Lab Assignment 08



Inspiring Excellence

Course Code:	CSE111
Course Title:	Programming Language II
Topic:	Static Variable & Static Method
Number of Tasks:	11

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

Task 1

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{
                                                                  Total Passenger: 0
 public static void main(String args[]){
                                                                  Total Fare: 0.0 TK
   System.out.println("Total Passenger: "+ Passenger.no_of_passenger);
                                                                  System.out.println("Total Fare: "+ Passenger.total_fare + " TK");
                                                                  Name: Lara
   System.out.println("=======1======");
                                                                  Fare: 112.0 TK
   Passenger p1 = new Passenger("Lara", 5.6);
                                                                  p1.passengerDetails();
                                                                  Name: Kevin
   System.out.println("======2======");
                                                                  Fare: 268.0 TK
   Passenger p2 = new Passenger("Kevin", 10.0);
                                                                  p2.setBaggageWeight(6.8);
                                                                  Name: Robin
   p2.passengerDetails();
                                                                  Fare: 96.0 TK
   System.out.println("======3======");
                                                                  Passenger p3 = new Passenger("Robin", 2.3);
                                                                  Total Passenger: 3
                                                                  Total Fare: 476.0 TK
   p3.setBaggageWeight(5);
   p3.passengerDetails();
   System.out.println("=======4======"):
   System.out.println("Total Passenger: "+ Passenger.no_of_passenger);
   System.out.println("Total Fare: "+ Passenger.total_fare + " TK");
 }
}
```

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
<pre>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("====================================</pre>	Total Books Sold: 0 Total Revenue: 0.0 TK ====================================

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

Driver Code	Output
<pre>public class StudentTester { public static void main(String[] args) { Student.printDetails(); System out println("</pre>	Total Student(s): 0 CSE Student(s): 0 Other Department Student(s): 0
<pre>System.out.println(""); Student mikasa = new Student("Mikasa", 3.75); mikasa.individualDetail(); System.out.println(""); Student.printDetails(); System.out.println("");</pre>	ID: 1 Name: Mikasa CGPA: 3.75 Department: CSE
Student harry = Student.createStudent("Harry", 2.5, "Charms"); harry.individualDetail(); System.out.println("");	Total Student(s): 1 CSE Student(s): 1 Other Department Student(s): 0
<pre>System.out.println(</pre>	ID: 2 Name: Harry CGPA: 2.5 Department: Charms
Student.printDetails(); } }	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

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
<pre>public class Tester{ public static void main(String args[]){ Borrower.bookStatus(); System.out.println("*************************); Borrower b1 = new Borrower("Nabila"); b1.borrowBook("Pather Panchali"); b1.borrowBook("Anandmath"); b1.borrowBook("Anandmath"); b1.borrowBook("Anandmath"); b2.borrowBook("Anandmath"); b2.borrowBook("Durgesh Nandini"); b2.borrowBook("Pather Panchali"); b2.borrowBook("Pather Panchali"); b2.borrowBook("Pather Panchali"); b2.borrowerDetails(); System.out.println("************************************</pre>	Available Books: Pather Panchali: 3 Durgesh Nandini: 3 Anandmath: 3 **************** 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
<pre>public class Borrower{ public static int book_count[] = {3, 3, 3}; public static String book_name[] = {"Pather Panchali", "Durgesh Nandini", "Anandmath"}; // Your Code here }</pre>	

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 {
                                                          Cargo Capacity: 10.0
  public static void main(String[] args) {
                                                          1=========
   System.out.println("Cargo Capacity: " + Cargo.capacity());
                                                          Cargo ID: 1, Contents: Industrial
   System.out.println("1=======");
                                                          Machinery, Weight: 4.5, Loaded: false
   Cargo a = new Cargo("Industrial Machinery", 4.5);
                                                          2============
                                                          Cargo 1 loaded for transport.
   a.details();
   System.out.println("2======="):
                                                          3==========
                                                          Cargo ID: 2, Contents: Steel Ingot,
   a.load();
   System.out.println("3=======");
                                                          Weight: 2.7, Loaded: false
   Cargo b = new Cargo("Steel Ingot", 2.7);
                                                          4==========
   b.details();
                                                          Cargo Capacity: 5.5
   System.out.println("4=======");
                                                          5==========
   System.out.println("Cargo Capacity: " + Cargo.capacity());
System.out.println("5==========");
                                                          Cargo 2 loaded for transport.
                                                          Cargo Capacity: 2.8
                                                          6=========
   b.load();
   System.out.println("Cargo Capacity: " + Cargo.capacity());
                                                          Cannot load cargo, exceeds weight
   System.out.println("6=======");
                                                          capacity.
   Cargo c = new Cargo("Tree Trunks", 3.6);
                                                          7==========
                                                          Cargo ID: 3, Contents: Tree Trunks,
   c.load();
   System.out.println("7========");
                                                          Weight: 3.6, Loaded: false
   c.details();
                                                          Cargo ID: 2, Contents: Steel Ingot,
   b.details():
                                                          Weight: 2.7, Loaded: true
   System.out.println("8=======");
                                                          8=========
   Cargo d = new Cargo("Processed Goods", 1.8);
                                                          Cargo 4 loaded for transport.
   d.load();
                                                          Cargo Capacity: 1.0
   System.out.println("Cargo Capacity: " + Cargo.capacity());
                                                          9==========
   System.out.println("9=======");
                                                          Cargo 2 unloaded.
                                                          Cargo Capacity: 3.7
   System.out.println("Cargo Capacity: " + Cargo.capacity());
                                                          10========
   System.out.println("10========");
                                                          Cargo 3 loaded for transport.
   c.load();
                                                          11=========
   System.out.println("11========");
                                                          Cargo ID: 2, Contents: Steel Ingot,
                                                          Weight: 2.7, Loaded: false
   b.details();
   System.out.println("Cargo Capacity: " + Cargo.capacity());
                                                          Cargo Capacity: 0.099999999999964
 }
}
```

 $\underline{Task\ 6}$ Complete the class Circle so that the desired outputs are generated properly.

Given Code	Expected Output
<pre>public class shapeTester { public static void main(String[] args) { Circle c = new Circle(); Circle restanting[]</pre>	Name: Circle Color: Red
<pre>System.out.println("==========="); c.name = "Circle"; c.color = "Red"; c.radius = 5; diaployInfo();</pre>	Area of Red Circle: 78.54
<pre>c.displayInfo(); System.out.println("=========="); c.area(); } </pre>	
<pre>public class Shape { public String name; public String color;</pre>	
<pre>public void displayInfo() { System.out.printf("Name: %s\nColor: %s\n", name, color); } </pre>	
<pre>public class Circle extends Shape { //Your Code Here }</pre>	

 $\underline{Task\ 7}$ Complete the class Dog so that the desired outputs are generated properly.

Given Code	Expected Output
<pre>public class AnimalTester{ public static void main(String args[]){ Animal a1 = new Animal(); System.out.println("1"); a1.details(); System.out.println("2"); Dog d1 = new Dog(); d1.name = "Pammy"; System.out.println("3"); System.out.println("Name: " + d1.getName()); d1.details(); System.out.println("4"); d1.updateSound("Bark"); System.out.println("5"); d1.details(); } }</pre>	1 Legs: 4 Sound: Not defined 2 The dog says hello! 3 Name: Pammy Legs: 4 Sound: Not defined 4 5 Legs: 4 Sound: Bark
<pre>public class Animal{ public int legs = 4; public String sound = "Not defined"; public void details(){ System.out.println("Legs: "+legs); System.out.println("Sound: "+sound); } } public class Dog extends Animal{ //Your Code Here }</pre>	

1.	<pre>public class Maze{</pre>	Output
2.	<pre>public static int x;</pre>	
3.	<pre>public void methodA(){</pre>	
4.	int m = 5;	
5.	x=11;	
6.	<pre>System.out.println(x+" "+m);</pre>	
7.	<pre>m=methodB(m-3)+x;</pre>	
8.	<pre>System.out.println(x+" "+(m));</pre>	
9.	<pre>methodB(x,m);</pre>	
10.	<pre>System.out.println(x+" "+m+x);</pre>	
11.	}	
12.	<pre>public int methodB(int y){</pre>	
13.	x=y*y;	
14.	<pre>System.out.println(x+" "+y);</pre>	
15.	return x+3;	
16.	}	
17.	<pre>public void methodB(int z, int x){</pre>	
18.	z=z-2;	
19.	x=x*1%z;	
20.	<pre>System.out.println(z+" "+x);</pre>	
21.	}	
22.	}	
23.	public class Test8{	
24.	<pre>public static void main(String [] args){</pre>	
25.	Maze c = new Maze();	
26.	c.methodA();	
27.	c.methodB(-11, 45);	
28.	}	
29.	}	

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

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

 $\underline{Task\ 11}$ Find the outputs after running the main() method in Test11 class.

1	public class Quiz1{	Output	s
2	<pre>public static int temp = 4;</pre>		
3	public int sum;		
4	public int y;		
5	<pre>public Quiz1(){</pre>		
6	y = temp - 1;		
7	sum = temp + 1;		
8	temp+=2;		
9	}		
10	<pre>public Quiz1(int p){</pre>	-	
11	y = temp + p;		
12	sum = p + temp + 1;		
13	temp-=1;		
14	}		
15	<pre>public void methodA(){</pre>		
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	<pre>public int methodB(int m, int n){</pre>		
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	}		
—	public class Test11{		
32	<pre>public static void main(String [] args){</pre>		
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	}	