🅢 - Single Responsibility Principle (SRP)

❌ **Bad Example:**

```
public class User {
    public String name;
    public String email;
    public void saveUser() {
        // Save to DB
    }
    public void sendEmail(String message) {
        // Send email
    }
}
```

✅ **Good Example:**

```
public class User {
    public String name;
    public String email;
}

public class UserRepository {
    public void saveUser(User user) {
        // Save to DB
    }
}
```

```
public class EmailService {
    public void sendEmail(String to, String message) {
        // Send email
    }
}
```

◉ - Open/Closed Principle (OCP)

❌ **Bad Example:**

```
public class DiscountCalculator {
    public double calculateDiscount(String type) {
        if (type.equals("Regular")) {
            return 0.1;
        } else if (type.equals("VIP")) {
            return 0.2;
        }
        return 0;
    }
}
```

✅ **Good Example:**

```
public interface DiscountStrategy {
    double getDiscount();
}
```

```
public class RegularCustomer implements DiscountStrategy {
    public double getDiscount() {
        return 0.1;
    }
}

public class VIPCustomer implements DiscountStrategy {
    public double getDiscount() {
        return 0.2;
    }
}

public class DiscountCalculator {
    public double calculateDiscount(DiscountStrategy customer) {
        return customer.getDiscount();
    }
}
```

🅛 - Liskov Substitution Principle (LSP)

❌ Bad Example:

```
public class Bird {
    public void fly() {
        System.out.println("Flying");
    }
```

```
}


public class Ostrich extends Bird {
    @Override
    public void fly() {
        throw new UnsupportedOperationException("Ostrich can't
fly");
    }
}
```

✅ **Good Example:**

```
public abstract class Bird {
    public abstract void move();
}
public class Sparrow extends Bird {
    public void move() {
        System.out.println("Flying");
    }
}
public class Ostrich extends Bird {
    public void move() {
        System.out.println("Running");
    }
}
```

❶ - Interface Segregation Principle (ISP)

❌ Bad Example:

```
public interface Worker {
    void work();
    void eat();
}
public class Robot implements Worker {
    public void work() {
        // Working
    }
    public void eat() {
        // Not applicable
        throw new UnsupportedOperationException();
    }
}
```

✅ Good Example:

```
public interface Workable {
    void work();
}
public interface Eatable {
    void eat();
}
public class Human implements Workable, Eatable {
```

```
    public void work() {
        // Working
    }
    public void eat() {
        // Eating
    }
}
public class Robot implements Workable {
    public void work() {
        // Working
    }
}
```

🅓 - Dependency Inversion Principle (DIP)

❌ Bad Example:

```
public class LightBulb {
    public void turnOn() {
        System.out.println("Bulb ON");
    }
    public void turnOff() {
        System.out.println("Bulb OFF");
    }
```

```
}
public class Switch {
    private LightBulb bulb = new LightBulb();
    public void operate() {
        bulb.turnOn();
    }
}
```

✅ **Good Example:**

```
public interface Switchable {
    void turnOn();
    void turnOff();
}
public class LightBulb implements Switchable {
    public void turnOn() {
        System.out.println("Bulb ON");
    }
    public void turnOff() {
        System.out.println("Bulb OFF");
    }
}
public class Switch {
    private Switchable device;
    public Switch(Switchable device) {
```

```
        this.device = device;
    }
    public void operate() {
        device.turnOn();
    }
}
```