```java
// Fruit and Apple Classes
// Topic: Basic Single Inheritance
// Problem Statement:
// Create a Fruit class with color and taste fields. Create an Apple class
that extends
// Fruit and adds variety field.
// Hints:
// • Use extends keyword for inheritance
// • Make fields protected in parent class
// • Test by creating Apple object and accessing inherited fields


// Parent class
class Fruit {
    protected String color;
    protected String taste;

    public Fruit(String color, String taste) {
        this.color = color;
        this.taste = taste;
    }

    public void displayFruitInfo() {
        System.out.println("Color: " + color);
        System.out.println("Taste: " + taste);
    }
}

// Child class
class Apple extends Fruit {
    private String variety;

    public Apple(String color, String taste, String variety) {
        super(color, taste); // Call parent constructor
        this.variety = variety;
    }

    public void displayAppleInfo() {
        displayFruitInfo(); // Access inherited method
        System.out.println("Variety: " + variety);
```

```
    }
}

// Test class
public class AppleInheritanceDemo {
    public static void main(String[] args) {
        Apple myApple = new Apple("Red", "Sweet", "Fuji");
        myApple.displayAppleInfo();
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> java AppleInheritanceDemo
Color: Red
Taste: Sweet
Variety: Fuji
```

```
// Phone and SmartPhone
// Constructors
// Topic: Constructor Chaining with super()
// Problem Statement:
// Create Phone class with brand and model. Create SmartPhone class
extending Phone with
// operatingSystem field. Use constructor chaining.
// Hints:
// • Add print statements in constructors to see execution order
// • Use super() in child constructor
// • Create objects using different constructor combinations


// Parent class
class Phone {
    protected String brand;
    protected String model;

    // Constructor
    public Phone(String brand, String model) {
        System.out.println("Phone constructor called");
        this.brand = brand;
```

```
        this.model = model;
    }

    public void displayPhoneInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
    }
}

// Child class
class SmartPhone extends Phone {
    private String operatingSystem;

    // Constructor chaining with super()
    public SmartPhone(String brand, String model, String operatingSystem)
{
        super(brand, model); // Call parent constructor
        System.out.println("SmartPhone constructor called");
        this.operatingSystem = operatingSystem;
    }

    public void displaySmartPhoneInfo() {
        displayPhoneInfo(); // Inherited method
        System.out.println("Operating System: " + operatingSystem);
    }
}

// Test class
public class SmartPhoneConstructorDemo {
    public static void main(String[] args) {
        System.out.println("Creating SmartPhone object...");
        SmartPhone myPhone = new SmartPhone("Samsung", "Galaxy S21",
"Android");
        myPhone.displaySmartPhoneInfo();
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> javac SmartPhoneConstructorDemo.java
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> java SmartPhoneConstructorDemo
Creating SmartPhone object...
Phone constructor called
SmartPhone constructor called
Brand: Samsung
Model: Galaxy S21
Operating System: Android
```

```java
// Bird Flying Behavior
// Topic: Method Overriding with @Override
// 1

// Problem Statement:
// Create Bird class with fly() method. Create Penguin and Eagle classes
that override
// fly() method differently.
// Hints:
// • Use @Override annotation
// • Make different implementations in each child class
// • Test polymorphism with array of Bird references


// Superclass
class Bird {
    public void fly() {
        System.out.println("Bird is flying...");
    }
}

// Subclass: Penguin
class Penguin extends Bird {
    @Override
    public void fly() {
        System.out.println("Penguin can't fly, it waddles instead.");
    }
}

// Subclass: Eagle
```

```java
class Eagle extends Bird {
    @Override
    public void fly() {
        System.out.println("Eagle soars high in the sky.");
    }
}

// Test class
public class BirdFlyingBehaviorDemo {
    public static void main(String[] args) {
        Bird[] birds = new Bird[3];
        birds[0] = new Bird();
        birds[1] = new Penguin();
        birds[2] = new Eagle();

        for (Bird b : birds) {
            b.fly(); // Polymorphic behavior
        }
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> javac BirdFlyingBehaviorDemo.java
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> java BirdFlyingBehaviorDemo
Bird is flying...
Penguin can't fly, it waddles instead.
Eagle soars high in the sky.
```

```java
// Phone and SmartPhone
// Constructors
// Topic: Constructor Chaining with super()
// Problem Statement:
// Create Phone class with brand and model. Create SmartPhone class
extending Phone with
// operatingSystem field. Use constructor chaining.
// Hints:
// • Add print statements in constructors to see execution order
// • Use super() in child constructor
// • Create objects using different constructor combinations
```

```java
// Parent class
class Phone {
    protected String brand;
    protected String model;

    // Constructor
    public Phone(String brand, String model) {
        System.out.println("Phone constructor called");
        this.brand = brand;
        this.model = model;
    }

    public void displayPhoneInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
    }
}

// Child class
class SmartPhone extends Phone {
    private String operatingSystem;

    // Constructor chaining with super()
    public SmartPhone(String brand, String model, String operatingSystem)
{
        super(brand, model); // Call parent constructor
        System.out.println("SmartPhone constructor called");
        this.operatingSystem = operatingSystem;
    }

    public void displaySmartPhoneInfo() {
        displayPhoneInfo(); // Inherited method
        System.out.println("Operating System: " + operatingSystem);
    }
}

// Test class
public class SmartPhoneConstructorDemo {
```

```java
    public static void main(String[] args) {
        System.out.println("Creating SmartPhone object...");
        SmartPhone myPhone = new SmartPhone("Samsung", "Galaxy S21",
"Android");
        myPhone.displaySmartPhoneInfo();
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> javac SmartPhoneConstructorDemo.java
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\lab-work> java SmartPhoneConstructorDemo
Creating SmartPhone object...
Phone constructor called
SmartPhone constructor called
Brand: Samsung
Model: Galaxy S21
Operating System: Android
```