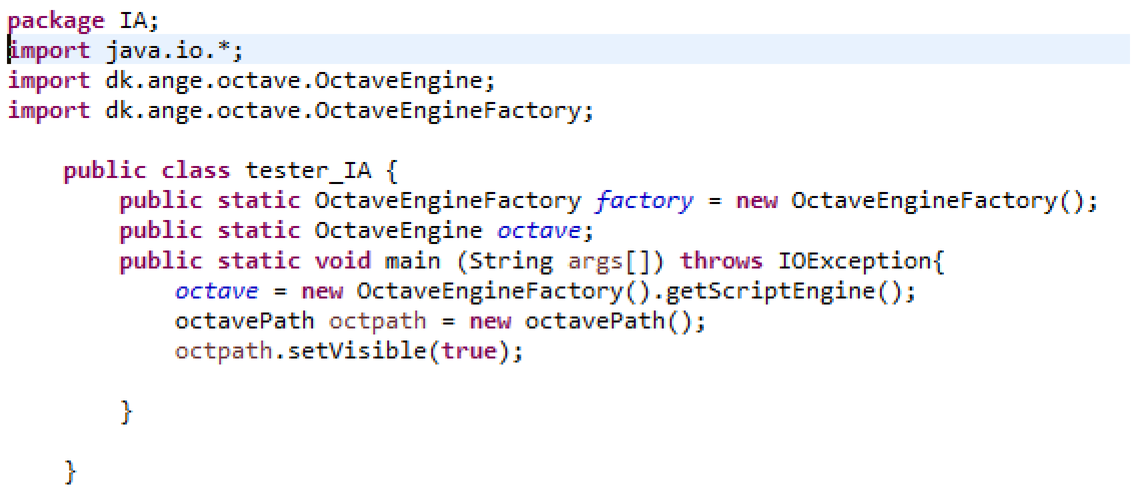
Criterion C: Development

The main techniques used are:

- Use filechooser to read and write data
- Use of 2d arrays and loops to output data
- Try and catch statements on possible errors
- Some basic GUI handling
- Parsing strings into doubles
- Use additional libraries
- Implementing a hierarchical composite data structure
- Inserting data to an ordered sequential file
- Merging two or more data structures
- Searching
- Implementation of basic logistic regression

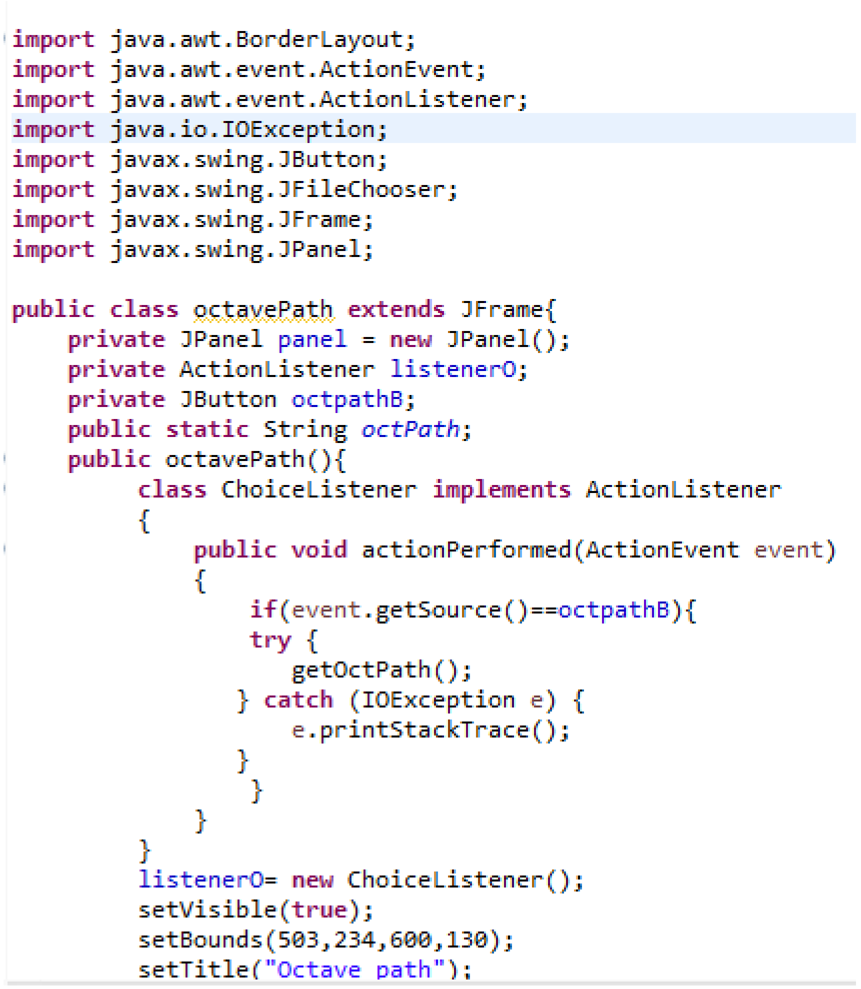
The libraries and variables used in tester_IA class:



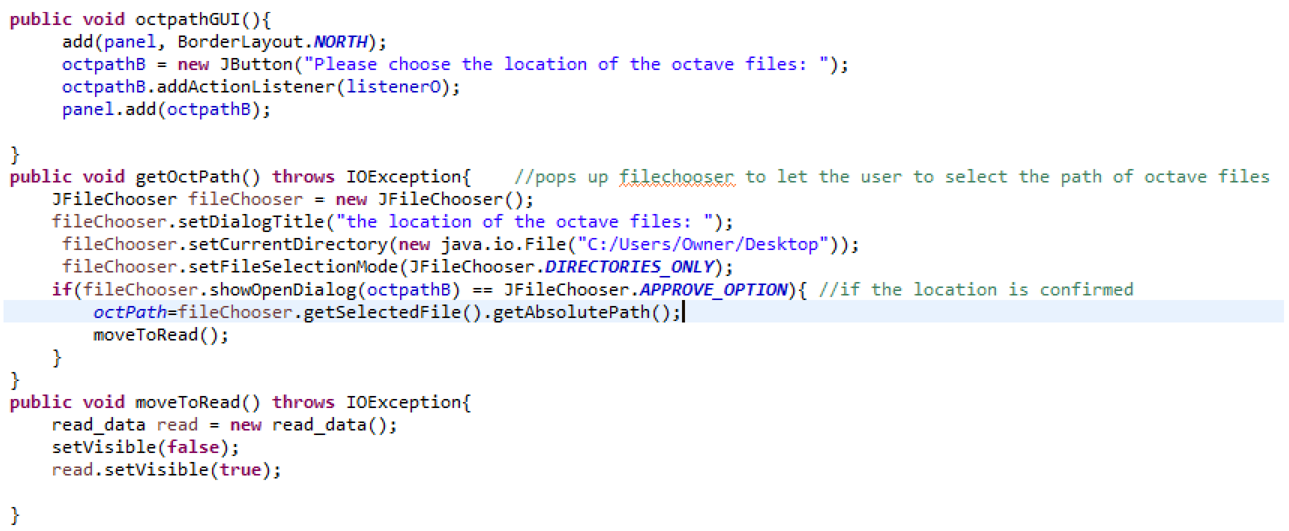
```
package IA;
import java.io.*;
import dk.ange.octave.OctaveEngine;
import dk.ange.octave.OctaveEngineFactory;

    public class tester_IA {
        public static OctaveEngineFactory factory = new OctaveEngineFactory();
        public static OctaveEngine octave;
        public static void main (String args[]) throws IOException{
            octave = new OctaveEngineFactory().getScriptEngine();
            octavePath octpath = new octavePath();
            octpath.setVisible(true);

        }

    }
```

The octave engine is called inside tester and is made static because only tester can implement the environmental variables for the launch configuration; the environmental variable can only be applied to tester. Therefore, the only solution is to declare the octave engine inside tester and make it static so call_octave class can use it.

The libraries used in octavePath class:



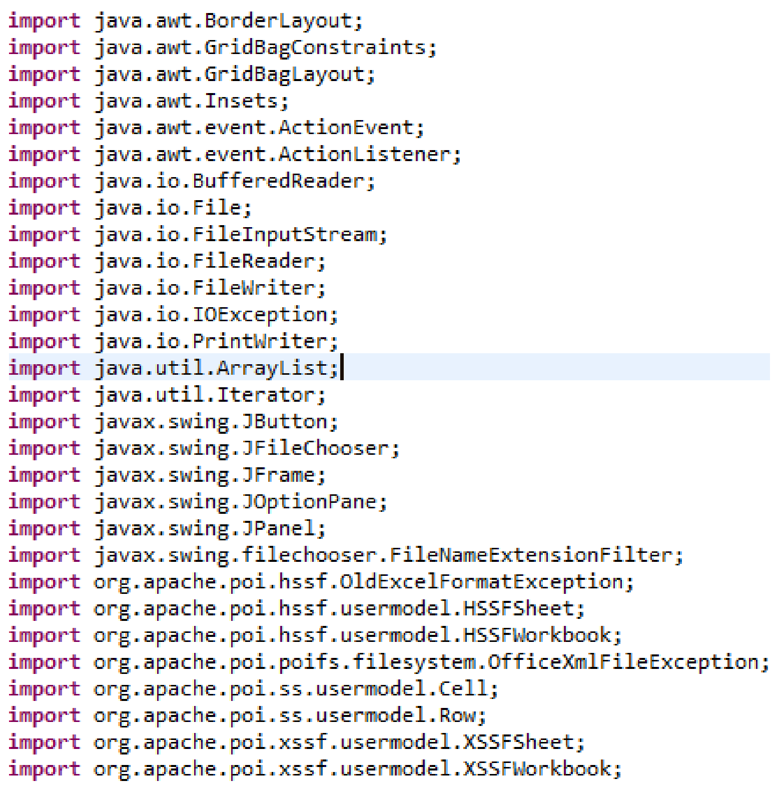
```java
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class octavePath extends JFrame{
    private JPanel panel = new JPanel();
    private ActionListener listener0;
    private JButton octpathB;
    public static String octPath;
    public octavePath(){
         class ChoiceListener implements ActionListener
         {
             public void actionPerformed(ActionEvent event)
             {
                 if(event.getSource()==octpathB){
                 try {
                    getOctPath();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                 }
             }
         }
         listener0= new ChoiceListener();
         setVisible(true);
         setBounds(503,234,600,130);
         setTitle("Octave path");
```



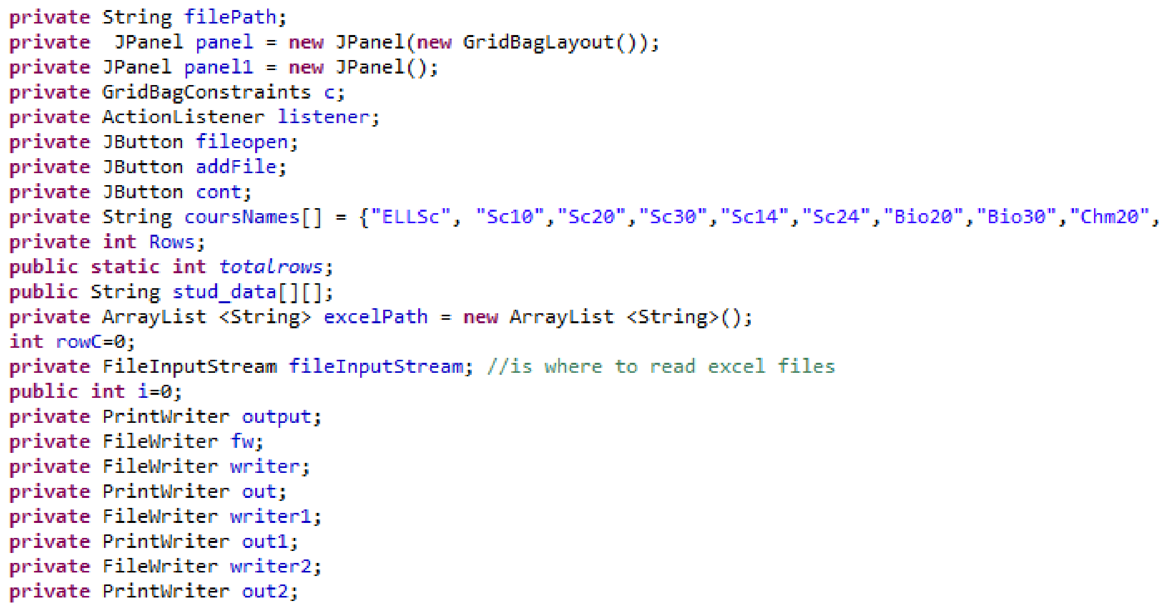
```java
public void octpathGUI(){
     add(panel, BorderLayout.NORTH);
     octpathB = new JButton("Please choose the location of the octave files: ");
     octpathB.addActionListener(listener0);
     panel.add(octpathB);

}
public void getOctPath() throws IOException{    //pops up filechooser to let the user to select the path of octave files
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("the location of the octave files: ");
     fileChooser.setCurrentDirectory(new java.io.File("C:/Users/Owner/Desktop"));
     fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    if(fileChooser.showOpenDialog(octpathB) == JFileChooser.APPROVE_OPTION){ //if the location is confirmed
        octPath=fileChooser.getSelectedFile().getAbsolutePath();
        moveToRead();
    }
}
public void moveToRead() throws IOException{
    read_data read = new read_data();
    setVisible(false);
    read.setVisible(true);

}
```

This file chooser can only open folder locations, not file locations. This is intentionally made since only the path of octave files is needed.

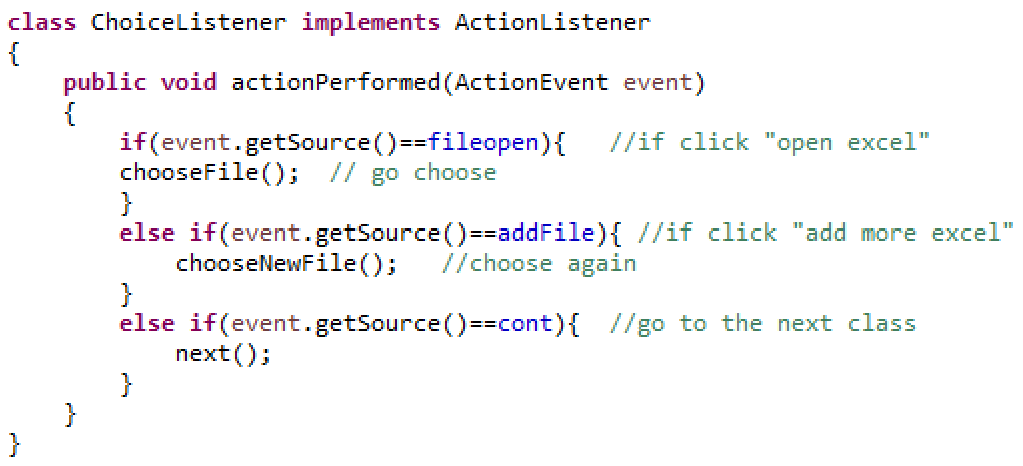
The libraries and variables used in read_data class:



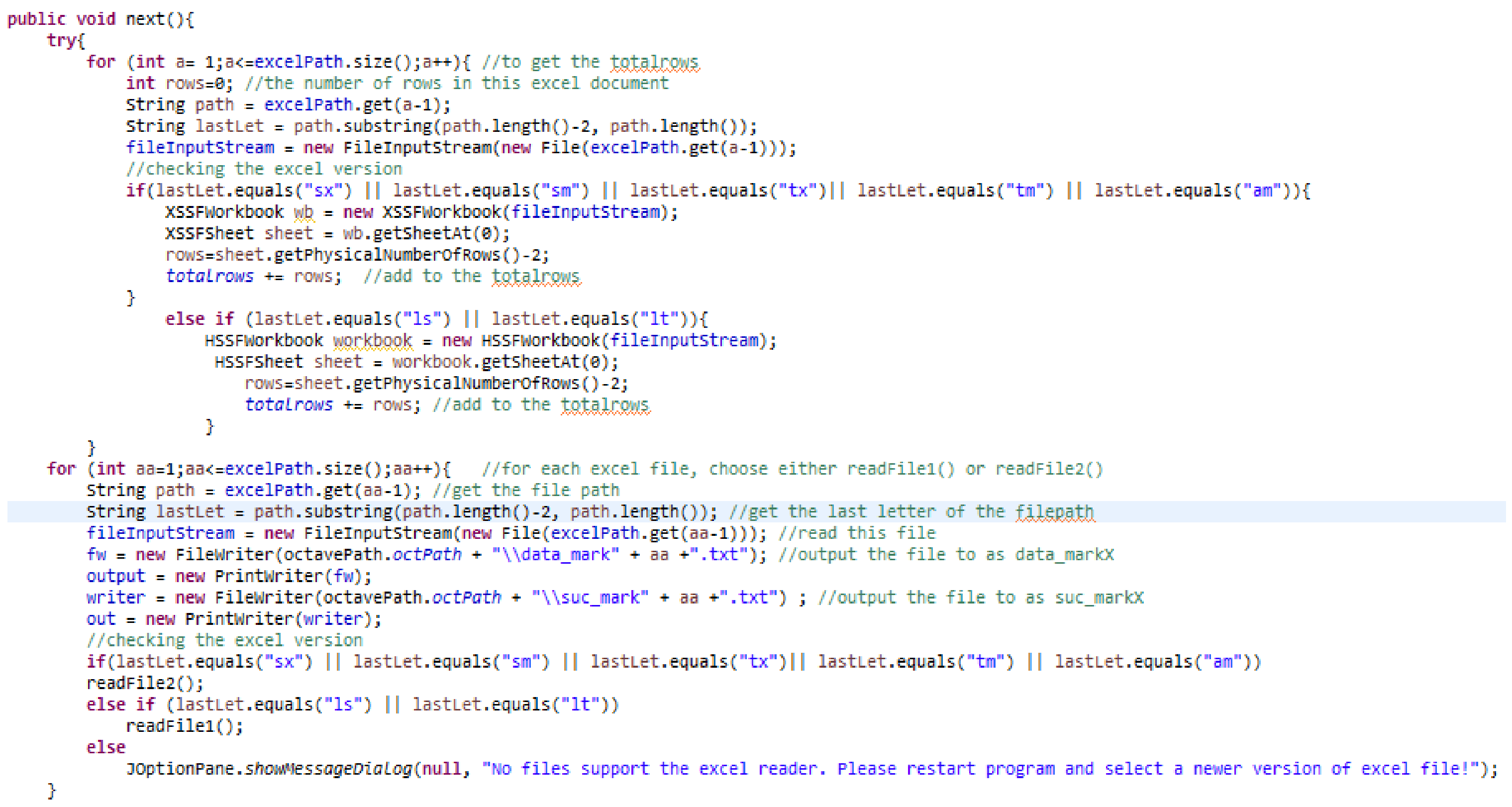
```java
import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.filechooser.FileNameExtensionFilter;
import org.apache.poi.hssf.OldExcelFormatException;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.poifs.filesystem.OfficeXmlFileException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```



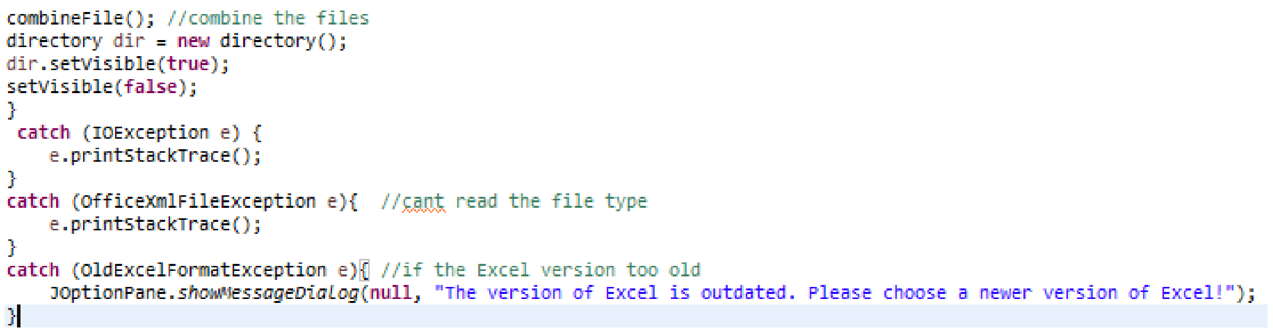
```java
private String filePath;
private  JPanel panel = new JPanel(new GridBagLayout());
private JPanel panel1 = new JPanel();
private GridBagConstraints c;
private ActionListener listener;
private JButton fileopen;
private JButton addFile;
private JButton cont;
private String coursNames[] = {"ELLSc", "Sc10","Sc20","Sc30","Sc14","Sc24","Bio20","Bio30","Chm20",
private int Rows;
public static int totalrows;
public String stud_data[][];
private ArrayList <String> excelPath = new ArrayList <String>();
int rowC=0;
private FileInputStream fileInputStream; //is where to read excel files
public int i=0;
private PrintWriter output;
private FileWriter fw;
private FileWriter writer;
private PrintWriter out;
private FileWriter writer1;
private PrintWriter out1;
private FileWriter writer2;
private PrintWriter out2;
```



```
class ChoiceListener implements ActionListener
{
    public void actionPerformed(ActionEvent event)
    {
        if(event.getSource()==fileopen){   //if click "open excel"
        chooseFile();  // go choose
        }
        else if(event.getSource()==addFile){ //if click "add more excel"
            chooseNewFile();   //choose again
        }
        else if(event.getSource()==cont){  //go to the next class
            next();
        }
    }
}
```

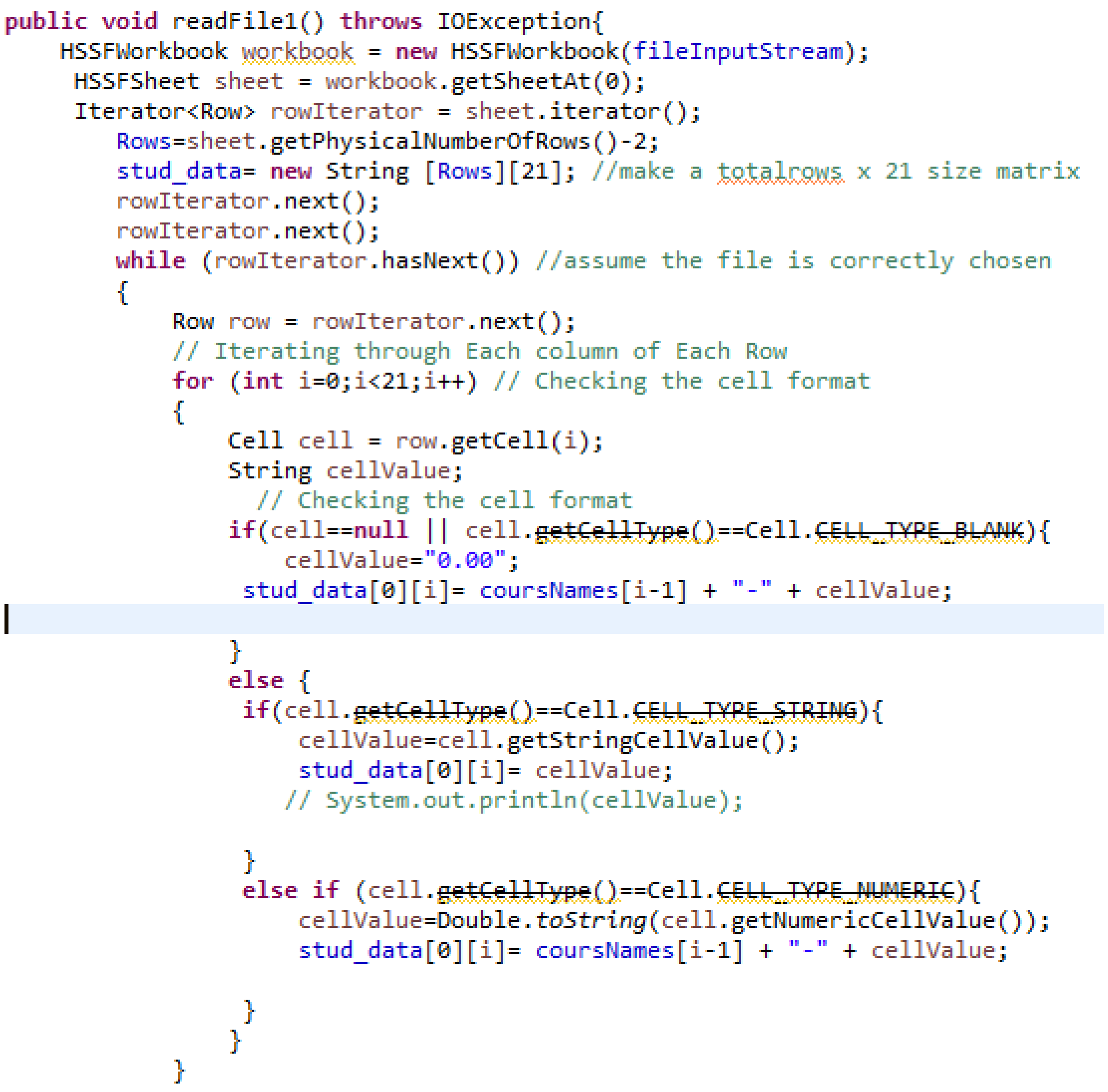


```
public void next(){
    try{
        for (int a= 1;a<=excelPath.size();a++){ //to get the totalrows
            int rows=0; //the number of rows in this excel document
            String path = excelPath.get(a-1);
            String lastLet = path.substring(path.length()-2, path.length());
            fileInputStream = new FileInputStream(new File(excelPath.get(a-1)));
            //checking the excel version
            if(lastLet.equals("sx") || lastLet.equals("sm") || lastLet.equals("tx")|| lastLet.equals("tm") || lastLet.equals("am")){
                XSSFWorkbook wb = new XSSFWorkbook(fileInputStream);
                XSSFSheet sheet = wb.getSheetAt(0);
                rows=sheet.getPhysicalNumberOfRows()-2;
                totalrows += rows;  //add to the totalrows
            }
                else if (lastLet.equals("ls") || lastLet.equals("lt")){
                    HSSFWorkbook workbook = new HSSFWorkbook(fileInputStream);
                     HSSFSheet sheet = workbook.getSheetAt(0);
                        rows=sheet.getPhysicalNumberOfRows()-2;
                        totalrows += rows; //add to the totalrows
                    }
        }
    for (int aa=1;aa<=excelPath.size();aa++){   //for each excel file, choose either readFile1() or readFile2()
        String path = excelPath.get(aa-1); //get the file path
        String lastLet = path.substring(path.length()-2, path.length()); //get the last letter of the filepath
        fileInputStream = new FileInputStream(new File(excelPath.get(aa-1))); //read this file
        fw = new FileWriter(octavePath.octPath + "\\data_mark" + aa +".txt"); //output the file to as data_markX
        output = new PrintWriter(fw);
        writer = new FileWriter(octavePath.octPath + "\\suc_mark" + aa +".txt") ; //output the file to as suc_markX
        out = new PrintWriter(writer);
        //checking the excel version
        if(lastLet.equals("sx") || lastLet.equals("sm") || lastLet.equals("tx")|| lastLet.equals("tm") || lastLet.equals("am"))
        readFile2();
        else if (lastLet.equals("ls") || lastLet.equals("lt"))
            readFile1();
        else
            JOptionPane.showMessageDialog(null, "No files support the excel reader. Please restart program and select a newer version of excel file!");
    }
```



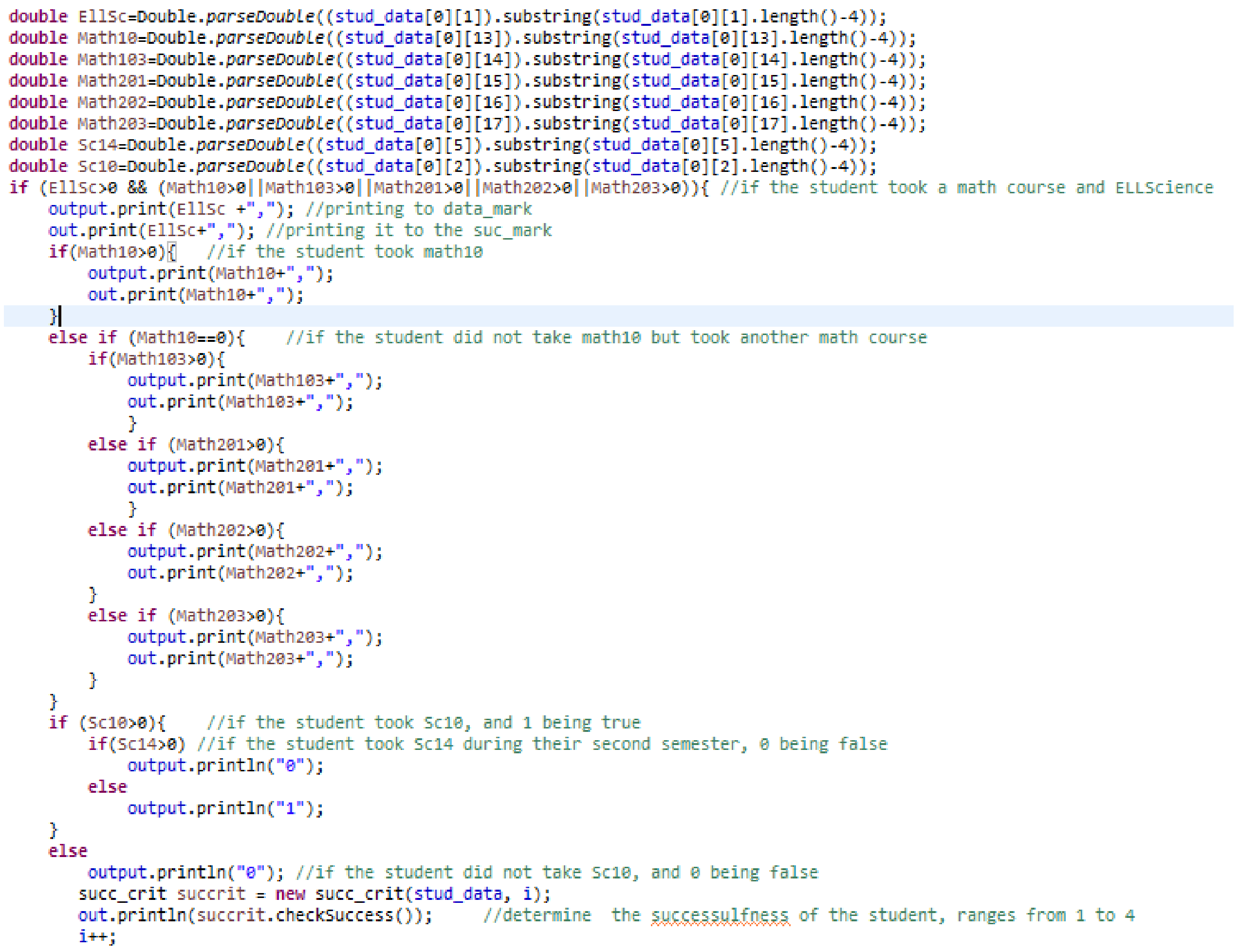
```
combineFile(); //combine the files
directory dir = new directory();
dir.setVisible(true);
setVisible(false);
}
 catch (IOException e) {
    e.printStackTrace();
}
catch (OfficeXmlFileException e){  //cant read the file type
    e.printStackTrace();
}
catch (OldExcelFormatException e){ //if the Excel version too old
    JOptionPane.showMessageDialog(null, "The version of Excel is outdated. Please choose a newer version of Excel!");
}
```

After the user finishes choosing his excel files, the next() method will first get the total row of all the excel files. The number total rows is important because 2d array that stores the student's successfulness needs to be created as a total rows*2 matrix. Then, PrintWriter *output* and *out* will be assigned as data_markX and suc_markX as temporary files. Later on, these temporary files will be merged together, and then deleted. next() will then assign which readFile method to read the excel file (Javainterviewpoint, 2017).



```java
public void readFile1() throws IOException{
    HSSFWorkbook workbook = new HSSFWorkbook(fileInputStream);
     HSSFSheet sheet = workbook.getSheetAt(0);
     Iterator<Row> rowIterator = sheet.iterator();
        Rows=sheet.getPhysicalNumberOfRows()-2;
        stud_data= new String [Rows][21]; //make a totalrows x 21 size matrix
        rowIterator.next();
        rowIterator.next();
        while (rowIterator.hasNext()) //assume the file is correctly chosen
        {
            Row row = rowIterator.next();
            // Iterating through Each column of Each Row
            for (int i=0;i<21;i++) // Checking the cell format
            {
                Cell cell = row.getCell(i);
                String cellValue;
                  // Checking the cell format
                if(cell==null || cell.~~getCellType~~()==Cell.~~CELL_TYPE_BLANK~~){
                    cellValue="0.00";
                 stud_data[0][i]= coursNames[i-1] + "-" + cellValue;

                }
                else {
                 if(cell.~~getCellType~~()==Cell.~~CELL_TYPE_STRING~~){
                     cellValue=cell.getStringCellValue();
                     stud_data[0][i]= cellValue;
                    // System.out.println(cellValue);

                 }
                 else if (cell.~~getCellType~~()==Cell.~~CELL_TYPE_NUMERIC~~){
                     cellValue=Double.toString(cell.getNumericCellValue());
                     stud_data[0][i]= coursNames[i-1] + "-" + cellValue;

                 }
                }
            }
```

Then, to read the file with less RAM, only one line of the student data is read each time, and outputting them into one line of data_markX and suc_markX.



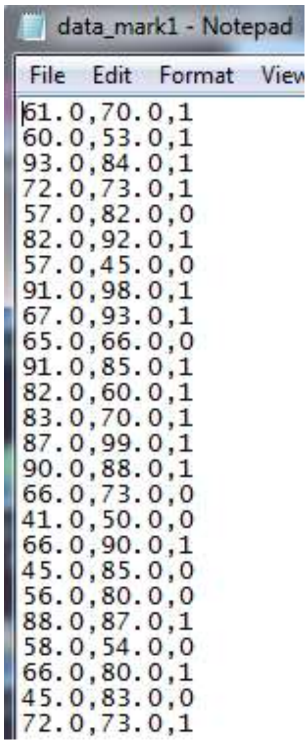
```
double EllSc=Double.parseDouble((stud_data[0][1]).substring(stud_data[0][1].length()-4));
double Math10=Double.parseDouble((stud_data[0][13]).substring(stud_data[0][13].length()-4));
double Math103=Double.parseDouble((stud_data[0][14]).substring(stud_data[0][14].length()-4));
double Math201=Double.parseDouble((stud_data[0][15]).substring(stud_data[0][15].length()-4));
double Math202=Double.parseDouble((stud_data[0][16]).substring(stud_data[0][16].length()-4));
double Math203=Double.parseDouble((stud_data[0][17]).substring(stud_data[0][17].length()-4));
double Sc14=Double.parseDouble((stud_data[0][5]).substring(stud_data[0][5].length()-4));
double Sc10=Double.parseDouble((stud_data[0][2]).substring(stud_data[0][2].length()-4));
if (EllSc>0 && (Math10>0||Math103>0||Math201>0||Math202>0||Math203>0)){ //if the student took a math course and ELLScience
    output.print(EllSc +","); //printing to data_mark
    out.print(EllSc+","); //printing it to the suc_mark
    if(Math10>0){   //if the student took math10
        output.print(Math10+",");
        out.print(Math10+",");
    }
    else if (Math10==0){    //if the student did not take math10 but took another math course
        if(Math103>0){
            output.print(Math103+",");
            out.print(Math103+",");
            }
        else if (Math201>0){
            output.print(Math201+",");
            out.print(Math201+",");
            }
        else if (Math202>0){
            output.print(Math202+",");
            out.print(Math202+",");
        }
        else if (Math203>0){
            output.print(Math203+",");
            out.print(Math203+",");
        }
    }
    if (Sc10>0){    //if the student took Sc10, and 1 being true
        if(Sc14>0) //if the student took Sc14 during their second semester, 0 being false
            output.println("0");
        else
            output.println("1");
    }
    else
        output.println("0"); //if the student did not take Sc10, and 0 being false
       succ_crit succrit = new succ_crit(stud_data, i);
       out.println(succrit.checkSuccess());     //determine  the successulfness of the student, ranges from 1 to 4
       i++;
```

```
     else {  //still put the student in successful crit, but not in data set
          succ_crit succrit = new succ_crit(stud_data, i);
           succrit.checkSuccess();
           i++;
     }


     //System.out.println(Arrays.deepToString(stud_data));

}
    fileInputStream.close();
    fw.close();
    output.close();
    writer.close();
    out.close();
}
```

Student's marks that are outputted into data_markX are also outputted into suc_markX. The int checksuccess() method in the succ_crit class will be called with the student data and integer i, which is a count for the remaining students. Since some students might drop out from school, the possibility of the student not taking ELLScience and Math exists. To exclude dropped out students (outliers) from the data set, a return value is added. It will add the student's successfulness to *success*[][], and also return a value (success level) ranged from 1 to 4. This way, the student took ELLScience and Math can get the success level , while the outliars can be added to the *success*[][] and not into the suc_markX and data_markX.

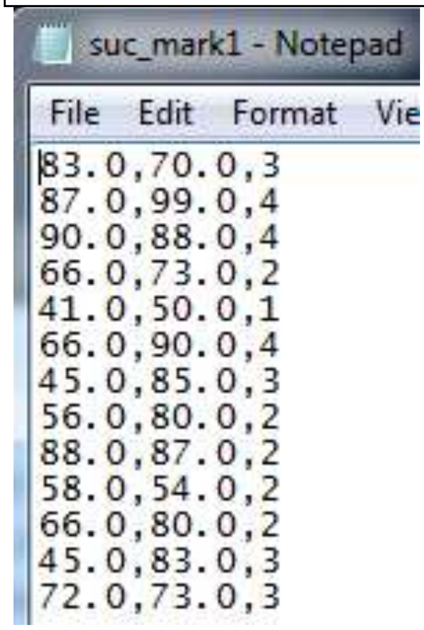
A hierarchical composite data structure is also used, since *stud_data* and the text file outputted to the octave file path are stored in a fixed data structure.

The data_mark1.txt that was outputted to the octave files will look like:



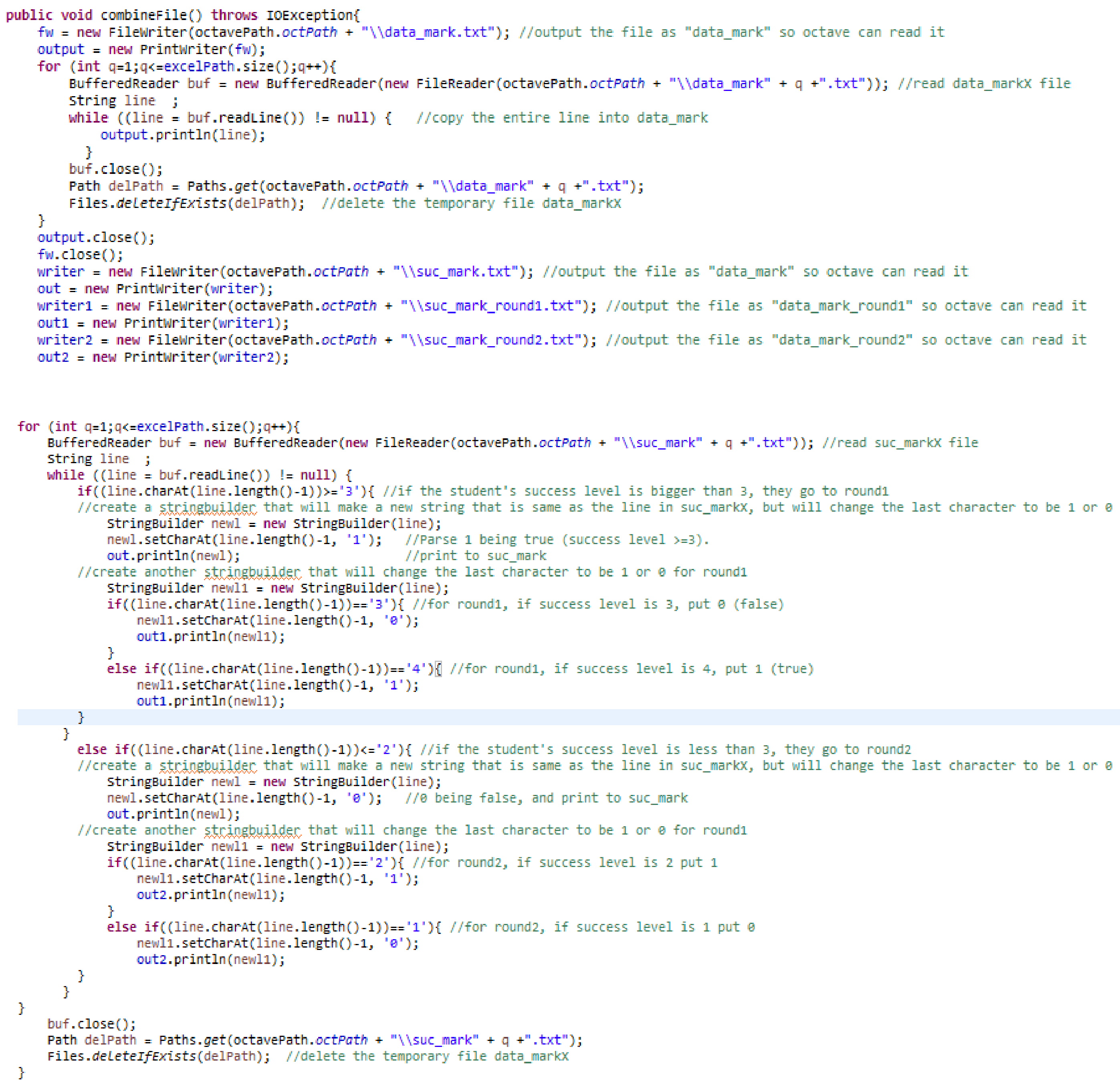
```
61.0,70.0,1
60.0,53.0,1
93.0,84.0,1
72.0,73.0,1
57.0,82.0,0
82.0,92.0,1
57.0,45.0,0
91.0,98.0,1
67.0,93.0,1
65.0,66.0,0
91.0,85.0,1
82.0,60.0,1
83.0,70.0,1
87.0,99.0,1
90.0,88.0,1
66.0,73.0,0
41.0,50.0,0
66.0,90.0,1
45.0,85.0,0
56.0,80.0,0
88.0,87.0,1
58.0,54.0,0
66.0,80.0,1
45.0,83.0,0
72.0,73.0,1
```

The suc_mark1.txt that was outputted to the octave files will look like:



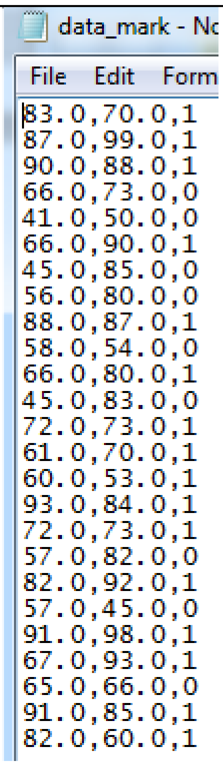
```
83.0,70.0,3
87.0,99.0,4
90.0,88.0,4
66.0,73.0,2
41.0,50.0,1
66.0,90.0,4
45.0,85.0,3
56.0,80.0,2
88.0,87.0,2
58.0,54.0,2
66.0,80.0,2
45.0,83.0,3
72.0,73.0,3
```

Then, the combineFile() method will combine the data_markX and suc_markX files together, which shows merging two data-structures, and inserting data into a sequential file:



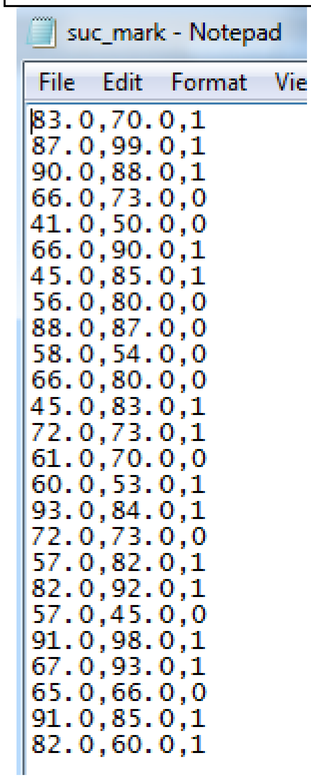
```java
public void combineFile() throws IOException{
    fw = new FileWriter(octavePath.octPath + "\\data_mark.txt"); //output the file as "data_mark" so octave can read it
    output = new PrintWriter(fw);
    for (int q=1;q<=excelPath.size();q++){
        BufferedReader buf = new BufferedReader(new FileReader(octavePath.octPath + "\\data_mark" + q +".txt")); //read data_markX file
        String line  ;
        while ((line = buf.readLine()) != null) {   //copy the entire line into data_mark
            output.println(line);
          }
        buf.close();
        Path delPath = Paths.get(octavePath.octPath + "\\data_mark" + q +".txt");
        Files.deleteIfExists(delPath);  //delete the temporary file data_markX
    }
    output.close();
    fw.close();
    writer = new FileWriter(octavePath.octPath + "\\suc_mark.txt"); //output the file as "data_mark" so octave can read it
    out = new PrintWriter(writer);
    writer1 = new FileWriter(octavePath.octPath + "\\suc_mark_round1.txt"); //output the file as "data_mark_round1" so octave can read it
    out1 = new PrintWriter(writer1);
    writer2 = new FileWriter(octavePath.octPath + "\\suc_mark_round2.txt"); //output the file as "data_mark_round2" so octave can read it
    out2 = new PrintWriter(writer2);



 for (int q=1;q<=excelPath.size();q++){
     BufferedReader buf = new BufferedReader(new FileReader(octavePath.octPath + "\\suc_mark" + q +".txt")); //read suc_markX file
     String line  ;
     while ((line = buf.readLine()) != null) {
         if((line.charAt(line.length()-1))>='3'){ //if the student's success level is bigger than 3, they go to round1
         //create a stringbuilder that will make a new string that is same as the line in suc_markX, but will change the last character to be 1 or 0
             StringBuilder newl = new StringBuilder(line);
             newl.setCharAt(line.length()-1, '1');   //Parse 1 being true (success level >=3).
             out.println(newl);                      //print to suc_mark
         //create another stringbuilder that will change the last character to be 1 or 0 for round1
             StringBuilder newl1 = new StringBuilder(line);
             if((line.charAt(line.length()-1))=='3'){ //for round1, if success level is 3, put 0 (false)
                 newl1.setCharAt(line.length()-1, '0');
                 out1.println(newl1);
             }
             else if((line.charAt(line.length()-1))=='4'){ //for round1, if success level is 4, put 1 (true)
                 newl1.setCharAt(line.length()-1, '1');
                 out1.println(newl1);
         }
       }
         else if((line.charAt(line.length()-1))<='2'){ //if the student's success level is less than 3, they go to round2
         //create a stringbuilder that will make a new string that is same as the line in suc_markX, but will change the last character to be 1 or 0
             StringBuilder newl = new StringBuilder(line);
             newl.setCharAt(line.length()-1, '0');   //0 being false, and print to suc_mark
             out.println(newl);
         //create another stringbuilder that will change the last character to be 1 or 0 for round1
             StringBuilder newl1 = new StringBuilder(line);
             if((line.charAt(line.length()-1))=='2'){ //for round2, if success level is 2 put 1
                 newl1.setCharAt(line.length()-1, '1');
                 out2.println(newl1);
             }
             else if((line.charAt(line.length()-1))=='1'){ //for round2, if success level is 1 put 0
                 newl1.setCharAt(line.length()-1, '0');
                 out2.println(newl1);
         }
       }
 }
     buf.close();
     Path delPath = Paths.get(octavePath.octPath + "\\suc_mark" + q +".txt");
     Files.deleteIfExists(delPath);  //delete the temporary file data_markX
 }
```

The data_mark.txt (final) will look like:



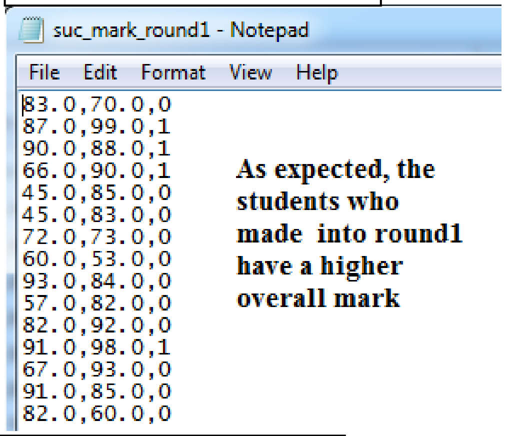
```
83.0,70.0,1
87.0,99.0,1
90.0,88.0,1
66.0,73.0,0
41.0,50.0,0
66.0,90.0,1
45.0,85.0,0
56.0,80.0,0
88.0,87.0,1
58.0,54.0,0
66.0,80.0,1
45.0,83.0,0
72.0,73.0,1
61.0,70.0,1
60.0,53.0,1
93.0,84.0,1
72.0,73.0,1
57.0,82.0,0
82.0,92.0,1
57.0,45.0,0
91.0,98.0,1
67.0,93.0,1
65.0,66.0,0
91.0,85.0,1
82.0,60.0,1
```

The suc_mark.txt (final) will look like:



```
83.0,70.0,1
87.0,99.0,1
90.0,88.0,1
66.0,73.0,0
41.0,50.0,0
66.0,90.0,1
45.0,85.0,1
56.0,80.0,0
88.0,87.0,0
58.0,54.0,0
66.0,80.0,0
45.0,83.0,1
72.0,73.0,1
61.0,70.0,0
60.0,53.0,1
93.0,84.0,1
72.0,73.0,0
57.0,82.0,1
82.0,92.0,1
57.0,45.0,0
91.0,98.0,1
67.0,93.0,1
65.0,66.0,0
91.0,85.0,1
82.0,60.0,1
```

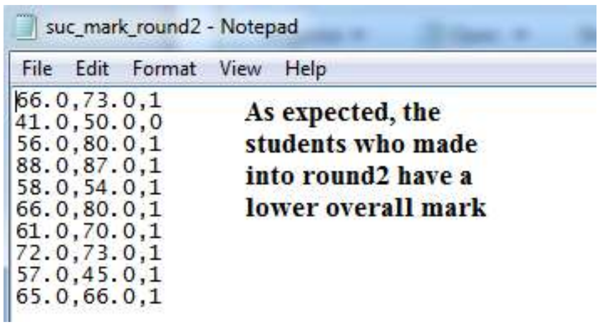
The suc_mark_round1.txt will look like:



```
83.0,70.0,0
87.0,99.0,1
90.0,88.0,1
66.0,90.0,1
45.0,85.0,0
45.0,83.0,0
72.0,73.0,0
60.0,53.0,0
93.0,84.0,0
57.0,82.0,0
82.0,92.0,0
91.0,98.0,1
67.0,93.0,0
91.0,85.0,0
82.0,60.0,0
```

As expected, the students who made into round1 have a higher overall mark

The suc_mark_round2.txt will look like:

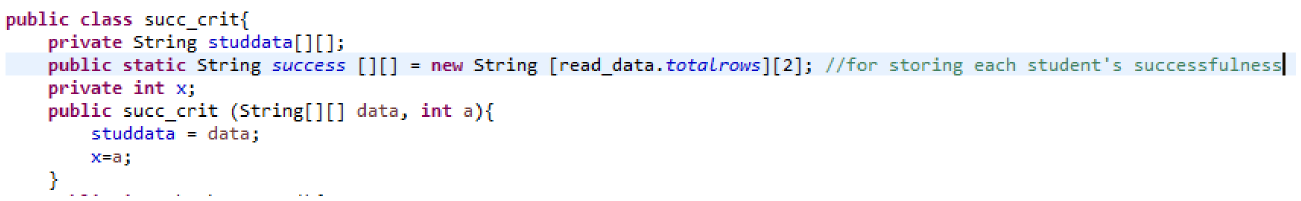


```
66.0,73.0,1
41.0,50.0,0
56.0,80.0,1
88.0,87.0,1
58.0,54.0,1
66.0,80.0,1
61.0,70.0,1
72.0,73.0,1
57.0,45.0,1
65.0,66.0,1
```

As expected, the students who made into round2 have a lower overall mark

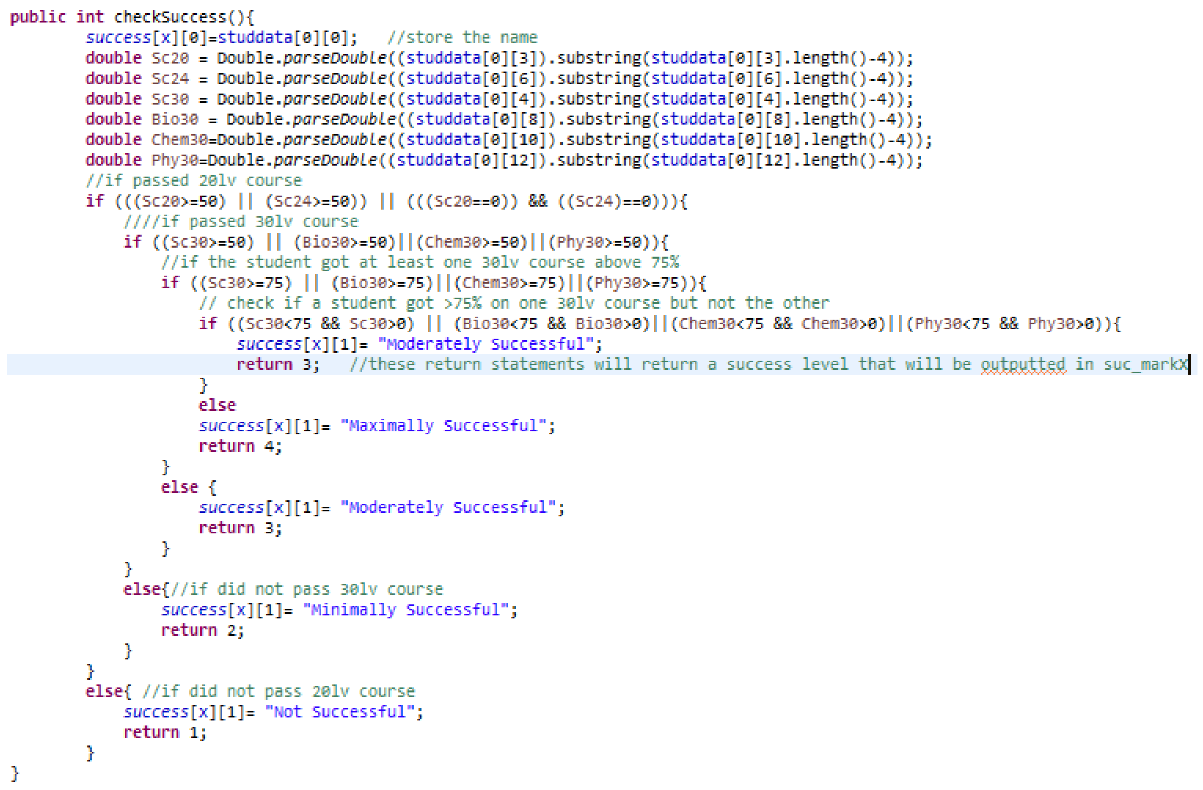
Creating temporary files and then merging them together is appropriate because it saves RAM, and they will be used in machine learning algorithms.

The variables used in succ_crit class:



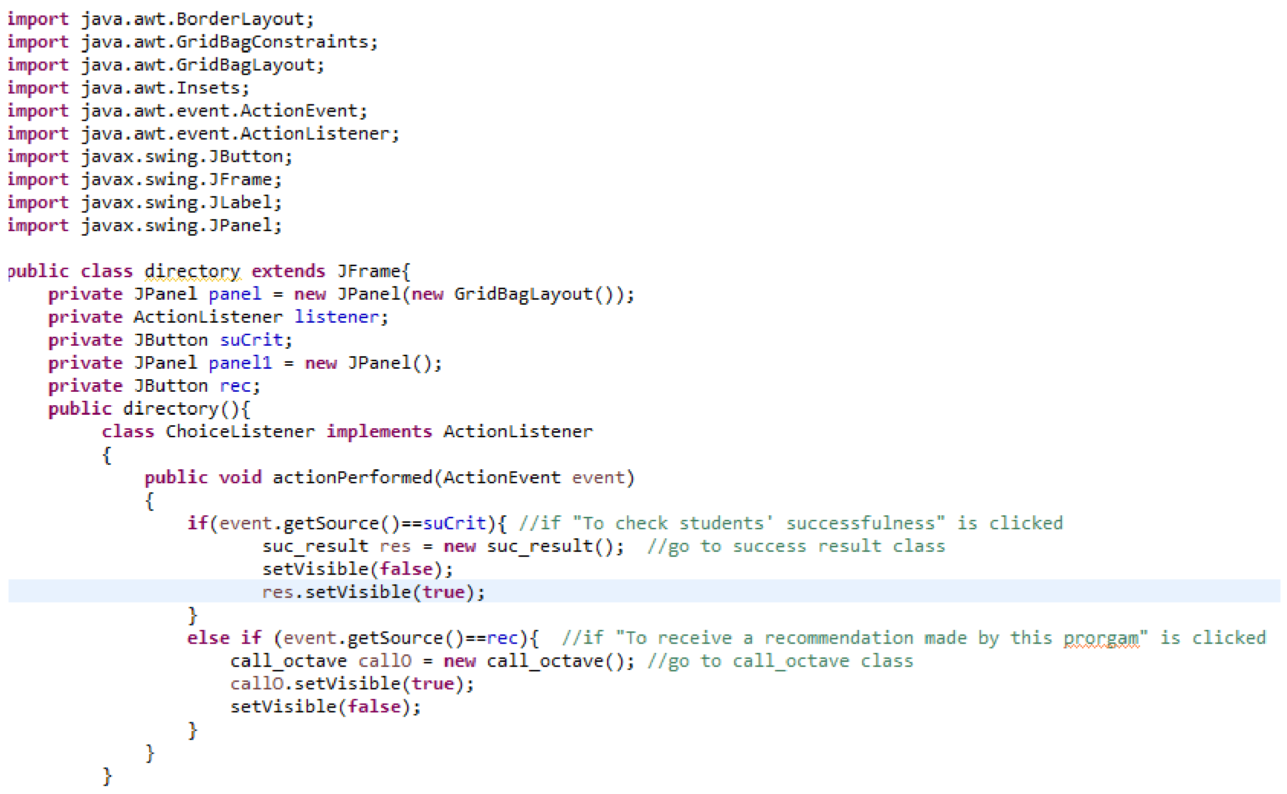
```java
public class succ_crit{
    private String studdata[][];
    public static String success [][] = new String [read_data.totalrows][2]; //for storing each student's successfulness
    private int x;
    public succ_crit (String[][] data, int a){
        studdata = data;
        x=a;
    }
```

Int x is assigned from int a, which is int i in read_data class, and will be used as a counter.



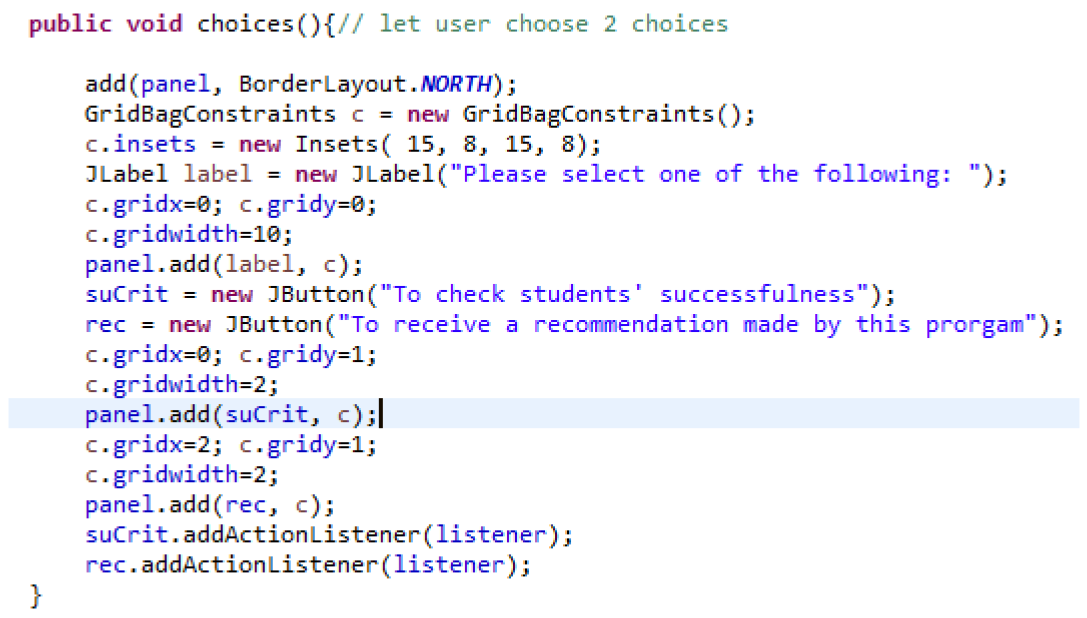
```java
public int checkSuccess(){
        success[x][0]=studdata[0][0];   //store the name
        double Sc20 = Double.parseDouble((studdata[0][3]).substring(studdata[0][3].length()-4));
        double Sc24 = Double.parseDouble((studdata[0][6]).substring(studdata[0][6].length()-4));
        double Sc30 = Double.parseDouble((studdata[0][4]).substring(studdata[0][4].length()-4));
        double Bio30 = Double.parseDouble((studdata[0][8]).substring(studdata[0][8].length()-4));
        double Chem30=Double.parseDouble((studdata[0][10]).substring(studdata[0][10].length()-4));
        double Phy30=Double.parseDouble((studdata[0][12]).substring(studdata[0][12].length()-4));
        //if passed 20lv course
        if (((Sc20>=50) || (Sc24>=50)) || (((Sc20==0)) && ((Sc24)==0))){
            ////if passed 30lv course
            if ((Sc30>=50) || (Bio30>=50)||(Chem30>=50)||(Phy30>=50)){
                //if the student got at least one 30lv course above 75%
                if ((Sc30>=75) || (Bio30>=75)||(Chem30>=75)||(Phy30>=75)){
                    // check if a student got >75% on one 30lv course but not the other
                    if ((Sc30<75 && Sc30>0) || (Bio30<75 && Bio30>0)||(Chem30<75 && Chem30>0)||(Phy30<75 && Phy30>0)){
                        success[x][1]= "Moderately Successful";
                        return 3;   //these return statements will return a success level that will be outputted in suc_markX
                    }
                    else
                    success[x][1]= "Maximally Successful";
                    return 4;
                }
                else {
                    success[x][1]= "Moderately Successful";
                    return 3;
                }
            }
            else{//if did not pass 30lv course
                success[x][1]= "Minimally Successful";
                return 2;
            }
        }
        else{ //if did not pass 20lv course
            success[x][1]= "Not Successful";
            return 1;
        }
}
```

The libraries used in directory class:

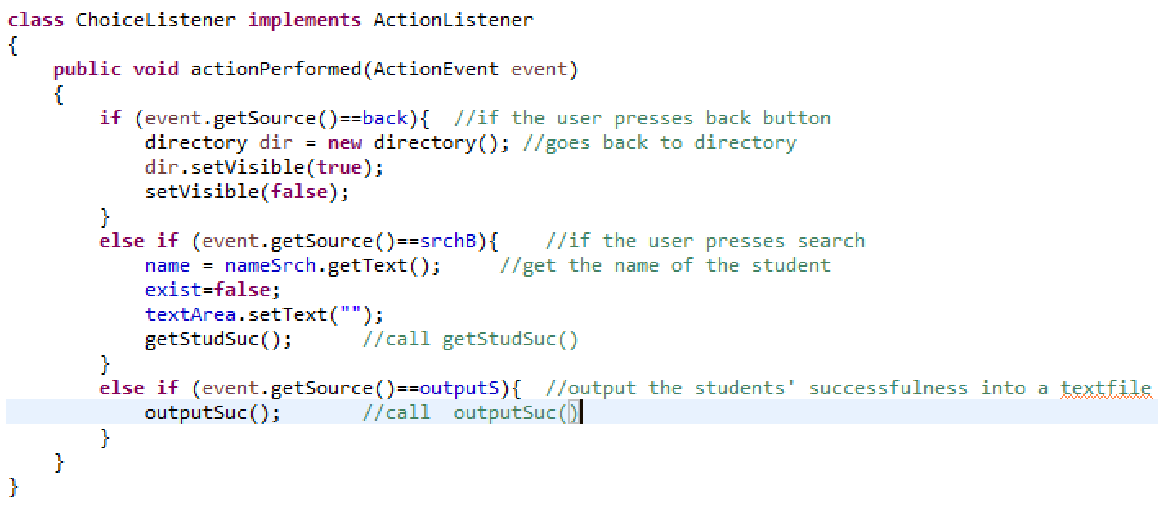


```java
import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class directory extends JFrame{
    private JPanel panel = new JPanel(new GridBagLayout());
    private ActionListener listener;
    private JButton suCrit;
    private JPanel panel1 = new JPanel();
    private JButton rec;
    public directory(){
        class ChoiceListener implements ActionListener
        {
            public void actionPerformed(ActionEvent event)
            {
                if(event.getSource()==suCrit){ //if "To check students' successfulness" is clicked
                        suc_result res = new suc_result();  //go to success result class
                        setVisible(false);
                        res.setVisible(true);
                }
                else if (event.getSource()==rec){  //if "To receive a recommendation made by this prorgam" is clicked
                    call_octave callO = new call_octave(); //go to call_octave class
                    callO.setVisible(true);
                    setVisible(false);
                }
            }
        }
```

GridBagLayout is used for the GUI (MrJavaHelp, 2009).

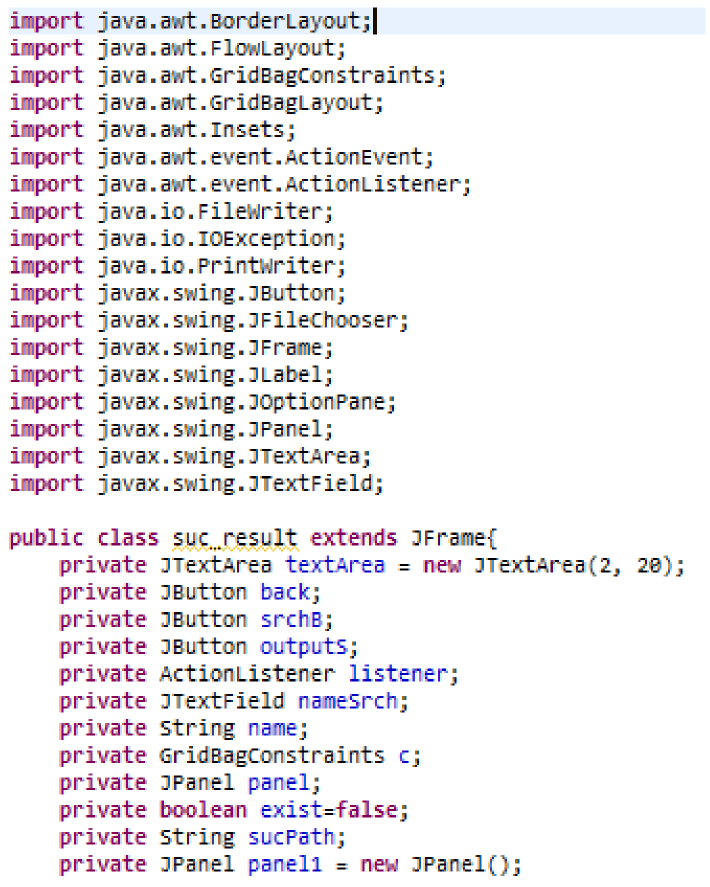


```java
    public void choices(){// let user choose 2 choices

        add(panel, BorderLayout.NORTH);
        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets( 15, 8, 15, 8);
        JLabel label = new JLabel("Please select one of the following: ");
        c.gridx=0; c.gridy=0;
        c.gridwidth=10;
        panel.add(label, c);
        suCrit = new JButton("To check students' successfulness");
        rec = new JButton("To receive a recommendation made by this prorgam");
        c.gridx=0; c.gridy=1;
        c.gridwidth=2;
        panel.add(suCrit, c);
        c.gridx=2; c.gridy=1;
        c.gridwidth=2;
        panel.add(rec, c);
        suCrit.addActionListener(listener);
        rec.addActionListener(listener);
    }
```

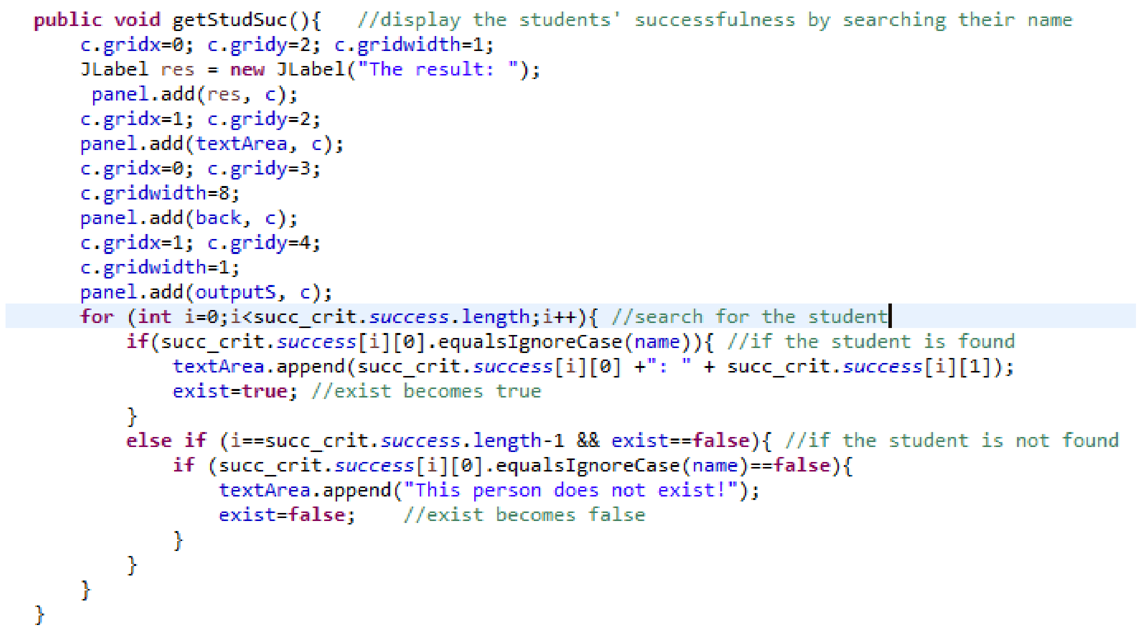


```java
class ChoiceListener implements ActionListener
{
    public void actionPerformed(ActionEvent event)
    {
        if (event.getSource()==back){  //if the user presses back button
            directory dir = new directory(); //goes back to directory
            dir.setVisible(true);
            setVisible(false);
        }
        else if (event.getSource()==srchB){    //if the user presses search
            name = nameSrch.getText();     //get the name of the student
            exist=false;
            textArea.setText("");
            getStudSuc();      //call getStudSuc()
        }
        else if (event.getSource()==outputS){  //output the students' successfulness into a textfile
            outputSuc();       //call  outputSuc()
        }
    }
}
```

The libraries and variables used in suc_result class:

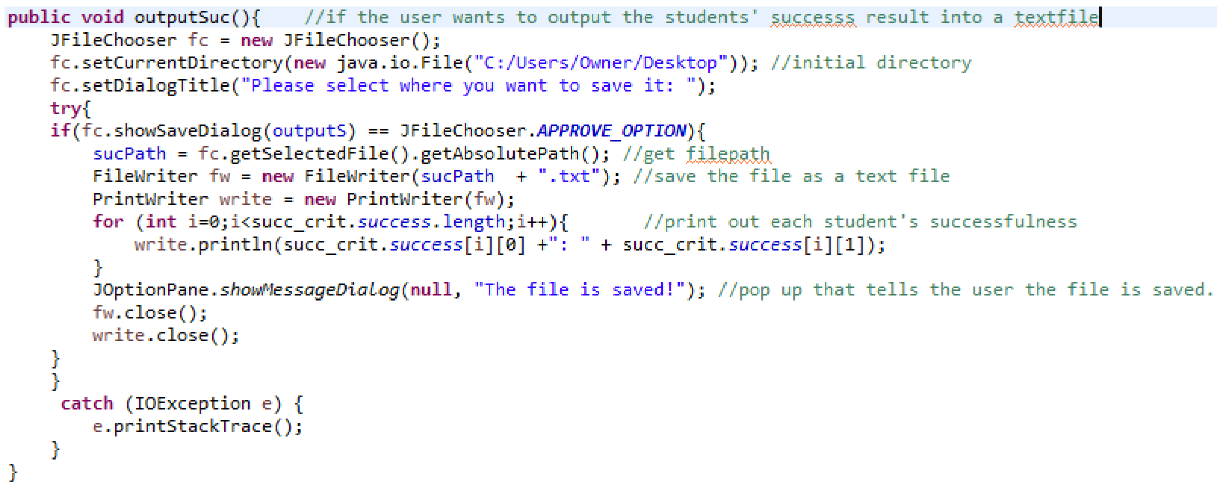


```java
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class suc_result extends JFrame{
    private JTextArea textArea = new JTextArea(2, 20);
    private JButton back;
    private JButton srchB;
    private JButton outputS;
    private ActionListener listener;
    private JTextField nameSrch;
    private String name;
    private GridBagConstraints c;
    private JPanel panel;
    private boolean exist=false;
    private String sucPath;
    private JPanel panel1 = new JPanel();
```



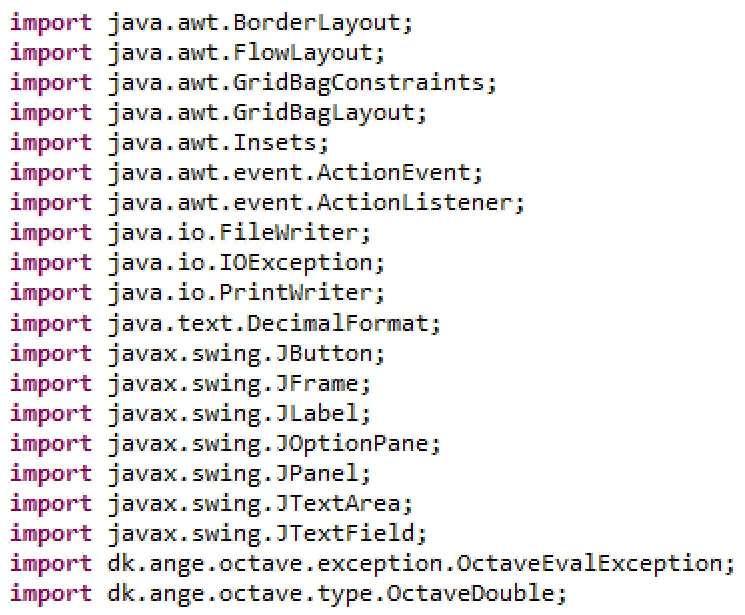
```
public void getStudSuc(){   //display the students' successfulness by searching their name
    c.gridx=0; c.gridy=2; c.gridwidth=1;
    JLabel res = new JLabel("The result: ");
     panel.add(res, c);
    c.gridx=1; c.gridy=2;
    panel.add(textArea, c);
    c.gridx=0; c.gridy=3;
    c.gridwidth=8;
    panel.add(back, c);
    c.gridx=1; c.gridy=4;
    c.gridwidth=1;
    panel.add(outputS, c);
    for (int i=0;i<succ_crit.success.length;i++){ //search for the student
        if(succ_crit.success[i][0].equalsIgnoreCase(name)){ //if the student is found
            textArea.append(succ_crit.success[i][0] +": " + succ_crit.success[i][1]);
            exist=true; //exist becomes true
        }
        else if (i==succ_crit.success.length-1 && exist==false){ //if the student is not found
            if (succ_crit.success[i][0].equalsIgnoreCase(name)==false){
                textArea.append("This person does not exist!");
                exist=false;    //exist becomes false
            }
        }
    }
}
```

In the getSuc() method, *success*[][] is called to receive the students' successfulness.

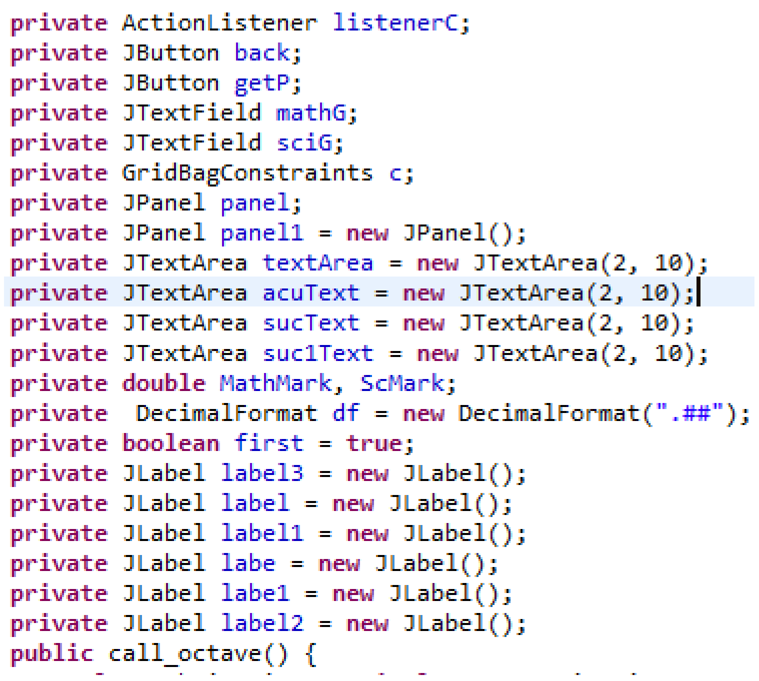


```
public void outputSuc(){    //if the user wants to output the students' successs result into a textfile
    JFileChooser fc = new JFileChooser();
    fc.setCurrentDirectory(new java.io.File("C:/Users/Owner/Desktop")); //initial directory
    fc.setDialogTitle("Please select where you want to save it: ");
    try{
    if(fc.showSaveDialog(outputS) == JFileChooser.APPROVE_OPTION){
        sucPath = fc.getSelectedFile().getAbsolutePath(); //get filepath
        FileWriter fw = new FileWriter(sucPath  + ".txt"); //save the file as a text file
        PrintWriter write = new PrintWriter(fw);
        for (int i=0;i<succ_crit.success.length;i++){      //print out each student's successfulness
            write.println(succ_crit.success[i][0] +": " + succ_crit.success[i][1]);
        }
        JOptionPane.showMessageDialog(null, "The file is saved!"); //pop up that tells the user the file is saved.
        fw.close();
        write.close();
    }
    }
     catch (IOException e) {
        e.printStackTrace();
    }
}
```

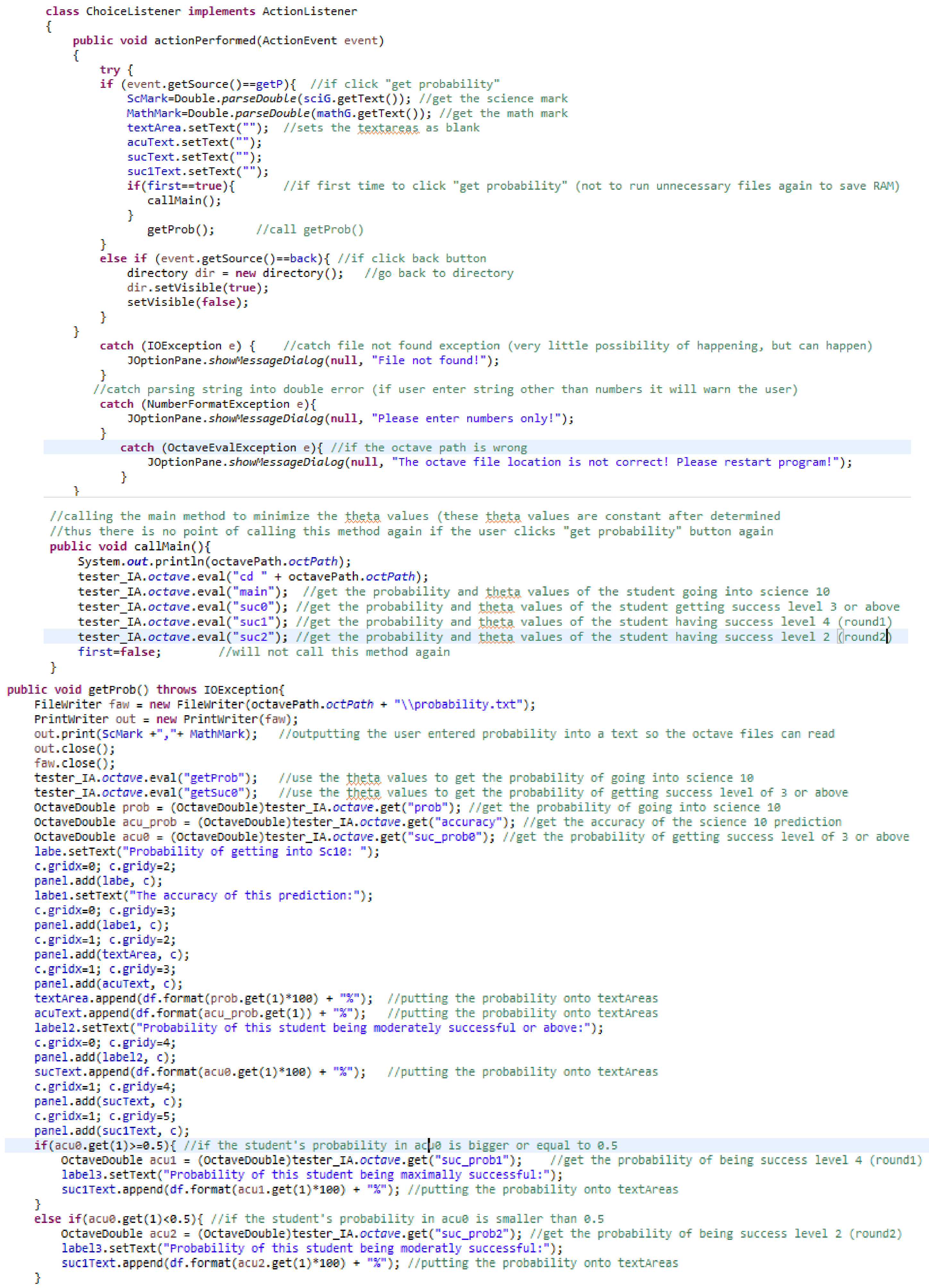
The libraries and variables used in call_octave class:



```java
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.DecimalFormat;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import dk.ange.octave.exception.OctaveEvalException;
import dk.ange.octave.type.OctaveDouble;
```

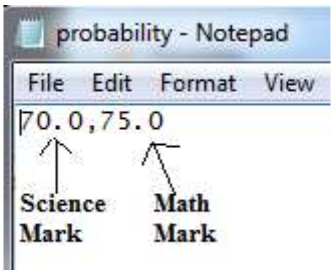


```java
private ActionListener listenerC;
private JButton back;
private JButton getP;
private JTextField mathG;
private JTextField sciG;
private GridBagConstraints c;
private JPanel panel;
private JPanel panel1 = new JPanel();
private JTextArea textArea = new JTextArea(2, 10);
private JTextArea acuText = new JTextArea(2, 10);
private JTextArea sucText = new JTextArea(2, 10);
private JTextArea suc1Text = new JTextArea(2, 10);
private double MathMark, ScMark;
private  DecimalFormat df = new DecimalFormat(".##");
private boolean first = true;
private JLabel label3 = new JLabel();
private JLabel label = new JLabel();
private JLabel label1 = new JLabel();
private JLabel labe = new JLabel();
private JLabel labe1 = new JLabel();
private JLabel label2 = new JLabel();
public call_octave() {
```



```java
class ChoiceListener implements ActionListener
{
    public void actionPerformed(ActionEvent event)
    {
        try {
        if (event.getSource()==getP){  //if click "get probability"
            ScMark=Double.parseDouble(sciG.getText()); //get the science mark
            MathMark=Double.parseDouble(mathG.getText()); //get the math mark
            textArea.setText("");  //sets the textareas as blank
            acuText.setText("");
            sucText.setText("");
            suc1Text.setText("");
            if(first==true){       //if first time to click "get probability" (not to run unnecessary files again to save RAM)
               callMain();
            }
               getProb();      //call getProb()
        }
        else if (event.getSource()==back){ //if click back button
            directory dir = new directory();   //go back to directory
            dir.setVisible(true);
            setVisible(false);
        }
    }
        catch (IOException e) {    //catch file not found exception (very little possibility of happening, but can happen)
            JOptionPane.showMessageDialog(null, "File not found!");
        }
       //catch parsing string into double error (if user enter string other than numbers it will warn the user)
        catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null, "Please enter numbers only!");
        }
           catch (OctaveEvalException e){ //if the octave path is wrong
               JOptionPane.showMessageDialog(null, "The octave file location is not correct! Please restart program!");
           }
    }

 //calling the main method to minimize the theta values (these theta values are constant after determined
 //thus there is no point of calling this method again if the user clicks "get probability" button again
 public void callMain(){
     System.out.println(octavePath.octPath);
     tester_IA.octave.eval("cd " + octavePath.octPath);
     tester_IA.octave.eval("main");  //get the probability and theta values of the student going into science 10
     tester_IA.octave.eval("suc0"); //get the probability and theta values of the student getting success level 3 or above
     tester_IA.octave.eval("suc1"); //get the probability and theta values of the student having success level 4 (round1)
     tester_IA.octave.eval("suc2"); //get the probability and theta values of the student having success level 2 (round2)
     first=false;        //will not call this method again
 }

public void getProb() throws IOException{
    FileWriter faw = new FileWriter(octavePath.octPath + "\\probability.txt");
    PrintWriter out = new PrintWriter(faw);
    out.print(ScMark +","+ MathMark);   //outputting the user entered probability into a text so the octave files can read
    out.close();
    faw.close();
    tester_IA.octave.eval("getProb");   //use the theta values to get the probability of going into science 10
    tester_IA.octave.eval("getSuc0");   //use the theta values to get the probability of getting success level of 3 or above
    OctaveDouble prob = (OctaveDouble)tester_IA.octave.get("prob"); //get the probability of going into science 10
    OctaveDouble acu_prob = (OctaveDouble)tester_IA.octave.get("accuracy"); //get the accuracy of the science 10 prediction
    OctaveDouble acu0 = (OctaveDouble)tester_IA.octave.get("suc_prob0"); //get the probability of getting success level of 3 or above
    labe.setText("Probability of getting into Sc10: ");
    c.gridx=0; c.gridy=2;
    panel.add(labe, c);
    label1.setText("The accuracy of this prediction:");
    c.gridx=0; c.gridy=3;
    panel.add(label1, c);
    c.gridx=1; c.gridy=2;
    panel.add(textArea, c);
    c.gridx=1; c.gridy=3;
    panel.add(acuText, c);
    textArea.append(df.format(prob.get(1)*100) + "%");  //putting the probability onto textAreas
    acuText.append(df.format(acu_prob.get(1)) + "%");   //putting the probability onto textAreas
    label2.setText("Probability of this student being moderately successful or above:");
    c.gridx=0; c.gridy=4;
    panel.add(label2, c);
    sucText.append(df.format(acu0.get(1)*100) + "%");   //putting the probability onto textAreas
    c.gridx=1; c.gridy=4;
    panel.add(sucText, c);
    c.gridx=1; c.gridy=5;
    panel.add(suc1Text, c);
    if(acu0.get(1)>=0.5){ //if the student's probability in acu0 is bigger or equal to 0.5
        OctaveDouble acu1 = (OctaveDouble)tester_IA.octave.get("suc_prob1");    //get the probability of being success level 4 (round1)
        label3.setText("Probability of this student being maximally successful:");
        suc1Text.append(df.format(acu1.get(1)*100) + "%"); //putting the probability onto textAreas
    }
    else if(acu0.get(1)<0.5){ //if the student's probability in acu0 is smaller than 0.5
        OctaveDouble acu2 = (OctaveDouble)tester_IA.octave.get("suc_prob2"); //get the probability of being success level 2 (round2)
        label3.setText("Probability of this student being moderatly successful:");
        suc1Text.append(df.format(acu2.get(1)*100) + "%"); //putting the probability onto textAreas
    }
```

The probability file looks like:



```
70.0,75.0
```

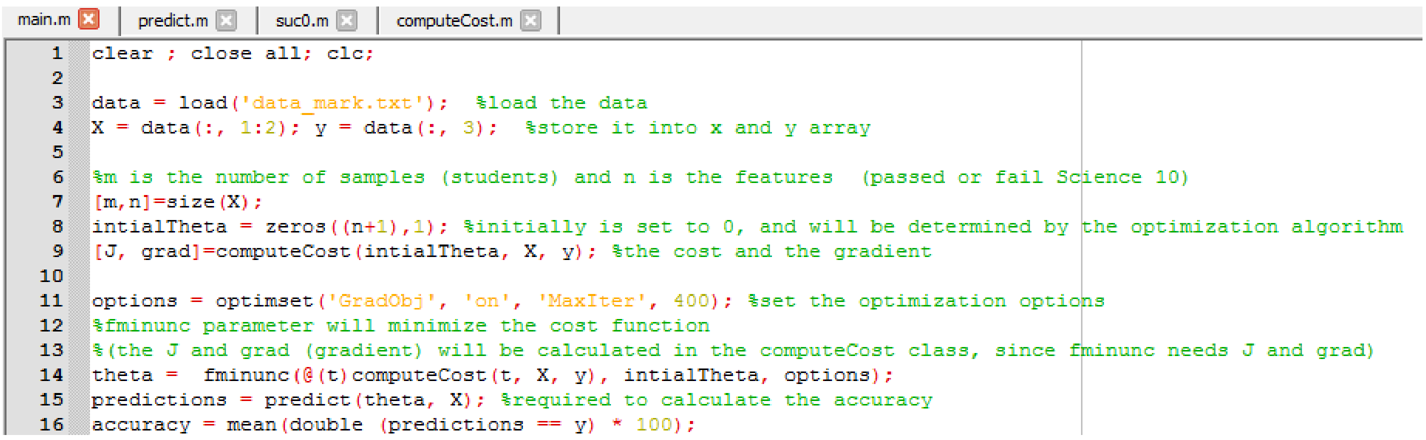
The Machine learning algorithms are implemented from the Machine Learning Class (Ng, 2017)

The hypothesis function is defined as: $$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + \exp -\theta^T x}$$

The sigmoid function is defined as: $$g(z) = \frac{1}{1 + e^{-z}}.$$

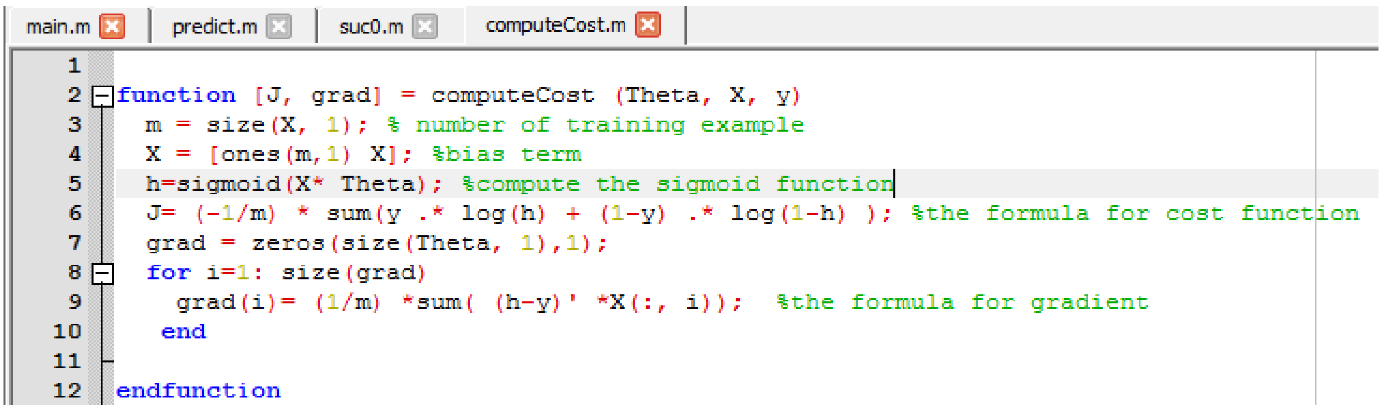
The cost function is defined as: $$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right],$$

The general idea is: since the cost function uses the hypothesis and sigmoid function, if the cost function is minimized then the theta values will also be minimized. Then, theta will be used to calculate the probabilities with the sigmoid function (Matt, 2017).



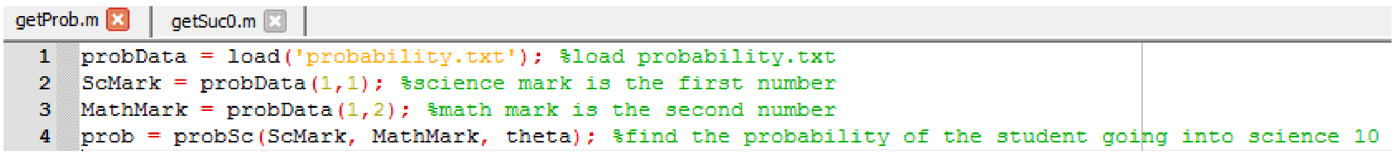
```
clear ; close all; clc;

data = load('data_mark.txt');  %load the data
X = data(:, 1:2); y = data(:, 3);  %store it into x and y array

%m is the number of samples (students) and n is the features  (passed or fail Science 10)
[m,n]=size(X);
intialTheta = zeros((n+1),1); %initially is set to 0, and will be determined by the optimization algorithm
[J, grad]=computeCost(intialTheta, X, y); %the cost and the gradient

options = optimset('GradObj', 'on', 'MaxIter', 400); %set the optimization options
%fminunc parameter will minimize the cost function
%(the J and grad (gradient) will be calculated in the computeCost class, since fminunc needs J and grad)
theta =  fminunc(@(t)computeCost(t, X, y), intialTheta, options);
predictions = predict(theta, X); %required to calculate the accuracy
accuracy = mean(double (predictions == y) * 100);
```

The same logic applies to suc0, suc1, and suc2, where they all calculate the minimized theta value.

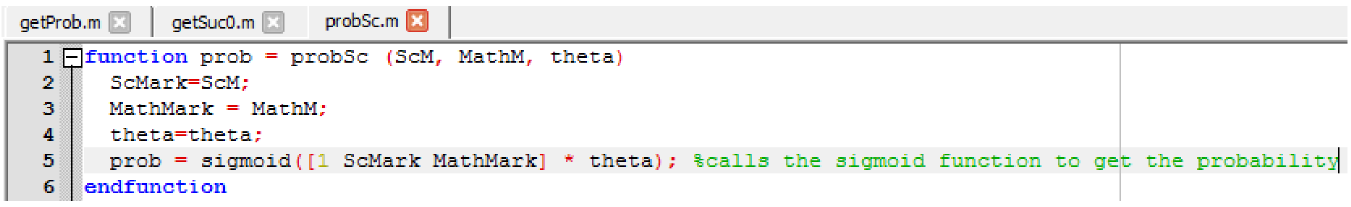


```

function [J, grad] = computeCost (Theta, X, y)
  m = size(X, 1); % number of training example
  X = [ones(m,1) X]; %bias term
  h=sigmoid(X* Theta); %compute the sigmoid function
  J= (-1/m) * sum(y .* log(h) + (1-y) .* log(1-h) ); %the formula for cost function
  grad = zeros(size(Theta, 1),1);
  for i=1: size(grad)
    grad(i)= (1/m) *sum( (h-y)' *X(:, i));  %the formula for gradient
   end

endfunction
```

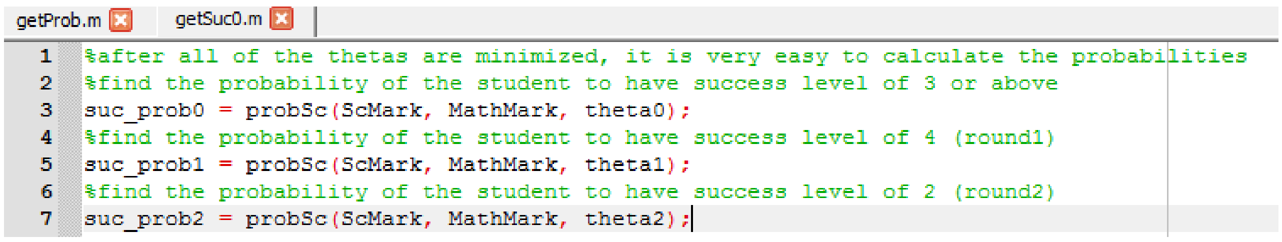
The machine learning code so far is made by the Machine Learning Class. I made getProb, probSc, and getSuc0 methods by myself:



```
probData = load('probability.txt'); %load probability.txt
ScMark = probData(1,1); %science mark is the first number
MathMark = probData(1,2); %math mark is the second number
prob = probSc(ScMark, MathMark, theta); %find the probability of the student going into science 10
```



```
function prob = probSc (ScM, MathM, theta)
  ScMark=ScM;
  MathMark = MathM;
  theta=theta;
  prob = sigmoid([1 ScMark MathMark] * theta); %calls the sigmoid function to get the probability
endfunction
```



```
%after all of the thetas are minimized, it is very easy to calculate the probabilities
%find the probability of the student to have success level of 3 or above
suc_prob0 = probSc(ScMark, MathMark, theta0);
%find the probability of the student to have success level of 4 (round1)
suc_prob1 = probSc(ScMark, MathMark, theta1);
%find the probability of the student to have success level of 2 (round2)
suc_prob2 = probSc(ScMark, MathMark, theta2);
```

Word count: 671

Marks	Awarded	Description
0		The response does not reach a standard described by the descriptors below.
1–4		The use of techniques demonstrates a low level of complexity and ingenuity or does not address the scenario identified in criterion A. It is characterized by limited use of existing tools. There is no explanation of why the techniques are used or how they are adequate for the task. Sources are used but are not identified.
5–8		The use of techniques demonstrates a moderate level of complexity and ingenuity in addressing the scenario identified in criterion A. It is characterized by some appropriate use of existing tools. There is some attempt to explain the techniques used and why they are adequate for the task. All sources are identified.
9–12		The use of techniques demonstrates a high level of complexity and ingenuity in addressing the scenario identified in criterion A. It is characterized by the appropriate use of existing tools. The techniques are adequate for the task and their use is explained. All sources are identified.