OBJECT ORIENTED PROGRAMMING WITH JAVA(22CS36)
UNIT 1: PRACTICE QUESTIONS

1. Illustrate Java being the Object-Oriented Programming language using OOP principles.
2. Write a simple Java program to display a welcome message and discuss each line of the code in detail. Write commands for compilation and execution with an example for each.
3. Illustrate the use of static method, static variable, and static initialization block with an example.
4. Demonstrate through code examples, the use of **this** keyword.
5. Discuss the Garbage collection technique and finalize() method of Java.
6. Write a Java program by using a class with the following specifications:
   Class name- Book
   Data members or Instance variables- String: title, author, Publisher Double: Cost.
   Member methods:
   i. void input ()- Accepts the title, author name, Publisher name and Cost of the book.
   ii. void display ()- display the title, author name, Publisher name, and Cost of the book.
7. Write a Java program to define an Employee class including fields such as empname, empid, age, occupation, and Income. Overload constructors to assign values to members of the object. Include a display method to print the employee details. In the main method create an array of employees and print those employee details with income greater than 15k.
8. Demonstrate the use of Constructor in Java. Explain the Parameterized Constructor with an example.
9. Differentiate the use of access modifiers in Java and their scope.
10. Can you overload a Constructor in Java? Justify with a suitable program.
11. Write a Java program using Class with the following specifications. Class name-Salary,
    Data members: private int basic, Member functions-void input(): to input basic pay, void display(): to calculate and print the following: da=30% of basic, hra=10% of basic, gross=basic+da+hra.
12. With suitable examples explain the types of Argument Passing in Java.
13. Develop a program to swap 2 objects of a Number class. The number class has an instance variable of type double, a constructor to initialize, and the swap method.
14. Design a student class with 2 instance variables Usn and name, a constructor, and a method to print student objects. Write driver class to test an array of student classes of size N
15. Design a person class with instance variables name and age and a constructor to initialize, a compare method to compare 2 person objects and return a person object with greater age. Write driver class to find the eldest person among N person objects
16. Write a Java program to Java Program to sort an anonymous int array.
17. Write a Java Program for Binary search. Util package may be used.
18. Write a program to input 2 strings, concatenate these strings, and reverse the concatenated string
19. How do you pass information to the program when you run it? Illustrate with an example.
20. Demonstrate with an example an Ambiguity when overloading a Vararg method.
21. Write a Java program to sort strings that are passed as varargs to a method.
22. Demonstrate the implementation of access modifiers on the Stack class restricting the Stack open to misuse of mischief.
23. Create a class namely Account with the data members(Accno : integer, name :String,
    Phone_No: integer, balance_amt:float), and following methods :

a. CreateAccount() method to create an account.
b. Deposit() method to deposit amount to an account.
c. Withdraw() method which gets the amount to be withdrawn from his/her account.
d. PrintAccount() method to display account details.

24. Imagine you are building a basic online shopping application. You need to create a Java class representing a product. Each product has a name, price, and quantity in stock. Implement the Product class with appropriate methods to set and get the product details.
Create a Java class named Product with instance variables for product name, price, and quantity in stock. Provide methods to set and get these variables. Additionally, create a method calculateTotalPrice() that calculates the total price of the products (price * quantity). Lastly, write a sample code snippet in the ProductDemo class to demonstrate the usage of the Product class.

25. Imagine you are building a basic library management system. Each book in the library has a unique identifier, a title, an author, and a publication year. You need to create a Java class to represent a book, allowing users to retrieve information about the book.
Create a Java class named Book with the following attributes:
id (int) - unique identifier for the book
title (String) - title of the book
author (String) - author of the book
publicationYear (int) - year when the book was published.

Provide a constructor to initialize these attributes and a method to display the book's information. Write a Java program to demonstrate the usage of the Book class by creating an instance of the class and printing its details.

26. Consider a scenario where you are developing a shopping cart application. In this application, customers can add items to their cart, and you want to implement method overloading to handle different types of items (e.g., books, electronics, and clothing). Assume suitable parameters for overloaded methods and test these methods.

27. Implement a class named **Stock** that contains:
■ A string data field named **symbol** for the stock's symbol.
■ A string data field named **name** for the stock's name.
■ A double data field named **previousClosingPrice** that stores the stock price for the previous day.
■ A double data field named **currentPrice** that stores the stock price for the current time.
■ A constructor that creates a stock with the specified symbol and name.
■ A method named **getChangePercent()** that returns the percentage changed from **previousClosingPrice** to **currentPrice.**
Write a test program that creates a Stock object with the stock symbol **ORCL**, the name **Oracle Corporation**, and the previous closing price of **34.5.** Set a new current price to **34.35** and display the price-change percentage.

28. Implement a class named **Fan** to represent a fan. The class contains:
■ Three constants named **SLOW, MEDIUM, and FAST** with the values 1, 2, and 3 to denote the fan speed.
■ A private int data field named **speed** that specifies the speed of the fan (the default is SLOW).
■ A private boolean data field named **on** that specifies whether the fan is on (the default is false).
■ A private double data field named **radius** that specifies the radius of the fan (the default is 5).
■ A string data field named **color** that specifies the color of the fan (the default is blue).
■ The accessor and mutator methods for all four data fields.
■ A no-arg constructor that creates a default fan.
■ A method named toString() that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string "fan is off" in one combined string.
Write a test program that creates two Fan objects. Assign maximum speed, radius 10, color yellow, and turn it on to the first object. Assign medium speed, radius 5, color blue, and turn it off to the second object. Display the objects by invoking their toString method.

29. Implement a class named MyPoint to represent a point with x and y-coordinates. The class contains:
■ The data fields x and y that represent the coordinates with get methods.

OBJECT ORIENTED PROGRAMMING WITH JAVA(22CS36)
UNIT 1: PRACTICE QUESTIONS

■ A no-arg constructor that creates a point (0, 0).
■ A constructor that constructs a point with specified coordinates.
■ Two get methods for the data fields x and y, respectively.
■ A method named distance that returns the distance from this point to another point of the MyPoint type.
■ A method named distance that returns the distance from this point to another point with specified x- and y-coordinates.
Write a test program that creates the two points (0, 0) and (10, 30.5) and displays the distance between them.

30. Some websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rules are as follows:

■ A password must have at least eight characters.
■ A password consists of only letters and digits.
■ A password must contain at least two digits.
Write a program that prompts the user to enter a password and displays Valid Password if the rules are followed or Invalid Password otherwise.