JAVA PRACTICE - SET

Basic Operations

1. Write a Java program to calculate the sum of two numbers.
2. Write a Java program to check if a given number is odd or even.
3. Write a Java program to check if a given number is divisible by 5.
4. ---
5. Basic Operations
6. Write a Java program to add two floating-point numbers.
7. Write a Java program to subtract one integer from another.
8. Write a Java program to multiply two double values.
9. Write a Java program to divide one integer by another and display the result.
10. Write a Java program to find the remainder of the division of two integers.
11. Write a Java program to calculate the power of a number.
12. Write a Java program to calculate the square root of a given number.
13. Write a Java program to find the absolute value of a number.
14. Write a Java program to round a floating-point number to the nearest integer.

15. Write a Java program to find the maximum of two numbers.

Control Statements

1. Write a Java program to check if a given number is prime.
2. Write a Java program to determine if a given character is an alphabet or digit.
3. Write a Java program using a switch statement to display a brand's slogan based on the brand name.
4. Write a Java program to calculate the provident fund (PF) from a given basic salary.
5. Write a Java program to check marriage eligibility based on age and salary.
6. Write a Java program to find the factorial of a given number.
7. Write a Java program to print the multiplication table of a given number.
8. Write a Java program to print numbers from 100 to 1.
9. Write a Java program to calculate the sum of the digits of a given number.
10. Write a Java program to check if a given number is an Armstrong number.
11. Write a Java program to print all prime numbers from 1 to 100.
12. ---
13. Write a Java program to check if a number is positive or negative.
14. Write a Java program to find the largest of three numbers using if-else statements.
15. Write a Java program to check if a number is divisible by both 3 and 5.
16. Write a Java program to determine if a given year is a leap year.
17. Write a Java program to print all even numbers between 1 and 100.
18. Write a Java program to print all odd numbers between 1 and 100.
19. Write a Java program to print the factorial of a number using a while loop.

20. Write a Java program to print the Fibonacci series up to a given number using a for loop.
21. Write a Java program to check if a number is prime.
22. Write a Java program to display a multiplication table for a given number.
23. Write a Java program to print the sum of digits of a given number using a for loop.
24. Write a Java program to check if a number is a palindrome.
25. Write a Java program to find the GCD (Greatest Common Divisor) of two numbers using a loop.
26. Write a Java program to find the LCM (Least Common Multiple) of two numbers using a loop.
27. Write a Java program to print a pattern of stars in a right-angled triangle shape.
28. Write a Java program to print a diamond pattern of stars.
29. Write a Java program to calculate the sum of numbers from 1 to 100 using a for loop.
30. Write a Java program to count the number of vowels in a given string.
31. Write a Java program to find the sum of even and odd numbers separately in a range using loops.
32. Write a Java program to reverse the digits of a number using a loop.

Arrays

1. Write a Java program to print the elements of an array.
2. Write a Java program to calculate the sum of the elements in an array.
3. Write a Java program to reverse the elements of an array.
4. Write a Java program to find the minimum and maximum values in an array.
5. String Methods
6. Write a Java program to demonstrate the use of the valueOf() method to convert an integer to a string.
7. Write a Java program to check if a given string starts with a particular character using the startsWith() method.

8. Write a Java program to check if a given string ends with a particular character using the endsWith() method.
9. Write a Java program to remove leading and trailing spaces from a string using the trim() method.
10. Write a Java program to demonstrate the use of the intern() method to access the object from the String Constant Pool (SCP).
11. Write a Java program to convert an object to its string representation using the toString() method.
12. Write a Java program to convert a string into a character array using the toCharArray() method.
13. Write a Java program to compare the contents of two strings using the equals() method.
14. Write a Java program to replace characters in a string using the replaceAll() method.
15. ---
16. Write a Java program to find the largest element in an array.
17. Write a Java program to find the smallest element in an array.
18. Write a Java program to calculate the sum of all elements in an array.
19. Write a Java program to calculate the average of elements in an array.
20. Write a Java program to reverse the elements of an array.
21. Write a Java program to sort an array in ascending order.
22. Write a Java program to merge two arrays into a single array.
23. Write a Java program to remove duplicates from an array.
24. Write a Java program to search for a specific element in an array using linear search.
25. Write a Java program to find the index of the first occurrence of a given element in an array.
26. Write a Java program to find the number of occurrences of a specific element in an array.
27. Write a Java program to rotate an array to the right by a given number of positions.
28. Write a Java program to rotate an array to the left by a given number of positions.
29. Write a Java program to find the second largest element in an array.
30. Write a Java program to check if two arrays are equal.

31. Write a Java program to find the common elements between two arrays.
32. Write a Java program to find the union of two arrays.
33. Write a Java program to find the intersection of two arrays.
34. Write a Java program to perform matrix multiplication.
35. Write a Java program to transpose a matrix.

String Manipulation

1. Write a Java program to combine two strings.
2. Write a Java program to reverse a given string.
3. Write a Java program to check if a given string is a palindrome.
4. Write a Java program to check if two given strings are anagrams (first way).
5. Write a Java program to check if two given strings are anagrams (second way).
6. Write a Java program to swap two strings without using a third variable (first way).
7. Write a Java program to swap two strings without using a third variable (second way).
8. Write a Java program to remove vowels from a given string.
9. ---
10. Concatenate two strings.
11. Check if a string contains a specific substring.
12. Count the number of vowels in a given string.
13. Reverse a given string.
14. Check if a given string is a palindrome.

15. Convert a given string to uppercase.
16. Convert a given string to lowercase.
17. Replace all occurrences of a specific character in a string with another character.
18. Split a string into an array of substrings based on a delimiter.
19. Remove all whitespace from a given string.
20. Count the number of words in a given string.
21. Find the length of a given string.
22. Compare two strings lexicographically.
23. Check if a string starts with a specific substring.
24. Check if a string ends with a specific substring.
25. Trim leading and trailing whitespace from a given string.
26. Convert a string into a character array.
27. Check if a specific character is present in a string.
28. Find the first occurrence of a specific character in a string.
29. Find the last occurrence of a specific character in a string.

Arrays

1. Write a program to find the second largest element in an array.
2. Write a program to check if an array contains a duplicate element.
3. Write a program to rotate an array by k positions.
4. Write a program to find the intersection of two arrays.
5. Write a program to merge two sorted arrays into one sorted array.
6. Write a program to find the missing number in an array containing numbers from 1 to n.
7. Write a program to find the sum of all elements in a 2D array.
8. Write a program to reverse the elements of a 2D array.
9. Write a program to find the most frequent element in an array.
10. Write a program to find the longest increasing subsequence in an array.
11. Write a program to find the maximum product of any two elements in an array.
12. Write a program to count the number of positive and negative numbers in an array.
13. Write a program to move all zeroes in an array to the end without changing the order of non-zero elements.
14. Write a program to find the longest consecutive sequence in an unsorted array.
15. Write a program to sort an array of strings by their lengths.

Strings

1. Write a program to count the number of vowels in a given string.
2. Write a program to check if two strings are anagrams of each other.
3. Write a program to remove all duplicate characters from a string.
4. Write a program to reverse each word in a given string.
5. Write a program to find the first non-repeated character in a string.
6. Write a program to convert a string to uppercase and lowercase.
7. Write a program to check if a string is a palindrome.
8. Write a program to find the longest palindromic substring in a given string.
9. Write a program to count the frequency of each character in a string.
10. Write a program to replace all occurrences of a given character in a string with another character.
11. Write a program to find the length of the longest substring without repeating characters.
12. Write a program to remove all spaces from a string.
13. Write a program to find and print all substrings of a given string.
14. Write a program to find the number of words in a string.

15. Write a program to extract a substring from a given string.

Control Statements

1. Write a program to print the Fibonacci series up to a given number of terms.
2. Write a program to print all prime numbers between 1 and 100.
3. Write a program to check if a number is a perfect number.
4. Write a program to generate a multiplication table for a given number.
5. Write a program to find the factorial of a given number using recursion.
6. Write a program to check if a given number is a Armstrong number.
7. Write a program to print a pattern of stars in a triangle shape.
8. Write a program to find the sum of digits of a given number.
9. Write a program to print all even numbers between 1 and 100.
10. Write a program to find the GCD (Greatest Common Divisor) of two numbers using Euclidean algorithm.

OOP Concepts

Class and Object

1 . Create a Java class called Person with attributes for name and age. Include methods to set and get these attributes, and create an object of this class to display the person's details.

Inheritance

2. Design a base class Animal with a method makeSound(). Create two subclasses, Dog and Cat, that override makeSound() to provide specific sounds for each animal. Demonstrate polymorphism by calling makeSound() on objects of both subclasses.

Encapsulation

3. Implement a class BankAccount with private attributes for account number and balance. Provide public methods to deposit and withdraw money, and to check the balance. Ensure that the balance cannot be directly modified from outside the class.

Abstraction

4. Define an abstract class Shape with an abstract method calculateArea(). Create two subclasses, Rectangle and Circle, that provide implementations for calculateArea(). Instantiate these subclasses and compute the area for both shapes.

Polymorphism

5. Write a Java program where you have a base class Vehicle with a method start(). Create derived classes Car and Bike that override the start() method. Demonstrate method overriding by creating instances of Car and Bike and calling start() on each.

Constructor Overloading

6. Create a class Book with multiple constructors to initialize the book's title, author, and price. Demonstrate constructor overloading by creating different instances of Book using various constructors.

Method Overloading

7. Implement a class Calculator with multiple overloaded methods named add(). Overload the method to handle different types of arguments (e.g., integers, doubles) and demonstrate method overloading by calling these methods with different data types.

Static Keyword

8. Design a class MathUtility with a static method add() that takes two integers and returns their sum. Create another class to call this static method and display the result.

Composition

9. Create a class Library that contains a collection of Book objects. The Book class should include attributes for the book's title and author. Demonstrate how the Library class can manage multiple Book

instances.

Inheritance and Constructor

10. Write a base class Employee with a constructor to initialize the employee's name and ID. Create a subclass Manager that inherits from Employee and includes additional attributes for department and salary. Demonstrate how constructors are called in the inheritance hierarchy.

Inheritance Hierarchy

11. Design a base class Person with attributes for name and age. Create a subclass Student that extends Person and includes additional attributes for student ID and major. Then create another subclass GraduateStudent that extends Student and includes an attribute for research topic.

Abstract Classes

12. Define an abstract class Account with an abstract method calculateInterest(). Implement this class in two subclasses, SavingsAccount and CurrentAccount, providing specific implementations for the calculateInterest() method.

Interfaces

13. Create an interface Playable with a method play(). Implement this interface in two classes, Guitar and Piano, with specific implementations of the play() method. Demonstrate the use of the Playable

interface by creating objects of Guitar and Piano.

Multiple Inheritance

14. Java does not support multiple inheritance directly through classes. Demonstrate how multiple inheritance can be achieved using interfaces by creating two interfaces, Flyable and Swimmable, and a class Duck that implements both interfaces.

Object Cloning

15. Implement a class Person with a clone() method that creates a copy of an object using the Cloneable interface. Demonstrate object cloning by creating a copy of a Person instance and displaying both the original and cloned objects.

Encapsulation with Getters and Setters

16. Create a class Rectangle with private attributes for length and width. Provide public getter and setter methods for these attributes and a method to calculate the area of the rectangle. Demonstrate encapsulation by setting and getting the values using these methods.

Method Overriding

17. Define a base class Animal with a method eat(). Override this method in subclasses Herbivore and Carnivore to provide specific eating behaviors. Demonstrate method overriding by calling eat() on

objects of these subclasses.

Class vs. Object Methods

18. Implement a class MathOperations with both instance methods and static methods for performing mathematical operations (e.g., addition, subtraction). Compare and demonstrate the use of class (static) methods and object (instance) methods.

Super Keyword

19. Write a base class Person with a constructor that initializes the name and age. Create a subclass Employee that extends Person and includes additional attributes for employee ID and department. Use the super keyword to call the base class constructor from the subclass constructor.

Abstract Method Implementation

20. Define an abstract class Appliance with an abstract method turnOn(). Create subclasses WashingMachine and Refrigerator that provide specific implementations of the turnOn() method. Instantiate these subclasses and call the turnOn() method.

Nested Classes

21. Create an outer class Outer with a private nested class Inner. The Inner class should have a method to display a message. Instantiate the Inner class from within the Outer class and call its method.

Enums

22. Define an enumeration Day with constants for each day of the week. Write a method that takes a Day enum as a parameter and prints whether it is a weekday or weekend.

Polymorphic Behavior

23. Implement a base class Shape with a method draw(). Create subclasses Circle and Square that override the draw() method. Create a list of Shape objects containing instances of Circle and Square, and iterate through the list to call the draw() method on each object.

Object Composition

24. Create a class House with attributes for the address and a collection of Room objects. The Room class should have attributes for room name and size. Demonstrate object composition by creating a House instance with multiple Room instances.

Method References

25. Create a class StringOperations with a method toUpperCase(String s) that converts a string to uppercase. Use method references to pass this method as a reference to a functional interface and invoke it.

Constructor Chaining

26. Implement a class Employee with multiple constructors that initialize different combinations of attributes (e.g., name, ID, department). Demonstrate constructor chaining by calling one constructor from another within the class.

Instance Initialization Blocks

27. Define a class Config with an instance initialization block that sets default values for attributes. Create an instance of this class and display the values of the attributes to see the effect of the initialization block.

Final Keyword

28. Create a class Constants with a final variable PI and a final method displayInfo(). Attempt to override the displayInfo() method in a subclass and observe the results. Explain the behavior of the final keyword in this context.

Composition vs. Inheritance

29. Compare and contrast composition and inheritance by creating two classes: Computer (with a Processor class as a component) and Smartphone (with Computer class as a base). Explain the advantages and disadvantages of each approach.

Instanceof Operator

30. Implement a class hierarchy with a base class Animal and subclasses Dog and Cat. Use the instanceof operator to check the type of an object at runtime and display appropriate messages based on the type.