Object-Oriented Programming in Java Exam

**Course:**   
**Author:**   
**Date:**   
**Additional Info:**

* Q1: Define the concept of Object-Oriented Programming (OOP) and list its four main principles.

A1: OOP is a programming paradigm based on the concept of objects. The four main principles are: Encapsulation, Inheritance, Polymorphism, and Abstraction.

* Q2: Explain the difference between abstraction and encapsulation with examples.

A2: Abstraction hides implementation details and shows only the functionality, e.g., using an interface. Encapsulation wraps data and methods into a single unit and restricts direct access to it, e.g., using private variables and getters/setters.

* Q3: What is inheritance? Write a simple Java example demonstrating single inheritance.

A3: Inheritance is a mechanism where one class acquires the properties and behaviors of another class. Example: class Animal { void makeSound() { System.out.println('Sound'); } } class Dog extends Animal { void bark() { System.out.println('Bark'); } }

* Q4: Differentiate between method overloading and method overriding in Java.

A4: Method overloading allows multiple methods with the same name but different parameter lists within a class. Method overriding allows a subclass to provide a specific implementation of a method already defined in its superclass.

* Q5: What is a constructor? How is it different from a method?

A5: A constructor is a special method used to initialize an object. Unlike methods, constructors: Have the same name as the class, Do not have a return type, Are called automatically when an object is created.

* Q6: Describe polymorphism in Java and provide a code snippet illustrating runtime polymorphism.

A6: Polymorphism allows objects to be treated as instances of their parent class. Example of runtime polymorphism: class Animal { void makeSound() { System.out.println('Sound'); } } class Dog extends Animal { void makeSound() { System.out.println('Bark'); } } Animal obj = new Dog(); obj.makeSound(); // Outputs: Bark

* Q7: Explain the use of the 'super' keyword in Java with an example.

A7: The 'super' keyword refers to the superclass of the current object. It is used to call superclass methods or constructors. Example: class Animal { Animal() { System.out.println('Animal created'); } } class Dog extends Animal { Dog() { super(); System.out.println('Dog created'); } }

* Q8: What is an interface in Java? How is it different from an abstract class?

A8: An interface in Java is a reference type that contains abstract methods. It differs from an abstract class as follows: Interfaces support multiple inheritance, abstract classes do not. Interfaces cannot have instance fields, while abstract classes can.

* Q9: Discuss the concept of exception handling in Java. Provide an example using try-catch-finally.

A9: Exception handling in Java manages runtime errors, ensuring normal flow of execution. Example: try { int a = 10/0; } catch (ArithmeticException e) { System.out.println('Cannot divide by zero'); } finally { System.out.println('End'); }

* Q10: What are inner classes in Java? Explain the types of inner classes with examples.

A10: Inner classes are classes defined within another class. Types: Static nested class, Inner class, Method-local inner class, Anonymous inner class.