---

1. Object-Oriented Programming with Java

Java Basics

- **History of Java**: Developed by James Gosling at Sun Microsystems in 1995. Key features: platform independence, robustness, and security.
- **Bytecode**: Java code is compiled into bytecode, which runs on the Java Virtual Machine (JVM).
- **Features of Java**:
    - Platform-independent (Write Once, Run Anywhere).
    - Object-oriented, robust, secure, and multithreaded.

Java Program Structure:
java
Copy
```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World!");
   }
```

- }
- **Data Types**:
    - Primitive: int, float, char, boolean, etc.
    - Non-primitive: String, Arrays, Classes.
- **Variables and Operators**:
    - Variables: int x = 10;
    - Operators: Arithmetic (+, -, *, /), Relational (==, !=), Logical (&&, ||).
- **Operator Precedence**: Determines the order of operations (e.g., * has higher precedence than +).
- **Control Statements**:
    - **Selection**: if, else, switch.
    - **Iteration**: for, while, do-while.
- **Scope of Variables**: Local (within a block), Instance (within a class), Static (shared across instances).

Classes and Objects

Defining Classes:
java
Copy
```
class Student {
   int marks;
   void display() {
      System.out.println("Marks: " + marks);
   }
```

- }

Creating Objects:
java
Copy
```
Student s1 = new Student();
s1.marks = 90;
```

- s1.display();
- **Automatic Garbage Collection**: Java automatically reclaims memory by destroying unused objects.

Arrays and Strings

Arrays:
java
Copy
```
int[] arr = {1, 2, 3};
```

- System.out.println(arr[0]); // Output: 1
- **Strings**:

String Class: Immutable.
java
Copy
```
String str = "Hello";
```

    - System.out.println(str.length()); // Output: 5

StringBuffer Class: Mutable.
java
Copy
```
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World");
```

- System.out.println(sb); // Output: Hello World

---

2. Classes and Inheritance

Inheritance

Single Inheritance:
java
Copy
```
class Animal {
   void eat() {
      System.out.println("Eating...");
   }
}
class Dog extends Animal {
   void bark() {
      System.out.println("Barking...");
   }
```

- }

Multiple Levels of Inheritance:
java
Copy
```
class BabyDog extends Dog {
   void weep() {
      System.out.println("Weeping...");
   }
```

- }

Abstract Classes:
java
Copy
```
abstract class Shape {
   abstract void draw();
}
class Circle extends Shape {
   void draw() {
      System.out.println("Drawing Circle");
   }
```

- }

- **Final Modifier**:
    - Prevents inheritance: final class A {}
    - Prevents method overriding: final void method() {}

Packages

Defining a Package:
java
Copy
package com.example;

- public class MyClass {}
- **Using Packages**:
  java
  Copy
  import com.example.MyClass;
- **CLASSPATH**: Specifies the location of user-defined classes.
- **Access Protection**:
    - public: Accessible everywhere.
    - protected: Accessible within the package and subclasses.
    - private: Accessible only within the class.

Exception Handling

- **Types of Exceptions**:
    - **Checked**: Compile-time (e.g., IOException).
    - **Unchecked**: Runtime (e.g., NullPointerException).

Handling Exceptions:
java
Copy
```
try {
   int x = 10 / 0;
} catch (ArithmeticException e) {
   System.out.println("Division by zero");
} finally {
   System.out.println("Finally block executed");
```

- }

Custom Exceptions:
java

Copy
```
class MyException extends Exception {
   MyException(String s) {
      super(s);
   }
```

- }

---

3. Practice Questions

Group-A (Very Short Answer Type)

Output of the Program:
java
Copy
```
public class JavaThreadsQuiz {
   public static void main(String[] args) {
      String name = Thread.currentThread().getName();
      System.out.println(name); // Output: main
   }
```

1. }
2. **Interfaces and Inner Classes**: FALSE (Interfaces cannot have inner classes).
3. **Final Class**: Yes, but it cannot be subclassed.
4. **Error in Code**: synchronized cannot be used with abstract methods.
5. **Feature of OOP**: Inheritance.
6. **Output of the Code**: 1111.
7. **Features of Java**: Platform independence, robustness, security.
8. **Types of Exceptions**: Checked and Unchecked.
9. **Static Members in Inner Classes**: Yes, static members of the outer class can be accessed.
10. **Member Inner Classes Inheritance**: FALSE (Inner classes are not inherited).

Group-B (Short Answer Type)

1. **Static Keyword**:
    - Used for memory management. Shared across all instances.

java
Copy
```
class Counter {
   static int count = 0;
```

```
Counter() { count++; }
```

2. }
3. **Data Types in Java**:
    - Primitive: int, float, char, boolean.
    - Non-primitive: String, Arrays, Classes.
4. **Abstraction vs Encapsulation**:
    - Abstraction: Hiding implementation details.
    - Encapsulation: Wrapping data and methods into a single unit.

Interface:
java
Copy
```
interface Animal {
   void eat();
}
class Dog implements Animal {
   public void eat() {
      System.out.println("Dog is eating");
   }
```

5. }
6. **JVM Architecture**:
    - Class Loader, Memory Area, Execution Engine.

Group-C (Long Answer Type)

Applet Skeleton:
java
Copy
```
import java.applet.Applet;
import java.awt.Graphics;
public class MyApplet extends Applet {
   public void paint(Graphics g) {
      g.drawString("Hello, World!", 50, 50);
   }
```

1. }
2. **Inheritance Types**:
    - Single, Multilevel, Hierarchical.

java
Copy
```
class A {}
```

class B extends A {}

3. class C extends B {}
4. **String vs StringBuffer vs StringBuilder**:
    - String: Immutable.
    - StringBuffer: Mutable, thread-safe.
    - StringBuilder: Mutable, not thread-safe.

Tips for Exam Preparation

1. Practice writing Java programs for all concepts.
2. Understand the differences between abstract classes and interfaces.
3. Focus on exception handling and multithreading.
4. Solve previous year's question papers.

Good luck with your exam! 🚀