

**SOFTWARE ENGINEERING PROGRAMME (HC03)**

**FACULTY OF COMPUTING AND INFORMATIC**

**KT14203 OBJECT-ORIENTED PROGRAMMING**

**SEMESTER 2, 2019/2020**

**NAME : BLASE LINUS**

**MATRIC NO. : BI19110081**

**PROJECT : 2**

**LECTERUR : SAMRY@MOHD SHAMRIE SAININ**

**SECTION : 1**

1. **JAVA CODE**

Main class :

public class MainProject {  
 public static final int width = 400, height = width / 12 \* 17;  
 public MainProject() {  
 new Window(width, height, "Course Grade Program", this);  
 }  
 public static void main(String[] args) {  
 new MainProject();   
 }  
}

Entry class :

import javax.swing.JOptionPane;  
  
public class Entry {  
 private String matric;  
   
 public Entry() {  
 do{  
 matric = JOptionPane.showInputDialog("Enter your matric number",null);   
   
 if(matric == null){  
 System.exit(0);   
 } else if(matric.length() != 10){  
 JOptionPane.showMessageDialog(null, "Please enter a valid matric number");   
 }  
   
 }while(matric.length() != 10);  
 }  
   
 public String getMatric() {  
 return matric;  
 }  
}

Window class :

import java.awt.BorderLayout;  
import java.awt.Dimension;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JMenu;  
import javax.swing.JMenuBar;  
import javax.swing.JMenuItem;  
import javax.swing.JPanel;  
  
public class Window {  
   
 private String matric;  
 private Entry entry;  
 private Panel panel;  
  
 public Window(int width,int height, String title, MainProject project) {  
 JFrame frm = new JFrame(title);  
   
 frm.setPreferredSize(new Dimension(width, height));  
 frm.setMaximumSize(new Dimension(width, height));  
 frm.setMinimumSize(new Dimension(width, height));  
   
 Header header = new Header("Course Peformance v.0.1b by Blase");  
 panel = new Panel();  
 entry = new Entry();  
   
 System.out.print(matric);  
 frm.add(header, BorderLayout.NORTH);  
 frm.add(panel, BorderLayout.CENTER);  
   
 panel.save.addActionListener(new Button());  
 panel.clear.addActionListener(new Button());  
 panel.total.addActionListener(new Button());  
   
 JMenuBar menuBar = new JMenuBar();   
 JMenu menu = new JMenu("Menu");   
   
 JMenuItem data = new JMenuItem("View Data");  
 data.addActionListener(new MenuActionListener());  
   
 JMenuItem exit = new JMenuItem("Exit");   
 exit.addActionListener((event) -> System.exit(0));  
   
 menu.add(data);  
 menu.add(exit);  
   
 menuBar.add(menu);  
   
 frm.setJMenuBar(menuBar);   
   
 frm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 frm.setResizable(false);  
 frm.setLocationRelativeTo(null);  
 frm.setVisible(true);  
 }  
   
 public String getMatric() {  
 return matric;  
 }  
   
 private class Button implements ActionListener{  
  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource() == panel.save) {  
 panel.setMatric(entry.getMatric());  
 panel.transferMatric();  
 panel.transferMark();  
 panel.printOutput();  
 panel.writeIndividualData();  
 } else if(e.getSource() == panel.clear) {  
 panel.clear();  
 } else if(e.getSource() == panel.total) {  
 panel.writeData();  
 panel.printOverallOutput();  
 }   
   
 }   
 }  
   
 private class MenuActionListener implements ActionListener{  
  
 public void actionPerformed(ActionEvent e) {  
 BufferedReader reader;  
 try {  
   
 reader = new BufferedReader(new FileReader("database.txt"));  
 String line = reader.readLine();  
 String output="<html>";  
 while (line != null) {  
 output += line + "<br>";  
 line = reader.readLine();  
 }  
 output += "<br>";  
 panel.setOutput(output);  
 reader.close();  
   
 } catch (IOException io) {  
 io.printStackTrace();  
 panel.setOutput(io.toString());  
 }   
 }   
 }  
}  
  
class Header extends JPanel{  
   
 private static final long serialVersionUID = 1L;  
  
 public Header(String headerTitle) {  
 JLabel header = new JLabel(headerTitle);  
 add(header);  
 }  
   
}

Panel class :

import java.awt.Color;  
import java.awt.Dimension;  
import java.awt.FlowLayout;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
import javax.swing.BorderFactory;  
import javax.swing.JButton;  
import javax.swing.JComboBox;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
import javax.swing.JScrollPane;  
import javax.swing.JTextField;  
import javax.swing.border.Border;  
  
  
public class Panel extends JPanel {  
   
 private static final long serialVersionUID = 1L;  
   
 private FlowLayout flow;  
 private JLabel lblCourse, lblResult, output;  
 private JComboBox<String> courseCode;  
 private JTextField mark;  
 private JScrollPane scrollPane;  
 private String[] courseList = {"KT14403 Struktur Diskrit", "UW00102 Hubungan Etnik", "UC01502 Makna Dalam Komunikasi"};  
 private int[] creditHour = {4 , 4 , 4};  
 private String matric;  
   
 private ReadWrite RW;  
   
 public JButton save, total, clear;  
   
 public Panel() {  
 flow = new FlowLayout(FlowLayout.LEFT);  
 setLayout(flow);  
   
 lblCourse = new JLabel("Course code : ");  
 lblCourse.setPreferredSize(new Dimension(90, 20));  
 add(lblCourse);  
   
 RW = new ReadWrite(courseList, creditHour);  
   
 courseCode = new JComboBox<String>(RW.getCourseList());  
 courseCode.setPreferredSize(new Dimension(265, 20));  
 add(courseCode);  
 courseCode.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent evt) {  
 RW.setCourse(courseCode.getSelectedItem().toString());  
 }  
 });  
   
   
 lblResult = new JLabel("Carry Mark : ");  
 lblResult.setPreferredSize(new Dimension(150, 20));  
 add(lblResult);  
   
 mark = new JTextField();  
 mark.setPreferredSize(new Dimension(150, 25));  
 mark.setHorizontalAlignment(JTextField.CENTER);  
 add(mark);  
   
 save = new JButton("Save");  
 save.setPreferredSize(new Dimension(80, 25));  
 add(save);  
   
 clear = new JButton("Clear ALL");  
 save.setPreferredSize(new Dimension(80, 25));  
 add(clear);  
   
 total = new JButton("Final Total");  
 total.setPreferredSize(new Dimension(125, 25));  
 add(total);  
   
 Border border = BorderFactory.createLineBorder(Color.BLACK, 2);  
 output = new JLabel();  
 output.setBorder(border);  
 output.setVerticalAlignment(JLabel.TOP);  
 System.out.print(RW.getCourseQuantity());  
   
 scrollPane = new JScrollPane(output);  
 scrollPane.setPreferredSize(new Dimension(375, 375));  
 scrollPane.getViewport().setBackground(Color.getHSBColor(185, 33, 86));  
 add(scrollPane);  
   
   
 }  
   
 public void setMatric(String matric) {  
 this.matric = matric;  
 }  
   
 public void setOutput(String output) {  
 this.output.setText(output);  
 scrollPane.getViewport().revalidate();  
 }  
   
 public void transferMatric() {  
 RW.setMatric(matric);  
 }  
   
 public void transferMark() {  
 double dnum = Double.parseDouble(mark.getText());  
 RW.setMark(dnum);  
 RW.setGrade(dnum);  
 }  
   
 public void printOutput() {  
 output.setText(RW.getOutput());  
 scrollPane.getViewport().revalidate();  
 }  
   
 public void printOverallOutput() {  
 output.setText(RW.getOverallOutput());  
 scrollPane.getViewport().revalidate();  
 }  
  
 public void clear() {  
 RW.clear();  
 output.setText("Please re-Enter all data from the begining");  
 }  
   
 public void writeIndividualData() {  
 RW.writeIndividual();  
 }  
   
 public void writeData() {  
 RW.writeOverall();  
 }  
}

Student class :

import java.math.RoundingMode;  
import java.text.DecimalFormat;  
  
public class Student {  
 private String[] courseCode;  
 private int[] creditHour;  
 private double[] mark = { 0, 0, 0}, coursePointer = {0, 0, 0};  
 private int courseQuantity, totalCreditHr, i=0;  
 private String grade, overallGrade, matric, output, overallOutput, course;  
 private double courseMark, pointer, overallMark, overallPointer;  
   
 private static DecimalFormat df = new DecimalFormat("#.##");  
   
   
 public Student(String [] courseCode, int[] creditHour) {  
 this.courseCode = courseCode;  
 this.creditHour = creditHour;  
   
 for(int i = 0; i < creditHour.length; i++)  
 totalCreditHr += creditHour[i];  
 }  
   
 public void setCourse(String course) {  
 this.course = course;  
 }  
   
 public String[] getCourseList() {  
 return courseCode;  
 }  
   
 public String getCourse() {  
 return courseCode[i-1];  
 }  
   
 public String getAllCourses() {  
 String input="";   
 for(int i = 0; i<courseCode.length; i++)  
 input += courseCode[i] + " ";   
   
 return input;  
 }  
   
 public int getCourseQuantity() {  
 for(int i = 0; i < courseCode.length; i++)  
 courseQuantity++;  
   
 return courseQuantity;  
 }  
   
 public void setMark(double mark) {  
 courseMark = mark;  
 this.mark[i] = courseMark;   
 i++;  
 System.out.println(i);  
 }  
   
 public double getMark() {  
 return courseMark;  
 }  
   
 public void clear() {  
 i = 0;  
 }  
   
 public void setMatric(String matric) {  
 this.matric = matric;  
 System.out.println(matric);  
   
 }  
   
 public String getMatric() {  
 return matric;  
 }  
   
 public void setGrade(double mark) {  
   
 if(mark >= 80) {  
 grade = "A";  
 } else if (mark >= 75 && mark <= 79) {  
 grade = "A-";  
 } else if (mark >= 70 && mark <= 74) {  
 grade = "B+";  
 } else if (mark >= 65 && mark <= 69) {  
 grade = "B";  
 } else if (mark >= 60 && mark <= 64) {  
 grade = "B-";  
 } else if (mark >= 55 && mark <= 59) {  
 grade = "C+";  
 } else if (mark >= 50 && mark <= 54) {  
 grade = "C";  
 } else if (mark >= 45 && mark <= 49) {  
 grade = "C-";  
 } else if (mark >= 40 && mark <= 44) {  
 grade = "D+";  
 } else if (mark >= 35 && mark <= 39) {  
 grade = "D";  
 } else if (mark >= 0 && mark <= 34) {  
 grade = "E";  
 } else {  
 grade = null;  
 }  
 setPointer();   
 }  
   
 public String getGrade() {  
 return grade;  
 }  
   
 public void setPointer() {  
 if(grade == "A")  
 pointer = 4.00;  
 else if(grade == "A-")  
 pointer = 3.67;  
 else if(grade == "B+")  
 pointer = 3.33;  
 else if(grade == "B")  
 pointer = 3.00;  
 else if(grade == "B-")  
 pointer = 2.67;  
 else if(grade == "C+")  
 pointer = 2.33;  
 else if(grade == "C")  
 pointer = 2.00;  
 else if(grade == "C-")  
 pointer = 1.67;  
 else if(grade == "D+")  
 pointer = 1.33;  
 else if(grade == "D")  
 pointer = 1.00;  
 else if(grade == "E")  
 pointer = 0.00;  
 }  
   
 public double getPointer() {  
 coursePointer[i-1] = pointer;  
 return pointer;  
 }  
   
 public void calculateOverallMark() {  
 overallMark = 0;  
 for(int i = 0; i < mark.length; i++) {  
 overallMark += (mark[i] \* creditHour[i]);  
 }  
 overallMark /= totalCreditHr;  
 }  
   
 public double getOverallMark() {  
 calculateOverallMark();  
 return overallMark;  
 }  
   
 public String getOverallGrade() {  
 if(overallPointer >= 4.00) {  
 overallGrade = "A";  
 } else if (overallPointer >= 3.67 && overallPointer < 4.00) {  
 overallGrade = "A-";  
 } else if (overallPointer >= 3.33 && overallPointer < 3.67) {  
 overallGrade = "B+";  
 } else if (overallPointer >= 3.00 && overallPointer < 3.33) {  
 overallGrade = "B";  
 } else if (overallPointer >= 2.67 && overallPointer < 3.00) {  
 overallGrade = "B-";  
 } else if (overallPointer >= 2.33 && overallPointer < 2.67) {  
 overallGrade = "C+";  
 } else if (overallPointer >= 2.00 && overallPointer < 2.33) {  
 overallGrade = "C";  
 } else if (overallPointer >= 1.67 && overallPointer < 2.00) {  
 overallGrade = "C-";  
 } else if (overallPointer >= 1.33 && overallPointer < 1.67) {  
 overallGrade = "D+";  
 } else if (overallPointer >= 1.00 && overallPointer < 1.33) {  
 overallGrade = "D";  
 } else if (overallPointer >= 0 && overallPointer < 1.00) {  
 overallGrade = "E";  
 } else {  
 overallGrade = null;  
 }  
   
 return overallGrade;  
 };  
   
 public double getOverallPointer() {  
 overallPointer = 0;  
 for(int i = 0; i < coursePointer.length; i++) {  
 overallPointer += coursePointer[i];  
 }  
 overallPointer /= coursePointer.length;  
   
 return overallPointer;  
 }  
   
 public String getOutput() {  
 output = "<html>";  
 output += "Entered Info <br><br>";  
 output += "Matric No. : " + matric + "<br>";  
 output += "Course : " + course + "<br>";  
 output += "Carry Mark : " + courseMark + "<br>";  
 output += "Grade : " + grade + "<br>";  
 output += "Pointer : " + pointer + "<br>";  
 output += "</htm>";  
   
 return output;  
 }  
   
 public String getOverallOutput() {  
 df.setRoundingMode(RoundingMode.UP);  
   
 overallOutput = "<html>";  
 overallOutput += "OVERALL <br><br>";  
 overallOutput += "Matric No. : " + matric + "<br>";  
 overallOutput += "Course : <br>";  
 for(int i=0; i<courseCode.length; i++)  
 overallOutput += courseCode[i] + "<br>";  
 overallOutput +="Overall Mark : " + df.format(overallMark) + "<br>";  
 overallOutput += "Grade : " + overallGrade + "<br>";  
 overallOutput += "Pointer : " + df.format(overallPointer) + "<br>";  
 overallOutput += "</htm>";  
   
 return overallOutput;  
 }  
}

ReadWrite class :

import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.math.RoundingMode;  
import java.text.DecimalFormat;  
  
public class ReadWrite extends Student{  
   
 private String filePath1 = "courses.txt";  
 private String filePath2 = "database.txt";  
 private static DecimalFormat df = new DecimalFormat("#.##");  
  
 public ReadWrite(String[] courseCode, int[] creditHour) {  
 super(courseCode, creditHour);  
 }  
   
 public void writeIndividual() {  
 File file = new File(filePath1);  
 FileWriter fw = null;  
 BufferedWriter bw = null;  
 PrintWriter pw = null;  
   
 String input = getMatric() + ", " + getCourse() + ", " + getMark() + ", " + getGrade() + ", " + getPointer() + "\n";  
 System.out.println(input);  
   
 try {  
 fw = new FileWriter (file, true);  
 bw = new BufferedWriter(fw);  
 pw = new PrintWriter(bw);  
   
 pw.println(input);  
   
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 pw.close();  
 bw.close();  
 fw.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }   
 }   
 }  
   
 public void writeOverall() {  
 File file = new File(filePath2);  
 FileWriter fw = null;  
 BufferedWriter bw = null;  
 PrintWriter pw = null;  
 df.setRoundingMode(RoundingMode.UP);  
   
 String input = getMatric() + ", " + getAllCourses() +"," + df.format(getOverallMark()) + ", " + df.format(getOverallPointer()) + ", " + getOverallGrade() + "\n";  
 System.out.println(input);  
   
 try {  
 fw = new FileWriter (file, true);  
 bw = new BufferedWriter(fw);  
 pw = new PrintWriter(bw);  
   
 pw.println(input);  
   
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 pw.close();  
 bw.close();  
 fw.close();  
 } catch (IOException e) {  
 e.printStackTrace();   
 }   
 }   
 }   
}

1. **Object Oriented Concept Implementation**
   1. Object and Classes

This is one of the basic concepts of object-oriented programming revolve around real life entities. An object consists of state, behavior and identity which are represented by attributes, methods, and unique name of an object. Classes are the blueprints or prototype from which objects are created. This concept is present in the making of the program as the are 6 classes that has their specific usage which some consists of modifiers and constructors. Such example are the Student, window, and panel classes.

* 1. Encapsulation

Encapsulation is a mechanism where you bind data and code together as a single unit such as the classes in this project. In other words, to hide data in order to make it safe from any modification. Encapsulation is achieved in this project by declaring the variables of a class as private. Public getter and setter methods are also provided in this project to modify or view the variables values.

* 1. Inheritance

Inheritance is the concept where the properties of one class is inherited by another class. The code is reusable between these classes where a relationship is established between different classes. The implementation of this concept in this project is shown by the ReadWrite class inheriting the class Student. This helps shortens the class and increase the reusability of methods in student class. Therefore, by calling upon ReadWrite class method of student can also be used without declaring Student class. This is known as a single inheritance.

* 1. Interface

Interface in Java is a blueprint of a class. Interface helps to achieve multiple inheritance in java. Therefore, an interface is basically a group of related methods with empty bodies. An interface is implemented in the project to allows a class to become more formal about the behaviour it promises to provide.

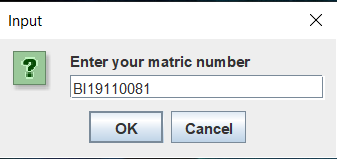
* 1. Inner Class

Inner class or nested class is a class which is declared inside the class or interface. This concept is used to group classes and interfaces in one place so that it can be more readable or maintainable. Such instances can be seen in the main class and several classes of the project. Inner classes can also access all the members of the outer classes including private data members and methods.

1. **Read and write implementation**

Read and write is the concept where data is either write into a file or read from the file and processed as input. In the case of this project write concept is use through buttons that are implemented with action listener. Whenever the save and total is pressed by the user. Information such as the matric number, marks, grades and so on are written or saved into files. This file then can be read through the “view data” in the menu tab above. This is the read implementation of this project that will read the data of the file and output at scroll pane.

1. **User Manual**



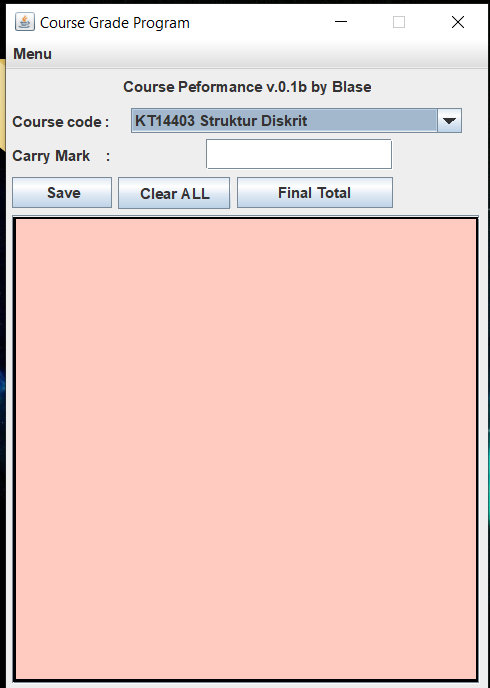
**3**

**1**

**2**

Figure 1: Starting window

1. Enter your matric number in the Input Area (1) so that it will be recorded. Only valid matric numbers can proceed.
2. Click OK button (2) to proceed or Cancel button to exit from program (3).



**5**

**4**

**7**

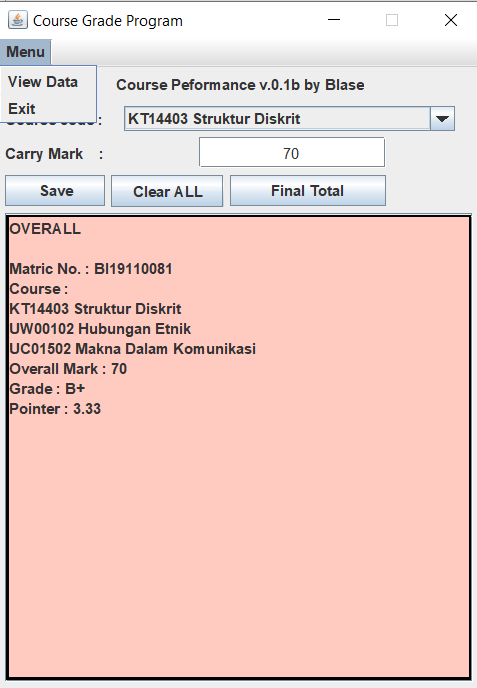
**6**

**8**

**8**

Figure 2: Main GUI

1. Click on combo box (4) to select your course.(compulsory to click in the beginning)
2. Enter your carry marks in the space (5).
3. Click on save button (6) to output your marks on the field below(9).
4. Any error can be cleared by clicking on Clear ALL button (7).
5. When Clear ALL button is selected all data is required to key in again.
6. Final Total (8) is click to show overall performance after all marks have been allocated to each course.
7. WARNING: Marks must be saved according to the order given in combo box (7).



**11**

**10**

**9**

Figure 3: Menu Tab

1. Click on menu (9) to bring down menu tab.
2. Click on “View Data” (10) to view all overall performance that has been recorded.
3. Click on Exit (11) to exit the program.