Sample Problems OOP – Java

1. Define a class 'Airplane'
   Define the following member variables:
   - model (String : public): represents the model of the airplane.
   - year (int : public): represents the manufacturing year of the airplane.
   - seatingCapacity (int : private): represents the maximum seating capacity of the airplane.
   - unit_cost (double : private) – represents the cost of each seat
   - availableSeats (int []: private) – an array to keep track which seat is available or not

   Define the following methods:

   - A constructor to initialize all the member variables
   - Another constructor to initialize only the "model", "year" and "seatingCapacity"
   - Necessary getter and setter methods for private variables
   - Booking(n:int, luggage:int) : This method will check if n consecutive seats are available or not.
     - If n consecutive seats are available then book those seats sequentially and mark as unavailable. A passenger can take at most 3 luggage. For each extra luggage 600tk will be added. Then compute the total cost and print the total cost.
     - Otherwise print "seats are not available".

   In main method of "MyBooking" class,

- Create an object of Airplane with the following info.
  (model: "boeing747", year=2012, seatingCapacity: 30)
- Set unit cost = 12000
- Call "Booking" method to book 5 seats and 5 luggage

……..

2. Implement the "Calculator" class so that the expected output is produced when the following code is executed.

```
class TestCalc{
   public static void main(String[] args) {
      Calculator c1=new Calculator(3,2);
      System.out.println(c1.getFactorial());
      System.out.println(c1.getFactorial(4));
      System.out.println(c1.pow());
      System.out.println(c1.pow(4,2));
      System.out.println(c1.multiply());
      System.out.println(c1.multiply(5,6));
   }
}
```

Expected output:

```
Factorial method with no arg
6
Factorial method with 1 arg
24
```

```
power method with no arg
9
pow method for 2 arg
16
multiple method with no arg
6
multiple method with 2 arg
30
```

……..

3. Write a program which will take employee information of an office. Where the number of employees will be provided by console as **n**. After this number, you will get names and ids and salaries of employees. Create a class **Employee** and store the information received from the console in an array of employees. Now, traverse and find the total number of employees who have a salary of at least 12000 or more, also print the maximum employee salary. Please check the following sample input/output along with the explanation.

Input	Output	Explanation
4 Forhad Hossain FH-101 10000 Emran Ahmed EA-102 20000 Riad Hasan RdH-103 15500 Sadia Tabassum SaTb-104 21000	Count: 3 Max = 21000	Emran, Riad and Sadia Have salaries >= 12000 and maximum salary is 21000

……..

4. We need to compute the efficiency percentage based on three efficiency ratings (each out of 100) for Machine A and four efficiency ratings (each out of 100) for Machine B. Create an abstract class 'Efficiency' with an abstract method 'getEfficiencyPercentage'. This abstract class is inherited by two subclasses, 'MachineA' and 'MachineB', each having a method with the same name that returns the efficiency percentage for the respective machine. The constructor of MachineA takes the efficiency ratings for three periods as its parameters, and the constructor of MachineB takes the efficiency ratings for four periods. Create an instance for each of the two classes and print the efficiency percentage for both machines.

Efficiency percentage for three periods: (totalEfficiency / (3 * 100)) * 100

Efficiency percentage for four periods: (totalEfficiency / (4 * 100)) * 100

`

```
public class Main {
    public static void main(String[] args) {
        // Create objects for MachineA and MachineB
        MachineA machineA = new MachineA(85, 88, 90);
        MachineB machineB = new MachineB(92, 89, 90, 87);

        // Calculate and print the efficiency percentage for both machines
        System.out.println("Efficiency percentage for MachineA: " + machineA.getEfficiencyPercentage() + "%");
        System.out.println("Efficiency percentage for MachineB: " + machineB.getEfficiencyPercentage() + "%");
    }
}
```

Sample Output:

Efficiency percentage for MachineA: _____

Efficiency percentage for MachineB: _____

……..

5. You have been engaged by an educational software company to develop a program that simulates a school management system. This program should allow users to manage various types of people in the school, such as students, teachers, and staff. Your task is to design and implement the necessary classes to represent these individuals and their behaviors.

Design a base class named Person with the following attributes:

- **name** (string): to store the person's name.

- **age** (integer): to store the person's age.

Ensure these attributes are encapsulated using appropriate access modifiers for data security. Implement the following methods in the Person class:

- **attend()**: void - This method should display a message indicating that the person is attending the school.

- **leave()**: void - This method should display a message indicating that the person is leaving the school.

Create three derived classes: Student, Teacher, and Staff, which inherit from the Person class. Each derived class should include specific attributes and methods:

- **Student**:

  - **Attribute**: gradeLevel (integer) - to store the student's grade level.

  - **Method**: Override the attend() method to display a message indicating that the student is attending a class.

- **Teacher**:

  - **Attribute**: subject (string) - to store the subject the teacher teaches.

  - **Method**: Overload the leave() method with a parameterized version that takes an additional argument representing the reason for leaving (e.g., meeting, end of day).

- **Staff**:

  - **Attribute**: position (string) - to store the staff member's position.

  - **Method**: Implement a new method called assist() that displays a message indicating that the staff member is assisting with school operations.

Inside the main method (Main Class), create instances of each derived class to represent different individuals in the school. Set the values of the attributes accordingly. Demonstrate method overloading by calling the leave() method on a teacher instance with different arguments (e.g., meeting, end of day) to observe the different leave reasons. Demonstrate method overriding by calling the attend() method on a student instance and observing the overridden behavior. Call the assist() method on a staff instance to demonstrate the unique behavior implemented in the Staff class.

……..

6. Write a java in your IDE code to:
    a. Create an odd sized double array.
    b. Initialize the array by taking inputs from the user (using Scanner)
    c. swap in the odd indices of the array. For example swap the values between index 1, index 3, and index 5 with index 7 etc.

……..

7. Create a class named 'Shape' with a method to print "This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.

……..

8. Create the Animal class, which is the abstract superclass of all animals.

    a. • Declare a public integer attribute called legs, which records the number of legs for this animal.
    b. • Define a public constructor that initializes the legs attribute.
    c. • Declare an abstract method eat.
    d. • Declare a concrete method walk that prints out something about how the animals walk (include the number of legs).

    Create the Spider class that extends the Animal class. Define a default constructor that calls the superclass constructor to specify that all spiders have eight legs. Implement the eat method.

    Create another class Bird that has the following features.

    • Declare an abstract method wings that takes the number of wings.

    • Declare an abstract method sing. Create the Parrot class that implements Bird class. This class has another method called color that prints “Parrots are green in color.”

    Create a Main class that contains the main method and executes all the other classes.

……..

9. Write the class named “Rectangle” which will have two attributes named length and width (both are integer in type). This class also contains two methods named CalculateArea and CalculatePerimeter (these two methods returns integer values when they are called).

    Now write a class named TestRectangle to test your code (contains main method).

    • In this class at first, print a line, “Enter dimensions of rectangle:”. Then take two integer values from user.

    • Again, print a line, “Please enter a choice: 1. Area of rectangle. 2. Perimeter of rectangle. “

    • Then take an integer number as a user input from keyboard.

    If the integer is 1, show the area of the Rectangle; If choice is 2, print the perimeter. Area = (width*length) Perimeter = 2(l+w)

……..

10. Your task is to develop a basic contact management application where users can add contacts to their address book. The requirements are as follows:

AddressBook Class:

- **Attributes:**

- `name` (string): The name of the address book.

- `contactList` (array of strings): A list of contact names.

The following methods must be implemented:

- `addContact(contact_name)`: Adds a contact to the address book. Ensure that the same contact does not get added to the list twice.

- `removeContact(contact_name)`: Removes a contact from the address book if it exists.

- `displayContacts()`: Displays all contacts in the address book.

- `searchContact(contact_name)`: Given the name of a contact, find and return the contact from the address book if it exists.

……..

11. Your task is to build a virtual pet simulator in Python. The pet can be any animal (e.g., cat, dog, dragon) with specific needs and behaviors. The requirements are as follows:

Pet Class:

a. Attributes: name (string), type (string: “cat”, “dog”, “dragon”), hungerLevel (integer), happinessLevel (integer)

Owner Class: a. Attributes: name (string), pet (object)

b. The following methods must be implemented:

i. displayOwnerInfo(): This function should return a string like “[Name] is the owner of [Pet’s Name].”

ii. feedPet(): Feeding the pet will decrease its hunger by 20 points. But the hunger cannot be negative. For example: if the hunger level is at 10 and you feed the pet, the hunger level should be at 0.

iii. playWithPet(): Playing with the pet will increase its happiness by 35 points. Again, the happiness level cannot go above 100. For example: if the pet’s happiness is at 80 and you play with it, it will go to 100

……..

12. BanglaCar is a security and electronic company which produces and assembles products for **cars**. It delivers any car electronic or security system you want, from airbags to GPS tracking system, reverse parking system etc. Big car companies use its products in their cars. The company uses a well defined object oriented approach to keep

track of their products using software which is developed and maintained by them only. They get the car, produce the system for it and assemble it into the car. Recently, they got new orders from **BMW(a car company)** to produce a **central locking** and **gear lock** system for their new **X5 model**. After a while, another car company **Mercedes (another car company)** asked them to produce a new system of **central locking** and **gear lock** for their **GLS** model. Please note that the **car** and the **product** should vary independently in order to make the software system easy to extend and reusable. We can separate the design into two different hierarchies. One **interface** is for the **product** which will be used as an implementer and the other will be an **abstraction** of **car** type. The implementer will be implemented by the concrete implementers and provides an implementation for it. On the other side, the abstraction will be extended by more refined abstraction.

Tasks:

. Create an interface of Product which contains String getProductName() and void produce() methods.

. Create a class named CentralLocking which implements Product interface. The class has a property as productName. Implement the methods where the produce method shows the “Producing Central Locking System” message. Implement the constructor also.

. Create a class named GearLocking which implements Product interface. Theclass has a property as productName. Implement the methods where the producemethod shows the “Producing Gear Locking System” message. Implement the

constructor also.

. Create an abstract class named Car which has two protected properties as product and carType. Create a constructor of this class. Implement a method as printDetails() which prints car type and product name. Look at the output of the

main method. Add two void abstract methods as assemble() and produceProduct().

. Create a BMW class which inherits the Car class. The assemble method prints the following Assembling message.

Example: Assembling [Central Locking System] for [BMW X5 Model]

The produceProduct method firstly produces the product and then shows the following Modifying message.

Example: Modifying product [Gear Locking System] according to [BMW X5 Model]

. Create a Mercedes class which inherits the Car class. The assemble method prints the following Assembling message.

Example: Assembling [Central Locking System] for [Mercedes GLS Model]

The produceProduct method firstly produces the product and then shows the following Modifying message.

Example: Modifying product [Gear Locking System] according to [Mercedes GLS Model]

Main Function

public static void main(String[] args)

```
{
Product product = new CentralLocking("Central Locking System");
Product product2 = new GearLocking("Gear Locking System");
Car car = new BMW(product, "BMW X5 Model");
car.produceProduct();
car.assemble();
car.printDetails();
System.out.println();
car = new BMW(product2, "BMW X5 Model");
car.produceProduct();
car.assemble();
car.printDetails();
System.out.println();
car = new Mercedes(product, "Mercedes GLS Model");
car.produceProduct();
car.assemble();
car.printDetails();
System.out.println();
car = new Mercedes(product2, "Mercedes GLS Model");
car.produceProduct();
car.assemble();
car.printDetails();
}
```

Output

```
Producing Central Locking System
Modifying product Central Locking System according to BMW X5 Model
Assembling Central Locking System for BMW X5 Model
Car: BMW X5 Model, Product: Central Locking System

Producing Gear Locking System
```

Modifying product Gear Locking System according to BMW X5 Model

Assembling Gear Locking System for BMW X5 Model

Car: BMW X5 Model, Product: Gear Locking System

Producing Central Locking System

Modifying product Central Locking System for Mercedes GLS Model

Assembling Central Locking System for Mercedes GLS Model

Car: Mercedes GLS Model, Product: Central Locking System

Producing Gear Locking System

Modifying product Gear Locking System for Mercedes GLS Model

Assembling Gear Locking System for Mercedes GLS Model

Car: Mercedes GLS Model, Product: Gear Locking System

……..

13. Write a program named CricketWorldCup.java following the instructions

below and submit the CricketWorldCup.java file only.

. Create a project named CricketWorldCup.

. Create two classes named Team and Match inside the CricketWorldCup.java file.

. Team class has the following private properties. Create constructors, getters and setters for the required properties. Use the proper data type. Make all of them private.

a. Country

b. Captain Name

c. Point

d. Total Match

. Match class has the following private properties. Create constructors, getters and setters for the required properties. Use the proper data type. Make all of them private.

a. TeamA

b. TeamB

. Create a method named void updateResult(int result) inside Team class following proper logic.

Here, the result value represents 0 (NO RESULT), 1 (WIN), 2 (LOSS). For winning a match points increases by 3.

. Create a method named void updateResult(int runsA, int runsB) inside Match

class following proper logic. Here runsA and runsB denote corresponding runs of

teamA and teamB. [3]

. Create an array of teams inside the main method. And generate the final point

table of all teams using proper logic and inputs. [5]

. Take the following inputs of teams and matches. Sample output is shown below.

Sample Input	Sample Output
4 Bangladesh Tamim Iqbal India Virat Kohli England Joe Root New Zealand Shane Bond 6 1 2 320 240 3 4 220 220 2 4 350 355 3 1 280 280 1 4 267 265 2 3 239 280	Points Table Team Name | Captain Name | Total Matches | Points Bangladesh | Tamim Iqbal| 3 | 6 India | Virat Kohli| 3 | 0 England | Joe Root| 3 | 3 New Zealand | Shane Bond| 3 | 3

……..

14. Write a Java program to implement the following tasks.

. Write an abstract class named Shape where it will have two properties as name and area. Keep both properties as protected.

. Write a protected constructor of Shape class which takes the name parameter.

. Write a concrete void method as printArea which prints the name of the object with its area.

. Keep an abstract method named calculateArea which calculates the area of different shapes.

. Write a class named Rectangle which inherits Shape class and keeps length and width properties as protected.

. Write a constructor for Rectangle class with length and width parameters. It passes the “Rectangle” parameter to superclass.

. Override the calculateArea method appropriately for Rectangle class.

. Write a class named Circle which inherits Shape class and keeps radius property as protected.

. Write a constructor for Circle class with radius parameter. It passes the “Circle” parameter to superclass.

.Override the calculateArea method appropriately for Circle class.

. Write an interface named Movable which has open and close void methods.

.Implement the Movable interface by Door class which shows “Opening the door” inside open method and “Closing the door” inside close method.

.Follow the main method and sample output for more clarity

Main Method:

```
public static void main(String[] args) {

Shape rectangle = new Rectangle(20,10);

rectangle.calculateArea();

rectangle.printArea();

Shape circle = new Circle(10);

circle.calculateArea();

circle.printArea();

Movable door = new Door();

door.open();

door.close();

}
```

Output

```
Rectangle Area: 200.0

Circle Area: 314.15997

Opening the door

Closing the door
```

……..

15. Write a program which will take books names of a library. Where the number of books will be provided by console as **n**. After this number, you will get names of books. Keep them in an array of strings. Find the total number of books whose name starts with a vowel. Please check the following sample input/output along with the explanation.

Sample Input	Sample Output	Explanation
3 **I Capture the Castle** American Dreamer To kill a mockingbard	Count: 2	Book 1 and 2 starts with vowel

……..

16. You are tasked with creating a Java program to manage points in both 2D and 3D space, and calculate distances and midpoints between them. The requirements are as follows:

Point Class (for 3D Points):

- **Attributes:**
  - `x`, `y`, `z` (double): Coordinates of the point.
  - `name` (String): Name of the point.
- **Constructors:**
  1. A constructor that takes only the name.
  2. A constructor that takes `x`, `y`, `z`.
  3. A constructor that takes `x`, `y`, `z`, and `name`.
- **Methods:**
  1. `distance(Point p)`: Calculates the distance to another point using `Math.sqrt(double value)`.
  2. `midpoint(Point p)`: Calculates the midpoint with another point.
  3. `midpoint(Point p, String name)`: Calculates the midpoint with another point, and sets the name of the midpoint.
  4. Override the `toString()` method to represent the point as a string.

Abstract Point Class (for 2D Points):

- **Attributes:**
  - `x`, `y` (double): Coordinates of the point.
- **Constructor:**

- A constructor that takes `x`, `y` and is protected.

- **Abstract Method:**

- `distance(Point p)`: Abstract method to calculate the distance to another point.

Point2D Class:

- **Inherits:** Point

- **Constructor:**

- A public constructor that takes `x`, `y`.

- **Methods:**

- Overrides `distance(Point p)` to calculate the distance to another 2D point using `Math.sqrt(double value)`.

Point3D Class:

- **Inherits:** Point

- **Constructor:**

- A public constructor that takes `x`, `y`, `z`.

- **Methods:**

- Overrides `distance(Point p)` to calculate the distance to another 3D point using `Math.sqrt(double value)`.

Main Method and Output

Implement the main method to create instances of `Point2D` and `Point3D`, calculate distances and midpoints, and display the results.

```
    public static void main(String[] args) {
        // Testing 3D Points
        Point p1 = new Point("origin");
        Point p2 = new Point(3, 4, 12, "corner");
        Point p3 = p1.midpoint(p2);
        Point p4 = p1.midpoint(p2, "mid point");
        double distance1 = p1.distance(p2);
```

```
        System.out.println(p3);
        System.out.println(p4);
        System.out.println("Distance: " + distance1);

        // Testing 2D Points
        AbstractPoint p5 = new Point2D(0, 0);
        AbstractPoint p6 = new Point2D(3, 4);
        double distance2 = p5.distance(p6);
        System.out.println("Distance 2D: " + distance2);

        Point3D p7 = new Point3D(0, 0, 0);
        Point3D p8 = new Point3D(3, 4, 12);
        double distance3 = p7.distance(p8);
        System.out.println("Distance 3D: " + distance3);
    }
}
```

Expected Output

```
(1.500000,2.000000,6.000000)
mid point (1.500000,2.000000,6.000000)
Distance: 13.0
Distance 2D: 5.0
Distance 3D: 13.0
```