KK14203 OBJECT ORIENTED PROGRAMMING

(SECTION 1)

SEMESTER 2, SESSION 2019/2020

PROJECT 2 REPORT

PROJECT TITLE: COURSE GRADE FOR HC00

NAME : LEVANNYAH A/P RAJASEGARAN

MATRIC NO.: BI19160337

LECTURER : DR MOHD SHAMRIE SAININ

CONTENT

1.0 JAVA CODE

1.1 Login.java

```
//To login, Username: Admin, Password: Admin
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;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class Login implements ActionListener{
   private static JLabel lbl_user;
   private static JTextField userText;
   private static JLabel lbl_password;
   private static JPasswordField passwordText;
   private static JButton button;
   private static JLabel success;

    public static void main(String[] args) {
        JPanel panel = new JPanel();
        JFrame frame = new JFrame("Login");
        frame.setSize(300, 180);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(panel);
        panel.setLayout(null);

        lbl_user = new JLabel("User");
        lbl_user.setBounds(10, 20, 80, 25);
        panel.add(lbl_user);

        userText = new JTextField(20);
        userText.setBounds(100, 20, 165, 25);
        panel.add(userText);

        lbl_password = new JLabel("Password");
        lbl_password.setBounds(10, 50, 80, 25);
        panel.add(lbl_password);

        passwordText = new JPasswordField();
        passwordText.setBounds(100, 50, 165, 25);
        panel.add(passwordText);
```

```
        button = new JButton("Login");
        button.setBounds(185, 80, 80, 25);
        button.addActionListener(new Login());
        panel.add(button);

        success = new JLabel("");
        success.setBounds(10, 110, 300, 25);
        panel.add(success);

        frame.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
      String user = userText.getText();
      String password = passwordText.getText();

         if (user.equals("Admin") && password.equals("Admin")) {
            success.setText("Login successful! Welcome, " + user + ".");
         } else {
            success.setText("Invalid username and password!");
         }
      }
  }
```

1.2 Project2.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Project2{

    private double totalMarks = 0.0;
    private String grade;
    private String gradeStatus;
    private double courseGp = 0.0;

    public Project2(){
    }

    //Accessors and Mutators
    public double getTotalMarks(){
       return totalMarks;
    }

    public void setTotalMarks(double totalMarks){
       this.totalMarks = totalMarks;
    }

    public String getGrade(){
       return grade;
    }

    public void setGrade(String grade){
       this.grade = grade;
    }

    public String getGradeStatus(){
       return gradeStatus;
    }

    public void setGradeStatus(String gradeStatus){
       this.gradeStatus = gradeStatus;
    }

    public double getCourseGp(){
       return courseGp;
    }
```

```
public void setCourseGpa(double courseGp){
   this.courseGp = courseGp;
}

//calculate total marks entered
public double calculateMarks(int assignment, int quizzes, int midterm, int finalexam){
   totalMarks = (assignment * 0.3) + (quizzes * 0.1) + (midterm * 0.2) + (finalexam *
   0.4);
   return totalMarks;
}

//get grade based on total marks
public String getGrade(double totalMarks){
   if (totalMarks >= 80.0){
        grade = "A";
   }else if (totalMarks >= 75.0){
        grade = "A-";
   }else if (totalMarks >= 70.0){
        grade = "B+";
   }else if (totalMarks >= 65.0){
        grade = "B";
   }else if (totalMarks >= 60.0){
        grade = "B-";
   }else if (totalMarks >= 55.0){
        grade = "C+";
   }else if (totalMarks >= 50.0){
        grade = "C";
   }else if (totalMarks >= 45.0){
        grade = "C-";
   }else if (totalMarks >= 40.0){
        grade = "D+";
   }else if (totalMarks >= 35.0){
        grade = "D";
   }else {
        grade = "E";
   }

    return grade;
 }

   //get grade status based on grade obtained
   public String getGradeStatus(String grade){
     if (grade == "A" || grade == "A-"){
          gradeStatus = "PASS WITH A DISTINCTION";
     }else if (grade == "B+" || grade == "B" || grade == "B-"){
          gradeStatus = "PASS WITH A CREDIT";
```

```
        }else if (grade == "C+" || grade == "C" || grade == "C-" || grade == "D+" || grade
          == "D"){
            gradeStatus = "PASS";
        }else {
            gradeStatus = "FAIL";
        }

        return gradeStatus;
      }

      //get grade points for course based on grade
       public double getCourseGp(double totalMarks) {
        if (totalMarks >= 80.0){
          courseGp = 4.00;
        }else if (totalMarks >= 75.0){
          courseGp = 3.67;
        }else if (totalMarks >= 70.0){
          courseGp = 3.33;
        }else if (totalMarks >= 65.0){
          courseGp = 3.00;
        }else if (totalMarks >= 60.0){
          courseGp = 2.67;
        }else if (totalMarks >= 55.0){
          courseGp = 2.33;
        }else if (totalMarks >= 50.0){
          courseGp = 2.00;
        }else if (totalMarks >= 45.0){
          courseGp = 1.67;
        }else if (totalMarks >= 40.0){
           courseGp = 1.33;
        }else if (totalMarks >= 35.0){
          courseGp = 1.00;
        }else {
          courseGp = 0.00;
       }

       return courseGp;
    }
}
```

1.3 Project2_GUI.java

```
//This program accepts marks from user and outputs total marks, grade, grade status and
grade points
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.BufferedWriter;
import java.io.BufferedReader;

import java.io.IOException;

 class MyPanel extends JPanel{
    private JLabel lbl_faculty;
    private JLabel lbl_programme;
    private JLabel lbl_name;
    private JTextField name;
    private JLabel lbl_session;
    private JLabel lbl_ic;
    private JTextField ic;
    private JLabel lbl_matric_no;
    private JTextField matric_no;
    private JLabel lbl_course;
    private JComboBox course;
    private JLabel lbl_marks;
    private JLabel lbl_assignment;
    private JTextField assignment;
    private JLabel lbl_quizzes;
    private JTextField quizzes;
    private JLabel lbl_midterm;
    private JTextField midterm;
    private JLabel lbl_finalexam;
    private JTextField finalexam;
    private JButton btn_submit;
    private JButton btn_clear;
    JTextArea textArea;
    JScrollPane jsp;

    Project2 p = new Project2();
```

```
String output = " ";
String course_selection = " ";
String filePath = "CourseGradeHC00.txt";

public MyPanel() {
    //construct preComponents
    String[] courseItems = {"[SELECT]", "KK14203 OBJECT ORIENTED PROGRAMMING",
    "KT14203 COMPUTER ARCHITECTURE & ORGANISATION", "KT14403 DISCRETE
     STRUCTURES", "UB02002 ENGLISH FOR EMPLOYMENT", "UC01202 NEGOTIATION
     SKILLS", "UW00102 HUBUNGAN ETNIK"};

    //adjust size and set layout
    setPreferredSize (new Dimension (666, 485));
    setLayout (null);

    //construct components, add components, set component bounds
    lbl_faculty = new JLabel ("FACULTY OF COMPUTING AND INFORMATICS");
    add (lbl_faculty);
    lbl_faculty.setBounds (190, -5, 300, 30);

    lbl_programme = new JLabel ("PROGRAMME: HC00 - SOFTWARE ENGINEERING");
    add (lbl_programme);
    lbl_programme.setBounds (180, 25, 300, 25);

    lbl_name = new JLabel ("NAME:");
    add (lbl_name);
    lbl_name.setBounds (15, 85, 100, 25);

    name = new JTextField (5);
    add (name);
    name.setBounds (95, 85, 545, 25);

    lbl_session = new JLabel ("SEMESTER 2 (2019/2020)");
    add (lbl_session);
    lbl_session.setBounds (245, 50, 145, 25);

    lbl_ic = new JLabel ("IC NO: ");
    add (lbl_ic);
    lbl_ic.setBounds (15, 115, 100, 25);

    ic = new JTextField (5);
    add (ic);
    ic.setBounds (95, 115, 175, 25);

    //TextField for ic will only accept integers
     ic.addKeyListener(new KeyAdapter(){
```

```
 public void keyTyped (KeyEvent e)
{
    char c = e.getKeyChar() ;

    if (!((c==KeyEvent.VK_BACK_SPACE) || (c==KeyEvent.VK_DELETE) ||  (c==
KeyEvent.VK_ENTER) || (c == KeyEvent.VK_TAB) ||  (Character.isDigit(c))))
    {
      e.consume() ;
    }
 }
});

lbl_matric_no = new JLabel ("MATRIC NO.:");
add (lbl_matric_no);
lbl_matric_no.setBounds (315, 115, 100, 25);

matric_no = new JTextField (5);
add (matric_no);
matric_no.setBounds (400, 115, 145, 25);

lbl_course = new JLabel ("COURSE:");
add (lbl_course);
lbl_course.setBounds (15, 150, 100, 25);

course = new JComboBox (courseItems);
add (course);
course.setBounds (95, 150, 360, 25);

//JComboBox action listener
course.addActionListener(new ActionListener(){
 public void actionPerformed(ActionEvent ae){
    course_selection = (String) course.getSelectedItem();
 }
});

lbl_marks = new JLabel ("ENTER MARKS FOR:");
add (lbl_marks);
lbl_marks.setBounds (15, 185, 145, 25);

lbl_assignment = new JLabel ("ASSIGNMENTS:");
add (lbl_assignment);
lbl_assignment.setBounds (15, 215, 100, 25);

assignment = new JTextField (5);
add (assignment);
assignment.setBounds (115, 215, 55, 25);
```

```
  assignment.addKeyListener(new KeyAdapter(){
 public void keyTyped (KeyEvent e)
{
    char c = e.getKeyChar() ;

    if (!((c==KeyEvent.VK_BACK_SPACE) || (c==KeyEvent.VK_DELETE) ||  (c==
        KeyEvent.VK_ENTER) || (c == KeyEvent.VK_TAB) ||  (Character.isDigit(c))))
    {
       e.consume() ;
    }
 }
});

lbl_quizzes = new JLabel ("QUIZZES:");
add (lbl_quizzes);
lbl_quizzes.setBounds (195, 215, 60, 25);

quizzes = new JTextField (5);
add (quizzes);
quizzes.setBounds (255, 215, 55, 25);

quizzes.addKeyListener(new KeyAdapter(){
 public void keyTyped (KeyEvent e)
{
    char c = e.getKeyChar() ;

    if (!((c==KeyEvent.VK_BACK_SPACE) || (c==KeyEvent.VK_DELETE) ||  (c==
        KeyEvent.VK_ENTER) || (c == KeyEvent.VK_TAB) ||  (Character.isDigit(c))))
    {
       e.consume() ;
    }
 }
});

lbl_midterm = new JLabel ("MIDTERM:");
add (lbl_midterm);
lbl_midterm.setBounds (335, 215, 100, 25);

midterm = new JTextField (5);
add (midterm);
midterm.setBounds (405, 215, 55, 25);

 midterm.addKeyListener(new KeyAdapter(){
 public void keyTyped (KeyEvent e)
{
    char c = e.getKeyChar() ;
```

```
        if (!((c==KeyEvent.VK_BACK_SPACE) || (c==KeyEvent.VK_DELETE) ||  (c==
            KeyEvent.VK_ENTER) || (c == KeyEvent.VK_TAB) ||  (Character.isDigit(c))))
        {
           e.consume() ;
        }
     }
   });

   lbl_finalexam = new JLabel ("FINAL EXAM:");
   add (lbl_finalexam);
   lbl_finalexam.setBounds (490, 215, 100, 25);

   finalexam = new JTextField (5);
   add (finalexam);
   finalexam.setBounds (575, 215, 55, 25);

   finalexam.addKeyListener(new KeyAdapter(){
    public void keyTyped (KeyEvent e)
   {
        char c = e.getKeyChar() ;

        if (!((c==KeyEvent.VK_BACK_SPACE) || (c==KeyEvent.VK_DELETE) ||  (c==
            KeyEvent.VK_ENTER) || (c == KeyEvent.VK_TAB) ||  (Character.isDigit(c))))
        {
           e.consume() ;
        }
     }
   });

   btn_submit = new JButton ("SUBMIT");
   add (btn_submit);
   btn_submit.setBounds (15, 255, 100, 25);

   btn_clear = new JButton ("CLEAR");
   add (btn_clear);
   btn_clear.setBounds (125, 255, 100, 25);

  btn_submit.addActionListener(new ActionListener(){
     public void actionPerformed(ActionEvent e){
        if(printOutput())
           writeInput();
     }
  });

 btn_clear.addActionListener(new ActionListener(){
```

```
    public void actionPerformed(ActionEvent e){
        textArea.setText(" ");
        name.setText(" ");
        ic.setText(" ");
        matric_no.setText(" ");
        course.setSelectedIndex(0);
        assignment.setText(" ");
        quizzes.setText(" ");
        midterm.setText(" ");
        finalexam.setText(" ");
    }
  });

    textArea = new JTextArea (5, 5);

    //add JScrollPane
    jsp = new JScrollPane(textArea);
    jsp.setBounds(15, 290, 640, 185);
    add(jsp);
}

     //print output to textArea and input validation to check for empty fields or selections
    public boolean printOutput(){

      if(name.getText().isEmpty()){
        JOptionPane.showMessageDialog(null, "Please enter a name. Thank you.");
        return false;
      }
      output = " NAME: " + name.getText() + "\n";

      if(ic.getText().isEmpty()){
        JOptionPane.showMessageDialog(null, "Please enter an IC no. Thank you.");
        return false;
      }
      output += " IC NO.: " + ic.getText() + "\n";

      if(matric_no.getText().isEmpty()){
        JOptionPane.showMessageDialog(null, "Please enter a matric no. Thank you.");
        return false;
      }
      output += " MATRIC NO: " + matric_no.getText() + "\n\n";

      if(course_selection.equals("[Select]") || course_selection.equals(" ")){
        JOptionPane.showMessageDialog(null, "Please select a course. Thank you.");
        return false;
      }
```

```
 output += " COURSE: " + course_selection + "\n\n";

 output += " MARKS ENTERED FOR: \n";

 if(assignment.getText().isEmpty()){
   JOptionPane.showMessageDialog(null, "Please enter marks for Assignment. Thank
   you.");
   return false;
}
 output += " ASSIGNMENT: " + assignment.getText() + "\n";

if(quizzes.getText().isEmpty()){
   JOptionPane.showMessageDialog(null, "Please enter marks for Quizzes. Thank
   you.");
   return false;
}
output += " QUIZZES: " + quizzes.getText() + "\n";

if(midterm.getText().isEmpty()){
   JOptionPane.showMessageDialog(null, "Please enter marks for Midterm. Thank
   you.");
   return false;
}
output += " MIDTERM: " + midterm.getText() + "\n";

if(finalexam.getText().isEmpty()){
   JOptionPane.showMessageDialog(null, "Please enter marks for Final Exam. Thank
   you.");
   return false;
}
output += " FINAL EXAM: " + finalexam.getText() + "\n\n";

try{
 p.calculateMarks(Integer.parseInt(assignment.getText()),
            Integer.parseInt(quizzes.getText()), Integer.parseInt(midterm.getText()),
            Integer.parseInt(finalexam.getText()));

 output += " TOTAL MARKS: " + p.getTotalMarks() + "\n";
 output += " GRADE: " + p.getGrade(p.getTotalMarks()) +"\n";
 output += " GRADE STATUS: " + p.getGradeStatus(p.getGrade()) + "\n";
 output += " GRADE POINTS: " + p.getCourseGp(p.getTotalMarks()) + "\n";

 } catch (NumberFormatException errorMsg) {
      JOptionPane.showMessageDialog(null, "Something went wrong! Please exit the
      program and run again if you wish to enter a new input.");
      return false;
```

```
        }

         output +=
"-------------------------------------------------------------------------------------------------------------
----------";

         textArea.setText(output);
         jsp.getViewport().revalidate();

         return true;
       }

     //write to file
     public void writeInput(){
      File file = new File(filePath);
                FileWriter fr = null;
                BufferedWriter br = null;
                PrintWriter pr = null;

      //exception implementation
                try{
                        fr = new FileWriter(file, true);
                        br = new BufferedWriter(fr);
                        pr = new PrintWriter(br);
                        pr.println(output);
        JOptionPane.showMessageDialog(null, "Input has been successfully saved.");
                } catch (IOException e) {
        textArea.setText(e.toString());
        JOptionPane.showMessageDialog(null, "Something went wrong. Please try again.");
                } finally {
                        try {
                              pr.close();
                              br.close();
                              fr.close();
                        } catch (IOException e) {
                                 textArea.setText(e.toString());
                        }
                }
         }

}

class MenuActionListener implements ActionListener {
   MyPanel pan;
   public MenuActionListener(MyPanel p){
      pan = p;
```

```
  }

  public void actionPerformed(ActionEvent e) {
     BufferedReader reader;
      try {
             reader = new BufferedReader(new FileReader(pan.filePath));
             String line = reader.readLine();
             String output = "Data :\n";
             while (line != null) {
             output += line + "\n";
             line = reader.readLine();
             }
             output += "\n";
             pan.textArea.setText(output);
             reader.close();
             } catch (IOException io) {
              pan.textArea.setText(io.toString());
             }

      }
}

class MenuActionListener2 implements ActionListener{
     MyPanel pan;
     public MenuActionListener2(MyPanel p){
     pan = p;
  }

 //for exit confirmation
 public void actionPerformed(ActionEvent e){
    int confirm = JOptionPane.showConfirmDialog(null, "Are you sure you want to exit?",
      "Exit Confirmation", JOptionPane.YES_NO_OPTION);
       if (confirm == JOptionPane.YES_OPTION)
       System.exit(0);
  }
}

public class Project2_GUI{
   public static void main (String[] args) {

       JFrame frame = new JFrame ("Course Grade for HC00");

       MyPanel pan = new MyPanel();
       JMenuBar mb = new JMenuBar();
       JMenu m = new JMenu("Menu");
```

```
        //View Data reads from file
        JMenuItem m1 = new JMenuItem("View Data");
        m1.addActionListener(new MenuActionListener(pan));
        JMenuItem m2 = new JMenuItem("Exit");
        m2.addActionListener(new MenuActionListener2(pan));
        m.add(m1);
        m.add(m2);
        mb.add(m);
        frame.setJMenuBar(mb);

        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add (new MyPanel());
        frame.pack();
        frame.setVisible (true);
    }
}
```

2.0 OBJECT ORIENTED CONCEPT IMPLEMENTATION

1. Encapsulation (Applied in public class Project2 (Project2.java))

```
public class Project2{

    private double totalMarks = 0.0;
    private String grade;
    private String gradeStatus;
    private double courseGp = 0.0;

    public Project2(){
    }

    //Accessors and Mutators
    public double getTotalMarks(){
      return totalMarks;
    }

    public void setTotalMarks(double totalMarks){
      this.totalMarks = totalMarks;
    }

    public String getGrade(){
      return grade;
    }

    public void setGrade(String grade){
      this.grade = grade;
    }

    public String getGradeStatus(){
      return gradeStatus;
    }

    public void setGradeStatus(String gradeStatus){
      this.gradeStatus = gradeStatus;
    }

    public double getCourseGp(){
      return courseGp;
    }

    public void setCourseGpa(double courseGp){
      this.courseGp = courseGp;
    }
```

2. Objects and Classes (Applied throughout the project)

3. Inheritance (Applied in class MyPanel (Project2_GUI.java))

```
class MyPanel extends JPanel{
```

4. Interfaces (Applied in class MenuActionListener and class MenuActionListener2 (Project2_GUI.java))

```
class MenuActionListener implements ActionListener{

class MenuActionListener2 implements ActionListener{
```

5. Abstraction (Applied in class MyPanel extends JPanel and class MenuActionListener implements ActionListener (Project2_GUI.java))

```
public void writeInput(){
    File file = new File(filePath);
            FileWriter fr = null;
            BufferedWriter br = null;
            PrintWriter pr = null;


public void actionPerformed(ActionEvent e) {
    BufferedReader reader;
     try {
        reader = new BufferedReader(new FileReader(pan.filePath));
```

3.0 READ AND WRITE IMPLEMENTATION

3.1 Read from file

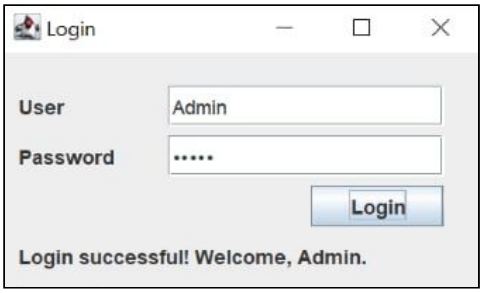
```
class MenuActionListener implements ActionListener{
   MyPanel pan;
   public MenuActionListener(MyPanel p){
      pan = p;
   }

   public void actionPerformed(ActionEvent e) {
      BufferedReader reader;
       try {
            reader = new BufferedReader(new FileReader(pan.filePath));
            String line = reader.readLine();
             String output = "Data :\n";
            while (line != null) {
                  output += line + "\n";
                  line = reader.readLine();
         }
           output += "\n";
           pan.textArea.setText(output);
           reader.close();
         } catch (IOException io) {
             pan.textArea.setText(io.toString());
         }

    }
}
```

3.2 Write to file

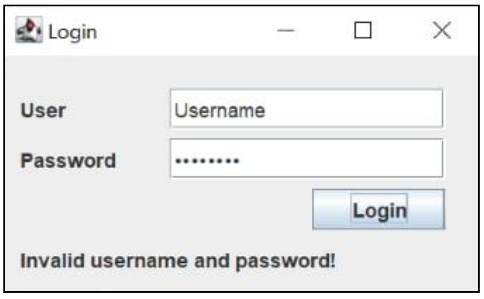
```
public void writeInput(){
    File file = new File(filePath);
    FileWriter fr = null;
    BufferedWriter br = null;
    PrintWriter pr = null;

    //exception implementation
        try{
            fr = new FileWriter(file, true);
            br = new BufferedWriter(fr);
            pr = new PrintWriter(br);
            pr.println(output);
            JOptionPane.showMessageDialog(null, "Input has been successfully saved.");
        } catch (IOException e) {
       textArea.setText(e.toString());
       JOptionPane.showMessageDialog(null, "Something went wrong. Please try again.");
      } finally {
            try {
                pr.close();
                br.close();
                fr.close();
             } catch (IOException e) {
                textArea.setText(e.toString());
            }
        }
  }
```

4.0 USER MANUAL

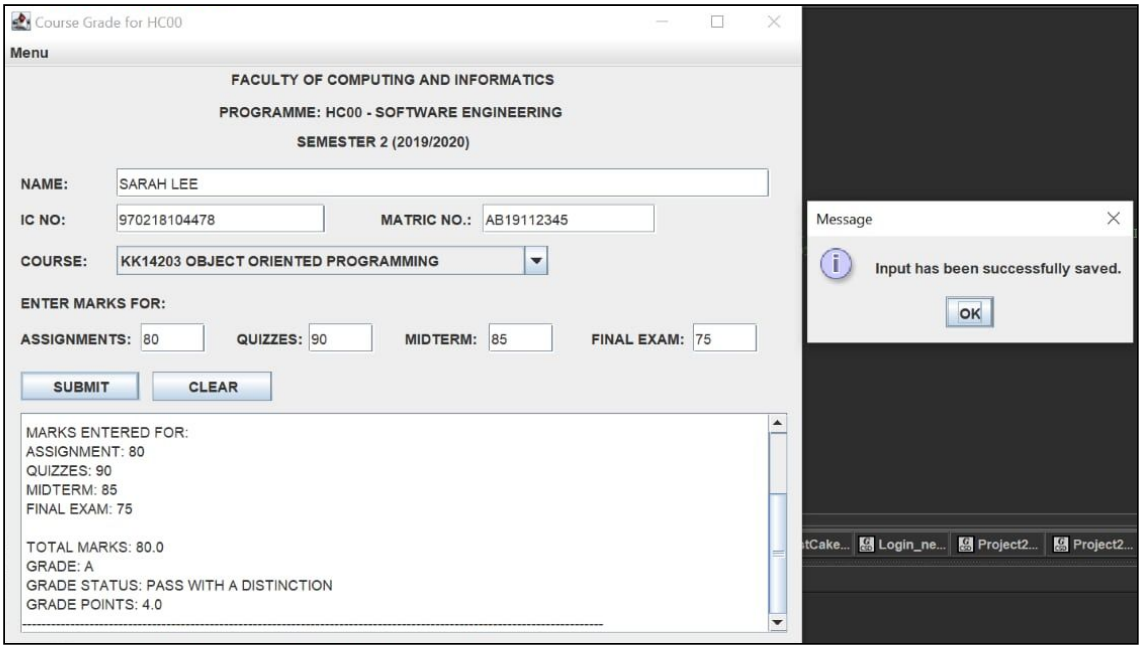
1. In Login.java, login with Username: Admin and Password: Admin.

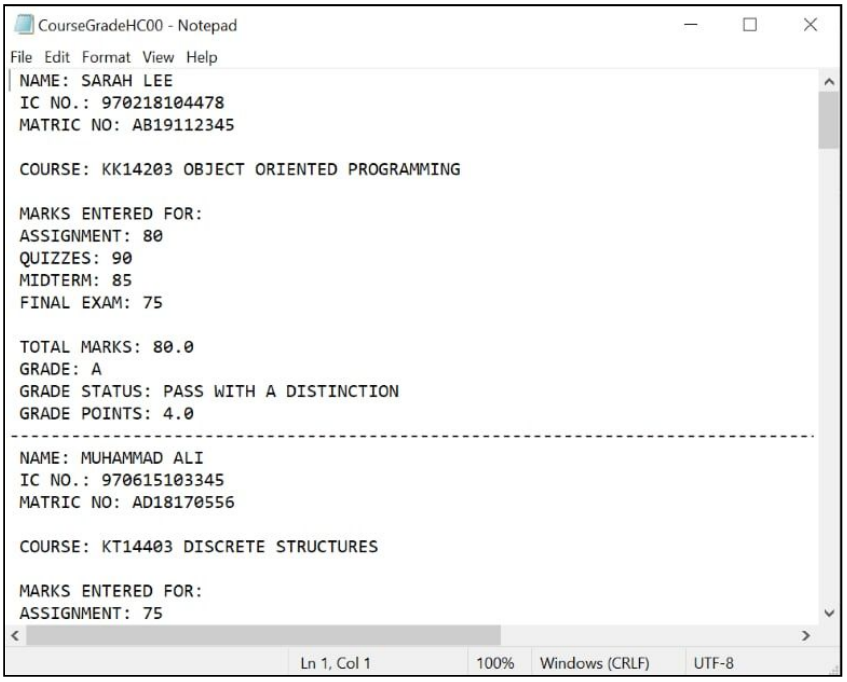


2. If Username and Password are incorrect, an error message is displayed. User may re enter Username and Password.

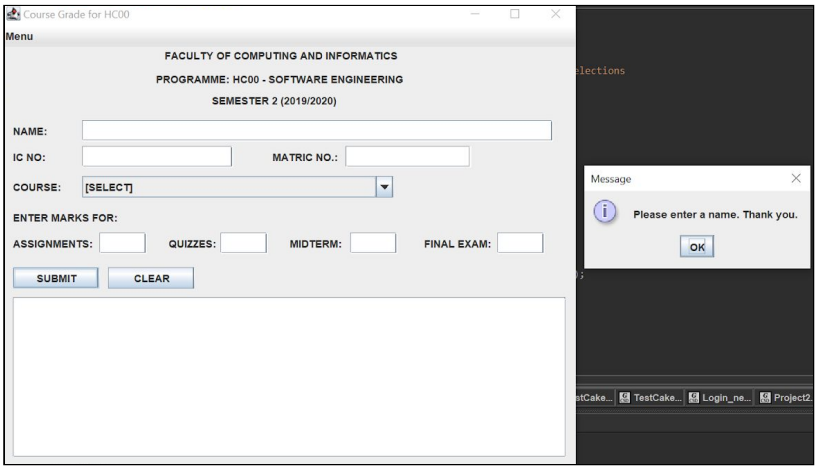


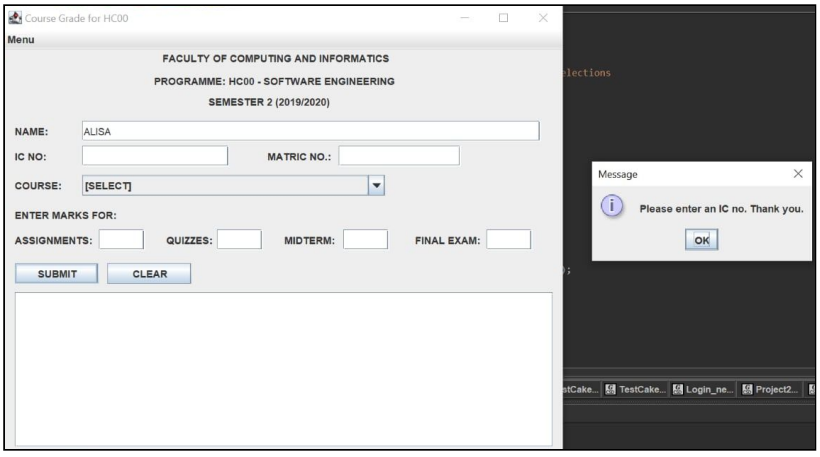
3. In Project2_GUI.java, enter name, IC no., matric no.. Choose a course. Enter marks for each field. Text field for IC no., assignments, quizzes, midterm and final exam will only accept integers. After user "SUBMITS" input, it is saved to the text file "CourseGradeHC00.txt" and a confirmation message is displayed. The total marks, grade, grade status and grade points will be displayed.

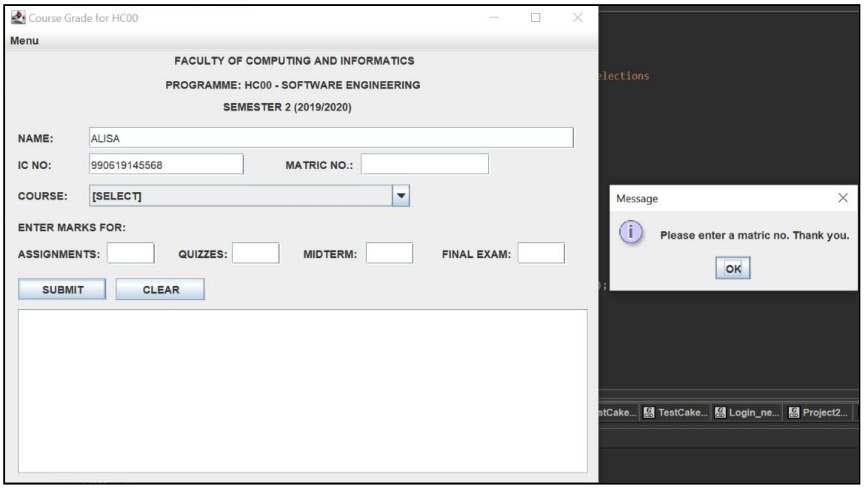


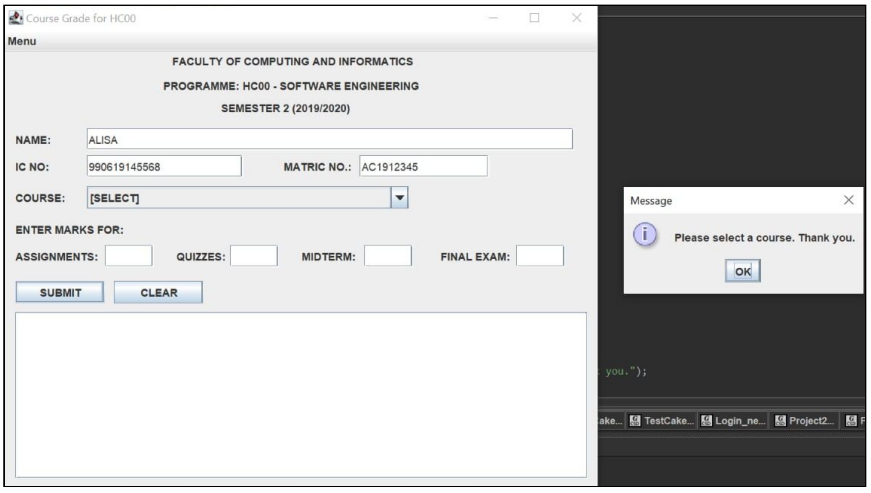


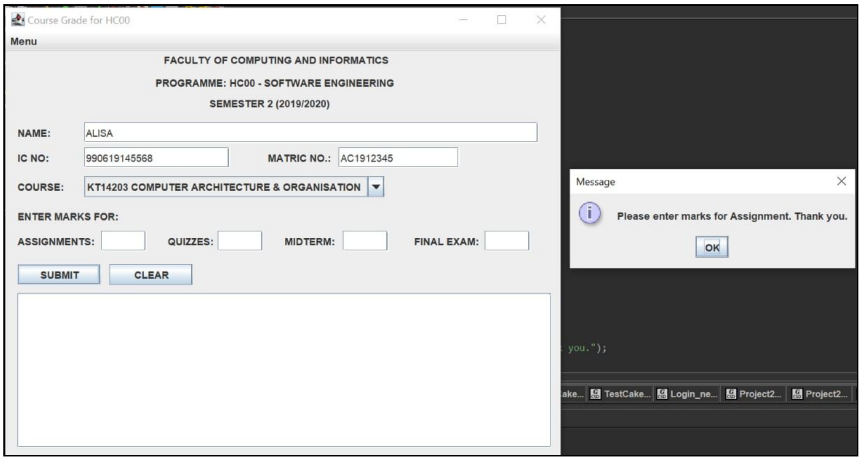
4. If any of the fields or selections are left empty, the user will be prompted to enter an input.

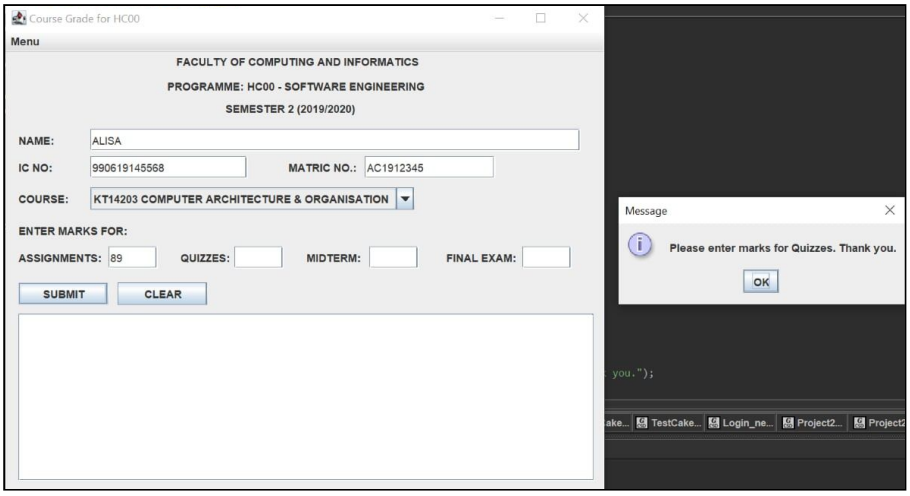


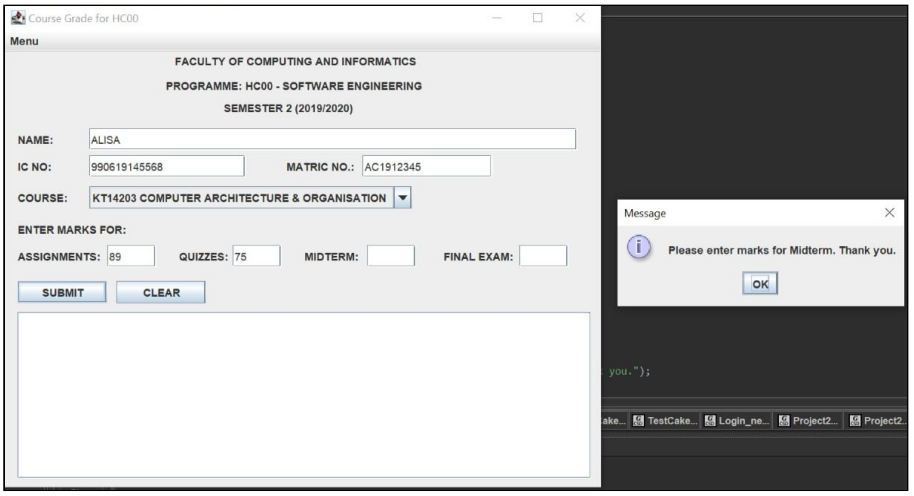


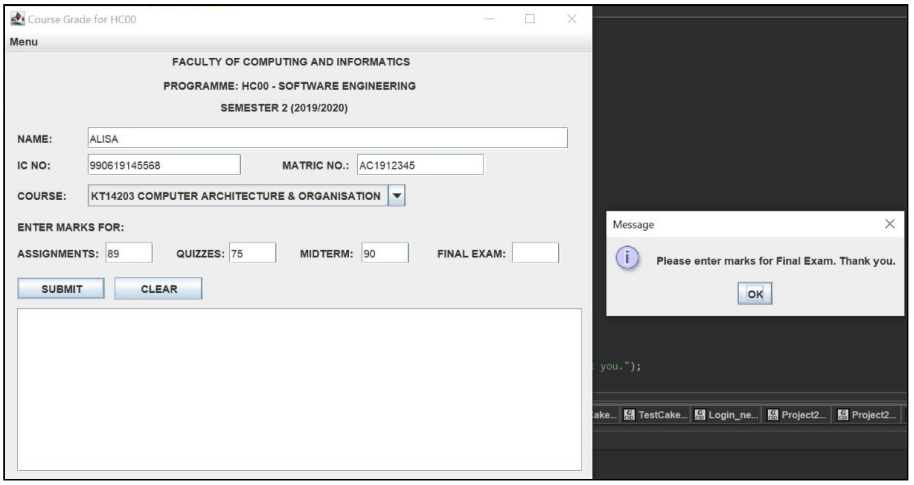

Course Grade for HC00
Menu
FACULTY OF COMPUTING AND INFORMATICS
PROGRAMME: HC00 - SOFTWARE ENGINEERING
SEMESTER 2 (2019/2020)
NAME:
ALISA
IC NO:
990619145568
MATRIC NO.:
COURSE:
[SELECT]
ENTER MARKS FOR:
ASSIGNMENTS:
QUIZZES:
MIDTERM:
FINAL EXAM:
SUBMIT
CLEAR
Message
Please enter a matric no. Thank you.
OK


Course Grade for HC00
Menu
FACULTY OF COMPUTING AND INFORMATICS
PROGRAMME: HC00 - SOFTWARE ENGINEERING
SEMESTER 2 (2019/2020)
NAME:
ALISA
IC NO:
990619145568
MATRIC NO.:
AC1912345
COURSE:
[SELECT]
ENTER MARKS FOR:
ASSIGNMENTS:
QUIZZES:
MIDTERM:
FINAL EXAM:
SUBMIT
CLEAR
Message
Please select a course. Thank you.
OK

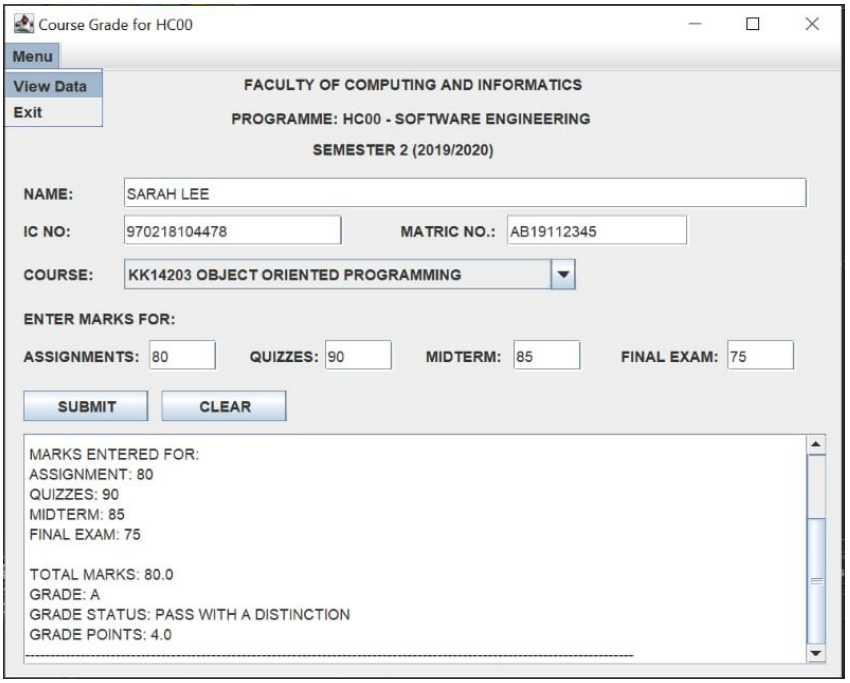

Course Grade for HC00
Menu
FACULTY OF COMPUTING AND INFORMATICS
PROGRAMME: HC00 - SOFTWARE ENGINEERING
SEMESTER 2 (2019/2020)
NAME:
ALISA
IC NO:
990619145568
MATRIC NO.:
AC1912345
COURSE:
KT14203 COMPUTER ARCHITECTURE & ORGANISATION
ENTER MARKS FOR:
ASSIGNMENTS:
QUIZZES:
MIDTERM:
FINAL EXAM:
SUBMIT
CLEAR
Message
Please enter marks for Assignment. Thank you.
OK


Course Grade for HC00
Menu
FACULTY OF COMPUTING AND INFORMATICS
PROGRAMME: HC00 - SOFTWARE ENGINEERING
SEMESTER 2 (2019/2020)
NAME: ALISA
IC NO: 990619145568
MATRIC NO.: AC1912345
COURSE: KT14203 COMPUTER ARCHITECTURE & ORGANISATION
ENTER MARKS FOR:
ASSIGNMENTS: 89
QUIZZES:
MIDTERM:
FINAL EXAM:
SUBMIT
CLEAR
Message
Please enter marks for Quizzes. Thank you.
OK

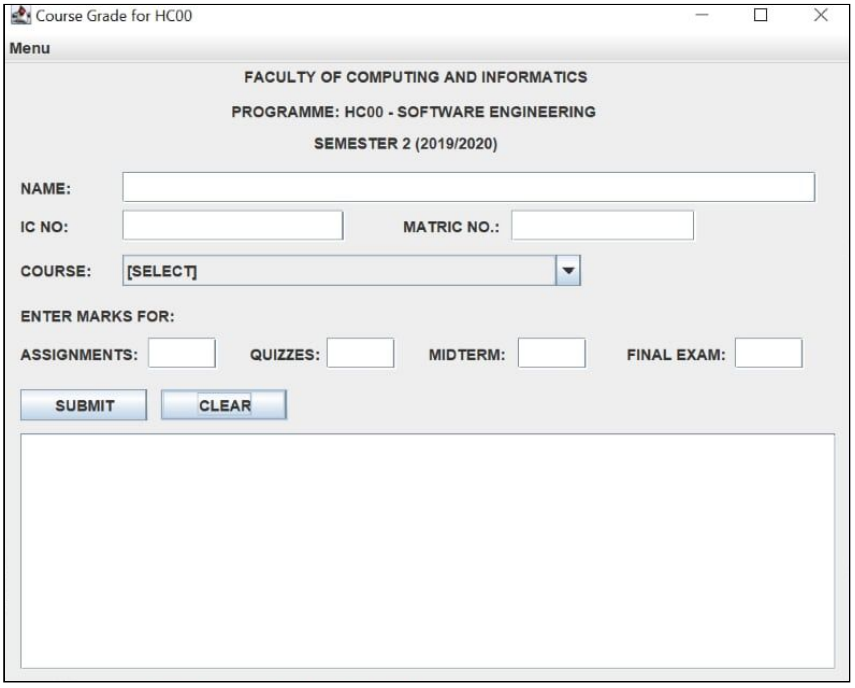

Course Grade for HC00
Menu
FACULTY OF COMPUTING AND INFORMATICS
PROGRAMME: HC00 - SOFTWARE ENGINEERING
SEMESTER 2 (2019/2020)
NAME: ALISA
IC NO: 990619145568
MATRIC NO.: AC1912345
COURSE: KT14203 COMPUTER ARCHITECTURE & ORGANISATION
ENTER MARKS FOR:
ASSIGNMENTS: 89
QUIZZES: 75
MIDTERM:
FINAL EXAM:
SUBMIT
CLEAR
Message
Please enter marks for Midterm. Thank you.
OK


Course Grade for HC00
Menu
FACULTY OF COMPUTING AND INFORMATICS
PROGRAMME: HC00 - SOFTWARE ENGINEERING
SEMESTER 2 (2019/2020)
NAME: ALISA
IC NO: 990619145568
MATRIC NO.: AC1912345
COURSE: KT14203 COMPUTER ARCHITECTURE & ORGANISATION
ENTER MARKS FOR:
ASSIGNMENTS: 89
QUIZZES: 75
MIDTERM: 90
FINAL EXAM:
SUBMIT
CLEAR
Message
Please enter marks for Final Exam. Thank you.
OK

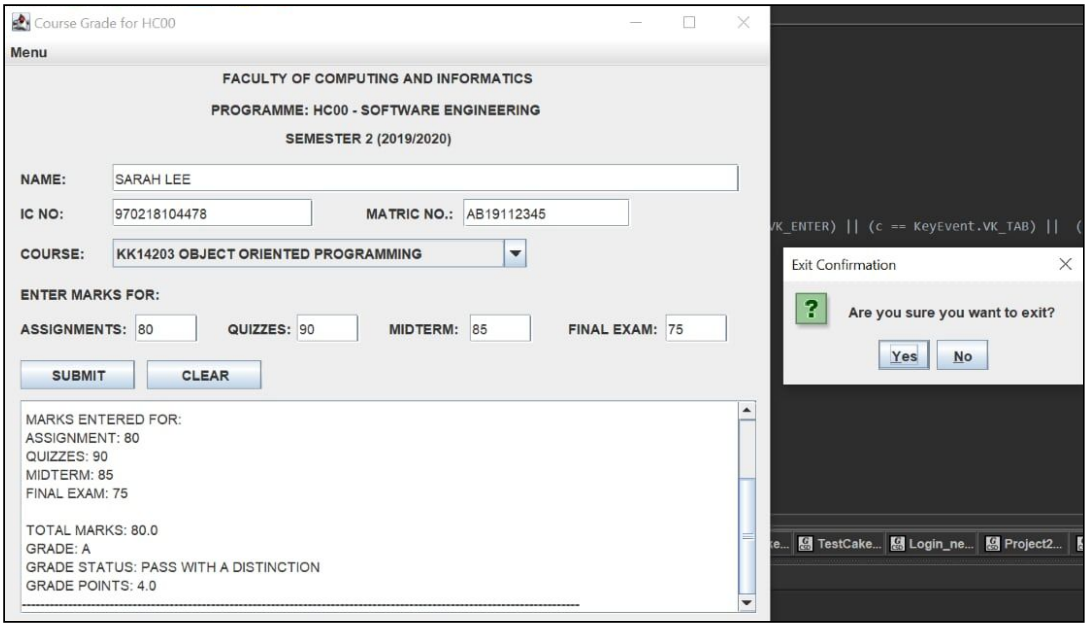
5. Go to Menu > View Data to read from text file.



6. Click the "CLEAR" button to empty text fields, selections and text area.



7. To submit a new input, exit the program and run again. Menu > Exit.



8. To make changes to the data, go to the text file "CourseGradeHC00.txt".