Abstraction

🎯 Direct Challenges

1. Create an abstract class Shape with an abstract method draw()

```
abstract class Shape {
    abstract void draw();
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing Circle");
    }
}

class Square extends Shape {
    void draw() {
        System.out.println("Drawing Square");
    }
}

public class Main {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Square();
        s1.draw();
        s2.draw();
    }
}
```

Output:

```
Drawing Circle
Drawing Square
```

2. Implement the abstract class in Circle and Square

```
abstract class Shape {
   abstract void draw();
}

class Circle extends Shape {
   void draw() {
      System.out.println("Drawing Circle");
   }
}

class Square extends Shape {
   void draw() {
      System.out.println("Drawing Square");
   }
}

public class Main {
   public static void main(String[] args) {
      Shape s1 = new Circle();
      Shape s2 = new Square();
      s1.draw();
      s2.draw();
   }
}
```

Output:

```
Drawing Circle
Drawing Square
```

3. Show partial abstraction using non-abstract and abstract methods

```
abstract class Vehicle {
    abstract void start();

    void fuelType() {
        System.out.println("Vehicle uses petrol or diesel");
    }
}

class Bike extends Vehicle {
    void start() {
        System.out.println("Bike starts with kick");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle v = new Bike();
        v.start();
        v.fuelType();
    }
}
```

Output:

```
Bike starts with kick
Vehicle uses petrol or diesel
```

🧩 Scenario-Based Challenges

1. Define a class Employee with abstract method calculateSalary() and implement in FullTime and PartTime subclasses

```
abstract class Employee {
    String name;
    Employee(String name) {
        this.name = name;
    }

    abstract void calculateSalary();
}

class FullTime extends Employee {
    FullTime(String name) {
        super(name);
    }

    void calculateSalary() {
        System.out.println(name + "'s salary is 50000 per month");
    }
}

class PartTime extends Employee {
    PartTime(String name) {
        super(name);
    }

    void calculateSalary() {
        System.out.println(name + "'s salary is 20000 per month");
    }
}

public class Main {
    public static void main(String[] args) {
        Employee e1 = new FullTime("Ravi");
        Employee e2 = new PartTime("Priya");
        e1.calculateSalary();
        e2.calculateSalary();
    }
}
```

Output:

```
Ravi's salary is 50000 per month
Priya's salary is 20000 per month
```

2. Create an abstract Appliance class and implement it for Fan and AC

```
abstract class Appliance {
    abstract void operate();
}

class Fan extends Appliance {
    void operate() {
        System.out.println("Fan is spinning");
    }
}

class AC extends Appliance {
    void operate() {
        System.out.println("AC is cooling the room");
    }
}

public class Main {
    public static void main(String[] args) {
        Appliance a1 = new Fan();
        Appliance a2 = new AC();
        a1.operate();
        a2.operate();
    }
}
```

Output:

```
Fan is spinning
AC is cooling the room
```