CSE-215

PROGRAMMING LANGUAGE-II

FACULTY: RRN

SECTION:9

FINAL ASSIGNMENT(ASSIGNMENT-4)

SUBMITTED BY:

NAME:CHOWDHURY NAFIS FAIYAZ

ID: 1931841642

Class: Course

```
public class Course {

	private String id;
	private String title;
	private int credit;
	private int tutionPerCredit;
	private int numberofStudent=0;		//initial no. of students is 0 for
every course
	private int seatCapacity=3 ;		//initial seat capacity for all courses
is 3

//	course constructor
	public Course(String id, String title, int credit, int tutionPerCredit) {
		this.id=id;
		this.title=title;
		this.credit=credit;
		this.tutionPerCredit=tutionPerCredit;

	}
//	another constructor which only takes in the id.
	public Course(String id) {
		this.id=id;
		//this.title=null;

	}
//	getters
	public String getID() {
		return id;
	}
	public String getTitle() {
		return title;
	}
	public int getCredit() {
		return credit;
	}
	public int getNumberOfStudent() {
		return numberofStudent;

	}
	public int getSeatCapacity() {
		return seatCapacity;
	}
	public int getTutionPerCredit() {
		return tutionPerCredit;
	}
//	setters
	public void setId(String id) {
		this.id=id;
	}
	public void setTitle(String title) {
		this.title=title;
	}
	public void setCredit(int credit) {
```

```
            this.credit=credit;
      }
      public void setNumberOfStudent(int numberofStudent) {
            this.numberofStudent=numberofStudent;

      }

//    add/remove number of student
      public void alterNoOfStudent(int student) {
            this.numberofStudent+=student;
      }
//    add/remove seats, by default seat is 3
      public void alterNoOfSeat(int seat) {
            this.seatCapacity+=seat;
      }
//    prints the course id seat capacity and number of students specifically (self
defined method)
      public void printcourse() {
            System.out.println("Course id: "+ id+ " ,Seat capacity: "+seatCapacity+"
,Number of students: "+numberofStudent);
      }

//    this method returns a specific course fee .
      public int getSubTotal(){
            return credit*tutionPerCredit;
      }


//    to string method
      @Override
      public String toString(){
            return "Course id: "+ id+", Course credit: "+credit+", Tution per
credit: "+tutionPerCredit+", Number of students registered: "+numberofStudent+", Seat
capacity: "+seatCapacity+"|";
      }
}
```

Class: CurrentOfferedCourse

```
import java.util.ArrayList;

public class CurrentOfferedCourse {

//    Arraylist declaration
      ArrayList<Course> cList = new ArrayList<Course>(); // all the offered courses
will be stored in the cList arrayList

//  blank constructor
      public CurrentOfferedCourse() {

      }

//    add course parameter to the array of the courses
      public void addCourse(Course course) {
```

```
            cList.add(course); // adds course using the inbuilt add method of
arrayLists

      }

//  returns a course object if it is offered in a semester (getter for course)
      public Course getCourse(Course course) {

            return course;
      }

//    returns an array of all the offered courses in a semester
      public ArrayList<Course> getCourseList() {

            return cList;
      }
}
```

Class: Student

```
public class Student {

	private String name;
	private int id;
	private double cgpa;
	private Registration reg;
	private char freedomFighterStatus;
	private char minorityGroupStatus;

	// getters and setters
	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public double getCgpa() {
		return cgpa;
	}

	public void setCgpa(double cgpa) {
		this.cgpa = cgpa;
	}

	public Registration getReg() {
		return reg;
	}

	public void setReg(Registration reg) {
		this.reg = reg;
	}

	public void setFreedomFighterStatus(char x) {
		freedomFighterStatus = x;
	}

	public char getFreedomFighterStatus() {
		return freedomFighterStatus;
	}

	public void setMinorityStatus(char x) {
		minorityGroupStatus = x;
```

```
	}

	public char getMinortyStatus() {
		return minorityGroupStatus;
	}
//	student constructor

	public Student(String name, int id, double cgpa, char freedomFighterStatus,
char minorityGroupStatus) {
		this.name = name;
		this.id = id;
		this.cgpa = cgpa;
		this.freedomFighterStatus = freedomFighterStatus;
		this.minorityGroupStatus = minorityGroupStatus;
	}

//	making new registration of class type REGISTRATION
	public void makeNewRegistration() {
		reg = new Registration();
	}

//	this method adds courses according to the student cgpa and seat availability
	public void addCourse(Course course) {

		if (course.getNumberOfStudent() != course.getSeatCapacity()) { // will
execute this if statement until the no of

					// students reaches 3 which is = the defaault

					// seat capacity

			if (this.cgpa >= 3.5) { // will execute this if clause when
student cgpa is > 3.5.

				while (reg.getCourseList().size() <= 6) { // the loop will
continue until the size of the courselist

		// array becomes 6courses(18 credits).

					if (reg.getCourseList().size() == 6) { // when the
course list size becomes 6 then it gives the

		// student a warning.
						System.out.println(
								getName() + ": You cannot take "
+ course.getID() + ". You exceeded 18 credits limit");

						break;
					} else { // or else it adds the course using the add
course method of the registration
								// class.
						reg.addCourse(course);
						course.alterNoOfStudent(1); // increases the
number of students by 1 everytime a course is added
						break;
```

```
                                        }
                                  }
                            }

                            else if (this.cgpa < 3.5) { // executes when cgpa is < 3.5

                                  while (reg.getCourseList().size() <= 4) { // the loop will
continue until the size of the courselist

                  // array becomes 4courses(12 credits).

                                        if (reg.getCourseList().size() == 4) { // when the
course list size becomes 4 then it gives the

                  // student a warning.
                                              System.out.println(
                                                          getName() + ": You cannot take "
+ course.getID() + ". You exceeded 12 credits limit");

                                              break;
                                        } else { // or else it adds the course using the add
course method of the registration
                                                          // class.
                                              reg.addCourse(course);
                                              course.alterNoOfStudent(1);
                                              break;
                                        }
                                  }
                            }
                  }

                  // when the no. of students = to the number of seats available then this
else
                  // clause is executed.
                  // more students can take that course only if the seats are increased by
the
                  // admin class.
                  else {
                        System.out.println(course.getID() + " cannot be added. Seat is
Full !!");
                  }
            }

//      method for dropping course
            public void dropCourse(Course course) {

                  reg.deleteCourse(course); // from registration class
                  course.alterNoOfStudent(-1); // decreases the number of student by 1
after dropping the course.

            }

//      this method return the registration object of a student created during
makeRegistration method call(getter of reg)
            public Registration getRegistration() {
```

```
		return reg;
	}

	public void printRegisteredCourse() { // prints the final course list of a
student
		// reg.getCourseList();
		System.out.println("Course ID:   Course Title");
		System.out.println("=============================================");
		for (Course list : reg.getCourseList()) {
			System.out.println(list.getID() + "           " +
list.getTitle());
		}
		System.out.println("=============================================");
	}

//	It will set different discounts applicable for a student
	public void setDiscount() {

		if (this.getFreedomFighterStatus() == 'Y') {
			reg.setApplicableDiscounts(new FreedomFighterDiscount());
		}
		if (this.getMinortyStatus() == 'Y') {
			reg.setApplicableDiscounts(new MinorityGroupDiscount());
		}
		if (this.getCgpa() > 3.5) {
			reg.setApplicableDiscounts(new AcademicExcellenceDiscount());
		}

	}

//	it will return the breakdown of the bill
	public String getBillingInfo() {
		return "Billing Info:     ( ID: " + this.id + " )" + "\n" + "----------
-----------------------------------"
				+ "\n" + "Total Course Fees:     " + reg.getTotal() + "\n"
+ "Extra Fees:        +     "
				+ reg.getExtraFeeAmount() + "\n" + "----------------------
-------------------------" + "\n"
				+ "Grand Total:           " + reg.getGrandTotal() + "\n" +
"Discount:     --    " + reg.getDiscountAmount()
				+ "\n" + "-----------------------------------------------"
+ "\n" + "Payable Amount:        "
				+ reg.getPayableAmount();
	}

	public void printBillingInfo() {
		System.out.println(getBillingInfo());
	}

//	this method prints all the basic information of a student including the
billing info and courses registered for.
	public void printRegistrationSlip() {
		System.out.println("Registration Time: " + reg.getLocalDateTime());
		System.out.println("-------------------------------------------");
```

```
            System.out.println("Name:" + this.name + ", ID: " + this.id + ", CGPA: "
+ this.cgpa);
            System.out.println();
            System.out.println("-----------------------------------------------");
            printRegisteredCourse();
            System.out.println("===============================================");
            System.out.println(getBillingInfo());
      }

//    to string
      @Override
      public String toString() {
            return "Student [name=" + name + ", id=" + id + ", cgpa=" + cgpa + ",
freedomFighterStatus="
                        + freedomFighterStatus + ", minorityGroupStatus=" +
minorityGroupStatus + "]";
      }

}
```

Class: Registration

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;

public class Registration {
//		declaring course array
	private ArrayList<Course> courseList = new ArrayList<Course>();
	private ArrayList<IDiscountStrategy> applicableDiscounts = new
ArrayList<IDiscountStrategy>();
	private IExtraFeeCalculator eFeeCalculator;

//		blank constructor
	public Registration() {

	}

//		this method adds a course given by the student to a students course list
	public void addCourse(Course course) {
		courseList.add(course);
	}

//		this deletes a course given by the student from the course list
	public void deleteCourse(Course course) {
		courseList.remove(course); // removes the desired course form the courselist array
by using .remove method
		course.alterNoOfStudent(-1); // refreshes the total number of student by subrtacting 1
student
	}

//		this will return all the courses the student registered for the semester
	public ArrayList<Course> getCourseList() {
		return courseList;
	}

	public String getLocalDateTime() {
		DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
		LocalDateTime now = LocalDateTime.now();
		return dtf.format(now);
	}

	// it will return the total amount based on a students registered course.
	public double getTotal() {
		int total = 0;
		for (Course list : courseList) { // enters inside each courses in the course list and
returns the amount of
                                                                                    // money for
each courses.

			total += list.getSubTotal();
		}
		double Total = total;
		return Total;
	}
```

```
//	it will return the extra fee that will be applied by the university depending on the semester
	public int getExtraFeeAmount() {

		Admin adminObject = Admin.getInstance();

		this.eFeeCalculator = adminObject.getExtraFeeCalculator();

		if (eFeeCalculator instanceof DevelopmentFeeCalculator) {

			return eFeeCalculator.getExtraAmount((int) this.getTotal());

		}
		if (eFeeCalculator instanceof BDTaxAdapter) {

			return eFeeCalculator.getExtraAmount((int) this.getTotal());

		} else
			return 0;

	}

//	it will return the grand total amount for a specific student.
	public int getGrandTotal() {
		double doubleExtraFee = getExtraFeeAmount();
		int extraFeeDouble = (int) doubleExtraFee;

		double doubleTotal = getTotal();
		int intTotalFee = (int) doubleTotal; // type casted from double to integer
		int sum = intTotalFee + extraFeeDouble;
		return sum;

	}

//	this method is used to add instances of discount classes to the arraylist<idiscountStrategy>
	public void setApplicableDiscounts(IDiscountStrategy discountStrategy) {

		applicableDiscounts.add(discountStrategy);

	}

//	this will return the discount amount for a student based on his status.
	public int getDiscountAmount() {

		int academicDiscount = 0;
		int freedomDiscount = 0;
		int minorityDiscount = 0;

		int max = 0;

		AcademicExcellenceDiscount Academic = new AcademicExcellenceDiscount();
		FreedomFighterDiscount Freedom = new FreedomFighterDiscount();
		MinorityGroupDiscount Minority = new MinorityGroupDiscount();

// the forloop will go throught the arraylist named applicable discount and will get the discounts
availed by the student and store them in the designated variables.
```

```
                for (int i = 0; i < applicableDiscounts.size(); i++) {
                        if (applicableDiscounts.get(i) instanceof AcademicExcellenceDiscount) {
                                academicDiscount = Academic.getTotal(this);

                        }
                        if (applicableDiscounts.get(i) instanceof FreedomFighterDiscount) {
                                freedomDiscount = Freedom.getTotal(this);

                        }

                        if (applicableDiscounts.get(i) instanceof MinorityGroupDiscount) {
                                minorityDiscount = Minority.getTotal(this);

                        }

                }

//this determines the maximum discount availed by the student and returns the maximum value.
                if (academicDiscount > minorityDiscount && academicDiscount > freedomDiscount) {
                        max = academicDiscount;

                } else if (freedomDiscount > academicDiscount && freedomDiscount >
minorityDiscount) {
                        max = freedomDiscount;
                } else if (minorityDiscount > academicDiscount && minorityDiscount >
freedomDiscount) {
                        max = minorityDiscount;
                }
                return max;
        }
        public int getPayableAmount() {
                return this.getGrandTotal() - this.getDiscountAmount();
        }

}
```

Interface IExtraFeeCalculator

```
public interface IExtraFeeCalculator {
        public abstract int getExtraAmount(int courseTotal);
}
```

Class: DevelopmentFeeCalculator

```
public class DevelopmentFeeCalculator implements IExtraFeeCalculator {
        @Override
        public int getExtraAmount(int courseTotal) {

                double DevelopmentfeeDouble = courseTotal * 0.10;
                int DevelopmentFee = (int) DevelopmentfeeDouble;
                return DevelopmentFee;

        }
}
```

Class: BDTaxAdapter

```
public class BDTaxAdapter extends BDTaxCalculator implements IExtraFeeCalculator {

	@Override
	public int getExtraAmount(int courseTotal) {
		BDTaxCalculator tax = new BDTaxCalculator();
		double taxDouble = tax.calculateVatAmount(courseTotal);
		int taxInt = (int) taxDouble;
		return taxInt;
	}

}
```

Class: BDTaxCalculator

```
public class BDTaxCalculator {
	public double calculateVatAmount(int total) {
		return total * 0.15;

	}
}
```

Interface: IDiscountStrategy

```
public interface IDiscountStrategy {
	public abstract int getTotal(Registration reg);}
```

Class: AcademicExcellenceDiscount

```
public class AcademicExcellenceDiscount implements IDiscountStrategy {

	@Override
	public int getTotal(Registration reg) {
		double discount = reg.getTotal() * 0.20;
		int Discount = (int) discount;
		return Discount;

	}

}
```

Class: FreedomFighterDiscount

```
public class FreedomFighterDiscount implements IDiscountStrategy {

	@Override
	public int getTotal(Registration reg) {
		// it will return the discount amount calculated for a specific student.
		double discount = reg.getTotal() * 0.25;
		int Discount = (int) discount;
		return Discount;
	}}
```

Class: MinorityGroupDiscount

```
public class MinorityGroupDiscount implements IDiscountStrategy {

	@Override
	public int getTotal(Registration reg) {

		double discount = reg.getTotal() * 0.10;
		int Discount = (int) discount;
		return Discount;

	}

}
```

Class: Admin

```
import java.util.ArrayList;

public class Admin {
	private IExtraFeeCalculator eFeeCalculator;

//	blank constructor
	public Admin() {

	}

//declaring arraylist of course type
	ArrayList<Course> courseOfferedinSemester = new ArrayList<>();

//takes a course of COURSE type from the user and adds it to the array list named
offered course
	public void offerCourse(Course course) {

		courseOfferedinSemester.add(course);
	}

//	prints all the courses offered in a semester with course id
	public void publishOfferedCourse() {

		for (int i = 1; i < 9; i++) { // the first for loop prints the serial
numbers.

			for (Course list : courseOfferedinSemester) { // enhanced for
loop is used to print elements form the array

		// list
				System.out.print(i + ". ");
				System.out.println("Course Id: " + list.getID());
				i++;
			}
		}
	}

//	increases the seat capacity of the desired course
	public void increaseSeatCapacity(Course course, int size) {
```

```
			course.alterNoOfSeat(size - 3);
		}

		public void seeCourseStatus() {
//			prints all the offered courses with course id , number of students and
seats
//			used a user-defined function called print course from the course class
			for (Course list : courseOfferedinSemester) {
				list.printcourse();
			}
		}

		public void setExtraFeeCalculator(IExtraFeeCalculator eFeeCalculator) {
			this.eFeeCalculator = eFeeCalculator;
		}

		public IExtraFeeCalculator getExtraFeeCalculator() {
			return eFeeCalculator;
		}

//this is used to set the type of extrafee to be charged.
		private static Admin instance;

		public static Admin getInstance() {
			if (instance == null) {
				instance = new Admin();
			}
			return instance;
		}

}
```

Class: DriverClass

```
public class DriverClass {

		public static void main(String[] args) {

			Course CSE115 = new Course("CSE115", "Programming Language-I", 3, 6000);
			Course CSE173 = new Course("CSE173", "Discrete Mathematics", 3, 6000);
			Course CSE215 = new Course("CSE215", "Programming Language-II", 3,
6000);
			Course CSE225 = new Course("CSE225", "Data Structures and Algorithms",
3, 6000);
			Course CSE231 = new Course("CSE231", "Digital Logic Design", 3, 6000);
			Course CSE311 = new Course("CSE311", "Database Systems", 3, 6000);
			Course CSE323 = new Course("CSE323", "Operating Systems Design", 3,
6000);
			Course CSE373 = new Course("CSE373", "Design and Analysis of
Algorithms", 3, 6000);

			Student s1 = new Student("Farhan Islam", 1631728042, 2.70, 'Y', 'N');
			Student s2 = new Student("Sadia Sultana", 1821347042, 3.44, 'N', 'Y');
			Student s3 = new Student("Sanjida Akter", 2021746042, 3.65, 'N', 'N');
			Student s4 = new Student("Farhan Bhuiyan", 1923147042, 3.94, 'N', 'N');
```

```
            Student s5 = new Student("Mahmudul Hoque", 1524137042, 2.14, 'Y', 'Y');

            Admin admin = Admin.getInstance(); // creating an Admin object

//SEGMENTS OF Assignment 3 INSTANTIATING STUDENT OBJECTS AND
//ADDING AND DROPPING COURSES ACCORDING TO THEIR CGPA.
            s1.makeNewRegistration();
            s2.makeNewRegistration();
            s3.makeNewRegistration();
            s1.addCourse(CSE115);
            s1.addCourse(CSE173);
            s2.addCourse(CSE115);
            s2.addCourse(CSE215);
            s2.addCourse(CSE225);
            s3.addCourse(CSE115);
            s3.addCourse(CSE225);
            s3.addCourse(CSE311);
            admin.increaseSeatCapacity(CSE115, 5);
            s4.makeNewRegistration();
            s5.makeNewRegistration();
            s4.addCourse(CSE115);
            s4.addCourse(CSE225);
            s5.addCourse(CSE115);
            s5.addCourse(CSE173);
            s5.addCourse(CSE215);
// adding 4 more course to s3
            s3.addCourse(CSE173);
            s3.addCourse(CSE215);
            s3.addCourse(CSE231);
            s3.addCourse(CSE323); // THIS COURSE WONT BE ADDED BECAUSE S3 ALREADY
REACHED 6 COURSE LIMIT
// adding 2 more course to s5
            s5.addCourse(CSE311);
            s5.addCourse(CSE373); // THIS COURSE WONT BE ADDED BECAUSE S3 ALREADY
REACHED 4 COURSE LIMIT
            s3.dropCourse(CSE311);
            System.out.println("\n\n");

//          ASSIGNMENT 3 ENDS.

//    ************************ ASSIGNMENT 4 ********************************:

//    PREPARING BILLING INFO FOR THE STUDENTS):

            DevelopmentFeeCalculator devFee = new DevelopmentFeeCalculator();//
CREATING OBJECTS OF DEVELOPMENTfeeCalculator

                                        // CLASS AND BDTAX.
            BDTaxAdapter tax = new BDTaxAdapter();

            admin.setExtraFeeCalculator(devFee); // DEVELOPMENT FEES WILL BE CHARGED
FOR S1 AND S2.

//    TASK 1 :
            s1.setDiscount();
```

```
            s1.printBillingInfo();
            System.out.println("\n\n");
//     TASK 2:
            s2.setDiscount();
            s2.printBillingInfo();
            System.out.println("\n\n");
//     TASK 3:

            admin.setExtraFeeCalculator(tax); // setExtraFee method is set to
calculate BD tax by passing tax as a parameter.
//TASK 4:
            s3.setDiscount();
            s3.printBillingInfo();
            System.out.println("\n\n");
//     task 5:
            s4.setDiscount();
            s4.printBillingInfo();
            System.out.println("\n\n");
//     task 6:
            s5.setDiscount();
            s5.printRegistrationSlip(); // prints the complete slip with the
courselist for student 5.
            System.out.println("\n\n");

      }

}
```