Lab Assignment 06

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Topic:	**Encapsulation, Static variable and Static Method**
Number of Tasks:	**11**

[Submit all the Coding Tasks (Task 1 to 8) in the Google Form shared on buX before the next lab. Submit the Tracing Tasks (Task 9 to 11) handwritten to your Lab Instructors at the beginning of the lab]

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

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();
  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 **Passenger** class in such a way that the following code provides the expected output.

- Passenger class has two static variables ***no_of_passenger*** and ***total_fare.***
- Each passenger has to pay 20 TK/Distance and extra 10 TK/BaggageWeight.

Given Code	Expected Output

public class PassengerTester{
  public static void main(String args[]){
  System.out.println("Total Passenger: "+ Passenger.no_of_passenger);
  System.out.println("Total Fare: "+ Passenger.total_fare + " TK");
  System.out.println("==========1==========");
  Passenger p1 = new Passenger("Lara", 5.6);
  p1.passengerDetails();
  System.out.println("==========2==========");
  Passenger p2 = new Passenger("Kevin", 10.0);
  p2.storeBaggageWeight(6.8);
  p2.passengerDetails();
  System.out.println("==========3==========");
  Passenger p3 = new Passenger("Robin", 2.3);
  p3.storeBaggageWeight(5.0);
  p3.passengerDetails();
  System.out.println("==========4==========");
  System.out.println("Total Passenger: "+ Passenger.no_of_passenger);
  System.out.println("Total Fare: "+ Passenger.total_fare + " TK");
  }
}
	Total Passenger: 0
	Total Fare: 0.0 TK
	==========1==========
	Name: Lara
	Fare: 112.0 TK
	==========2==========
	Name: Kevin
	Fare: 268.0 TK
	==========3==========
	Name: Robin
	Fare: 96.0 TK
	==========4==========
	Total Passenger: 3
	Total Fare: 476.0 TK

Task 3

Design a **Book** class in such a way that the following code provides the expected output.

- The Book class has two static variables: total_books_sold and total_revenue.
- Each book has a base price of 150 TK. If the discountPercentage is applied, the book's price is reduced by that percentage.
- The Book class should have a method to calculate the price after the discount

Given Code	Expected Output

public class BookTester {
 public static void main(String[] args) {
System.out.println("Total Books Sold: " + Book.total_books_sold);
System.out.println("Total Revenue: "+Book.total_revenue + " TK");
System.out.println("==========1==========");
 	
Book b1 = new Book("Java Programming", 10); // 10% discount
b1.bookDetails();
 	
System.out.println("==========2==========");
 	
Book b2 = new Book("Python Programming", 15); // 15% discount
b2.bookDetails();
 	
System.out.println("==========3==========");
 	
Book b3 = new Book("Data Structures", 5); // 5% discount
b3.bookDetails();
 	
System.out.println("==========4==========");
System.out.println("Total Books Sold: " + Book.total_books_sold);
System.out.println("Total Revenue: "+Book.total_revenue + " TK");
}
}
	Total Books Sold: 0
	Total Revenue: 0.0 TK
	==========1==========
	Title: Java Programming
	Price after Discount: 135.0 TK
	==========2==========
	Title: Python Programming
	Price after Discount: 127.5 TK
	==========3==========
	Title: Data Structures
	Price after Discount: 142.5 TK
	==========4==========
	Total Books Sold: 3
	Total Revenue: 405.0 TK

Task 4

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 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) {
  System.out.println("Total Circle: "+ Circle.count);
  Circle c1 = new Circle(4);
  System.out.println("1---------------");
  System.out.println("Total Circle: "+ Circle.count);
  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("Total Circle: "+ Circle.count);
  System.out.println("Second circle radius: " + c2.getRadius());
  System.out.println("Second circle area: " + c2.area());
  System.out.println("3---------------");
  }
}
	Total Circle: 0
	1---------------
	Total Circle: 1
	First circle radius: 4.0
	First circle area:
	50.26548245743669
	2---------------
	Total Circle: 2
	Second circle radius: 5.0
	Second circle area:
	78.53981633974483
	3---------------

Task 5

Suppose you have opened a new library, from where your friends can borrow books. Initially you have bought 3 books (Pather Panchali, Durgesh Nandini & Anandmath) each of 3 copies only. Design the **Borrower** class in such a way that the following code provides the expected output.

- You are given the arrays ***book_count*** and ***book_name*** to keep track of the number of books available. For simplicity, assume that there will be no other books in the library.
- You must reuse the ***remainingBooks()*** method when needed.

Given Code	Expected Output

public class Tester{
  public static void main(String args[]){
  Borrower.bookStatus();
  System.out.println("*********1*********");
  Borrower b1 = new Borrower("Nabila");
  b1.borrowBook("Pather Panchali");
  b1.borrowBook("Anandmath");
  b1.borrowerDetails();
  System.out.println("*********2*********");
  Borrower b2 = new Borrower("Sadia");
  b2.borrowBook("Anandmath");
  b2.borrowBook("Durgesh Nandini");
  b2.borrowBook("Pather Panchali");
  b2.borrowerDetails();
  System.out.println("*********3*********");
 	
System.out.println(Borrower.remainingBooks("Anandmath")+"
copies of Anandmath is remaining.");
  System.out.println("*********4*********");
  Borrower b3 = new Borrower("Anika");
  b3.borrowBook("Anandmath");
  Borrower.bookStatus();
  System.out.println("*********5*********");
  Borrower b4 = new Borrower("Oishi");
  b4.borrowBook("Anandmath");
  b4.borrowBook("Durgesh Nandini");
  b4.borrowerDetails();
  }
}
 	
public class Borrower{
  public static int book_count[] = {3, 3, 3};
  public static String book_name[] = {"Pather Panchali",
"Durgesh Nandini", "Anandmath"};
 	
  // Your Code here
}
	Available Books:
	Pather Panchali: 3
	Durgesh Nandini: 3
	Anandmath: 3
	*********1*********
	Name: Nabila
	Books Borrowed:
	Pather Panchali
	Anandmath
	*********2*********
	Name: Sadia
	Books Borrowed:
	Anandmath
	Durgesh Nandini
	Pather Panchali
	*********3*********
	1 copies of Anandmath is
	remaining.
	*********4*********
	Available Books:
	Pather Panchali: 1
	Durgesh Nandini: 2
	Anandmath: 0
	*********5*********
	This book is not available.
	Name: Oishi
	Books Borrowed:
	Durgesh Nandini

Task 6

For this task, you need to design the **Cargo** class with appropriate static and non-static variables and methods to produce this given output for the given tester code.

Note: .load() method marks an object as selected for transport, and .unload() method unmarked it. At a time, the transport capacity is 10.0 Tonnes. Each Cargo object is initialized with 2 attributes from the constructor - the contents and the weight. Carefully observe the outputs to identify the other attributes and design the class.

Given Code	Expected Output

public class CargoTester {
  public static void main(String[] args) {
  System.out.println("Cargo Capacity: "+ Cargo.capacity());
  System.out.println("1====================");
  Cargo a = new Cargo("Industrial Machinery", 4.5);
  a.details();
  System.out.println("2====================");
  a.load();
  System.out.println("3====================");
  Cargo b = new Cargo("Steel Ingot", 2.7);
  b.details();
  System.out.println("4====================");
  System.out.println("Cargo Capacity: "+ Cargo.capacity());
  System.out.println("5====================");
  b.load();
  System.out.println("Cargo Capacity: "+ Cargo.capacity());
  System.out.println("6====================");
  Cargo c = new Cargo("Tree Trunks", 3.6);
  c.load();
  System.out.println("7====================");
  c.details();
  b.details();
  System.out.println("8====================");
  Cargo d = new Cargo("Processed Goods", 1.8);
  d.load();
  System.out.println("Cargo Capacity: "+ Cargo.capacity());
  System.out.println("9====================");
  b.unload();
  System.out.println("Cargo Capacity: "+ Cargo.capacity());
  System.out.println("10====================");
  c.load();
  System.out.println("11====================");
  b.details();
  System.out.println("Cargo Capacity: "+ Cargo.capacity());
  }
}
	Cargo Capacity: 10.0
	1====================
	Cargo ID: 1, Contents: Industrial Machinery, Weight: 4.5, Loaded: false
	2====================
	Cargo 1 loaded for transport.
	3====================
	Cargo ID: 2, Contents: Steel Ingot, Weight: 2.7, Loaded: false
	4====================
	Cargo Capacity: 5.5
	5====================
	Cargo 2 loaded for transport.
	Cargo Capacity: 2.8
	6====================
	Cannot load cargo, exceeds weight capacity.
	7====================
	Cargo ID: 3, Contents: Tree Trunks, Weight: 3.6, Loaded: false
	Cargo ID: 2, Contents: Steel Ingot, Weight: 2.7, Loaded: true
	8====================
	Cargo 4 loaded for transport.
	Cargo Capacity: 1.0
	9====================
	Cargo 2 unloaded.
	Cargo Capacity: 3.7
	10====================
	Cargo 3 loaded for transport.
	11====================
	Cargo ID: 2, Contents: Steel Ingot, Weight: 2.7, Loaded: false
	Cargo Capacity: 0.09999999999999964

Task 7

Design a **Student** class in such a way that the following code provides the expected output.

Driver Code	Output

public class StudentTester {
  public static void main(String[] args) {
  Student.printDetails();
  System.out.println("--------------------");
  Student mikasa = new Student("Mikasa", 3.75);
  mikasa.individualDetail();
  System.out.println("--------------------");
  Student.printDetails();
  System.out.println("--------------------");
  Student harry = new Student("Harry", 2.5, "Charms");
  harry.individualDetail();
  System.out.println("--------------------");
  Student.printDetails();
  System.out.println("--------------------");
  Student levi = new Student("Levi", 3.33);
  levi.individualDetail();
  System.out.println("--------------------");
  Student.printDetails();
  }
}
	Total Student(s): 0
	CSE Student(s): 0
	Other Department Student(s): 0
	-------------------- 	
	ID: 1
	Name: Mikasa
	CGPA: 3.75
	Department: CSE
	-------------------- 	
	Total Student(s): 1
	CSE Student(s): 1
	Other Department Student(s): 0
	-------------------- 	
	ID: 2
	Name: Harry
	CGPA: 2.5
	Department: Charms
	-------------------- 	
	Total Student(s): 2
	CSE Student(s): 1
	Other Department Student(s): 1
	-------------------- 	
	ID: 3
	Name: Levi
	CGPA: 3.33
	Department: CSE
	-------------------- 	
	Total Student(s): 3
	CSE Student(s): 2
	Other Department Student(s): 1

Task 8

Design the Player class with the necessary property to produce the output from the given driver code. **Hint: The total number of players is maximum 11**

Driver Code	Output

public class PlayerTester{
  public static void main(String[] args) {
  System.out.println("Total number of players: " + Player.total);
  System.out.println("1------------------");
  Player p1 = new Player("Neymar", "Brazil",5);
  System.out.println(p1.player_detail());
  System.out.println("====================");
  Player.info();
  System.out.println("2------------------");
  Player p2 = new Player("Ronaldo", "Portugal", 7);
  System.out.println(p2.player_detail());
  System.out.println("====================");
  Player.info();
  System.out.println("3------------------");
  Player p3 = new Player("Messi", "Argentina", 6);
  System.out.println(p3.player_detail());
  System.out.println("====================");
  Player.info();
  System.out.println("4------------------");
  Player p4 = new Player("Mbappe", "France", 10);
  System.out.println(p4.player_detail());
  System.out.println("====================");
  Player.info();
  }
}
	Total number of players: 0
	1--------------------------
	Player Name: Neymar
	Jersey Number: 5
	Country: Brazil
	========================
	Total number of players: 1
	Players enlisted so far: Neymar
	2--------------------------
	Player Name: Ronaldo
	Jersey Number: 7
	Country: Portugal
	========================
	Total number of players: 2
	Players enlisted so far: Neymar, Ronaldo
	3--------------------------
	Player Name: Messi
	Jersey Number: 6
	Country: Argentina
	========================
	Total number of players: 3
	Players enlisted so far: Neymar, Ronaldo, Messi
	4--------------------------
	Player Name: Mbappe
	Jersey Number: 10
	Country: France
	========================
	Total number of players: 4
	Players enlisted so far: Neymar, Ronaldo, Messi, Mbappe

Task 9

		Output		
1.	**public class Tracing {**			
2.	public static int x= 0, y = 0;			
3.	public int a, b;			
4.	public Tracing(int a, int b){			
5.	this.a = a;			
6.	this.b = b;			
7.	x+=1;			
8.	y+=2;			
9.	}			
10.	public void methodA(int a){			
11.	this.a = x+a;			
12.	this.b = this.b+ this.a +this.methodB();			
13.	System.out.println(this.a+" "+this.b+" "+x);			
14.	}			
15.	public int methodB(){			
16.	this.b = y - this.b + this.a;			
17.	System.out.println(this.a+" "+this.b+" "+x);			
18.	x += this.b;			
19.	return this.b;			
20.	}			
21.	public void methodB(Tracing t1){			
22.	t1.b = this.y - t1.b + this.b;			
23.	System.out.println(t1.a+" "+t1.b+" "+x);			
24.	}			
25.	}			
26.	**public class Test9{**			
27.	public static void main(String [] args){			
28.	Tracing t1= new Tracing(2, 3);			
29.	t1.methodA(1);			
30.	Tracing t2= new Tracing(3, 4);			
31.	t2.methodA(2);			
32.	t1.methodB(t2);			
33.	t2.methodB(t2);			
34.	}			
35.	}			

Task 10

		Outputs		
1	**public class FinalT6A{**			
2	public static int temp = 3;			
3	public int sum;			
4	public int y = 2;			
5	public FinalT6A(int x, int p){			
6	temp+=3;			
7	y = temp - p;			
8	sum = FinalT6A.temp + x;			
9	System.out.println(x + " " + y+ " " + sum);			
10	}			
11	public void methodA(){			
12	int x=0, y =0;			
13	y = y + this.y;			
14	x = this.y + 2 + temp;			
15	sum = x + y + methodB(temp, y);			
16	System.out.println(x + " " + y+ " " + sum);			
17	}			
18	public int methodB(int temp, int n){			
19	int x = 0;			
20	y = y + (++temp);			
21	x = x + 2 + n;			
22	sum = sum + x + y;			
23	System.out.println(x + " " + y+ " " + sum);			
24	return sum;			
25	}			
26	}			
27	**public class Test10{**			
28	public static void main(String [] args){			
29	FinalT6A q1 = new FinalT6A(2,1);			
30	q1.methodA();			
31	q1.methodA();			
32	}			
33	}			

Task 11

```
1  public class B{
2    public static int x;
3    public int y = 4;
4    public int temp = -5;
5    public int sum = 2;
6    public B(){
7      y = temp + 3 ;
8      sum = 3 + temp + 3;
9      temp-=2;
10   }
11   public B(B b){
12     sum = b.sum;
13     x = b.x;
14     b.methodB(1,3);//
15   }
16    public void methodA(int m, int n){
17     int x = 2;
18     y = y + m + (temp++);
19     x = x + 7 +  n;
20     sum = sum + x + y;
21     System.out.println(x + " " + y+ " " + sum);
22   }
23   public void methodB(int m, int n){
24     int  y = 0;
25     y = y + this.y;
26     x = this.y + 3 + temp;
27     methodA(x, y);
28     sum = x + y + sum;
29     System.out.println(x + " " + y+ " " + sum);
30   }
31 }
```

Consider the following code:

```
B b1 = new B();
B b2 = new B(b1);
b1.methodA(3, 2);
b2.methodB(1, 2);
```

x	y	sum

Ungraded Tasks (Optional)

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

Task 1

Design the **SultansDine** class with the necessary property to produce the output from the given driver code.

Subtaks:

1. Create SultansDine class
2. Create 2 static variable and 1 static array
3. Create 1 static method
4. Calculation of branch sell is given below
    a. If sellQuantity < 10:
        i. Branch_sell = quantity * 300
    b. Else if sellQuantity < 20:
        i. Branch_sell = quantity * 350
    c. Else
        i. Branch_sell = quantity * 400
5. Calculation of branch's sell percentage = (branch's sell / total sell) * 100

Driver Code	Output

public class SultansDineTester {
  public static void main(String[] args) {
  SultansDine.details();
  System.out.println("1==========");
  SultansDine dhanmondi = new SultansDine("Dhanmondi");
  dhanmondi.sellQuantity(25);
  dhanmondi.branchInformation();
  System.out.println("2==========");
  SultansDine.details();
  System.out.println("3==========");
  SultansDine baily_road = new SultansDine("Baily Road");
  baily_road.sellQuantity(15);
  baily_road.branchInformation();
  System.out.println("4==========");
  SultansDine.details();
  System.out.println("5==========");
  SultansDine gulshan = new SultansDine("Gulshan");
  gulshan.sellQuantity(9);
  gulshan.branchInformation();
  System.out.println("6==========");
  SultansDine.details();
  }
}
	Total Number of branch(s): 0
	Total Sell: 0 Taka
	1==========
	Branch Name: Dhanmondi
	Branch Sell: 10000 Taka
	2==========
	Total Number of branch(s): 1
	Total Sell: 10000 Taka
	Branch Name: Dhanmondi, Branch Sell: 10000 Taka
	Branch consists of total sell's 100.00
	3==========
	Branch Name: Baily Road
	Branch Sell: 5250 Taka
	4==========
	Total Number of branch(s): 2
	Total Sell: 15250 Taka
	Branch Name: Dhanmondi, Branch Sell: 10000 Taka
	Branch consists of total sell's 65.57
	Branch Name: Baily Road, Branch Sell: 5250 Taka
	Branch consists of total sell's 34.43
	5==========
	Branch Name: Gulshan
	Branch Sell: 2700 Taka
	6==========
	Total Number of branch(s): 3
	Total Sell: 17950 Taka
	Branch Name: Dhanmondi, Branch Sell: 10000 Taka
	Branch consists of total sell's 55.71

	Branch Name: Baily Road, Branch Sell: 5250 Taka Branch consists of total sell's 29.25 Branch Name: Gulshan, Branch Sell: 2700 Taka Branch consists of total sell's 15.04

Task 2

Implement the design of the **Travel** class so that the following output is produced. Use Encapsulation to protect the variables. [Hint: Assign all the variables in private]

Driver Code	Output

public class TravelTester {
 public static void main(String[] args) {
 System.out.println("No. of Traveller = " + Travel.getCount());
 System.out.println("1===============");
 	
 Travel t1 = new Travel("Dhaka", "India");
 System.out.println(t1.displayTravelInfo());
 System.out.println("2===============");
 	
 Travel t2 = new Travel("Kuala Lampur", "Dhaka");
 t2.setTime(23);
 System.out.println(t2.displayTravelInfo());
 System.out.println("3===============");
 	
 Travel t3 = new Travel("Dhaka", "New_Zealand");
 t3.setTime(15);
 t3.setDestination("Germany");
 System.out.println(t3.displayTravelInfo());
 System.out.println("4===============");
 	
 Travel t4 = new Travel("Dhaka", "India");
 t4.setTime(9);
 t4.setSource("Malaysia");
 t4.setDestination("Canada");
 System.out.println(t4.displayTravelInfo());
 System.out.println("5===============");
 	
 System.out.println("No. of Traveller = " + Travel.getCount());
 }
}
	No. of Traveller = 0
	1===============
	Source: Dhaka
	Destination: India
	Flight Time: 1:00
	2===============
	Source: Kuala Lumpur
	Destination: Dhaka
	Flight Time: 23:00
	3===============
	Source: Dhaka
	Destination: Germany
	Flight Time: 15:00
	4===============
	Source: Malaysia
	Destination: Canada
	Flight Time: 9:00
	5===============
	No. of Traveller = 4

Task 3

		Output	
1.	**public class Maze{**		
2.	public static int x;		
3.	public void methodA(){		
4.	int m = 5;		
5.	x=11;		
6.	System.out.println(x+" "+m);		
7.	m=methodB(m-3)+x;		
8.	System.out.println(x+" "+(m));		
9.	methodB(x,m);		
10.	System.out.println(x+" "+m+x);		
11.	}		
12.	public int methodB(int y){		
13.	x=y*y;		
14.	System.out.println(x+" "+y);		
15.	return x+3;		
16.	}		
17.	public void methodB(int z, int x){		
18.	z=z-2;		
19.	x=x*1%z;		
20.	System.out.println(z+" "+x);		
21.	}		
22.	}		
23.	**public class TestU3{**		
24.	public static void main(String [] args){		
25.	Maze c = new Maze();		
26.	c.methodA();		
27.	c.methodB(-11, 45);		
28.	}		
29.	}		

Task 4

Find the outputs after running the main() method in **Test11** class.

		Outputs		
1	**public class Quiz1{**			
2	public static int temp = 4;			
3	public int sum;			
4	public int y;			
5	public Quiz1(){			
6	y = temp - 1;			
7	sum = temp + 1;			
8	temp+=2;			
9	}			
10	public Quiz1(int p){			
11	y = temp + p ;			
12	sum = p + temp + 1;			
13	temp-=1;			
14	}			
15	public void methodA(){			
16	int x=0, y =0;			
17	y = y + this.y;			
18	x = this.y + 2 + temp;			
19	sum = x + y + methodB(x, y);			
20	System.out.println(x + " " + y+ " " + sum);			
21	}			
22	public int methodB(int m, int n){			
23	int x = 0;			
24	y = y + m + (++temp);			
25	x = x + 2 + n;			
26	sum = sum + x + y;			
27	System.out.println(x + " " + y+ " " + sum);			
28	return sum;			
29	}			
30	}			
31	**public class TestU4{**			
32	public static void main(String [] args){			
33	Quiz1 q1 = new Quiz1();			
34	q1.methodA();			
35	q1.methodA();			
36	Quiz1.temp+= 2;			
37	Quiz1 q2 = new Quiz1(2);			

```
38          q2.methodA();
39          q2.methodA();
40      }
41  }
```