Practical Skills Assessment #2
Java Source Code

Author

Mr. Adam Torok - B00798824

Mr. Mateusz Tynkiewicz - B00798825

Jordanstown, April 2021

Uni.java

```java
package group100;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 * @author group100
 * This is the main class of the application, instantiate itself
 * call the filehandler, if file exists, load them, if not, create them
 * //Mr. Adam Torok - B00798824 Mr. Mateusz Tynkiewicz - B00798825
 * displays the menu */
public class Uni {
    private String courseFileName = "CourseDetails.txt";
    private FileHandler courseHandler;
    private Course course;
    Scanner Scan;
    Uni() throws IOException{
        this.Scan = new Scanner(System.in);
        try {
            courseHandler = new FileHandler(courseFileName);
            course = courseHandler.loadCourseFromFile();
        } catch (IOException ex) {
            Logger.getLogger(Uni.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public void displayMenu() throws IOException{//Displays a menu
        System.out.println("Application Started");
        String menu = "1. Enrolled Students\n"
            + "2. List Courses Details\n"
            + "3. Add Student\n"
            // + "4. Add Course\n"
            + "4. Search Student\n"
            + "5. Delete a Student\n"
            + "6.Exit\n"
            + "----------------------";

        while(true){ //inf loop
            System.out.println(menu);//display menu
            String choice = Scan.nextLine(); //ask user choice
            switch(choice){ //switch it
                case "1":                                       // list student
                    course.displayEnrolledStudents();
                    break;
                case "2":                                       // list student
                    course.prettifyCourse();
                    break;
                case "3":                                   // create a new student
                    if (course.isMaxStudentEnrolled() == false){
                        //only enter interactive mode if student can enroll
                        Student tmpStudent = new Student();
                        if (tmpStudent.valid == true) {
                            course.interactiveEnrollStudent(tmpStudent);
                        }
                    }
                    break;
                    // case "10":                               // create a new course
                    //      course = new Course();
                    //     if (course.valid == true) {
                    //         courseHandler.saveCourse(course);
                    //     }
                    //     break;
                case "4":                                       //search a student
                    course.searchStudent();
                    break;
                case "5":                                     //Delete a student
                    course.listStudentForDelete();
```

```
69                      break;
70                  case "6":
71                      System.out.println("Bye bye");
72                      System.exit(0);
73                  default:
74                      System.out.println("This is not a choice");
75                      break;
76              }
77          }
78      }
79
80      public static void main (String[] args) throws IOException{
81          //Program starts here
82          Uni app = new Uni();     // instantiation Uni class
83          app.displayMenu();       //throw the menu
84      }
85  }
```

Course.java

```java
package group100;
import java.io.IOException;
import java.util.ArrayList;
import java.util.*;
//describe a course, containing students in a list
//Mr. Adam Torok - B00798824 Mr. Mateusz Tynkiewicz - B00798825
public class Course {
    private String name,lecturer;
    private int totalStudents = 0;
    private int maleCounter = 0;
    private int femaleCounter = 0;
    private float malePercent = 0f;
    private float femalePercent = 0f;
    final int MAXSTUDENTS = 20;
    protected boolean valid;
    private ArrayList<Student> students = new ArrayList<>();
    private FileHandler studentHandler;
    final String studentFileName = "StudentDetails.txt";
    Scanner Scan = new Scanner(System.in);
    //Standrad constructor. So we can do this:
    //new Course('Drink Chiller','Lecturer Bill')
    public Course(String name, String lecturer) throws IOException{
        studentHandler = new FileHandler(studentFileName);
        this.name = name;
        this.lecturer = lecturer;
        validateCourse();
        loadStudents();
    }

    //Constructor Overload, if no arguments were supplied,
    //we step into 'interactive mode', so we can do this: new Srudent();
    public Course() throws IOException{
        System.out.print("Course Add. Interactive mode (press 'x' to exit)\n");
        addCourse();
        loadStudents();
    }

    private void loadStudents() throws IOException{
        //get the stored students from the filehandler
        //every student passed to enrollStudent (data integrity)
        ArrayList<Student> studentFromFile = studentHandler.loadStudentsFromFile();
        Iterator i = studentFromFile.iterator();
        while(i.hasNext()){
            enrollStudent((Student) i.next());
        }
    }

    public void saveStudents() throws IOException{
        //pass all the students<> to the student handler and save2
        FileHandler studentHandler = new FileHandler("StudentDetails.txt");
        studentHandler.saveStudents(students);
    }

    public String getName(){
        return name; //returns a course name
    }

    public String getLecturer(){
        return lecturer; //returns a lecturer
    }

    public ArrayList<Student> getEnrolledStudents(){
        return students; //get all students who enrolled
    }

    public void interactiveEnrollStudent(Student student) throws IOException{
        //enroll a student and save the new student list
        enrollStudent(student);
```

```
69          saveStudents();
70      }
71
72      public boolean isMaxStudentEnrolled(){
73          //this is a pre-check to avoid on those cases where the course is full
74          //and we want to avoid straight enetring into interaction mode
75          if (students.size() >= 20) {
76              System.out.println("-------------------------------");
77              System.out.println("This course is full ("+ MAXSTUDENTS+" students)");
78              System.out.println("-------------------------------");
79              return true;
80          }else{
81              return false;
82          }
83      }
84
85      private void enrollStudent(Student student) throws IOException{
86      //enroll a student
87      //automatic pre-populate calls this function
88      //interactive enroll calls this function
89          if (totalStudents < MAXSTUDENTS) {              //can we store more?
90              students.add(student);                      //add the student
91              totalStudents++;                            //increase the courseCount
92              if ("male".equals(student.getGender())) {
93                  this.maleCounter++;                     //register the gender
94              }else{
95                  this.femaleCounter++;
96              }
97              calculatePercent();                         //update the percentage
98          }else{//max number of students reached
99              System.out.println("Maximum number of students reached");
100         }
101     }
102
103     private void calculatePercent(){
104         //calculate the male percent or dis
105         if (this.femaleCounter > 0 && this.maleCounter > 0) {
106                 //avoid div0, cast to float, calulate the gender%
107                 this.malePercent = 100 * (float) this.maleCounter / this.totalStudents;
108                 this.femalePercent = 100 - this.malePercent;
109         }else if(this.maleCounter > 0 && this.femaleCounter == 0){
110             //course is pure males
111             this.malePercent = 100f;
112         }else if(this.femaleCounter > 0 && this.maleCounter == 0){
113             //course is pure males
114             this.malePercent = 0f;
115             this.femalePercent = 100f;
116         }else{
117             //pure females or division 0
118                 this.malePercent = 0f;          //if one of the oprands 0, keep 0
119             }
120     }
121
122     private boolean addCourse(){
123         //call the necessary methods, interactive mode
124         //if the file is empty
125         this.name = askQuestion("Course name: ");
126         this.lecturer = askQuestion("Lecturer: ");
127         return validateCourse();
128     }
129
130     private String askQuestion(String question){
131         //asks the question an returns a String
132         String tmpInput = "";
133         while(tmpInput.length() == 0){
134             System.out.print(question);
135             tmpInput = Scan.nextLine();
136             exitOnX(tmpInput); //if user send x, exit
137         }
138         return tmpInput;
```

```
    }

    private void exitOnX(String input){
        //if the user sends an x it will terminate the program
        if ("x".equals(input)) {
            System.out.print("Program has been terminated by user\n");
            System.exit(0);
        }
    }

    private boolean validateCourse(){
        //true if it is a valid course (all field filled)
        if(this.name.length() > 0 && this.lecturer.length() > 0){
            this.valid = true;
            return true;
        }else{
            System.out.print("Invalid Course data");
            this.valid = false;
            return false; //if this is not a valid course, return false
        }
    }

    @Override
    public String toString(){
        //represents a student (used while writing a txt file)
        return ""+this.name+","+this.lecturer+"\n";
    }

    public void displayEnrolledStudents(){
        //display students in the main menu
        System.out.println("Enrolled Students: ");
        students.forEach((singleStudent) -> {
            System.out.println(singleStudent.prettifyStudent());
        });
    }

    public String prettifyCourse(){
        //displays a pretty version of course details
        System.out.println();
        String output = "Course Name:\t\t" +this.name+"\n"
                + "Lecturer:\t\t"+this.lecturer+"\n"
                + "Number of Students:\t"+this.totalStudents+"\n"
                + "Males:\t\t\t" + this.maleCounter +"\n"
                + "Females:\t\t" + this.femaleCounter +"\n"
                + "Male%:\t\t\t" + Math.round(this.malePercent) +"%\n"
                + "Female%:\t\t" + Math.round(this.femalePercent) +"%\n";
        System.out.print(output);
        System.out.println();
        return output;
    }

    public void searchStudent(){
        //search a student
        if (!this.students.isEmpty()) {
            System.out.print("Enter a student name: ");
            String query = Scan.nextLine();
            searchStudent(query);
        }else{
            System.out.println("There is no student in this course");
        }
    }

    public void searchStudent(String query){
        //search a student and returns with his/her name
        boolean isFound = false;
        for (Student student : students){
            if (student.getName().contains(query)) {
                System.out.println("-------------");
                System.out.println("Student found");
                System.out.println("-------------");
```

```
209                 System.out.println(student.prettifyStudent());
210                 System.out.println("-----------------");
211                 isFound = true;
212                 break;
213             }
214         }
215         if (isFound == false) {
216             System.out.println("-----------------");
217             System.out.println("Student not found");
218             System.out.println("-----------------");
219         }
220     }
221 
222     public void listStudentForDelete() throws IOException{
223         //deletes a student
224         if (students.isEmpty()) { // if students empty
225             System.out.println("");
226             System.out.println("Student File is empty, "
227                                + "add students to the course first");
228             System.out.println("Falling back to main menu");
229             System.out.println("");
230             return;
231         }
232        int i = 0;
233        System.out.println("(\"x\") to quit");
234 
235        //render the output: id name
236        for (Student student : students){
237             System.out.println("Id: "+ i + "\t"+student.getName());
238             i++;
239         }
240 
241        //waiting for an id to initiate deletion or X to quit
242        while(true){
243            //ask user input
244             System.out.print("Enter an id to delete a student:  ");
245             String userChoice = Scan.nextLine();
246             if (userChoice.equals("x")) { //quit
247                System.out.println("-------------------------");
248                System.out.println("Falling back to The Main Menu");
249                System.out.println("-------------------------");
250                break;
251             }
252               try{
253                   //make an int from the input
254                   int userInt = Integer.parseInt(userChoice);
255                   deleteStudentAtIndex(userInt);
256                   return;
257               }
258               catch(IndexOutOfBoundsException e){ //input > student.size()-1
259                   System.out.println("Wrong Id");
260               }
261               catch(NumberFormatException e){ //if its cannot be casted to int
262                   //fall back to the cycle, ask the input again
263                   System.out.println("Input Must Be String");
264               }
265        }
266     }
267 
268     public boolean deleteStudentAtIndex(int index) throws IOException, IndexOutOfBoundsException{
269         //actually delete the student from the course by the given index
270         // listStudentForDelete() check the student existance therefore, no needed
271         //access the student for his/her name
272          Student student = this.students.get(index);
273 
274          System.out.println("-----------------------------------------------");
275          System.out.println("Student removed from the course: " + student.getName());
276          System.out.println("-----------------------------------------------");
277          students.remove(index); //remove the student
278          this.totalStudents--;   //one student less
```

```
279
280         //decrease the gendercounter as well
281         if ("male".equals(student.getGender())) {
282             this.maleCounter--;
283         }else{
284              this.femaleCounter--;
285          }
286
287         //update the percentage
288         calculatePercent();
289         //save
290         saveStudents();
291         return true;
292     }
293     public int getNumOfStudents(){
294         return students.size();
295     }
296 }
```

FileHandler.java

```java
package group100;
import java.io.*;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * @author group100
 * Handles the file reading and writing
 * Checks file existance, Create New files if needed
 * Load a course call loadCourseFromFile returns Course
 * Load students call loadStudentsFromFile returns ArrayList<Student>
 * Save a course call saveCourse(Course);
 * Save students call loadStudentsFromFile(ArrayList<Student>)
 * //Mr. Adam Torok - B00798824 Mr. Mateusz Tynkiewicz - B00798825
 */

public class FileHandler {
    //Some class variables
    private final File f;
    private final String fileName;
    private boolean exists;

    //sets up the class variables and try to read a file if exists
    public FileHandler(String fileName) throws IOException{
        this.f = new File(fileName);
        this.fileName = fileName;
        this.exists = this.f.isFile();
        if (this.exists == false) { // ... or create the file
            createFile();
        }
    }

    //creates the file of not exists
    private void createFile() throws IOException{
        if (this.exists == false) {
            f.createNewFile();
            System.out.println("Creating New file: " +this.fileName);
            this.exists = true;
        }
    }

    //saves the file this.fileName by iterating the given ArrayList
    public int saveStudents(ArrayList fileData) throws IOException{
        //deleting the existing contents and create a new file since
        //all data come through the argument
        this.f.delete();
        this.f.createNewFile();

        //setting up the writer
        PrintWriter writer = new PrintWriter(new FileWriter("StudentDetails.txt"));

        //iterating the ArrayList and write them to a file
        fileData.forEach((line) -> {
            writer.write(line.toString());
        });
        writer.close();
        return 0;
    }

    //loads a file
    public ArrayList<Student> loadStudentsFromFile() throws FileNotFoundException, IOException{
        BufferedReader in = new BufferedReader(new FileReader("StudentDetails.txt"));
        //Local List for students
        ArrayList<Student> studentList = new ArrayList<>();

            //iterating through the line
            for (String line = in.readLine(); line != null; line = in.readLine()) {
```

```
69
70                  if (line.length() > 0) {
71                      String studentData[] = line.split(","); //explode it!
72                   //make new students
73                   studentList.add(new Student(studentData[0],
74                                               studentData[1],
75                                               studentData[2],
76                                               studentData[3]));
77                  }
78              }
79          in.close();
80          return studentList;
81      }
82
83      //loads a course from file
84      public Course loadCourseFromFile() throws FileNotFoundException, IOException{
85              Course course; //initialize just for safety
86              BufferedReader in = new BufferedReader(new FileReader("CourseDetails.txt"));
87              String line = in.readLine();
88              try{
89                   if(line.length() > 0){
90                  String[] details = line.split(",");
91                  course = new Course(details[0],details[1]);
92
93              }
94              else{ //create a new file and adds a new course
95                  f.delete();
96                  createFile();
97                  System.out.println("No courses exists..");
98                  course = new Course();
99                  saveCourse(course);
100             }
101             }catch(java.lang.NullPointerException e){
102                 //create a new file and adds a new course
103                 f.delete();
104                 createFile();
105                 System.out.println("No courses exists..");
106                 course = new Course();
107                 saveCourse(course);
108             }
109             in.close();
110         return course;
111     }
112
113     public void saveCourse(Course courseDetails)
114                 throws FileNotFoundException, IOException{
115         //saves a course name, lecturer
116             PrintWriter writer = new PrintWriter(new FileWriter("CourseDetails.txt"));
117             String tmpString = courseDetails.getName() + "," +
118                                courseDetails.getLecturer()+"\n";
119             writer.write(tmpString);
120             writer.close();
121     }
122 }
```

Student.java

```
package group100;

/**
 *
 * //Mr. Adam Torok - B00798824 Mr. Mateusz Tynkiewicz - B00798825
 * Describe a single student and its features
 */
import java.util.*;

public class Student {
    private String name,gender, address, dob;
    protected boolean valid;
    static int numOfStudent = 0;
    static int male = 0;        //class var to keep track all males
    static int female = 0;      //class var to keep track all males
    Scanner Scan = new Scanner(System.in);

    //Standrad constructor. So we can do this:
    //new Student('John Doe','Male','Jordanstown',01/01/1991')
    public Student(String name, String gender, String address, String dob){
        //all variables are String, valiadates only on empty string
        this.name = name;
        this.gender = gender;
        this.address = address;
        this.dob = dob;
        isValidStudent();
    }

    //Constructor Overload, if no arguments were supplied, we step into
    //'interactive mode', so we can do this: new Srudent();
    public Student(){
        System.out.print("Stundent Add. Interactive mode (press 'x' to exit)\n");
        addStudent();
    }

    Student(Object next) {
        throw new UnsupportedOperationException("Not supported yet.");
        //To change body of generated methods, choose Tools | Templates.
    }

    private boolean addStudent(){//call the necessary methods, interactive mode
        this.name = askQuestion("Student name: ", "name");
        this.gender = askQuestion("Student gender (type \"male\" or \"female\")"
                + ": ", "gender");
        this.address = askQuestion("Student Address: ", "address");
        this.dob = askQuestion("Student DOB: ", "dob");
        return isValidStudent();
    }

    private String askQuestion(String question, String field){
    //asks a question, with
        String tmpInput = "";
        while(tmpInput.length() == 0){
            System.out.print(question);
            tmpInput = Scan.nextLine();
            if (field.equals("gender")){
                if (tmpInput.equals("male") || tmpInput.equals("female")) {
                    switch(tmpInput){
                            case "male":
                                male++;
                                break;
                            case "female":
                                female++;
                                break;
                    }
                }else{
                    System.out.println("Invalid gender. (Type \"male\" or"
                            + " \"female\")");
```

```
                    tmpInput = "";
                }
            }
            exitOnX(tmpInput); //if user send x, exit
        }
        return tmpInput;
    }

    private void exitOnX(String input){
        //if the user sends an x it will terminate the program
        if ("x".equals(input)) {
            System.out.print("Program has been terminated by user\n");
            System.exit(0);
        }
    }

    private boolean isValidStudent(){
        //true if it is a valid student
        //(all field filled)
        if(this.name.length() > 0 &&
           this.gender.length() > 0 &&
           this.address.length() > 0 &&
           this.dob.length() > 0 && canStoreStudent() == true){
                this.valid = true;
                numOfStudent++;
                return true;
        }else{
            System.out.print("Invalid Data or the number of maximum "
                    + "student is reached. No student added...");
            this.valid = false;
            return false; //if this is not a valid student, return false
        }
    }

    @Override
    public String toString(){
        //represents a student (used while writing a txt file)
        return ""+this.name+","+this.gender+","+this.address+","+this.dob+"\n";
    }

    public String prettifyStudent(){ //displays a pretty versin of students
        String output = "Name:\t\t" +this.name
                        +"\nGender:\t\t"+this.gender
                        +"\nAddress:\t"+this.address
                        +"\nDate:\t\t"+this.dob+"\n";
        return output;
    }

    public static int getNumOfStudent(){
        //returns the number of Students
        return numOfStudent;
    }

    private static boolean canStoreStudent(){
        //maximum of 20 student can enrolled
        return numOfStudent < 20;
    }

    public String getName(){
        //returns with a student name (mostly for displays)
        return this.name;
    }

    public void rename(String newName){
        //rename the student
        this.name = newName;
    }

    public String getGender(){
        //get the gender of the student
```

```
139         return this.gender;
140     }
141 }
```

TestCreateStudentDetails.java

```java
/*
 * This file just create the 19 Dummy Student
 */
import group100.*;
import java.io.IOException;
import java.util.ArrayList;
import org.junit.Test;
import static org.junit.Assert.*;
/**
 *
 * @author Adam
 */
public class TestCreateStudentDetails {

    private FileHandler studentHandler;
    final String studentFileName = "StudentDetails.txt";
    private ArrayList<Student> student = new ArrayList<Student>();

    @Test
    public void create19Student() throws IOException{
        /* It will create the student file and fills up with 19 dummy */
        studentHandler = new FileHandler(studentFileName);
        student.add(new Student("Adam","male","Newry","25/10/2000"));
        student.add(new Student("Troy Munoz","male","New York","25/10/2004"));
        student.add(new Student("Israel Mcghee","male","New York","5/7/1998"));
        student.add(new Student("Eduard Sellers","male","New York","10/4/1994"));
        student.add(new Student("Osman Thorne","male","New York","5/10/2006"));
        student.add(new Student("Rico Edmonds","male","New York","11/1/1990"));
        student.add(new Student("Vicky Blaese","female","New York","25/10/2008"));
        student.add(new Student("Gruffydd Dixon","male","New York","25/10/2000"));
        student.add(new Student("Geraldine Powell","female","New York","2/10/1970"));
        student.add(new Student("Shelby Caldwell","male","New York","2/1/1960"));
        student.add(new Student("Gordon Key","male","New York","6/8/2000"));
        student.add(new Student("Ajwa Shaw","female","New York","25/11/1998"));
        student.add(new Student("Bilal Connor","male","New York","25/12/1999"));
        student.add(new Student("Jasleen Mccann","female","New York","25/3/1992"));
        student.add(new Student("Jamal Prosser","male","New York","1/1/1994"));
        student.add(new Student("Giorgia Southern","female","New York","5/10/1998"));
        student.add(new Student("Umer Guest","male","New York","5/1/2007"));
        student.add(new Student("Franklin Casey","male","New York","25/6/2002"));
        student.add(new Student("Jena Nicholson","female","New York","11/09/2009"));
        student.add(new Student("20th Student","female","New York","11/09/2009"));
        studentHandler.saveStudents(student);

        student.forEach((stud) -> {
            System.out.println(stud.getName());
        });

        assertEquals(0,studentHandler.saveStudents(student));
    }
}
```

StudentTest.java

```
package group100;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

//Mr. Adam Torok - B00798824 Mr. Mateusz Tynkiewicz - B00798825
public class StudentTest {

    public StudentTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of toString method, of class Student.
     */
    @Test
    public void testToString() {
        System.out.println("toString");
        Student instance = new Student("Adam","male","Newry","25/10/2000");
        String expResult = "Adam,male,Newry,25/10/2000\n";
        String result = instance.toString();
        assertEquals(expResult, result);
        instance = null;

    }

    /**
     * Test of prettifyStudent method, of class Student.
     */
    @Test
    public void testPrettifyStudent() {
        System.out.println("prettifyStudent");
        Student instance = new Student("Adam","male","Newry","25/10/2000");
        String expResult = "Name:       Adam\n" +
                           "Gender:         male\n" +
                           "Address:    Newry\n" +
                           "Date:       25/10/2000\n";
        String result = instance.prettifyStudent();
        assertEquals(expResult, result);
        instance = null;

    }

    /**
     * Test of getNumOfStudent method, of class Student.
     */
    @Test
    public void testGetNumOfStudent() {
        System.out.println("getNumOfStudent");
        Student instance = new Student("Adam","male","Newry","25/10/2000");
```

```
69          int expResult = 3;
70          int result = Student.getNumOfStudent();
71          assertEquals(expResult, result);
72          instance = null;
73
74      }
75
76      /**
77       * Test of getName method, of class Student.
78       */
79      @Test
80      public void testGetName() {
81          System.out.println("getName");
82          Student instance = new Student("Adam","male","Newry","25/10/2000");
83          String expResult = "Adam";
84          String result = instance.getName();
85          assertEquals(expResult, result);
86          instance = null;
87      }
88
89      /**
90       * Test of getGender method, of class Student.
91       */
92      @Test
93      public void testGetGender() {
94          System.out.println("getGender");
95          Student instance = new Student("Adam","male","Newry","25/10/2000");
96          String expResult = "male";
97          String result = instance.getGender();
98          assertEquals(expResult, result);
99          instance = null;
100     }
101
102      /**
103      * Test of render method, of class Student.
104      */
105
106     @Test
107     public void rename() {
108         System.out.println("rename");
109         Student instance = new Student("Adam","male","Newry","25/10/2000");
110         String expResult = "New Name";
111         instance.rename("New Name");
112         String result = instance.getName();
113         assertEquals(expResult, result);
114         instance = null;
115     }
116
117 }
```

CourseTest.java

```java
//Mr. Adam Torok - B00798824 Mr. Mateusz Tynkiewicz - B00798825
package group100;

import java.io.IOException;
import java.util.ArrayList;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

public class CourseTest {

    public CourseTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }




    /**
     * Test of getName method, of class Course.
     */
    @Test
    public void testGetName() throws IOException {
        System.out.println("getName");
        Course instance = new Course("Course name","Lecturer");
        String expResult = "Course name";
        String result = instance.getName();
        assertEquals(expResult, result);
        instance = null;
    }

    /**
     * Test of getLecturer method, of class Course.
     */
    @Test
    public void testGetLecturer() throws IOException {
        System.out.println("getLecturer");
        Course instance = new Course("Course name","Lecturer");
        String expResult = "Lecturer";
        String result = instance.getLecturer();
        assertEquals(expResult, result);
        instance = null;

    }

    /**
     * Test of isMaxStudentEnrolled method, of class Course.
     * @throws java.io.IOException
     */
    @Test
    public void testIsMaxStudentEnrolled() throws IOException {
```

```
69          System.out.println("isMaxStudentEnrolled");
70          Course instance = new Course("Course name","Lecturer");
71          boolean expResult = false;
72          boolean result = instance.isMaxStudentEnrolled();
73          assertEquals(expResult, result);
74          instance = null;
75      }
76
77      /**
78       * Test of toString method, of class Course.
79       */
80      @Test
81      public void testToString() throws IOException {
82          System.out.println("toString");
83          Course instance = new Course("Course name","Lecturer");
84          String expResult = "Course name,Lecturer\n";
85          String result = instance.toString();
86          assertEquals(expResult, result);
87          instance = null;
88      }
89
90  }
```

Test Cases

#	Description	Expected	Actual	Pass
1	Student representation in file	"Adam,male,Newry,25/10/2000"	"Adam,male,Newry,25/10/2000"	Yes
2	Display Student Details	multiline	multiline	Yes
3	testGetNumOfStudent	3	3	Yes
4	testGetName	Adam	Adam	Yes
5	testGetGender	male	male	Yes
6	renameStudent	New Name	New Name	Yes
7	getName (course)	Course name	Course name	Yes
8	getLecturer	Lecturer	Lecturer	Yes
9	testGetName (Course)	Course name	Course name	Yes
10	testGetLecturer	Lecturer	Lecturer	Yes
11	testIsMaxStudentEnrolled	false	false	Yes
12	testToString	multiline	multiline	Yes

Implementation

Requirement	Implemented	Filename	LineNo.
Prepopulate the application with any previously stored data	Yes	FileHandler.java 3	62
Report the details of the course	Yes	Course.java 2	175
Ability to add new student to course	Yes	Uni.java 1	52
Ability to delete a student from the course	Yes	Course.java 2	267
Ability to search student by name and display the details	Yes	Course.java 2	208
Changes all stored to files	Yes	Course.java 2	48
Allow max up to 20 students to be enrolled to the course	Yes	Course.java 2	72

Test Plan

Tests involved manual testing (testing functionality) and a with some test classes (listed in this document).

This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version 6.3.2LaTeX

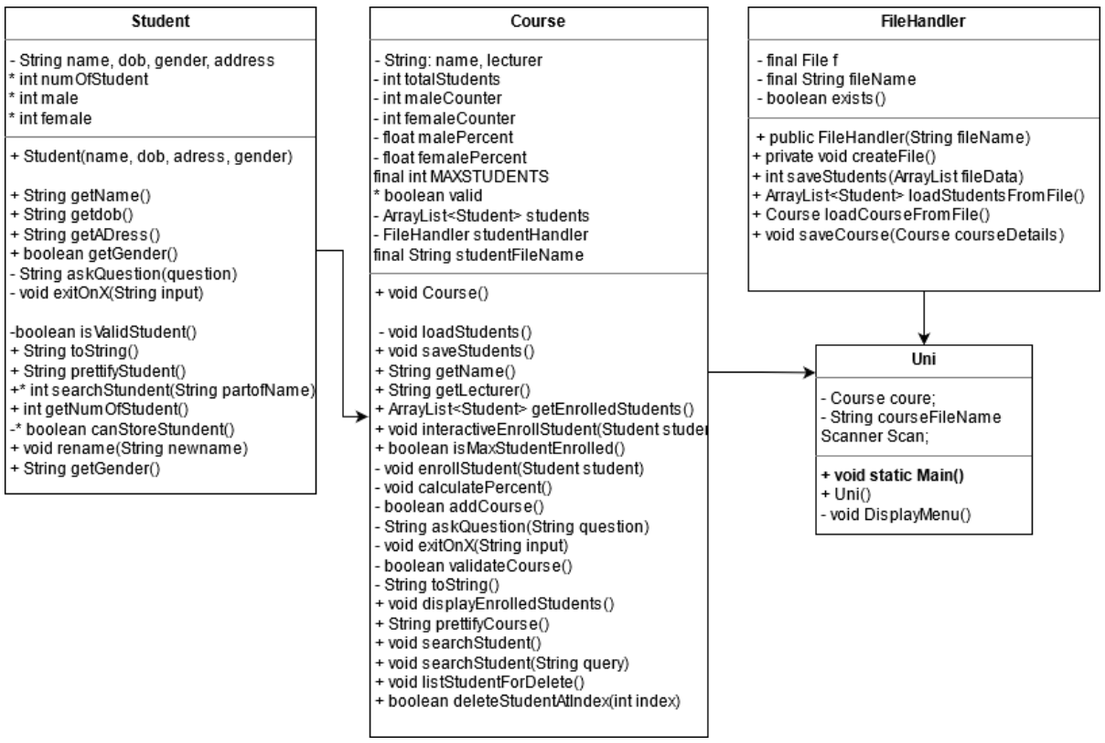

Figure 1: UML Diagram