**Department of Computer Science & Engineering**  
**B.Tech – 3rd Semester Examination**  
**Course Title:** Object Oriented Programming Using JAVA  
**Course Code:** CSE30220 **Time Allowed:** 2 Hours 30 Minutes  
**Max. Marks:** 50  **Min. Marks:** 20

**Instructions:**

* Attempt **all questions** from **Section A** and **Section B**.
* Attempt **any two** questions from **Section C**.

**Section A – Short Answer Type Questions**

**(10 × 1 = 10 Marks)**

1.

1. What is the difference between & and && operators?
2. List one use of the static keyword.
3. Show how the ?: operator works.
4. What is an Interface?
5. What is an Abstract class?
6. Define Thread.
7. What does AWT stand for?
8. What is Swing?
9. Name one standard stream.
10. What does JDBC stand for?

✅ **Answer Key – Section A**

1. & is a bitwise AND operator; && is a logical AND operator used in conditional statements.
2. To define class-level variables or methods that can be accessed without creating an object.
3. condition ? value\_if\_true : value\_if\_false — Example: int x = (a > b) ? a : b;
4. An interface is a reference type in Java that can contain abstract methods and constants.
5. An abstract class is a class that cannot be instantiated and may contain abstract methods.
6. A thread is a lightweight process that enables concurrent execution in Java.
7. AWT stands for Abstract Window Toolkit.
8. Swing is a GUI toolkit in Java that provides more powerful and flexible components than AWT.
9. System.out is a standard output stream.
10. JDBC stands for Java Database Connectivity.

**Section B – Medium Answer Type Questions**

**(4 × 5 = 20 Marks)**

2.

1. Write a program to demonstrate method-overloading in Java.
2. Write a program to demonstrate method-overriding in Java.
3. Differentiate between AWT and Swing.
4. Write a program to copy a file in Java using I/O Streams.

**Section C – Long Answer Type Questions**

**(2 × 10 = 20 Marks)**

3.

1. Explain arithmetic, relational, and logical operators available in Java.
2. Discuss exception-handling using try-catch-finally in Java with the help of code samples.
3. Discuss event-handling in Java with the help of diagrams and examples.
4. Discuss any five Collection classes available in Java with the help of code samples.

If you'd like, I can help you format this with your university logo, watermark, or footer. I can also prepare a matching PowerPoint or worksheet version for classroom use. Just say the word!

🧠 **Section A – Short Answer Type Questions (1 × 10 = 10 Marks)**

1. **Difference between & and && operators**
   * &: Bitwise AND (operates on bits)
   * &&: Logical AND (used in boolean expressions, short-circuits)
2. **One use of the static keyword**
   * To define class-level variables or methods that can be accessed without creating an object.
3. **How the ?: operator works**
   * Syntax: condition ? value\_if\_true : value\_if\_false
   * Example: int max = (a > b) ? a : b;
4. **What is an Interface?**
   * A reference type in Java containing abstract methods. It defines a contract that implementing classes must follow.
5. **What is an Abstract class?**
   * A class that cannot be instantiated and may contain abstract methods. Used for partial implementation.
6. **Define Thread**
   * A thread is a lightweight process that enables concurrent execution in Java.
7. **What is Swing?**
   * A GUI toolkit in Java that provides rich components like buttons, tables, and trees. It is part of javax.swing.
8. **Name one standard stream**
   * System.out (standard output stream)
9. **What does JDBC stand for?**
   * Java Database Connectivity
10. **What is a package?**

* A namespace that organizes classes and interfaces. Example: java.util, java.io

🧪 **Section B – Medium Answer Type Questions (2 × 5 = 10 Marks)**

1. **Method Overloading in Java**

class OverloadDemo { void show(int a) { System.out.println("Integer: " + a); } void show(String s) { System.out.println("String: " + s); } public static void main(String[] args) { OverloadDemo obj = new OverloadDemo(); obj.show(10); obj.show("Hello"); } }

2. **Use of Constructors**

class Student { String name; int age; Student(String n, int a) { name = n; age = a; } void display() { System.out.println(name + " is " + age + " years old."); } public static void main(String[] args) { Student s1 = new Student("Mushtaq", 20); s1.display(); } }

3. **File Handling using I/O Streams**

import java.io.\*; public class FileCopy { public static void main(String[] args) throws IOException { FileInputStream in = new FileInputStream("source.txt"); FileOutputStream out = new FileOutputStream("dest.txt"); int c; while ((c = in.read()) != -1) { out.write(c); } in.close(); out.close(); System.out.println("File copied successfully."); } }

📘 **Section C – Long Answer Type Questions (2 × 15 = 30 Marks)**

1. **try-catch-finally Example**

public class ExceptionDemo { public static void main(String[] args) { try { int a = 10 / 0; } catch (ArithmeticException e) { System.out.println("Cannot divide by zero."); } finally { System.out.println("Finally block executed."); } } }

2. **Inheritance Example**

class Animal { void sound() { System.out.println("Animal makes sound"); } } class Dog extends Animal { void sound() { System.out.println("Dog barks"); } public static void main(String[] args) { Dog d = new Dog(); d.sound(); // Method overriding } }

3. **Object Arrays Example**

class Student { String name; int marks; Student(String n, int m) { name = n; marks = m; } void display() { System.out.println(name + ": " + marks); } public static void main(String[] args) { Student[] arr = new Student[3]; arr[0] = new Student("Ali", 85); arr[1] = new Student("Zara", 90); arr[2] = new Student("Mushtaq", 95); for (Student s : arr) { s.display(); } } }

Method overloading:

class OverloadDemo {

void show(int a) {

System.out.println("Integer: " + a);

}

void show(String s) {

System.out.println("String: " + s);

}

public static void main(String[] args) {

OverloadDemo obj = new OverloadDemo();

obj.show(10);

obj.show("Hello");

}

}

Use of constructor:

class Student {

String name;

int age;

Student(String n, int a) {

name = n;

age = a;

}

void display() {

System.out.println(name + " is " + age + " years old.");

}

public static void main(String[] args) {

Student s1 = new Student("Mushtaq", 20);

s1.display();

}

}

**File handling and I/O:**

import java.io.\*;

public class FileCopy {

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

FileInputStream in = new FileInputStream("source.txt");

FileOutputStream out = new FileOutputStream("dest.txt");

int c;

while ((c = in.read()) != -1) {

out.write(c);

}

in.close();

out.close();

System.out.println("File copied successfully.");

}

}

**Try catch Finally:**

**public class ExceptionDemo {**

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

**try {**

**int a = 10 / 0;**

**} catch (ArithmeticException e) {**

**System.out.println("Cannot divide by zero.");**

**} finally {**

**System.out.println("Finally block executed.");**

**}**

**}**

**}**

**Inheritance Example:**

class Animal {

void sound() {

System.out.println("Animal makes sound");

}

}

class Dog extends Animal {

void sound() {

System.out.println("Dog barks");

}

public static void main(String[] args) {

Dog d = new Dog();

d.sound(); // Method overriding

}

}

**Object-Array Example**

class Student {

String name;

int marks;

Student(String n, int m) {

name = n;

marks = m;

}

void display() {

System.out.println(name + ": " + marks);

}

public static void main(String[] args) {

Student[] arr = new Student[3];

arr[0] = new Student("Ali", 85);

arr[1] = new Student("Zara", 90);

arr[2] = new Student("Mushtaq", 95);

for (Student s : arr) {

s.display();

}

}

}

🧠 Section C – Long Answer Type Questions

**(2 × 15 = 30 Marks)**

**1. Exception Handling using try-catch-finally**

✅ Explanation:

Java uses try, catch, and finally blocks to handle exceptions gracefully:

* try: encloses code that might throw an exception.
* catch: handles the exception.
* finally: executes code regardless of exception outcome.

✅ Code Example:

public class ExceptionDemo { public static void main(String[] args) { try { int[] arr = {1, 2, 3}; System.out.println(arr[5]); // ArrayIndexOutOfBoundsException } catch (ArrayIndexOutOfBoundsException e) { System.out.println("Exception caught: " + e); } finally { System.out.println("This block always executes."); } } }

**2. Inheritance in Java**

✅ Explanation:

Inheritance allows one class to acquire properties and behaviors of another. Java supports **single**, **multilevel**, and **hierarchical** inheritance.

✅ Code Example:

class Vehicle { void start() { System.out.println("Vehicle starts"); } } class Car extends Vehicle { void drive() { System.out.println("Car is driving"); } } public class TestInheritance { public static void main(String[] args) { Car c = new Car(); c.start(); // inherited method c.drive(); // own method } }

**3. Object Arrays in Java**

✅ Explanation:

An array of objects stores multiple instances of a class. Useful for managing structured data like student records, books, etc.

✅ Code Example:

class Student { String name; int marks; Student(String n, int m) { name = n; marks = m; } void display() { System.out.println(name + " scored " + marks); } public static void main(String[] args) { Student[] students = new Student[3]; students[0] = new Student("Ali", 85); students[1] = new Student("Zara", 90); students[2] = new Student("Mushtaq", 95); for (Student s : students) { s.display(); } } }