");
b4.setBounds(150,600,200,50);
f3.add(b4);
b4.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e2){
    String na = tf1.getText();
    String le = tf3.getText();
    String da = tf5.getText();
    String ti = tf6.getText();
    Manager.writeOtxt(na,le,da,ti);
  }
});
```

Figure 9: Page.java

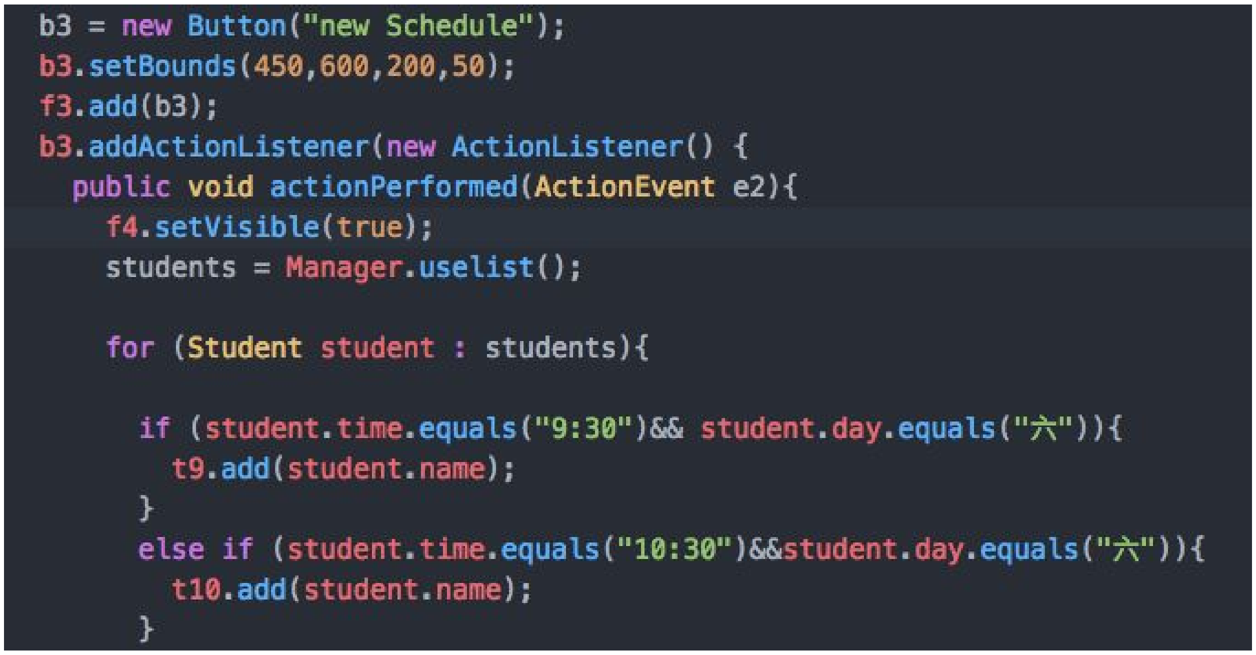
This is the action listener applied to the add student button. It would get the text from the text field I created and make them into four different strings. Then the program will run the writeOtxt method, which is written in the Manager.java.



```java
    public static void writeOtxt(String na, String le, String da, String ti){
    try {
        File file = new File("student.txt");

        FileWriter fw = new FileWriter(file.getAbsoluteFile(),true);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.append(na+" ");
        bw.append(le+" ");
        bw.append(da+" ");
        bw.append(ti+"\n");
        bw.close();
        fw.close();

    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 10: Manager.java

This is the writeOtxt method in Manager.java. It first decide which file we are writing to. A file writer and a bufferedwriter has been created. Then the bufferedwriter write all the information into the file and have a space between each of them and enter into the next line. Then we close the filewriter and the bufferedwriter.

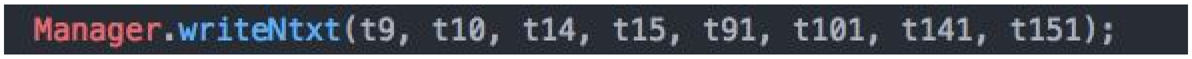


```
b3 = new Button("new Schedule");
b3.setBounds(450,600,200,50);
f3.add(b3);
b3.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e2){
    f4.setVisible(true);
    students = Manager.uselist();

    for (Student student : students){

      if (student.time.equals("9:30")&& student.day.equals("六")){
        t9.add(student.name);
      }
      else if (student.time.equals("10:30")&&student.day.equals("六")){
        t10.add(student.name);
      }
```

Figure 10: Page.java

This is where the button new schedule is created. If the user click on it, it will write a new schedule for the user.

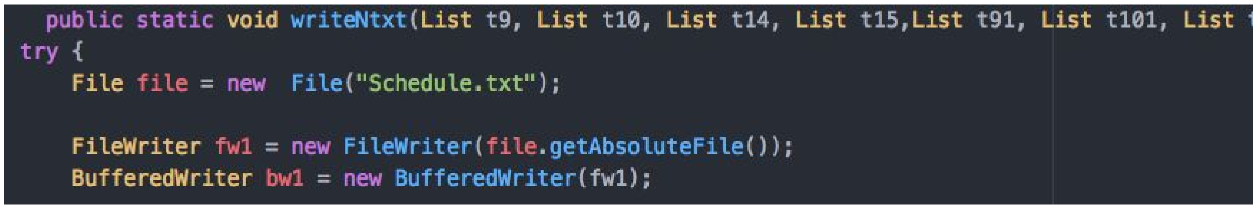
The action listener let the program loop through all the student objects in the list students. Then it will categorize them into into different arraylists based the time and date the students prefer. There are more if statement after this.



```
Manager.writeNtxt(t9, t10, t14, t15, t91, t101, t141, t151);
```

Figure 11: Page.java

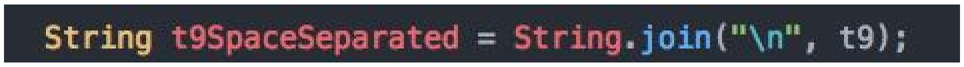
After all the students are put into different arraylists, the program will call on the writeNtxt method, which is written in the Manage.java. The method would use all the arraylists that are present in the parenthesis.



```
  public static void writeNtxt(List t9, List t10, List t14, List t15,List t91, List t101, List
try {
    File file = new  File("Schedule.txt");

    FileWriter fw1 = new FileWriter(file.getAbsoluteFile());
    BufferedWriter bw1 = new BufferedWriter(fw1);
```

Figure 12: Manager.java

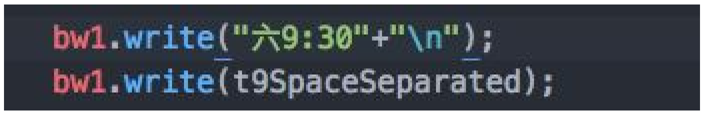
This writeNtxt is mostly the same as the writeOtxt file. It also requires a filewriter and a bufferedwriter.



```
  String t9SpaceSeparated = String.join("\n", t9);
```

Figure 13: Manager.java

This is how the list is converted into a string. This step is important because we can’t write a arraylist into a txt file.



```
  bw1.write("六9:30"+"\n");
  bw1.write(t9SpaceSeparated);
```

Figure 14: Manager.java

This is how the txt file is structured. It will first have the wordings about the date and time, and then it would write down the name of the students that would be in that class.

Word count: 773