Lab Assignment 06

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Topic:	**Encapsulation and Multi-class Design**
Number of Tasks:	**9 (Classwork: 04, Homework: 05)**

[Submit all the Coding Tasks (Homework: Task 1 to 5) in the Google Form shared on buX before the next lab.]

[You are not allowed to change the driver codes of any of the tasks]

CLASSWORK

Task 1

Write a class called Circle with the required constructor and methods to get the following output.

Subtasks:

1. Create a class called Circle.
2. Create the required constructor. Use Encapsulation to protect the variables. [Hint: Assign the **radius** variable in **private**]
3. Create a getRadius() and setRadius() method to access variables.
4. Create a method called area to calculate the area of circles.

Given Code	Expected Output

public class CircleTester {
  public static void main(String[] args) {
  Circle c1 = new Circle(4);
  System.out.println("1---------------");
  System.out.println("First circle radius: " + c1.getRadius());
  System.out.println("First circle area: " + c1.area());
  System.out.println("2---------------");
  Circle c2 = new Circle(5);
  System.out.println("Second circle radius: " + c2.getRadius());
  System.out.println("Second circle area: " + c2.area());
  System.out.println("3---------------");
  }
}
	1---------------
	First circle radius: 4.0
	First circle area: 50.26548245743669
	2---------------
	Second circle radius: 5.0
	Second circle area: 78.53981633974483
	3---------------

Task 2

Design the required class/es so that the following output is generated. Read the following description:

1. You may assume that to board a bus, a student must have the bus pass, and his/her location must match the route of the bus.
2. Additionally, the default maximum capacity of the bus is 2.
3. The location attribute of the Student class will be **private**

Driver Code	Output

public class BracuStudentTester {
 public static void main(String[] args) {
  BracuStudent st1 = new BracuStudent("Afif", "Mirpur");
  System.out.println("1==============");
  BracuStudent st2 = new BracuStudent("Shanto",
"Motijheel");
  BracuStudent st3 = new BracuStudent("Taskin", "Mirpur");
  st1.showDetails();
  st2.showDetails();
  System.out.println("2==============");
  st3.showDetails();
  System.out.println("3==============");
  BracuBus bus1 = new BracuBus("Mirpur");
  BracuBus bus2 = new BracuBus("Azimpur", 5);
  bus1.showDetails();
  bus2.showDetails();
  System.out.println("4==============");
  st2.collectPass();
  st3.collectPass();
  System.out.println("5==============");
  st2.showDetails();
  st3.showDetails();
  System.out.println("6==============");
  bus1.board();
  System.out.println("7==============");
  bus1.board(st1, st2);
  System.out.println("8==============");
  st1.collectPass();
  st2.setLocation("Mirpur");
  st1.showDetails();
  st2.showDetails();
  System.out.println("9==============");
  bus1.board(st1);
  bus1.board(st2, st3);
  System.out.println("10==============");
  bus1.showDetails();
 }
}
	1==============
	Student Name: Afif
	Lives in Mirpur
	Have Bus Pass? false
	Student Name: Shanto
	Lives in Motijheel
	Have Bus Pass? false
	2==============
	Student Name: Taskin
	Lives in Mirpur
	Have Bus Pass? false
	3==============
	Bus Route: Mirpur
	Passenger Count: 0 (Max: 2)
	Passengers on Board:
	Bus Route: Azimpur
	Passenger Count: 0 (Max: 5)
	Passengers on Board:
	4==============
	5==============
	Student Name: Shanto
	Lives in Motijheel
	Have Bus Pass? true
	Student Name: Taskin
	Lives in Mirpur
	Have Bus Pass? true
	6==============
	No passengers
	7==============
	You don't have a bus pass!
	You got on the wrong bus!
	8==============
	Student Name: Afif
	Lives in Mirpur
	Have Bus Pass? true
	Student Name: Shanto
	Lives in Mirpur
	Have Bus Pass? true
	9==============
	Afif boarded the bus.
	Shanto boarded the bus.
	Bus is full!
	10==============
	Bus Route: Mirpur
	Passenger Count: 2 (Max: 2)
	Passengers on Board:
	Afif Shanto

Task 3

Design the **Student** and the **Connect** class so that the following output is produced.
Note:

- A student's email, password, and login status are null by default while creating an object of the Student class.
- The password and login status attributes of the Student class will be **private**
- Your code should satisfy the conditions mentioned in the output only.
- Connect class will have two instance variables: totalAdvisee and an array of Student type to store the student object. The array will be updated inside the advising() method only when the advising is successful.
- Connect can take at most 5 advisees.

Driver Code	Expected Output

public class ConnectTester {
  public static void main(String[] args) {
  Student rakib = new Student("Rakib", 12301455, "CSE");
  Student roy = new Student("Roy", 12501345, "CS");
  System.out.println("1*************");
  Connect connectObj = new Connect();
  System.out.println("2*************");
  connectObj.login(rakib);
  System.out.println("3*************");
  connectObj.advising(rakib);
  System.out.println("4*************");
  rakib.email = "rakib@hotmail.com";
  rakib.setPassword("1234");
  System.out.println("5*************");
  connectObj.login(rakib);
  System.out.println("6*************");
  connectObj.advising(rakib);
  System.out.println("7*************");
  connectObj.advising(rakib, "CSE110", "PHY111", "MAT110", "CSE260");
  System.out.println("8*************");
  connectObj.advising(rakib, "CSE110", "PHY111","MAT110");
  System.out.println("9*************");
  connectObj.allAdviseeInfo();
  System.out.println("10*************");
  roy.email = "roy@hotmail.com";
  roy.setPassword("abcd");
  connectObj.login(roy);
  System.out.println("11*************");
  connectObj.advising(roy, "CSE110", "ENG101", "PHY112");
  System.out.println("12*************");
  connectObj.allAdviseeInfo();
  }
}
	Student object is created
	Student object is created
	1*************
	Connect is ready to use!
	2*************
	Email and password need to be set.
	3*************
	Please login to advise courses!
	4*************
	5*************
	Login successful
	6*************
	You haven't selected any courses.
	7*************
	You need special approval to take more than 3 courses.
	8*************
	Advising successful!
	9*************
	Total Advisee: 1
	Name: Rakib ID: 12301455
	Department: CSE
	Advised Courses:
	CSE110 PHY111 MAT110
	==============
	10*************
	Login successful
	11*************
	Advising successful!
	12*************
	Total Advisee: 2
	Name: Rakib ID: 12301455
	Department: CSE
	Advised Courses:
	CSE110 PHY111 MAT110
	==============
	Name: Roy ID: 12501345
	Department: CS
	Advised Courses:
	CSE110 ENG101 PHY112
	==============

Task 4

```
1  public class ExamClass {
2      public int ques;
3      public int sum;
4      public void methodA() {
5          System.out.println(ques + " " + 0 + " " + 0);
6      }
7  }
8  class QuizA {
9      public int x, y;
10     public int sum = 1;
11     public QuizA(int x, int y) {
12         this.x = y;
13         this.y = x;
14     }
15     public void methodA() {
16         int x = 3;
17         y = this.y + x;
18         ExamClass exam = new ExamClass();
19         exam.sum = x;
20         exam.ques = this.y;
21         x = this.x + x + exam.sum;
22         this.y = this.sum + methodB(exam.ques, exam);
23         System.out.println(x + " " + this.y + " " + sum);
24         sum = x % 2 + this.x;
25         y = x + y + exam.sum;
26         System.out.println(x + " " + y + " " + sum);
27     }
28     public int methodB(int x1, ExamClass x2) {
29         int y = 0;
30         y = this.y + x2.sum;
31         x2.ques = x1 + x2.ques;
32         sum = sum + x + y;
33         System.out.println(this.x + " " + this.y + " " + sum);
34         return x2.sum;
35     }
36 }
```

Driver Code	Output		

`public class QuizTesterA{`
`  public static void main(String []args){`
`  QuizA q1 = new QuizA(3,4);`
`  q1.methodA();`
`  }`
`}`			

HOMEWORK

Task 1

Write the "**Product**" class to show the following output
Note: Make sure to use proper ***Encapsulation concepts*** **for the setter & getter methods. All the attributes should have Private access.**

Driver Code	Output

public class ProductTester{
  public static void main(String[] args) {
  System.out.println("< --------1---------->");
  Product product1 = new Product();
  product1.displayInfo();
  System.out.println("< --------2---------->");
  Product product2 = new Product("Laptop", 1200.00);
  product2.setQuantity(10);
  product2.displayInfo(true);
  System.out.println("< --------3---------->");
  System.out.println("Retrieved Price: $" + product2.getPrice());
  System.out.println("Retrieved Quantity: " + product2.getQuantity());
  }
}
	< --------1---------->
	Product Name: Unknown
	Price: $0.0
	< --------2---------->>
	Product Name: Laptop
	Price: $1200.0
	Quantity: 10
	< --------3---------->>
	Retrieved Price: $1200.0
	Retrieved Quantity: 10

Task 2

Design the Company and Employee classes so that the Tester1 class produces the given outputs. [All attributes of Employee class should be **Private**]
Restriction: Company class can't have more than 1 array.

Driver Code	Output

public class Tester1{
  public static void main(String args[]){
  Employee e1 = new Employee();
  Employee e2 = new Employee("Alif", 34, "Fulltime");
  System.out.println("1-------------");
  Company c1 = new Company();
  c1.details();
  System.out.println("2-------------");
  Employee e3 = new Employee("Akter", 36,"Part-time");
  Employee e4 = new Employee("Ria", 38, "Fulltime");
  System.out.println("3-------------");
  c1.addEmployee(e2);
  c1.addEmployee(e3);
  System.out.println("4-------------");
  c1.details();
  System.out.println("5-------------");
  c1.addEmployee(e4);
  c1.addEmployee(e1);
  System.out.println("6-------------");
  c1.details();
  System.out.println("7-------------");
  c1.removeEmployee(e4);
  System.out.println("6-------------");
  c1.details();
  }
}
	A default employee has been created
	1-------------
	Company Name: ABC Company
	Total Employee: 0
	Fulltime Employees:
	Part-Time Employees:
	2-------------
	3-------------
	Alif has joined the company
	Akter has joined the company
	4-------------
	Company Name: ABC Company
	Total Employee: 2
	Fulltime Employees:
	Name: Alif, ID: 34
	Part-Time Employees:
	Name: Akter, ID: 36
	5-------------
	Ria has joined the company
	No more vacancy
	6-------------
	Company Name: ABC Company
	Total Employee: 3
	Fulltime Employees:
	Name: Alif, ID: 34
	Name: Ria, ID: 38
	Part-Time Employees:
	Name: Akter, ID: 36
	7-------------
	Ria has left the company
	6-------------
	Company Name: ABC Company
	Total Employee: 2
	Fulltime Employees:
	Name: Alif, ID: 34
	Part-Time Employees:
	Name: Akter, ID: 36

Task 3

Please write the **Student** and **Department** class with the necessary properties so that the provided driver code generates the output given below. The id of the Student class will be **private**. For simplicity, assume that a department can add a maximum of 5 students.

Driver Code

```
public class Tester1 {
 public static void main(String[] args) {
  Student s1 = new Student("Akib", 10, 3.29);
  Student s2 = new Student("Reza", 15, 3.45);
  Student s3 = new Student("Kabir", 20,4.0);
  System.out.println("1==========");
  Department cse = new Department("CSE");
  cse.findStudent(-100);
  System.out.println("2==========");
  cse.addStudent(s1, s2, s3);
  System.out.println("3==========");
  cse.details();
  System.out.println("4==========");
  cse.findStudent(15);
  System.out.println("5==========");
  Student s4 = new Student("Nakib", 15,3.22);
  cse.addStudent(s4);
  System.out.println("6==========");
  s4.setId(25);
  cse.addStudent(s4);
  System.out.println("7==========");
  cse.details();
  System.out.println("8==========");
  Student s5 = new Student("Sakib", 30,2.29);
  cse.addStudent(s5);
  System.out.println("9==========");
  cse.details();
 }
}
```

Output

```
1==========
Student with this ID doesn't exist, Please
give a valid ID
2==========
Welcome to CSE department, Akib
Welcome to CSE department, Reza
Welcome to CSE department, Kabir
3==========
Department Name: CSE
Number of student:3
Details of the students:
Student name: Akib, ID: 10, cgpa: 3.29
Student name: Reza, ID: 15, cgpa: 3.45
Student name: Kabir, ID: 20, cgpa: 4.0
4==========
Student info:
Student Name: Reza
ID: 15
CGPA:  3.45
5==========
Student with the same ID already exists,
Please try with another ID
6==========
Welcome to CSE department, Nakib
7==========
Department Name: CSE
Number of student:4
Details of the students:
Student name: Akib, ID: 10, cgpa: 3.29
Student name: Reza, ID: 15, cgpa: 3.45
Student name: Kabir, ID: 20, cgpa: 4.0
Student name: Nakib, ID: 25, cgpa: 3.22
8==========
Welcome to CSE department, Sakib
9==========
Department Name: CSE
Number of student:5
Details of the students:
Student name: Akib, ID: 10, cgpa: 3.29
Student name: Reza, ID: 15, cgpa: 3.45
Student name: Kabir, ID: 20, cgpa: 4.0
Student name: Nakib, ID: 25, cgpa: 3.22
Student name: Sakib, ID: 30, cgpa: 2.29
```

Task 4

Spaceship: This class represents a spaceship. Each spaceship has a **name** and a **capacity** (the maximum weight it can carry).
Cargo: This class represents a piece of cargo. Each cargo item has a **name** and a **weight**. Both attributes should be **private** which means they cannot be accessed directly from outside of the class.
A Spaceship contains Cargo. That means each spaceship can carry multiple cargo items, but the total weight of the cargo cannot exceed the spaceship's capacity. Also, the maximum number of cargo items is 100. Your task is to design the **Spaceship** and **Cargo** class with necessary properties so that the given output is produced for the provided driver code.

Driver Code	Output

`public class Tester2 {`
` public static void main(String[] args) {`
`  Spaceship falcon = new Spaceship("Falcon", 50000);`
`  Spaceship apollo = new Spaceship("Apollo", 100000);`
`  Spaceship enterprise = new Spaceship("Enterprise",220000);`
`  System.out.println("1.==========");`
`  Cargo gold = new Cargo("Gold", 20000);`
`  Cargo platinum = new Cargo("Platinum", 25000);`
`  Cargo dilithium = new Cargo("Dilithium", 50000);`
`  Cargo trilithium = new Cargo("Trilithium", 70000);`
`  Cargo neutronium = new Cargo("Neutronium", 80000);`
`  System.out.println("2.==========");`
`  falcon.loadCargo(gold);`
`  falcon.loadCargo(platinum);`
`  falcon.displayDetails();`
`  System.out.println("3.==========");`
`  apollo.loadCargo(gold);`
`  apollo.displayDetails();`
`  System.out.println("4.==========");`
`  falcon.loadCargo(neutronium);`
`  System.out.println("5.==========");`
`  enterprise.loadCargo(dilithium);`
`  enterprise.loadCargo(trilithium);`
`  enterprise.loadCargo(neutronium);`
`  enterprise.displayDetails();`
` }`
`}`
	`1.==========`
	`2.==========`
	`Spaceship Name: Falcon`
	`Capacity: 50000`
	`Current Cargo Weight: 45000`
	`Cargo:Gold Platinum`
	`3.==========`
	`Spaceship Name: Apollo`
	`Capacity: 100000`
	`Current Cargo Weight: 20000`
	`Cargo:Gold`
	`4.==========`
	`Warning: Unable to load Neutronium inside Falcon. Exceeds capacity by 75000.`
	`5.==========`
	`Spaceship Name: Enterprise`
	`Capacity: 220000`
	`Current Cargo Weight: 200000`
	`Cargo:Dilithium Trilithium Neutronium`

Task 5

```
1  public class Foo{
2    public int bar, buz;
3    public Foo(int bar, int buz){
4      this.bar = bar;
5      this.buz = buz;
6    }
7  }
8  class Quiz5{
9    public int sum = 12, x = 2, y = 6;
10   public Foo foo;
11   public Quiz5(Foo f){
12     foo = f;
13     int x = this.foo.buz + y;
14     sum = sum + (f.bar--) + y;
15     System.out.println(foo.bar + " " + sum + " " + x);
16     sum -= 10;
17   }
18   public void methodA(int bar, int buz){
19     bar = 3 + bar - this.foo.bar;
20     x = bar + 12 + y;
21     y = foo.buz + buz + bar;
22     sum = y + methodB(foo.buz, foo) + foo.buz;
23     System.out.println(bar + " " + y + " " + sum);
24   }
25   public int methodB(int bar, Foo buz){
26     int sum = bar + buz.bar + x;
27     buz.buz = sum + this.sum;
28     System.out.println(bar + " " + buz.buz + " " + sum);
29     return sum;
30   }
31 }
```

Driver Code	Output		

`public class LabTester{`
`  public static void main(String []args){`
`  Foo p = new Foo(3, 4);`
`  Quiz5 q = new Quiz5(p);`
`  q.methodA(4, 8);`
`  }`
`}`			

Ungraded Tasks (Optional)

(You don't have to submit the ungraded tasks)

Task 1

Design the **Vaccine** and **Person** class so that the following expected output is generated.
[N.B: Students will get vaccines on a priority basis. So, age doesn't matter for students. All **attributes of Vaccine** class should be **Private**.**]**

Driver Code	Output

public class VaccineTester {
  public static void main(String[] args) {
  Vaccine astra = new Vaccine("AstraZeneca", "UK", 60);
  Vaccine modr = new Vaccine("Moderna", "UK", 30);
  Vaccine sin = new Vaccine("Sinopharm", "China", 30);
 	
  Person p1 = new Person("Bob", 21, "Student");
  System.out.println("================");
  p1.pushVaccine(astra);
  System.out.println("================");
  p1.showDetail();
  System.out.println("================");
  p1.pushVaccine(sin, "2nd Dose");
  System.out.println("================");
  p1.pushVaccine(astra, "2nd Dose");
  System.out.println("================");
  p1.showDetail();
  System.out.println("================");
  p1.pushVaccine(astra, "2nd Dose");
  System.out.println("================");
  p1.showDetail();
  System.out.println("================");
 	
  Person p2 = new Person("Carol", 23, "Actor");
  System.out.println("================");
  p2.pushVaccine(sin);
  System.out.println("================");
 	
  Person p3 = new Person("David", 34);
  System.out.println("================");
  p3.pushVaccine(modr, "2nd Dose");
  System.out.println("================");
  p3.pushVaccine(modr, "1st Dose");
  System.out.println("================");
  p3.showDetail();
  System.out.println("================");
  p3.pushVaccine(modr, "2nd Dose");
  }
}
	================
	1st dose done for Bob
	================
	Name: Bob Age: 21 Type: Student
	Vaccine name: AstraZeneca
	1st dose: Given
	2nd dose: Please come after 60 days
	================
	Sorry Bob, you can't take 2 different vaccines
	================
	2nd dose done for Bob
	================
	Name: Bob Age: 21 Type: Student
	Vaccine name: AstraZeneca
	1st dose: Given
	2nd dose: Given
	================
	Sorry Bob, you already received both doses.
	================
	Name: Bob Age: 21 Type: Student
	Vaccine name: AstraZeneca
	1st dose: Given
	2nd dose: Given
	================
	================
	Sorry Carol, Minimum age for taking vaccines is 25 years now.
	================
	================
	Sorry David, invalid dose request.
	================
	1st dose done for David
	================
	Name: David Age: 34 Type: General Citizen
	Vaccine name: Moderna
	1st dose: Given
	2nd dose: Please come after 30 days
	================
	2nd dose done for David

Task 2

Design the **Library** and **Reader** class so that the following output is generated.

Read the following description:

- The Library class has two pairs of arrays: one pair contains borrower information(the name of borrowers and the number of books they borrowed) and the other contains book availability information (book type and their remaining number)
- A reader cannot borrow more than 5 books.
- If a book's availability is 0 in the Library, then the reader cannot borrow that book.
- The readerInfo method in the Reader class prints the type and the number of all books borrowed if no parameter is passed, else it prints the number of books borrowed of the specific type mentioned in the parameter.

Driver Code	Output

public class LibraryTester {
 public static void main(String[] args) {
  String [] genres = {"Arts", "Fiction", "Politics", "Science", "Poetry"};
  int [] available = {15, 135, 2, 11, 15};
  Library L1 = new Library("Dhaka", genres,available);
  L1.details();
  System.out.println("1----------");
  Reader r1 = new Reader("Aladdin", L1.getBookTypes());
  r1.borrow(L1, "Arts", "Fiction", "Fiction", "Politics");
  System.out.println("2----------");
  r1.borrow(L1, "Politics", "Fiction");
  System.out.println("3----------");
  r1.readerInfo();
  System.out.println("4----------");
  r1.readerInfo("Fiction");
  System.out.println("5----------");
  L1.details();
  System.out.println("6----------");
  Reader r2 = new Reader("Jasmine", L1.getBookTypes());
  r2.borrow(L1, "Politics", "Poetry");
  System.out.println("7----------");
  r2.readerInfo();
  System.out.println("8----------");
  L1.details();
 }
}
	Dhaka Library details
	Borrower details:
	No borrowers yet.
	Books availability:
	Arts: 15, Fiction: 135, Politics: 2, Science: 11, Poetry: 15
	1----------
	Arts book is borrowed successfully.
	Fiction book is borrowed successfully.
	Fiction book is borrowed successfully.
	Politics book is borrowed successfully.
	2----------
	Politics book is borrowed successfully.
	You cannot borrow more than 5 books.
	3----------
	Aladdin, you have 5 book(s) with you.
	Books on Arts: 1
	Books on Fiction: 2
	Books on Politics: 2
	4----------
	Aladdin, you have 2 Fiction book(s) with you.
	5----------
	Dhaka Library details
	Borrower details:
	Aladdin: 5
	Books availability:
	Arts: 14, Fiction: 133, Politics: 0, Science: 11, Poetry: 15
	6----------
	Politics books are not available at the moment.
	Poetry book is borrowed successfully.

	7---------- Jasmine, you have 1 book(s) with you. Books on Poetry: 1 8---------- Dhaka Library details Borrower details: Aladdin: 5, Jasmine: 1 Books availability: Arts: 14, Fiction: 133, Politics: 0, Science: 11, Poetry: 14