

Lab Assignment 06

HOMEWORK

Task 1

Design the **Product** class such that it produces the following output.

Tester Code	Output
<pre>public class ProductTester{ public static void main(String[] args) { Product p1 = new Product("Table", 10); Product p2 = new Product("Chair", 15); Product p3 = new Product("Sofa", 20); Product p4 = new Product("Divan", 8); System.out.println("-----1-----"); Product.displayProducts(); System.out.println("-----2-----"); Product.buy("Chair", 5); System.out.println("-----3-----"); Product.displayProducts(); System.out.println("-----4-----"); Product.buy("Sofa", 25); System.out.println("-----5-----"); Product.displayProducts(); System.out.println("-----6-----"); Product.buy("Bed", 10); } }</pre>	<pre>Stored: Table Stored: Chair Stored: Sofa Storage is full! Cannot add Divan -----1----- ==== Stored Products === Table - Qty: 10 Chair - Qty: 15 Sofa - Qty: 20 -----2----- Product Sold -----3----- ==== Stored Products === Table - Qty: 10 Chair - Qty: 10 Sofa - Qty: 20 -----4----- Quantity low -----5----- ==== Stored Products === Table - Qty: 10 Chair - Qty: 10 Sofa - Qty: 20 -----6----- Product not found</pre>

Task 2

Design the **Character** class such that it produces the following output.

Tester Code	Output
<pre>public class WeirderStuffTester { public static void main(String[] args) { Character.printStats(); System.out.println("-----1-----"); Character twelve = new Character("Twelve", "Kid", 100); twelve.printDetails(); System.out.println("-----2-----"); Character spike = new Character("Spike", "Kid", 50); spike.printDetails(); System.out.println("-----3-----"); Character.printStats(); System.out.println("-----4-----"); Character reeve = new Character("Reeve", 70); reeve.printDetails(); System.out.println("-----5-----"); Character chopper = new Character("Chopper", "Adult", 120); chopper.printDetails(); System.out.println("-----6-----"); Character.printStats(); } }</pre>	<pre>Total Characters: 0 Kids: 0 Teens: 0 Adults: 0 Average Health: 0 Strongest Character: None -----1----- ID: 1, Name: Twelve Group: Kid Health: 100 -----2----- ID: 2, Name: Spike Group: Kid Health: 50 -----3----- Total Characters: 2 Kids: 2 Teens: 0 Adults: 0 Average Health: 75.0 Strongest Character: Twelve (Health 100) -----4----- ID: 3, Name: Reeve Group: Teen Health: 70 -----5----- ID: 4, Name: Chopper Group: Adult Health: 120 -----6----- Total Characters: 4 Kids: 2 Teens: 1 Adults: 1 Average Health: 85.0 Strongest Character: Chopper (Health 120)</pre>

Task 3

Design the **Artifact** class where all the attributes of the class are classified i.e. private. The “Vault” can only store a maximum of 4 artifacts. The power of the artifacts are calculated as: For artifact with,

- Even length name => Summation of all characters in even index of name.
- Odd length name => Summation of all characters in odd index of name.

Tester Code	Output
<pre>public class TesterArtifact{ public static void main(String[] args) { Artifact a = new Artifact("Phone Microwave", "Kurisu"); System.out.println("-----1-----"); Artifact.AddtoVault(a); Artifact.AddtoVault(new Artifact("D-Mail Capsule", "Mayuri")); System.out.println("-----2-----"); Artifact c = new Artifact("C204 Chip"); Artifact d = new Artifact("Divergence Meter"); Artifact e = new Artifact("M4A2 Robot", "Okabe"); Artifact.AddtoVault(c); Artifact.AddtoVault(d); Artifact.AddtoVault(e); System.out.println("-----3-----"); Artifact.labReport(); System.out.println("-----4-----"); System.out.println("Power of "+c.GetName()+" is "+c.CalcPower()); System.out.println("-----5-----"); System.out.println("Strongest Artifact: "+Artifact.strongest()); System.out.println("-----6-----"); a.revealArtifact(); System.out.println("-----7-----"); a.changeName("Banana Microwave"); System.out.println("-----8-----"); Artifact.labReport(); System.out.println("-----9-----"); System.out.println("Strongest Artifact: "+Artifact.strongest()); } }</pre>	<pre>-----1----- Kurisu added Phone Microwave successfully to the vault. Mayuri added D-Mail Capsule successfully to the vault. -----2----- Okabe added C204 Chip successfully to the vault. Okabe added Divergence Meter successfully to the vault. !!Okabe unsuccessful in adding artifact to the vault!! -----3----- == Future Gadget Lab == Phone Microwave added by Kurisu has power of 702. D-Mail Capsule added by Mayuri has power of 602. C204 Chip added by Okabe has power of 274. Divergence Meter added by Okabe has power of 734. -----4----- Power of C204 Chip is 274 -----5----- Strongest Artifact: Divergence Meter -----6----- Phone Microwave added by Kurisu has power of 702. -----7----- Name changed and power recalculated. -----8----- == Future Gadget Lab == Banana Microwave added by Kurisu has power of 774. D-Mail Capsule added by Mayuri has power of 602. C204 Chip added by Okabe has power of 274. Divergence Meter added by Okabe has power of 734. -----9----- Strongest Artifact: Banana Microwave</pre>

Task 4

Design the **AnimalKeepers** class with the following requirements:

- The Animal Keepers have private IDs starting from 101.
- The Safari has an array named, Animals = {"Lion", "Tiger", "Seal", "Gorilla", "Deer"}
- Only one task is assigned per animal and so the tasks are overridden when reassigned for the same animal.

[Hint: You can call the static method printTasks() from inside details()]

Tester Code	Output
<pre>import java.util.Arrays; public class KeeperTester { public static void main(String[] args) { Animalkeepers.details(); System.out.println("-----1-----"); System.out.println(Arrays.toString(Animalkeepers.Animals)); System.out.println("-----2-----"); Animalkeepers leo = new Animalkeepers("Leo"); Animalkeepers theo = new Animalkeepers("Theo"); Animalkeepers mochi = new Animalkeepers("Mochi"); System.out.println("-----3-----"); Animalkeepers.printTasks(); System.out.println("-----4-----"); leo.doTask("Lion", "Feed"); System.out.println("-----5-----"); leo.doTask("Monkey", "Feed"); System.out.println("-----6-----"); Animalkeepers.details(); System.out.println("-----7-----"); theo.doTask("Tiger", "Bathe"); mochi.doTask("Seal", "Clean Pen"); mochi.doTask("Deer", "Add Food"); System.out.println("-----8-----"); Animalkeepers.printTasks(); System.out.println("-----9-----"); leo.doTask("Deer", "Play"); System.out.println("-----10-----"); Animalkeepers.details(); } }</pre>	<pre>No Animal Keepers working yet. -----1----- [Lion, Tiger, Seal, Gorilla, Deer] -----2----- Leo with ID 101 got the job! Theo with ID 102 got the job! Mochi with ID 103 got the job! -----3----- No tasks assigned. -----4----- Task assigned to Leo -----5----- Animal not in the Safari -----6----- Total Animal Keeper: 3 Total Task assigned: 1 Feed (Keeper - Leo) === Lion -----7----- Task assigned to Theo Task assigned to Mochi Task assigned to Mochi -----8----- Feed (Keeper - Leo) === Lion Bathe (Keeper - Theo) === Tiger Clean Pen (Keeper - Mochi) === Seal Add Food (Keeper - Mochi) === Deer -----9----- Task assigned to Leo -----10----- Total Animal Keeper: 3 Total Task assigned: 4 Feed (Keeper - Leo) === Lion Bathe (Keeper - Theo) === Tiger Clean Pen (Keeper - Mochi) === Seal Play (Keeper - Leo) === Deer</pre>

Task 5

Design the **Event** and **Organizer** classes in such a way that the following code provides the expected output. Hint:

- Make the name instance variable of the Event class **private**
- For simplicity assume that the Event class can create a maximum of 5 event objects and an Organizer can organize a maximum of 4 events.

Driver Code	Output
<pre>public class EventTester{ public static void main(String args[]){ Event.allEventInfo(); System.out.println("1-----"); Event ev1 = new Event("HP Day", "7/12/24"); Event ev2 = new Event("TechConnect", "10/12/24"); System.out.println(ev1.details()); System.out.println("2-----"); Organizer uni = new Organizer(); Organizer bracu = new Organizer("BRACU"); Organizer buet = new Organizer("BUET"); System.out.println("3-----"); Event.allEventInfo(); System.out.println("4-----"); bracu.organizeEvent(ev1); bracu.organizeEvent(ev2); System.out.println("5-----"); Event ev3 = new Event("From Earth to Orbit", "15/12/24"); Event ev4 = new Event("NSysS 2024", "21/12/24"); System.out.println("6-----"); buet.organizeEvent(ev4); bracu.organizeEvent(ev3); System.out.println("7-----"); bracu.searchEventByDate("21/12/24"); System.out.println("8-----"); bracu.searchEventByDate("15/12/24"); System.out.println("9-----"); Event.allEventInfo(); } }</pre>	<pre>Total Events: 0 Event Details: 1----- Name: HP Day Date: 7/12/24 2----- Please provide the organizer's name 3----- Total Events: 2 Event Details: Event 1: Name: HP Day Date: 7/12/24 Event 2: Name: TechConnect Date: 10/12/24 4----- BRACU successfully organized HP Day BRACU successfully organized TechConnect 5----- 6----- BUET successfully organized NSysS 2024 BRACU successfully organized From Earth to Orbit 7----- No event is scheduled for 21/12/24 8----- From Earth to Orbit 9----- Total Events: 4 Event Details: Event 1: Name: HP Day Date: 7/12/24 Event 2: Name: TechConnect Date: 10/12/24 Event 3: Name: From Earth to Orbit Date: 15/12/24 Event 4: Name: NSysS 2024 Date: 21/12/24</pre>

Task 6

```
1 class Trace {  
2     public static int[] x = {3, -4};  
3     public int y = 4;  
4     public static int temp = -5;  
5     private int sum = 2;  
6     public Trace(){  
7         y = temp + 3 ;  
8         sum = 3 + temp + x[1];  
9         temp-=2;  
10        x[0] = ++x[1] - 2;  
11    }  
12    public Trace(Trace trace){  
13        sum = trace.sum;  
14        x = trace.x;  
15        trace.methodB(1,3);  
16    }  
17    public void methodA(int m, int n){  
18        int x = 2 - this.x[0] - Trace.x[1];  
19        y = y + m + (temp++);  
20        x = x + 7 + n;  
21        sum = sum + x + y;  
22        System.out.println(x + " " + y+ " " + sum);  
23    }  
24    public void methodB(int m, int n){  
25        int y = 0;  
26        y = y + this.y;  
27        Trace.x[0] = this.y + 3 + temp;  
28        methodA(x[1], y);  
29        sum = Trace.x[1] + y + sum;  
30        System.out.println(this.x[0] + " " + y+ " " + sum);  
31    }  
32    public static void methodC(Trace trace1, Trace trace2){  
33        temp = x[0] - Trace.x[1];  
34        x = new int[]{trace1.y, trace2.y};  
35    }  
36}
```

Consider the following driver code and find the output.

```
Trace trace1 = new Trace();  
Trace trace2 = new Trace(trace1);  
trace1.methodA(3, 2);  
Trace.methodC(trace1, trace2);  
trace2.methodB(1, 2);
```

	Output 1	Output 2	Output 3

Task 7

```
1 class Tracing {  
2     public static int x = 0, y = 0;  
3     public int a;  
4     private int b = 3;  
5     public Tracing(int a, int b) {  
6         this.a = a;  
7         this.b = b - this.b;  
8         x += 1;  
9         y += Tracing.y - 2;  
10    }  
11    public void set_b(int b) {  
12        this.b = b;  
13    }  
14    public int get_b() {  
15        return this.b;  
16    }  
17    public void methodA(int x) {  
18        this.a = x + this.x - Tracing.x;  
19        this.b = this.a + this.methodB() - this.b;  
20        System.out.println(this.a + " " + this.b + " " + x);  
21    }  
22    public int methodB() {  
23        int y = -3;  
24        this.b = y - this.y + this.a;  
25        System.out.println(this.a + " " + this.b + " " + x);  
26        this.y -= y;  
27        x += this.b + this.y;  
28        return this.b;  
29    }  
30    public void methodB(Tracing t1) {  
31        int t = this.y - t1.get_b() + this.b;  
32        t1.set_b(t);  
33        t1.a = this.x - t1.a + this.a;  
34        System.out.println(t1.a + " " + t1.get_b() + " " + x);  
35    }  
36 }
```

Consider the following driver code and find the output.

```
Tracing t1 = new Tracing(2, 3);  
t1.methodA(1);  
Tracing t2 = new Tracing(3, 4);  
t2.methodA(2);  
t1.methodB(t2);  
t2.methodB(t2);
```

	Output 1	Output 2	Output 3