### **Workshop Explanation:**

This workshop is designed to give students a practical understanding of Object-Oriented Programming concepts, focusing on **inheritance** in Java. Students will learn how to create and use inheritance hierarchies, work with base and derived classes, and understand overriding, polymorphism, and the super keyword.

### **Flow:**

1. **Introduce the Concept of Inheritance** (short discussion):

• Explain the purpose of inheritance.

• Discuss key terms like parent class (superclass), child class (subclass), and polymorphism.

• Demonstrate real-world examples (e.g., Animal -> Dog and Cat).

2. **Hands-On Implementation**:

• Implement and extend classes with inheritance.

• Work on practical examples to solidify understanding.

### **Workshop Structure**

**1. Shape Hierarchy (Shape.java, Circle.java, Rectangle.java)**

**Purpose**: Demonstrates basic inheritance and overriding of methods.

**Implementation**:

• Students implement a Shape base class with attributes like color and methods like area() and perimeter() (both abstract).

• Extend the Shape class to create Circle and Rectangle classes.

**Activity**:

• Calculate and print the area and perimeter for different shapes.

• Compare outputs for various shapes.

• Discuss the benefits of using abstract classes.

**2. Employee Hierarchy (Employee.java, Manager.java, Developer.java)**

**Purpose**: Shows how inheritance can model real-world hierarchies.

**Implementation**:

• Create an Employee base class with attributes like name, id, and salary, and a method calculateBonus().

• Extend the Employee class to create Manager and Developer classes, overriding the calculateBonus() method based on role.

**Activity**:

• Instantiate objects for different employees and calculate their bonuses.

• Add a toString() method to demonstrate polymorphism during object printing.

**3. Zoo Management System (Animal.java, Mammal.java, Bird.java)**

**Purpose**: Combines inheritance and polymorphism for a more complex application.

**Implementation**:

• Build an Animal base class with attributes like name and age, and methods like makeSound() and move().

• Extend Animal to create Mammal and Bird classes, adding unique methods (fly() for birds, run() for mammals).

**Activity**:

• Create a zoo by instantiating animals and invoking their methods.

• Use polymorphism to iterate over a list of animals and call makeSound() dynamically.

**General Guidelines for Students**

• **Plan before coding**: Understand the class structure and relationships.

• **Focus on DRY**: Avoid code duplication by properly utilizing inheritance.

• **Experiment with Overriding**: Try calling overridden methods from the base class using the super keyword.

• **Explore Polymorphism**: Use base class references to work with derived class objects.