**1. Static Variables**

class Example {

static int count = 0; // Static variable

Example() {

count++; // Increment count whenever an instance is created

}

}

// Usage

Example obj1 = new Example();

Example obj2 = new Example();

System.out.println(Example.count); // Outputs: 2

**2. Static Methods**

class Utility {

static int add(int a, int b) { // Static method

return a + b;

}

}

// Usage

int result = Utility.add(5, 10); // Outputs: 15

**3. Static Blocks**

class Config {

static int value;

static {

value = 10; // Static block for initialization

System.out.println("Static block executed");

}

}

// Usage

System.out.println(Config.value); // Outputs: 10

**4. Static Classes**

class Outer {

static int outerStaticVar = 10;

static class Inner { // Static nested class

void display() {

System.out.println("Outer static variable: " + outerStaticVar);

}

}

}

// Usage

Outer.Inner inner = new Outer.Inner();

inner.display(); // Outputs: Outer static variable: 10

**5. Static Import**

import static java.lang.Math.\*; // Importing static methods

// Usage

double area = PI \* pow(5, 2); // Using static members without Math prefix

**6. Static Interface Methods (Java 8 and above)**

interface MathOperations {

static int multiply(int a, int b) {

return a \* b;

}

}

// Usage

int result = MathOperations.multiply(5, 6); // Outputs: 30

**7. Static Final Variables (Constants)**

class Constants {

static final double PI = 3.14159; // Static final variable

}

// Usage

System.out.println(Constants.PI); // Outputs: 3.14159

1. **Static Counter in a Class**

class Counter {

static int instanceCount = 0; // Static variable to count instances

Counter() {

instanceCount++; // Increment count when an instance is created

}

static int getCount() { // Static method to get the count

return instanceCount;

}

}

// Usage

public class TestCounter {

public static void main(String[] args) {

Counter obj1 = new Counter();

Counter obj2 = new Counter();

Counter obj3 = new Counter();

System.out.println("Number of instances created: " + Counter.getCount()); // Outputs: 3

}

}

**9. Static Factory Method**

class Person {

private String name;

private Person(String name) { // Private constructor

this.name = name;

}

public static Person createPerson(String name) { // Static factory method

return new Person(name);

}

public String getName() {

return name;

}

}

// Usage

public class TestFactory {

public static void main(String[] args) {

Person person1 = Person.createPerson("Alice");

System.out.println("Person name: " + person1.getName()); // Outputs: Person name: Alice

}

}

**10. Static Synchronization**

class SynchronizedCounter {

private static int count = 0; // Static variable to count

public static synchronized void increment() { // Synchronized static method

count++;

}

public static int getCount() {

return count;

}

}

// Usage

public class TestSynchronizedCounter {

public static void main(String[] args) throws InterruptedException {

Thread t1 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

SynchronizedCounter.increment();

}

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

SynchronizedCounter.increment();

}

});

t1.start();

t2.start();

t1.join();

t2.join();

System.out.println("Final count: " + SynchronizedCounter.getCount()); // Outputs: Final count: 2000

}

}

**11. Initialize and Print Elements of an Array**

public class ArrayExample {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

for (int i = 0; i < arr.length; i++) {

System.out.println("Element at index " + i + ": " + arr[i]);

}

}

}

**12. Find the Sum of All Elements in an Array**

public class SumOfArray {

public static void main(String[] args) {

int[] arr = {10, 20, 30, 40, 50};

int sum = 0;

for (int i = 0; i < arr.length; i++) {

sum += arr[i];

}

System.out.println("Sum of elements: " + sum);

}

}

**13. Find the Maximum Element in an Array**

public class MaxElement {

public static void main(String[] args) {

int[] arr = {5, 8, 12, 7, 3};

int max = arr[0];

for (int i = 1; i < arr.length; i++) {

if (arr[i] > max) {

max = arr[i];

}

}

System.out.println("Maximum element: " + max);

}

}

**14. Find the Minimum Element in an Array**

public class MinElement {

public static void main(String[] args) {

int[] arr = {15, 22, 3, 9, 8};

int min = arr[0];

for (int i = 1; i < arr.length; i++) {

if (arr[i] < min) {

min = arr[i];

}

}

System.out.println("Minimum element: " + min);

}

}

**15. Reverse the Elements of an Array**

public class ReverseArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

System.out.println("Reversed array:");

for (int i = arr.length - 1; i >= 0; i--) {

System.out.print(arr[i] + " ");

}

}

}

**16. Count Even and Odd Numbers in an Array**

public class CountEvenOdd {

public static void main(String[] args) {

int[] arr = {10, 21, 32, 43, 54};

int evenCount = 0, oddCount = 0;

for (int value : arr) {

if (value % 2 == 0) {

evenCount++;

} else {

oddCount++;

}

}

System.out.println("Even count: " + evenCount);

System.out.println("Odd count: " + oddCount);

}

}

**17. Sort an Array in Ascending Order**

import java.util.Arrays;

public class SortArray {

public static void main(String[] args) {

int[] arr = {5, 2, 8, 3, 1};

Arrays.sort(arr);

System.out.println("Sorted array:");

for (int value : arr) {

System.out.print(value + " ");

}

}

}

**18. Copy Elements of One Array to Another**

public class CopyArray {

public static void main(String[] args) {

int[] original = {10, 20, 30, 40, 50};

int[] copy = new int[original.length];

System.arraycopy(original, 0, copy, 0, original.length);

System.out.println("Copied array:");

for (int value : copy) {

System.out.print(value + " ");

}

}

}

**19. Check if an Array Contains a Specific Value**

public class ContainsValue {

public static void main(String[] args) {

int[] arr = {5, 7, 9, 2, 4};

int target = 9;

boolean found = false;

for (int value : arr) {

if (value == target) {

found = true;

break;

}

}

System.out.println("Array contains " + target + ": " + found);

}

}

**20. Find the Second Largest Element in an Array**

public class SecondLargest {

public static void main(String[] args) {

int[] arr = {15, 24, 36, 48, 29};

int largest = Integer.MIN\_VALUE;

int secondLargest = Integer.MIN\_VALUE;

for (int value : arr) {

if (value > largest) {

secondLargest = largest;

largest = value;

} else if (value > secondLargest && value != largest) {

secondLargest = value;

}

}

System.out.println("Second largest element: " + secondLargest);

}

}

**From CS 05.ppt (Conceptual & UML Questions)**

**1. Basics of Object-Oriented Design**

* What are the qualities of a good software? Explain **operational, transitional, and maintenance** qualities.
* Explain the **Object-Oriented Life Cycle Model** with advantages.
* Differentiate between **OOA, OOD, and OOP** with examples.
* Define **object** and **object class**. How are they related?
* What are the **advantages of encapsulation and modularity** in OOD?

**2. UML and Diagrams**

* What is UML? Why is it considered a standard for OO modeling?
* List the **9 UML diagrams**. Classify them into **structural** and **behavioral** categories.
* Draw and explain a **use case diagram** for an order management system.
* Differentiate between **<<include>>** and **<<extend>>** in use case diagrams.
* What is a **class diagram**? Explain with an example.
* Define **attributes, operations, and visibility symbols (+, -, #)** in UML.
* Explain **object diagrams** with an ATM system example.
* What is a **sequence diagram**? Show one for **ATM withdrawal**.

**3. Object-Oriented Design Process**

* Explain steps of an **Object-Oriented Design process**.
* What are the challenges in **object identification**? List approaches for identifying objects.
* Explain **static vs. dynamic models** in OOD.
* What are **subsystem models, state machine models, and aggregation models**?

**4. Case Study – Weather Station**

* Explain the **weather data collection system** requirements.
* Describe the **layered architecture** of the weather station.
* Identify possible **objects and classes** in the weather station system.
* Write a **simple Java code** to represent the weather station system.
* Explain **use-cases and actors** for weather reporting.
* Draw the **system context model** for the weather station.

**📙 From CS 05 – Programs.docx (Programming & Application Questions)**

**1. Static Keyword in Java**

* What is a **static variable**? Write a program to show how it counts object instances.
* What is a **static method**? Write an example.
* Explain the role of a **static block** in initialization.
* What is a **static nested class**? Demonstrate with a program.
* What is a **static import**? Show with Math class example.
* How do **static methods in interfaces** (Java 8+) work? Show with multiplication example.
* Define **static final variable (constant)**. Why is it useful?
* Write a program for a **static factory method** (Person example).
* What is **static synchronization**? Why is it needed in multithreading?

**2. Array Programs**

* Write a program to **initialize and print elements** of an array.
* Write a program to **find sum of elements** in an array.
* Write a program to find the **maximum** and **minimum** element in an array.
* Write a program to **reverse elements** of an array.
* Count **even and odd numbers** in an array.
* Write a program to **sort an array** in ascending order.
* Write a program to **copy elements** from one array to another.
* Write a program to **check if an array contains a specific value**.
* Write a program to find the **second largest element** in an array.