|  |
| --- |
| Student Mark + Additional |

# Brief

A simple UI linking to a dataset of student marks where each student has a name, an id and a range of marks for their modules. The UI allows new students to be added; new marks to be added and provides various lookup functions and facilities to check the average mark & number of passes/fails for a given student. The additional section deals with the module statistics. The UI provides various lookup functions and facilities to check the average mark & number of passes/fails for a given module.

# Tools & techniques

To create my student marks system I needed a way to link my students to the system. I created a course class that holds an array list of students and an array list of modules. Then I defined three different course constructors. The first one is empty. The next one has a code, a title and the array list of modules. The final one has a code, a title, the array list of modules and the array list of students. This covered all types of courses as some many only have modules but no students.

In the student class I created a hashmap of marks for the student because it was easier to store them so I could grab a certain mark later on. In the student class is also where I created the code that deals with adding a module mark, getting the average mark for the student and the number of passes or fails for the student. As I had used a hashmap for the marks I could loop through the map entries grabbing the marks as I need them. In the module class I have also created methods that calculate the averages, passes and fails per module so the additional section of the problem can be implemented.

My setup for my GUI main is based off my phonebook application because I thought it is an effective way of displaying the information. The students are displayed in a way much like the contacts for the phone book application, but they are listed by student number instead. To add a mark for a student you have to be selected on a student from the list otherwise it will not work. My discussion with my tutors on what to include and to take inspiration from the Teesside University’s Evision records help greatly with my layout planning. This helped me gain an idea on what classes I would need to implement. For this application I have used the multiton pattern because I have implemented hashmaps and arraylists to deal with my data for marks, students and modules.

# Results

My application performs all of the necessary functionality for the student marks problem. The GUI displays and updates the student list when a new student is added. When a new mark is added for a student the list updates again to reflect the input. The course details are also updated. I have conducted decision table testing below to prove that all of my buttons and validation work effectively.

## Test Results

### Junit Tests

I have created Junit tests for every class in my application however I did not get time to fully implement all the tests necessary for the application. This means that a lot of my tests are failing at the minute. If I had more time to improve my application I would get these fully functional and all passing.

### Decision Table

|  |  |  |  |
| --- | --- | --- | --- |
| Screen | Feature | Expected Output | Actual Output |
| MainUI | Student List | A list of students within the application are displayed. Clicking on a student will bring up the information in the display JList. | A list of students within the application are displayed. Clicking on the student number S6110375 brings up the information in the display JList. |
| MainUI | Add Student Button | The Add Student UI screen will load up after the button is pressed. | The Add Student UI screen loads up after the button is pressed. |
| MainUI | Add Mark Button | Click on a student number from the list and pressing the add mark button will load up the Add Mark UI screen. | Clicking on the student number and then pressing the button loads up the Add Mark UI screen. |
| MainUI | View Course Info Button | The View Course UI screen will load up after the button is pressed. | The View Course UI screen loads up after the button is pressed. |
| MainUI | Close Button | The application will close and the code will stop running. | The application closes and the code stops running. |
| Add Student UI | All Text Fields are empty | If all text fields are left empty a pop up box will appear stating a message to fill out all of the boxes. | A message dialog box pops up with the message to fill in all boxes appears. |
| Add Student UI | Student ID Text Field | If the student id text field has an input with too many characters a message dialog box will appear. | A message dialog box appears with the message that the input must be 8 digits. |
| Add Student UI | First Name Text Field | If the first name text field has an input with too many characters a message dialog box will appear. | A message dialog box appears with the message that the input must be between 1 and 25 characters. |
| Add Student UI | Last Name Text Field | If the last name text field has an input with too many characters a message dialog box will appear. | A message dialog box appears with the message that the input must be between 1 and 25 characters. |
| Add Student UI | Course Combo Box | All the courses will be displayed here in the drop down. | All the courses are displayed in the combo box drop down. |
| Add Student UI | Add Button | If all validation is met for the text field inputs then a new student will be created and a message dialog box will appear stating such. The user will be taken back to the Main UI screen. | A message dialog box appears stating that a new student has been added and the user is taken back to the Main UI screen. |
| Add Student UI | Back Button | The Main UI screen will load up after the button is pressed. | The Main UI screen loads up after the button is pressed. |
| Add Mark UI | Student Details Displayed in JLabels |  |  |
| Add Mark UI | Modules Combo Box | All the modules for the course will be displayed here in the drop down unless the student already has a mark for the module. | All the modules for the course are displayed in the combo box drop down unless the student has a mark already. |
| Add Mark UI | Mark Text Field | If the mark text field has an input with too many digits a message dialog box will appear. | Error Appears. This will need to be looked at in detail to prevent this problem occurring again. |
| Add Mark UI | Add Button | If all validation is met for the text field inputs then a new mark for the module will be created and a message dialog box will appear stating such. The user will be taken back to the Main UI screen. | A message dialog box appears stating that a new mark has been added and the user is taken back to the Main UI screen. |
| View Course UI | Course Combo Box | All the courses will be displayed in the drop down box. | All the courses are displayed in the combo box drop down. |
| View Course UI | Module List | All the modules for the course selected will be displayed in the JList. | All the modules for the course selected are displayed in JList. |
| View Course UI | Module Details | After selecting a module code from the list the module data (title, pass mark, number of students, average, number of passes and fails) will be displayed. | After selecting a module code from the list the module data (title, pass mark, number of students, average, number of passes and fails) is displayed. |
| View Course UI | Back Button | The Main UI screen will load up after the button is pressed. | The Main UI screen loads up after the button is pressed. |

# Critique

The decision to layout my code that has courses, which have modules and students on those modules made it easier to link all my classes together to grab the relevant data. I have also taken inspiration from my phonebook application, like I have stated earlier. This is an effective layout that displays all the relevant information for user to see. I add the row of buttons along the top of the GUI so they are easier to see what they do. I believe my code is well presented using appropriate class names and method names. All classes and methods are fully commented with explanations where necessary.

I would like to improve my GUIs look by adding more labels where necessary and playing around with my GUI layout graphics. This would help create a better visual effect of the overall student marks system. This is a minor change that would not affect the functionality of the application it would just make some things clearer for the user. I would also like to implement my Junit test so they work properly.

# Other details

# Java code

StudentMark :

|  |
| --- |
| /\*  \* Student Marks Problem + Additional.  \* A simple UI linking to a dataset of student marks where each student has a  \* name, an id and a range of marks for their modules. The UI allows new  \* students to be added; new marks to be added and provides various lookup  \* functions and facilities to check the average mark & number of passes/fails  \* for a given student.  \* Additional : Module statistics; the UI provides various lookup functions and  \* facilities to check the average mark & number of passes/fails for a given module.  \*/  package studentmark;  import java.util.ArrayList;  /\*\*  \* @author Heidi Portwine (s6110438)  \*/  public class StudentMark  {  // Variables declared.  static ArrayList<Course> courses = new ArrayList<>();  static Course c1, c2;    /\*\*  \* Main method of Student Mark class.  \* @param args the command line arguments  \*/  public static void main(String[] args)  {  // Calls initCourse and initStudent to grab data.  initCourse();  initStudent();  StudentMainUI main = new StudentMainUI(courses);  main.setVisible(true);  }      /\*\*  \* Initalises the courses for the system.  \*/  public static void initCourse()  {  // Creates an arrayList of module.  ArrayList<Module> mod1 = new ArrayList<>();  // Creates the module objects used for the system.  Module m1 = new Module("432","Object Oriented Programming",40,0);  Module m2 = new Module("754","Machine Learning",40,0);  Module m3 = new Module("287","3D Animation",40,0);  Module m4 = new Module("486","Android App Development",40,0);  // Adds the modules to the array lists.  mod1.add(m1);  mod1.add(m2);  mod1.add(m3);  mod1.add(m4);  // Creates a new course.  c1 = new Course("COM4987\_N","Computer Science",mod1);  courses.add(c1);  // For the second course and its modules.  ArrayList<Module> mod2 = new ArrayList<>();  Module m5 = new Module("123","Games Prototyping",40,0);  Module m6 = new Module("453","Games Studies",40,0);  Module m7 = new Module("876","Mission Design",40,0);  Module m8 = new Module("543","Games Future",40,0);  mod2.add(m5);  mod2.add(m6);  mod2.add(m7);  mod2.add(m8);  c2 = new Course("COM9512\_N","Computer Games Design",mod2);  courses.add(c2);  }    /\*\*  \* Initalises the students for the system.  \*/  public static void initStudent()  {  // Creates an arrayList of students.  ArrayList<Student> stu1 = new ArrayList<>();  ArrayList<Student> stu2 = new ArrayList<>();    // Creates a new student  Student s1 = new Student("S6464921", "Bob", "Green", "COM9512\_N");  // Adds the module marks for the student.  s1.addModMark("123", 50);  s1.addModMark("453", 30);    // Calls updateModStu to update the system.  updateModStu("123", "COM9512\_N");  updateModStu("453", "COM9512\_N");    // For the second student.  Student s2 = new Student("S6110375", "Sally", "Brown", "COM4987\_N");  s2.addModMark("754", 45);  updateModStu("754", "COM4987\_N");  stu1.add(s1);  stu2.add(s2);    // Sets cIndex a certain course and sets the students on the course.  int cIndex1 = courses.indexOf(c1);  courses.get(cIndex1).setStudents(stu2);  int cIndex2 = courses.indexOf(c2);  courses.get(cIndex2).setStudents(stu1);  }    /\*\*  \* This method is called to update the module students.  \* @param mc  \* @param cID  \*/  public static void updateModStu(String mc, String cID)  {  // Initalises the variables.  int cIndex = 0;  int mIndex = 0;  Course course = null;    // Loops through the courses.  for(Course c : courses)  {  // checks if the course code is equal to the current course code.  if(c.getCode().equals(cID))  {  course = c;  cIndex = courses.indexOf(c);  }  }    // Sets the module variable to null.  Module mod = null;    // Loops through the modules.  for(Module m : course.getModule())  {  // Checks if the module code is equal to the current module code.  if(m.getCode().equals(mc))  {  mod = m;  }  }  mIndex = courses.get(cIndex).getModule().indexOf(mod);    // Updates the stident number.  int studNum = courses.get(cIndex).getModule().get(mIndex).getStudentNumber() +1;  courses.get(cIndex).getModule().get(mIndex).setStudentNumber(studNum);  }  } |

Student:

|  |
| --- |
| /\*  \* Students for the student marking system.  \* This class creates the fields needed for a student entry to be added to the system.  \*/  package studentmark;  import java.util.HashMap;  import java.util.Map;  /\*\*  \* @author Heidi Portwine (S6110438)  \*/  public class Student  {  // Variable for the class declared.  private String studentID;  private String fName;  private String lName;  private String course;  HashMap<String, Integer> marks;    /\*\*  \* Empty constructor method created.  \*/  public Student()  {    }    /\*\*  \* Student created with a student ID, first name, last name, course.  \* @param st  \* @param fn  \* @param ln  \* @param c  \*/  public Student(String st, String fn, String ln, String c)  {  studentID = st;  fName = fn;  lName = ln;  course = c;  marks = new HashMap<>();  }  /\*\*  \* Gets the students first name.  \* @return fName  \*/  public String getfName()  {  return fName;  }  /\*\*  \* Sets the students first name.  \* @param fName  \*/  public void setfName(String fName)  {  this.fName = fName;  }  /\*\*  \* Gets the students last name.  \* @return lName  \*/  public String getlName()  {  return lName;  }  /\*\*  \* Sets the students last name.  \* @param lName  \*/  public void setlName(String lName)  {  this.lName = lName;  }  /\*\*  \* Gets the course the student is on.  \* @return course  \*/  public String getCourse()  {  return course;  }  /\*\*  \* Sets the course the student is on.  \* @param course  \*/  public void setCourse(String course)  {  this.course = course;  }  /\*\*  \* Gets the marks for the course module in a hashmap.  \* @return marks.  \*/  public HashMap<String, Integer> getMarks()  {  return marks;  }  /\*\*  \* Sets the students marks.  \* @param marks  \*/  public void setMarks(HashMap<String, Integer> marks)  {  this.marks = marks;  }  /\*\*  \* Gets the students ID.  \* @return studentID  \*/  public String getStudentID()  {  return studentID;  }  /\*\*  \* Sets the students ID.  \* @param studentID  \*/  public void setStudentID(String studentID)  {  this.studentID = studentID;  }    /\*\*  \* Adds a module mark for the student.  \* @param c  \* @param ma  \*/  public void addModMark(String c, int ma)  {  // Checks is the course mark is null.  if(marks.get(c) == null)  {  // Adds the mark to the hashmap.  marks.put(c, ma);  }  }    /\*\*  \* Gets the students average mark for all marks.  \* @return total/count  \*/  public double getAverage()  {  // Initialises the variables.  int total = 0;  int count = 0;    // Loops through the hashmap checking the marks.  for(Map.Entry m : marks.entrySet())  {  // Adds together all the marks to gain a total.  total += Integer.parseInt(m.getValue().toString());  count++;  }  // Divides the total of marks by the count of how many marks there where.  return total/count;  }    /\*\*  \* Gets the students total number of passes.  \* @param c  \* @return total  \*/  public int getPasses(Course c)  {  // Initalises variable.  int total = 0;    // Loops through the hashmap checking the marks.  for(Map.Entry m : marks.entrySet())  {  // Sets variables to equal data gathered from the map.  String code = m.getKey().toString();  int ma = Integer.parseInt(m.getValue().toString());    // Loops through the course modules.  for(Module mod : c.getModule())  {  // Checks if the module code is equal to the module code from the map.  if(mod.getCode().equals(code))  {  // Sets pass as the module pass mark.  int pass = mod.getPassMark();    // Checks if the students mark is greater than or equal  // to the pass mark of the module.  if(ma >= pass)  {  // Increments the total.  total++;  }  }  }  }  return total;  }    /\*\*  \* Getst the total number of fails for the student.  \* @param c  \* @return total  \*/  public int getFails(Course c)  {  // Initalises the varibale.  int total = 0;    // Loops through the hashmap checking the marks.  for(Map.Entry m : marks.entrySet())  {  // Sets variables to equal data gathered from the map.  String code = m.getKey().toString();  int ma = Integer.parseInt(m.getValue().toString());    // Loops through the course modules.  for(Module mod : c.getModule())  {  // Checks if the module code is equal to the module code from the map.  if(mod.getCode().equals(code))  {  // Sets pass as the module pass mark.  int pass = mod.getPassMark();    // Checks if the students mark is less than the pass mark for the module.  if(ma < pass)  {  // Increments the total.  total++;  }  }  }  }  return total;  }  } |

Module:

|  |
| --- |
| /\*  \* Modules for the student marking system.  \* This class creates the fields needed for a module entry to be added to the system.  \*/  package studentmark;  import java.util.ArrayList;  import java.util.Map;  /\*\*  \* @author Heidi Portwine (S6110438)  \*/  public class Module  {  // Variable for the class declared.  String code;  String moduleTitle;  int passMark;  int studentNumber;    /\*\*  \* Empty Constructor Created.  \*/  public Module()  {    }    /\*\*  \* Module created with module code, module title, pass mark and a total of students.  \* @param c  \* @param t  \* @param p  \* @param s  \*/  public Module(String c, String t, int p, int s)  {  code = c;  moduleTitle = t;  passMark = p;  studentNumber = s;  }    /\*\*  \* Adds a Student to a module.  \*/  public void addStudent()  {  // Increments student number by 1.  studentNumber++;  }    /\*\*  \* Gets the average mark for a module.  \* @param students  \* @return total/studentNumber  \*/  public double getAverage(ArrayList<Student> students)  {  // Initialise the variable.  int total = 0;    // Loops through students.  for(Student stu : students)  {  // Loops through the hashmap checking the student marks.  for(Map.Entry m : stu.getMarks().entrySet())  {  // Checks if the map key equals the module code.  if(m.getKey().equals(code))  {  // Adds together all the marks to gain a total.  total += Integer.parseInt(m.getValue().toString());  }  }  }  // Divides the total of all marks by the number of students.  return total/studentNumber;  }    /\*\*  \* Gets the number of students who passed the module.  \* @param students  \* @return total  \*/  public int getPass(ArrayList<Student> students)  {  // Initalises the variable.  int total = 0;    // Loops through the students.  for(Student stu : students)  {  // Checks the map for the student marks.  for(Map.Entry m : stu.getMarks().entrySet())  {  // Checks if the map key equals the module code.  if(m.getKey().equals(code))  {  // Sets the mark as the value from the map.  int mark = Integer.parseInt(m.getValue().toString());    // Checks if the mark is greater than or equal to the pass mark.  if(mark >= passMark)  {  // Increments the total by 1.  total++;  }  }  }  }  return total;  }    /\*\*  \* Gets the number of fails for a module.  \* @param students  \* @return total  \*/  public int getFail(ArrayList<Student> students)  {  // Initalises the variable.  int total = 0;    // Loops through the students  for(Student stu : students)  {  // Checks the map for student marks.  for(Map.Entry m : stu.getMarks().entrySet())  {  // If the map key is equal to the module code.  if(m.getKey().equals(code))  {  // Set mark as the student mark from the map/  int mark = Integer.parseInt(m.getValue().toString());    // Checks if the mark is less than the pass mark.  if(mark < passMark)  {  // Increments the total.  total++;  }    }  }  }  return total;  }  /\*\*  \* Gets the module code.  \* @return code  \*/  public String getCode()  {  return code;  }  /\*\*  \* Sets the module code.  \* @param code  \*/  public void setCode(String code)  {  this.code = code;  }  /\*\*  \* Gets the module title.  \* @return moduleTitle  \*/  public String getModuleTitle()  {  return moduleTitle;  }  /\*\*  \* Sets the module title.  \* @param moduleTitle  \*/  public void setModuleTitle(String moduleTitle)  {  this.moduleTitle = moduleTitle;  }  /\*\*  \* Gets the pass mark for a module.  \* @return passMark  \*/  public int getPassMark()  {  return passMark;  }  /\*\*  \* Sets the pass mark.  \* @param passMark  \*/  public void setPassMark(int passMark)  {  this.passMark = passMark;  }  /\*\*  \* Gets the student number for a module.  \* @return studentNumber  \*/  public int getStudentNumber()  {  return studentNumber;  }  /\*\*  \* Sets the student number for a module.  \* @param studentNumber  \*/  public void setStudentNumber(int studentNumber)  {  this.studentNumber = studentNumber;  }    } |

Course:

|  |
| --- |
| /\*  \* Course for the student marking system.  \* This class creates the fields needed for a course entry to be added to the system.  \*/  package studentmark;  import java.util.ArrayList;  /\*\*  \* @author Heidi Portwine (s6110438)  \*/  public class Course  {  // Initialises the array lists students and modules.  ArrayList<Student> students;  ArrayList<Module> module;  String code;  String title;    /\*\*  \* Empty Constructor.  \*/  public Course()  {    }    /\*\*  \* Course created with a course code, course title and modules.  \* @param c  \* @param t  \* @param mod  \*/  public Course(String c, String t,ArrayList mod)  {  code = c;  title = t;  module = mod;  students = new ArrayList();  }    /\*\*  \* Course created with a course code, course title, modules and students.  \* @param c  \* @param t  \* @param mod  \* @param stu  \*/  public Course(String c, String t,ArrayList mod, ArrayList stu)  {  code = c;  title = t;  module = mod;  students = stu;  }    /\*\*  \* Adds a student to the course.  \* @param stu  \*/  public void addStudent(Student stu)  {  students.add(stu);  }  /\*\*  \* Gets a list of students for the course.  \* @return students  \*/  public ArrayList<Student> getStudents()  {  return students;  }  /\*\*  \* Sets students on the course.  \* @param students  \*/  public void setStudents(ArrayList<Student> students)  {  this.students = students;  }  /\*\*  \* Gets the modules for the course.  \* @return module  \*/  public ArrayList<Module> getModule()  {  return module;  }  /\*\*  \* Sets the modules on the course.  \* @param module  \*/  public void setModule(ArrayList<Module> module)  {  this.module = module;  }  /\*\*  \* Gets the course code.  \* @return code  \*/  public String getCode()  {  return code;  }  /\*\*  \* Sets the course code.  \* @param code  \*/  public void setCode(String code)  {  this.code = code;  }  /\*\*  \* Gets the course title.  \* @return title  \*/  public String getTitle()  {  return title;  }  /\*\*  \* Sets the course title.  \* @param title  \*/  public void setTitle(String title)  {  this.title = title;  }    } |

StudentMain UI:

|  |
| --- |
| /\*  \* The student mark systems main screen when the GUI loads up.  \*/  package studentmark;  import java.util.ArrayList;  import java.util.Map;  import javax.swing.DefaultListModel;  /\*\*  \* @author Heidi Portwine (s6110438)  \*/  public class StudentMainUI extends javax.swing.JFrame  {  // Variables Declared.  private static ArrayList<Course> courses = new ArrayList();  private static ArrayList<Student> students = new ArrayList();  public DefaultListModel studentListModel = new DefaultListModel();  public DefaultListModel infoListModel = new DefaultListModel();  public static Student cStu;    /\*\*  \* Creates new form StudentMainUI passing through the array list course  \*/  public StudentMainUI(ArrayList<Course> c)  {  courses = c;  initComponents();  initStudentList();  displayInfoList.setEnabled(false);  }  /\*\*  \* Initalises the student list.  \*/  public void initStudentList()  {  // Clears the student list model to prevent duplicates.  studentListModel.clear();    // Loops through the courses.  for(Course c : courses)  {  // Loops through the students on the course.  for(Student stu : c.getStudents() )  {  // Adds the stdent IDs to the student list model.  students.add(stu);  studentListModel.addElement(stu.getStudentID());  }  }  // Sets the student list object to display the student list model.  studentList.setModel(studentListModel);  }    /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents() {  jPanel1 = new javax.swing.JPanel();  jScrollPane1 = new javax.swing.JScrollPane();  studentList = new javax.swing.JList<>();  jScrollPane2 = new javax.swing.JScrollPane();  displayInfoList = new javax.swing.JList<>();  addButton = new javax.swing.JButton();  AddMarkButton = new javax.swing.JButton();  viewCourseButton = new javax.swing.JButton();  closeButton = new javax.swing.JButton();  jLabel1 = new javax.swing.JLabel();  setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  studentList.setModel(new javax.swing.AbstractListModel<String>() {  String[] strings = { "Student 1", "Student 2", "Student 3", "Student 4" };  public int getSize() { return strings.length; }  public String getElementAt(int i) { return strings[i]; }  });  studentList.addListSelectionListener(new javax.swing.event.ListSelectionListener() {  public void valueChanged(javax.swing.event.ListSelectionEvent evt) {  studentListValueChanged(evt);  }  });  jScrollPane1.setViewportView(studentList);  displayInfoList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE\_SELECTION);  jScrollPane2.setViewportView(displayInfoList);  addButton.setText("Add Student");  addButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  addButtonActionPerformed(evt);  }  });  AddMarkButton.setText("Add Mark");  AddMarkButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  AddMarkButtonActionPerformed(evt);  }  });  viewCourseButton.setText("View Course Info");  viewCourseButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  viewCourseButtonActionPerformed(evt);  }  });  closeButton.setText("Close");  closeButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  closeButtonActionPerformed(evt);  }  });  jLabel1.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N  jLabel1.setText("Welcome to the Student Marking System.");  javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  jPanel1.setLayout(jPanel1Layout);  jPanel1Layout.setHorizontalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addContainerGap()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(0, 0, Short.MAX\_VALUE)  .addComponent(closeButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 70, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGroup(javax.swing.GroupLayout.Alignment.LEADING, jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(jScrollPane1)  .addComponent(addButton, javax.swing.GroupLayout.DEFAULT\_SIZE, 163, Short.MAX\_VALUE))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addComponent(AddMarkButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 137, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(18, 18, 18)  .addComponent(viewCourseButton, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addComponent(jScrollPane2))))  .addGap(25, 25, 25))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(58, 58, 58)  .addComponent(jLabel1)  .addContainerGap(98, Short.MAX\_VALUE))  );  jPanel1Layout.setVerticalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addGap(7, 7, 7)  .addComponent(jLabel1)  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(viewCourseButton, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(addButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 32, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addComponent(AddMarkButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 34, javax.swing.GroupLayout.PREFERRED\_SIZE)))  .addGap(11, 11, 11)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 193, Short.MAX\_VALUE)  .addComponent(jScrollPane2))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  .addComponent(closeButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addContainerGap())  );  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  pack();  }// </editor-fold>  /\*\*  \* The action performed event for the add student button on the main UI.  \* This will take the user to the Add Students UI class,  \* while passing through the courses array.  \* @param evt  \*/  private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {  AddStudentsUI addSt = new AddStudentsUI(courses);  addSt.setVisible(true);  this.setVisible(false);  }  /\*\*  \* The action performed event for the add mark button on the main UI.  \* This will take the user to the Add Mark UI class,  \* while passing through the contacts array and the current student selected.  \* @param evt  \*/  private void AddMarkButtonActionPerformed(java.awt.event.ActionEvent evt) {  AddMarksUI addMa = new AddMarksUI(courses, cStu);  addMa.setVisible(true);  this.setVisible(false);  }  /\*\*  \* The action performed event for the close button on the main UI.  \* This will completely close the system and stop the code running.  \* @param evt  \*/  private void closeButtonActionPerformed(java.awt.event.ActionEvent evt) {  System.exit(0);  }  /\*\*  \* The method triggered when the student list value is changed.  \* This method displays the students in the array on to the list  \* so the user can select one to view.  \* @param evt  \*/  private void studentListValueChanged(javax.swing.event.ListSelectionEvent evt) {    // Clears the model to prevent duplicates.  infoListModel.clear();    // Checks if the student list selection is not empty.  if(!studentList.isSelectionEmpty())  {  if(evt.getValueIsAdjusting() == false)  {  // Clears the model to prevent duplicates.  infoListModel.clear();    // Loops through the students using a for loop.  for(Student stu : students)  {  // Sets the variables to get the students data.  String studentID = stu.getStudentID();  String fName = stu.getfName();  String lName = stu.getlName();  String course = stu.getCourse();  double av = 0;  int p = 0;  int f = 0;  // If the the student list select item equals the student ID.  if(studentList.getSelectedValue().equals(studentID))  {  // Clears the model to prevent duplicates.  infoListModel.clear();    // Sets the current student to student.  cStu = stu;    // Checks if the student has marks.  if(!stu.getMarks().isEmpty())  {  // Sets the average variable to be the students average mark.  av = stu.getAverage();    // Loops through the courses.  for(Course c : courses)  {  // Checks if the course code is equal to the students course.  if(c.getCode().equals(stu.getCourse()))  {  // Sets the variables to be the students passess and fails.  p = stu.getPasses(c);  f = stu.getFails(c);  }  }  }    // Adds the elements to the model so they can be displayed in the correct layout.  infoListModel.addElement("Student Id : " + studentID);  infoListModel.addElement("First Name : " + fName);  infoListModel.addElement("Last Name : " + lName);  infoListModel.addElement("Course : " + course);  infoListModel.addElement("Average : " + av);    infoListModel.addElement("Passes : " + p);  infoListModel.addElement("Fails : " + f);  infoListModel.addElement("------------------------------- ");  infoListModel.addElement("Module Marks ");    // Checks if the student has marks  if(!stu.getMarks().isEmpty())  {  // Checks the map for the students marks.  for(Map.Entry m : stu.getMarks().entrySet())  {  // Adda it to the model.  infoListModel.addElement("Module Code : " + m.getKey() + " " + " Mark : " + m.getValue().toString());  }  }  }  }  // Sets the display info list to display the info list model.  displayInfoList.setModel(infoListModel);  }  }  }  /\*\*  \* The action performed event for the view course button on the main UI.  \* This will take the user to the View Course UI class,  \* while passing through the contacts array and the current student selected.  \* @param evt  \*/  private void viewCourseButtonActionPerformed(java.awt.event.ActionEvent evt) {  ViewCourseUI viewC = new ViewCourseUI(courses);  viewC.setVisible(true);  this.setVisible(false);  }  /\*\*  \* The main method for the main UI.  \* @param args the command line arguments  \*/  public static void main(String args[])  {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Nimbus".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(StudentMainUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(StudentMainUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(StudentMainUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(StudentMainUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>  /\* Create and display the form \*/  java.awt.EventQueue.invokeLater(new Runnable()  {  public void run()  {  new StudentMainUI(courses).setVisible(true);  }  });  }  // Variables declaration - do not modify  private javax.swing.JButton AddMarkButton;  private javax.swing.JButton addButton;  private javax.swing.JButton closeButton;  private javax.swing.JList<String> displayInfoList;  private javax.swing.JLabel jLabel1;  private javax.swing.JPanel jPanel1;  private javax.swing.JScrollPane jScrollPane1;  private javax.swing.JScrollPane jScrollPane2;  private javax.swing.JList<String> studentList;  private javax.swing.JButton viewCourseButton;  // End of variables declaration  } |

AddStudentsUI

|  |
| --- |
| /\*  \* The student mark systems Add Students UI class that loads up when the add student  \* button on the main ui is pressed.  \*/  package studentmark;  import java.util.ArrayList;  import javax.swing.DefaultComboBoxModel;  import javax.swing.JOptionPane;  /\*\*  \* @author Heidi Portwine s6110438  \*/  public class AddStudentsUI extends javax.swing.JFrame  {  // Array list course that is passed through and model declared.  private static ArrayList<Course> courses = new ArrayList();  public DefaultComboBoxModel coursesModel = new DefaultComboBoxModel();    /\*\*  \* Creates new form AddStudentsUI passing through the array list course.  \*/  public AddStudentsUI(ArrayList<Course> c)  {  courses = c;  initComponents();  initComboBox();  }    /\*\*  \* Initalises the combo box that lists the courses.  \*/  public void initComboBox()  {  // Loops through the courses.  for(Course c : courses)  {  // Adds the course codes to the course model.  coursesModel.addElement(c.code);  }  // Sets the course combo box to display the model.  courseCombo.setModel(coursesModel);  }  /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents() {  jPanel1 = new javax.swing.JPanel();  addButton = new javax.swing.JButton();  backButton = new javax.swing.JButton();  stuLabel = new javax.swing.JLabel();  fNameLabel = new javax.swing.JLabel();  lNameLabel = new javax.swing.JLabel();  stuLabel2 = new javax.swing.JTextField();  fNameLabel2 = new javax.swing.JTextField();  lNameLabel2 = new javax.swing.JTextField();  courseLabel = new javax.swing.JLabel();  courseCombo = new javax.swing.JComboBox<>();  setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  addButton.setText("Add");  addButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  addButtonActionPerformed(evt);  }  });  backButton.setText("Back");  backButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  backButtonActionPerformed(evt);  }  });  stuLabel.setText("Student ID");  fNameLabel.setText("First Name:");  lNameLabel.setText("Last Name:");  courseLabel.setText("Course: ");  courseCombo.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Please select a course" }));  javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  jPanel1.setLayout(jPanel1Layout);  jPanel1Layout.setHorizontalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(66, 66, 66)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  .addComponent(fNameLabel)  .addComponent(lNameLabel)  .addGroup(jPanel1Layout.createSequentialGroup()  .addComponent(courseLabel)  .addGap(13, 13, 13))  .addGroup(javax.swing.GroupLayout.Alignment.LEADING, jPanel1Layout.createSequentialGroup()  .addComponent(stuLabel)  .addGap(3, 3, 3)))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10, Short.MAX\_VALUE)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(lNameLabel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(fNameLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 144, Short.MAX\_VALUE)  .addComponent(stuLabel2))  .addComponent(courseCombo, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addContainerGap(75, Short.MAX\_VALUE))  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addGap(39, 39, 39)  .addComponent(addButton)  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(backButton)  .addGap(41, 41, 41))  );  jPanel1Layout.setVerticalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addGap(37, 37, 37)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(stuLabel)  .addComponent(stuLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(fNameLabel)  .addComponent(fNameLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(lNameLabel)  .addComponent(lNameLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(courseLabel)  .addComponent(courseCombo, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 35, Short.MAX\_VALUE)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(backButton)  .addComponent(addButton))  .addGap(50, 50, 50))  );  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  pack();  }// </editor-fold>  /\*\*  \* The action performed event for the add button on the add Student UI.  \* This will add the users input of the student data fields to the  \* course for the student.  \* @param evt  \*/  private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {    // Sets the variables to get the data from the text fields and combo box.  String stuID = stuLabel2.getText();  String fName = fNameLabel2.getText();  String lName = lNameLabel2.getText();  String courseCom = courseCombo.getSelectedItem().toString();    // Initalises variabes.  Course course = null;  int cIndex = 0;    // Checks that no fields are left empty.  if(stuID.equals("") || fName.equals("") || lName.equals(""))  {  // Displays a message diaglog box asking the user to fill  // out all of the fields required.  JOptionPane.showMessageDialog(this, "Please fill in all the boxes");  }  // If all the fields are complete then it will check the inputs.  else  {  // Checks that the home phone number is 8 digits long.  if(!(stuID.length() == 8))  {  // Displays a message diaglog box asking the user to enter  // a student number that is 8 digits.  JOptionPane.showMessageDialog(this, "Please enter a Student ID of 8 characters");  }  // Checks that the first name entered is greater than 0 and  // less than or equal to 25 characters.  else if(!(fName.length() <= 25 && fName.length() > 0))  {  // Displays a message dialog box asking the user to enter a first  // name that is between 0 and 25 characters long.  JOptionPane.showMessageDialog(this, "Please enter a student first name between 1 and 25 characters.");  }  // Checks that the last name entered is greater than 0  // and less than or equal to 25 characters.  else if(!(lName.length() <= 25 && lName.length() > 0))  {  // Displays a message dialog box asking the user to enter a  // last name between 0 and 25 characters long.  JOptionPane.showMessageDialog(this, "Please enter a student last name between 1 and 25 characters.");  }  // If all of the validation is passed then it will create the student.  else  {  // Creates a student object using the user inputs given.  Student s1 = new Student(stuID,fName,lName,courseCom);    // Loops throught the courses array using a for loop.  for(Course cou : courses)  {  // Checks if the course code equals the current coude selectect in the combo box.  if(cou.getCode().equals(courseCom))  {  course = cou;  cIndex = courses.indexOf(cou);  }  }    // Updates the student number for the course  courses.get(cIndex).getStudents().add(s1);  // Displays a message dialog box informing the user that the student has been added.  JOptionPane.showMessageDialog(this, "The Student has been added.");  // Takes the user back to the main UI screen.  StudentMainUI stMain = new StudentMainUI(courses);  stMain.setVisible(true);  this.setVisible(false);  }  }  }  /\*\*  \* The action performed event for the back button on the add student UI.  \* This method will take the user back to the main UI screen when the button is pressed.  \* @param evt  \*/  private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {  StudentMainUI stMain = new StudentMainUI(courses);  stMain.setVisible(true);  this.setVisible(false);  }  /\*\*  \* The main method for the add student UI.  \* @param args the command line arguments  \*/  public static void main(String args[])  {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Nimbus".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(AddStudentsUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(AddStudentsUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(AddStudentsUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(AddStudentsUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>  /\* Create and display the form \*/  java.awt.EventQueue.invokeLater(new Runnable()  {  public void run()  {  new AddStudentsUI(courses).setVisible(true);  }  });  }  // Variables declaration - do not modify  private javax.swing.JButton addButton;  private javax.swing.JButton backButton;  private javax.swing.JComboBox<String> courseCombo;  private javax.swing.JLabel courseLabel;  private javax.swing.JLabel fNameLabel;  private javax.swing.JTextField fNameLabel2;  private javax.swing.JPanel jPanel1;  private javax.swing.JLabel lNameLabel;  private javax.swing.JTextField lNameLabel2;  private javax.swing.JLabel stuLabel;  private javax.swing.JTextField stuLabel2;  // End of variables declaration  } |

AddMarksUI:

|  |
| --- |
| /\*  \* The student mark systems Add Mark UI class that loads up when the add mark  \* button on the main ui is pressed.  \*/  package studentmark;  import java.util.ArrayList;  import javax.swing.DefaultComboBoxModel;  import javax.swing.JOptionPane;  /\*\*  \* @author Heidi Portwine (s6110438)  \*/  public class AddMarksUI extends javax.swing.JFrame  {  // Variables declared.  private static ArrayList<Course> courses = new ArrayList();  static Student student;  public DefaultComboBoxModel modulesModel = new DefaultComboBoxModel();  static Course course;  static Module module;    /\*\*  \* Creates new form AddMarksUI passing through the array list course and student.  \*/  public AddMarksUI(ArrayList<Course> c, Student stu)  {  student = stu;  courses = c;  initComponents();  initComboBox();    // Sets the labels.  stuLabel2.setText(student.getStudentID());  fNameLabel2.setText(student.getfName());  lNameLabel2.setText(student.getlName());  }    /\*\*  \* Initalises the combo box.  \*/  public void initComboBox()  {  // Loops through the courses.  for(Course c : courses)  {  // Checks if the course code is equal to the students course.  if(c.getCode().equals(student.getCourse()))  {  course = c;    // Loops through modules.  for(Module mod : c.getModule())  {  // Checks if the student does not have marks for the module already.  if(!student.getMarks().containsKey(mod.getCode()))  {  // Adds the module code to the model.  module = mod;  modulesModel.addElement(mod.getCode());  }  }  }  }  // Sets the module box to dsplay the model.  moduleBox.setModel(modulesModel);  }  /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents() {  jPanel1 = new javax.swing.JPanel();  addButton = new javax.swing.JButton();  backButton = new javax.swing.JButton();  moduleBox = new javax.swing.JComboBox<>();  moduleLabel = new javax.swing.JLabel();  markField = new javax.swing.JTextField();  markLabel = new javax.swing.JLabel();  stuLabel = new javax.swing.JLabel();  stuLabel2 = new javax.swing.JLabel();  fnameLabel = new javax.swing.JLabel();  jLabel1 = new javax.swing.JLabel();  lNameLabel2 = new javax.swing.JLabel();  fNameLabel2 = new javax.swing.JLabel();  setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  addButton.setText("Add");  addButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  addButtonActionPerformed(evt);  }  });  backButton.setText("Back");  backButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  backButtonActionPerformed(evt);  }  });  moduleBox.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Module 1", "Module 2", "Module 3", "Module 4" }));  moduleLabel.setText("Modules:");  markLabel.setText("Mark:");  stuLabel.setText("Student ID");  stuLabel2.setText("studentID");  fnameLabel.setText("First Name : ");  jLabel1.setText("Last Name :");  lNameLabel2.setText("lastName");  fNameLabel2.setText("firstName");  javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  jPanel1.setLayout(jPanel1Layout);  jPanel1Layout.setHorizontalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(61, 61, 61)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(moduleLabel)  .addComponent(markLabel))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(markField, javax.swing.GroupLayout.PREFERRED\_SIZE, 159, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addComponent(moduleBox, javax.swing.GroupLayout.PREFERRED\_SIZE, 159, javax.swing.GroupLayout.PREFERRED\_SIZE)))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(stuLabel)  .addComponent(jLabel1)  .addComponent(fnameLabel))  .addGap(41, 41, 41)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(fNameLabel2)  .addComponent(lNameLabel2)  .addComponent(stuLabel2))))  .addContainerGap(73, Short.MAX\_VALUE))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(35, 35, 35)  .addComponent(addButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(backButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(43, 43, 43))  );  jPanel1Layout.setVerticalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(34, 34, 34)  .addComponent(stuLabel2))  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addContainerGap()  .addComponent(stuLabel)))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(fnameLabel)  .addComponent(fNameLabel2))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 20, Short.MAX\_VALUE)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel1)  .addComponent(lNameLabel2))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(moduleLabel)  .addComponent(moduleBox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(18, 18, 18)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(markField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addComponent(markLabel))  .addGap(33, 33, 33)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(backButton)  .addComponent(addButton))  .addGap(35, 35, 35))  );  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  pack();  }// </editor-fold>  /\*\*  \* The action performed event for the add button on the add marks UI.  \* This will add the users input of mark to the correct module for the student.  \* @param evt  \*/  private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {    // Sets the variables to get the data from the text field and combo box.  int m = Integer.parseInt(markField.getText());  String mb = moduleBox.getSelectedItem().toString();    int cIndex = courses.indexOf(course);  int mIndex = courses.get(cIndex).getModule().indexOf(module);  int stuIndex = courses.get(cIndex).getStudents().indexOf(student);    // Sets the student number for the course to increase by one student and adds the marks.  int studNum = courses.get(cIndex).getModule().get(mIndex).getStudentNumber() +1;  courses.get(cIndex).getModule().get(mIndex).setStudentNumber(studNum);  courses.get(cIndex).getStudents().get(stuIndex).addModMark(mb, m);    // Displays a message dialog stating that the mark has been added.  JOptionPane.showMessageDialog(this, "The Mark has been added.");  // Returns to the main UI.  StudentMainUI stMain = new StudentMainUI(courses);  stMain.setVisible(true);  this.setVisible(false);    }  /\*\*  \* The action performed event for the back button on the add mark UI.  \* This method will take the user back to the main UI screen when the button is pressed.  \* @param evt  \*/  private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {  StudentMainUI stMain = new StudentMainUI(courses);  stMain.setVisible(true);  this.setVisible(false);  }  /\*\*  \* The main method for the add marks UI.  \* @param args the command line arguments  \*/  public static void main(String args[])  {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Nimbus".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(AddMarksUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(AddMarksUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(AddMarksUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(AddMarksUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>  /\* Create and display the form \*/  java.awt.EventQueue.invokeLater(new Runnable()  {  public void run()  {  new AddMarksUI(courses, student).setVisible(true);  }  });  }  // Variables declaration - do not modify  private javax.swing.JButton addButton;  private javax.swing.JButton backButton;  private javax.swing.JLabel fNameLabel2;  private javax.swing.JLabel fnameLabel;  private javax.swing.JLabel jLabel1;  private javax.swing.JPanel jPanel1;  private javax.swing.JLabel lNameLabel2;  private javax.swing.JTextField markField;  private javax.swing.JLabel markLabel;  private javax.swing.JComboBox<String> moduleBox;  private javax.swing.JLabel moduleLabel;  private javax.swing.JLabel stuLabel;  private javax.swing.JLabel stuLabel2;  // End of variables declaration  } |

ViewCourseUI

|  |
| --- |
| /\*  \* The student mark systems View Course UI class that loads up when the view course  \* button on the main ui is pressed.  \*/  package studentmark;  import java.util.ArrayList;  import javax.swing.DefaultComboBoxModel;  import javax.swing.DefaultListModel;  /\*\*  \* @author Heidi Portwine (S6110438)  \*/  public class ViewCourseUI extends javax.swing.JFrame  {  // Variables Declared.  private static ArrayList<Course> courses = new ArrayList();  private static ArrayList<Student> students = new ArrayList();  public DefaultComboBoxModel coursesModel = new DefaultComboBoxModel();  public DefaultListModel infoListModel = new DefaultListModel();  public DefaultListModel infoListModel2 = new DefaultListModel();  static Course course;  static Module module;    /\*\*  \* Creates new form ViewCourseUI passing through the array list course.  \*/  public ViewCourseUI(ArrayList<Course> c)  {  courses = c;  initComponents();  initComboBox();  viewModuleList.setEnabled(false);  }    /\*\*  \* Initalises the combo box for the courses.  \*/  public void initComboBox()  {  // Adds a message to the model first.  coursesModel.addElement("Select ");    // Loops through the courses.  for(Course c : courses)  {  // Add the course codes to the model.  coursesModel.addElement(c.code);  }  // Sets the course combo object to display the courses model.  courseCombo.setModel(coursesModel);  }  /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents() {  jPanel1 = new javax.swing.JPanel();  backButton = new javax.swing.JButton();  courseCombo = new javax.swing.JComboBox<>();  jScrollPane1 = new javax.swing.JScrollPane();  viewList = new javax.swing.JList<>();  jScrollPane2 = new javax.swing.JScrollPane();  viewModuleList = new javax.swing.JList<>();  setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  backButton.setText("Back");  backButton.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  backButtonActionPerformed(evt);  }  });  courseCombo.addItemListener(new java.awt.event.ItemListener() {  public void itemStateChanged(java.awt.event.ItemEvent evt) {  courseComboItemStateChanged(evt);  }  });  viewList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE\_SELECTION);  viewList.addListSelectionListener(new javax.swing.event.ListSelectionListener() {  public void valueChanged(javax.swing.event.ListSelectionEvent evt) {  viewListValueChanged(evt);  }  });  jScrollPane1.setViewportView(viewList);  viewModuleList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE\_SELECTION);  jScrollPane2.setViewportView(viewModuleList);  javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  jPanel1.setLayout(jPanel1Layout);  jPanel1Layout.setHorizontalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addContainerGap()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addGap(0, 0, Short.MAX\_VALUE)  .addComponent(backButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 97, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)  .addComponent(courseCombo, 0, 109, Short.MAX\_VALUE))  .addGap(18, 18, 18)  .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 225, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(0, 8, Short.MAX\_VALUE)))  .addContainerGap())  );  jPanel1Layout.setVerticalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addContainerGap()  .addComponent(courseCombo, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT\_SIZE, 144, Short.MAX\_VALUE)  .addComponent(jScrollPane1))  .addGap(18, 18, 18)  .addComponent(backButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 23, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(38, 38, 38))  );  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addContainerGap()  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addContainerGap())  );  pack();  }// </editor-fold>  /\*\*  \* The action performed event for the back button on the view course UI.  \* This method will take the user back to the main UI screen when the button is pressed.  \* @param evt  \*/  private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {  StudentMainUI stMain = new StudentMainUI(courses);  stMain.setVisible(true);  this.setVisible(false);  }  /\*\*  \* The method triggered when the view list value is changed.  \* This method displays the modules in the array on to the list  \* so the user can select one to view.  \* @param evt  \*/  private void viewListValueChanged(javax.swing.event.ListSelectionEvent evt) {  // Clears the model to prevent duplicates.  infoListModel2.clear();    // Checks if the view list selection is not empty.  if(!viewList.isSelectionEmpty())  {  if(evt.getValueIsAdjusting() == false)  {  // Loops through the courses.  for(Course cou : courses)  {  // Loops through the modules.  for(Module mod : cou.getModule())  {  // Checks if the view list selection is not empty  if(!viewList.isSelectionEmpty())  {  // Sets the variables to get the modules data.  String modT = mod.getModuleTitle();  int passM = mod.getPassMark();  int stN = mod.getStudentNumber();  double a = 0;  int p = 0;  int f = 0;    // Checks that the item selected equals the module code.  if(viewList.getSelectedValue().equals(mod.getCode()))  {  // Sets students to be the courses students.  students = cou.getStudents();    // Checks that the student number is not 0.  if(stN != 0)  {  // Sets the averages, passes, fails.  a = mod.getAverage(students);  p = mod.getPass(students);  f = mod.getFail(students);  }  // Adds the elements to the model so they can be displayed in the correct layout.  infoListModel2.addElement("Module Title : " + modT);  infoListModel2.addElement("Pass Mark : " + passM);  infoListModel2.addElement("Number of Students : " + stN);  infoListModel2.addElement("Average Mark : " + a);  infoListModel2.addElement("Number of Passes : " + p);  infoListModel2.addElement("Number of Fails : " + f);  }  }  }  }  // Sets the view module list object to dsplay the model.  viewModuleList.setModel(infoListModel2);  }  }    }  /\*\*  \* This method deals with the combo box xhange in state.  \* @param evt  \*/  private void courseComboItemStateChanged(java.awt.event.ItemEvent evt) {    // Clears the model to prevent duplicates.  infoListModel.clear();  // Checks that the event is due to the combo box.  if(evt.getSource() == courseCombo)  {  // Sets the variables to combo box item.  String courseCom = courseCombo.getSelectedItem().toString();  int cIndex = 0;  int mIndex = 0;    // Loops through the courses.  for(Course cou : courses)  {  // Checks that the course code is equal to the combo box.  if(cou.getCode().equals(courseCom))  {  // Loops through the modules.  for(Module mod : cou.getModule())  {  // Adds the module codes to the model.  infoListModel.addElement(mod.getCode());  }  }  // Sets the view list to display the model.  viewList.setModel(infoListModel);    }  }  }  /\*\*  \* The main method for the view course UI.  \* @param args the command line arguments  \*/  public static void main(String args[])  {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Nimbus".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(ViewCourseUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(ViewCourseUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(ViewCourseUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(ViewCourseUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>  /\* Create and display the form \*/  java.awt.EventQueue.invokeLater(new Runnable()  {  public void run()  {  new ViewCourseUI(courses).setVisible(true);  }  });  }  // Variables declaration - do not modify  private javax.swing.JButton backButton;  private javax.swing.JComboBox<String> courseCombo;  private javax.swing.JPanel jPanel1;  private javax.swing.JScrollPane jScrollPane1;  private javax.swing.JScrollPane jScrollPane2;  private javax.swing.JList<String> viewList;  private javax.swing.JList<String> viewModuleList;  // End of variables declaration  } |