Lab 03 & 04: Inheritance and Polymorphism

Objectives:

- Understand the concept of inheritance and its benefits in code reusability.
- Implement inheritance using `extends` keyword.
- Practice method overriding and polymorphism.
- Explore real-world applications of inheritance.

Theory:

- **Inheritance:** The process of creating a new class (subclass or derived class) from an existing class (superclass or base class). It promotes code reusability and hierarchical relationships between classes.
- **Method Overriding:** The ability of a subclass to provide a specific implementation of a method that is already defined in its superclass.
- **Polymorphism:** The ability of objects of different types to be treated as if they were of the same type. It allows for flexible code design.

Example:

```
class Animal {
    String name;

    public void makeSound() {
        System.out.println("Generic animal sound");
    }
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Woof!");

    }
}

class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Meow!");


    }
}
```

Exercises:

1. Animal Kingdom:

- Create classes for different animals (e.g., Dog, Cat, Bird) inheriting from an Animal class.
- Implement specific sounds for each animal.
- Create an array of Animal objects and demonstrate polymorphism by calling the `makeSound()` method on each object.

2. Shape Hierarchy:

- Create a class named `Shape` with a method to calculate area (abstract).
- Create subclasses for `Circle`, `Rectangle`, and `Triangle` inheriting from `Shape`.
- Implement the `calculateArea()` method in each subclass accordingly.
- Create an array of `Shape` objects and calculate the total area of all shapes.

3. Employee Management:

- Create a class named `Employee` with attributes like name, ID, and salary.
- Create subclasses for `Manager`, `Developer`, and `Tester` inheriting from `Employee`.
- Add specific attributes and methods for each subclass (e.g., `bonus` for Manager, `projectsHandled` for Developer).
- Calculate the total salary for all employees, considering any bonuses or allowances.

4. Vehicle Inheritance:

- Create a class named `Vehicle` with attributes like model, year, and color.
- Create subclasses for `Car`, `Motorcycle`, and `Truck` inheriting from `Vehicle`.
- Add specific attributes and methods for each vehicle type (e.g., `numberOfDoors` for Car, `engineCapacity` for Motorcycle).
- Implement a method to display vehicle details for each type.

5. Shape Calculator with Polymorphism:

- Create a class named `Shape` with an abstract method `calculateArea()`.
- Create subclasses for `Circle`, `Rectangle`, and `Triangle` implementing the `calculateArea()` method.
- Create an array of `Shape` objects and calculate the total area of all shapes using polymorphism.

Roll No: _____

Name: _____

Subject: _____

Teacher Name:_____