Exercice 1 : Définir une Interface et Implémenter une Classe

1. **Définissez une interface `Animal` avec les méthodes suivantes :**
   - `void makeSound();`
   - `void eat();`
2. **Créez une classe `Dog` qui implémente l'interface `Animal`. Dans cette classe, implémentez les méthodes :**
   - `makeSound()` qui affiche "Woof"
   - `eat()` qui affiche "Dog is eating"
3. **Créez une classe `Cat` qui implémente l'interface `Animal`. Dans cette classe, implémentez les méthodes :**
   - `makeSound()` qui affiche "Meow"
   - `eat()` qui affiche "Cat is eating"
4. **Créez une classe principale `Main` avec une méthode `main` pour tester vos classes `Dog` et `Cat`.**

```java
// Define the Animal interface
interface Animal {
    void makeSound();
    void eat();
}

// Implement the Dog class
class Dog implements Animal {
    @Override
    public void makeSound() {
        System.out.println("Woof");
    }

    @Override
    public void eat() {
        System.out.println("Dog is eating");
    }
}

// Implement the Cat class
class Cat implements Animal {
    @Override
    public void makeSound() {
        System.out.println("Meow");
```

```
    }

    @Override
    public void eat() {
        System.out.println("Cat is eating");
    }
}

// Main class to test Dog and Cat classes
public class Main {
    public static void main(String[] args) {
        Animal dog = new Dog();
        dog.makeSound();
        dog.eat();

        Animal cat = new Cat();
        cat.makeSound();
        cat.eat();
    }
}
```

Exercice 2 : Interface avec Paramètres et Valeurs de Retour

1. **Définissez une interface `Shape` avec les méthodes suivantes :**
   - `double getArea();`
   - `double getPerimeter();`
2. **Créez une classe `Circle` qui implémente l'interface `Shape`. Dans cette classe :**
   - Ajoutez un champ `radius`.
   - Implémentez les méthodes `getArea()` et `getPerimeter()`. Utilisez la formule (Area = PI ×radius^2) et (Perimeter = 2 ×PI ×radius).
3. **Créez une classe `Rectangle` qui implémente l'interface `Shape`. Dans cette classe :**
   - Ajoutez les champs `width` et `height`.
   - Implémentez les méthodes `getArea()` et `getPerimeter()`. Utilisez la formule (Area = width ×height) et (Perimeter = 2 ×(width + height)).
4. **Créez une classe principale `Main` avec une méthode `main` pour tester vos classes `Circle` et `Rectangle`.**

```java
// Define the Shape interface
interface Shape {
    double getArea();
    double getPerimeter();
}

// Implement the Circle class
class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }

    @Override
    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }
}

// Implement the Rectangle class
class Rectangle implements Shape {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    @Override
    public double getArea() {
        return width * height;
    }
```

```
    @Override
    public double getPerimeter() {
        return 2 * (width + height);
    }
}

// Main class to test Circle and Rectangle classes
public class Main {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        System.out.println("Circle Area: " + circle.getArea());
        System.out.println("Circle Perimeter: " + circle.getPerimeter());

        Shape rectangle = new Rectangle(4, 7);
        System.out.println("Rectangle Area: " + rectangle.getArea());
        System.out.println("Rectangle Perimeter: " + rectangle.getPerimeter());
    }
}
```

Exercice 3 : Interface Héritée

1. **Définissez une interface `Movable` avec les méthodes suivantes :**
   - `void move(int x, int y);`
   - `void stop();`
2. **Définissez une interface `Vehicle` qui hérite de `Movable` et ajoute les méthodes suivantes :**
   - `void start();`
   - `void accelerate(int speed);`
3. **Créez une classe `Car` qui implémente l'interface `Vehicle`. Dans cette classe :**
   - Implémentez les méthodes `start()`, `accelerate(int speed)`, `move(int x, int y)`, et `stop()` avec des affichages appropriés.
4. **Créez une classe principale `Main` avec une méthode `main` pour tester votre classe `Car`.**

```
// Define the Movable interface
interface Movable {
    void move(int x, int y);
    void stop();
```

```
}

// Define the Vehicle interface
interface Vehicle extends Movable {
    void start();
    void accelerate(int speed);
}

// Implement the Car class
class Car implements Vehicle {
    @Override
    public void start() {
        System.out.println("Car started");
    }

    @Override
    public void accelerate(int speed) {
        System.out.println("Car is accelerating to " + speed + " km/h");
    }

    @Override
    public void move(int x, int y) {
        System.out.println("Car is moving to coordinates (" + x + ", " + y + ")");
    }

    @Override
    public void stop() {
        System.out.println("Car stopped");
    }
}

// Main class to test Car class
public class Main {
    public static void main(String[] args) {
        Vehicle car = new Car();
        car.start();
        car.accelerate(60);
        car.move(10, 20);
        car.stop();
    }
```

```
}
```