**Summary of Key OOP Principles and Concepts:**

1. **Encapsulation**
   * Encapsulates data (variables) and methods in a single unit (class).
   * Restricts access to class components using access modifiers (e.g., private, public).
   * Access to private variables is provided through public getter and setter methods.
2. **Abstraction**
   * Hides implementation details while exposing essential features.
   * Achieved using:
     + **Abstract Classes**: Contain abstract methods (no implementation).
     + **Interfaces**: Define methods that must be implemented by the class.
3. **Inheritance**
   * Allows a child class to inherit methods and fields from a parent class.
   * Promotes code reusability and hierarchical relationships.
   * Use the extends keyword to implement inheritance.
4. **Polymorphism**
   * Enables methods to behave differently based on the object.
   * Types:
     + **Method Overloading**: Same method name, different parameter lists.
     + **Method Overriding**: Subclass provides a specific implementation of a superclass method.
5. **Classes and Objects**
   * **Class**: A blueprint for creating objects.
   * **Object**: An instance of a class with specific attributes and behavior.
6. **this Keyword**
   * Refers to the current instance of a class.
   * Used to differentiate between instance variables and parameters with the same name.
7. **Constructor and Constructor Overloading**
   * **Constructor**: Special method to initialize objects when created.
   * **Overloading**: Multiple constructors with different parameter lists to initialize objects in various ways.
8. **Static Members**
   * **Static Variables**: Shared among all instances of a class.
   * **Static Methods**: Can be called without creating an object of the class.
9. **Nested and Inner Classes**
   * **Nested Class**: A class defined inside another class.
   * **Inner Class**: Non-static nested class that can access the outer class's members.

**Examples of Each Concept**

* **Encapsulation**: Use private variables with public getters/setters.
* **Abstraction**: Abstract classes (e.g., abstract class Animal) or interfaces (e.g., interface Flyable).
* **Inheritance**: class Dog extends Animal inherits Animal methods.
* **Polymorphism**: Animal animal = new Dog(); calls overridden sound() method in Dog.
* **Static Members**: Counter.count is shared across all objects.
* **Inner Classes**: OuterClass.InnerClass accesses outer class variables.

4o