Advanced Object-Oriented Programming (AOOP)

Most Important Questions and Well-Formatted Answers

## Q1. Explain features of Object-Oriented Programming.

Features of OOP include:  
1. Encapsulation: Bundling data and methods into a single unit (class).  
2. Inheritance: One class can inherit fields and methods from another.  
3. Polymorphism: Same method behaves differently based on the object.  
4. Abstraction: Hiding internal details and showing only essential information.  
5. Modularity: Code is divided into modules or classes.

## Q2. Compare Procedural and Object-Oriented Programming.

Procedural Programming vs Object-Oriented Programming:  
- Focuses on functions vs Focuses on objects and classes  
- No access specifiers vs Uses public, private, etc.  
- Hard to modify large code vs Easy to manage with modular code  
- Example: C vs Example: Java, C++

## Q3. Define class and object with example in Java.

A class is a blueprint for objects.  
  
Java Example:  
class Car {  
 String color;  
 void display() {  
 System.out.println(color);  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 Car c1 = new Car();  
 c1.color = "Red";  
 c1.display();  
 }  
}

## Q4. Explain constructor and its types with example.

A constructor is a special method used to initialize objects.  
Types:  
- Default Constructor  
- Parameterized Constructor  
- Constructor Overloading  
  
Java Example:  
class Student {  
 Student() { System.out.println("Default Constructor"); }  
 Student(String name) { System.out.println("Name: " + name); }  
}

## Q5. Explain method overloading and method overriding with example.

- Method Overloading: Same method name, different parameters.  
Example: int add(int a, int b); float add(float a, float b);  
- Method Overriding: Subclass redefines parent method.  
Example:  
class A { void show(){} }  
class B extends A { void show(){} }

## Q6. Difference between compile time and run time polymorphism.

Compile-time vs Run-time Polymorphism:  
- Method Overloading vs Method Overriding  
- Resolved during compilation vs Resolved during execution

## Q7. Explain inheritance and its types in Java.

Types of Inheritance:  
1. Single  
2. Multilevel  
3. Hierarchical  
4. Multiple (via interfaces)  
  
Example:  
class A {}  
class B extends A {}

## Q8. Explain interfaces with suitable example.

An interface contains abstract methods only.  
  
Java Example:  
interface Animal { void sound(); }  
class Dog implements Animal {  
 public void sound() { System.out.println("Bark"); }  
}

## Q9. What is exception? Explain try, catch, finally with example.

An exception is an event that disrupts program flow.  
  
Java Example:  
try {  
 int x = 5/0;  
} catch (ArithmeticException e) {  
 System.out.println("Error");  
} finally {  
 System.out.println("Done");  
}

## Q10. Explain multithreading in Java.

Multithreading allows executing multiple threads simultaneously.  
  
Java Example:  
class MyThread extends Thread {  
 public void run() {  
 for(int i=1;i<=5;i++) {  
 System.out.println(i);  
 }  
 }  
}

## Q11. Explain Java Virtual Machine (JVM).

JVM is a part of Java Runtime Environment that executes Java bytecode.

## Q12. What is static keyword in Java?

The static keyword defines class-level variables or methods.  
  
Example:  
static int count = 0;

## Q13. Difference between constructor and method.

Constructor vs Method:  
- Called when object is created vs Called manually  
- No return type vs Has return type

## Q14. What is dynamic method dispatch?

Dynamic method dispatch is runtime polymorphism using overridden methods.

## Q15. Difference between abstract class and interface.

Abstract Class vs Interface:  
- Can have concrete methods vs All methods abstract (Java 7)  
- Can have constructors vs Cannot have constructors

## Q16. What is final keyword? Explain with example.

Used to restrict modification.  
  
Example:  
final int x = 10;

## Q17. Explain thread lifecycle in Java.

Thread Lifecycle States: New, Runnable, Running, Blocked, Terminated.

## Q18. Explain types of exceptions in Java.

Types:  
- Checked Exception (e.g., IOException)  
- Unchecked Exception (e.g., ArithmeticException)