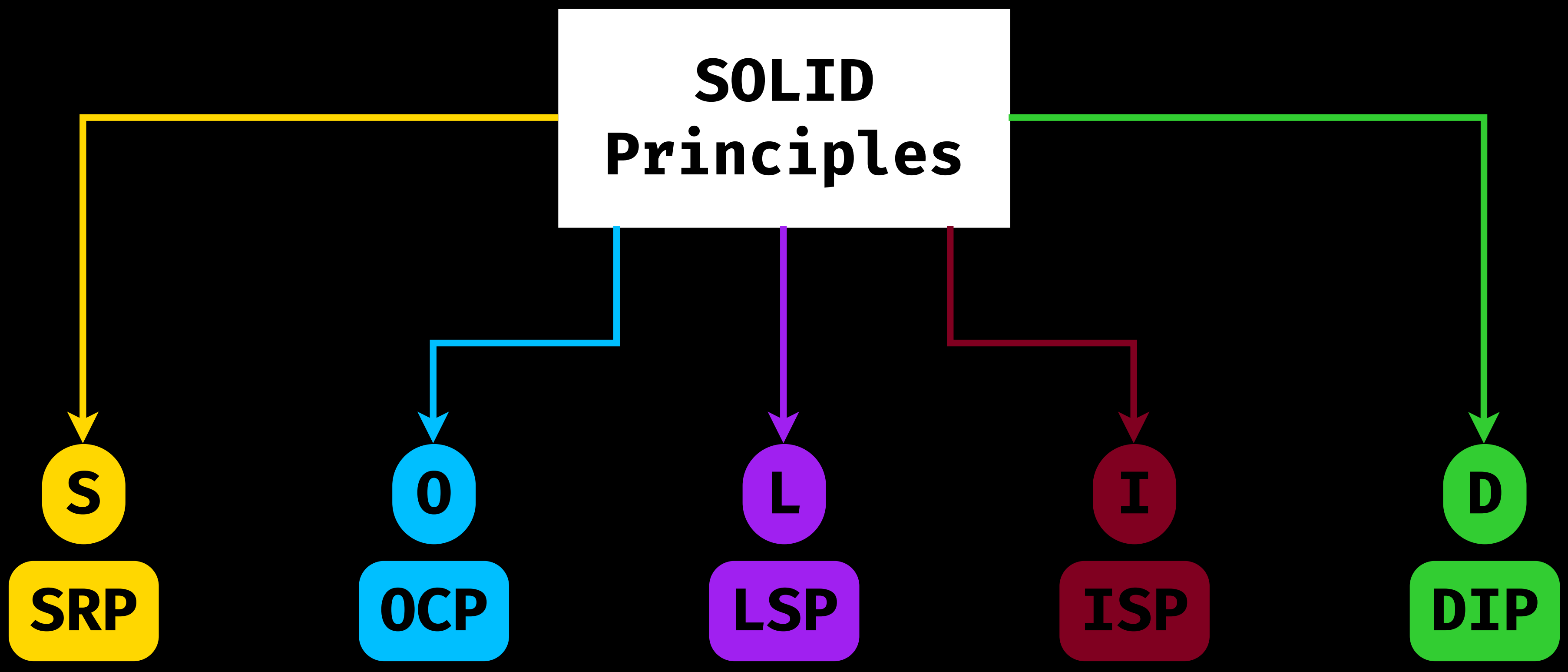

Single Responsibility Principle

يجب أن تكون كل class
مسؤولة عن سبب واحد فقط
للتغيير، أي أن تقوم بمهمة واحدة
فقط

Violation

☒

```
class Report {
    public void generateReport() {

    }

    public void printReport() {

    }
}
```

Compliance

☑

```
class Report {
    public void generateReport() {

    }

    public void printReport() {

    }
}
```

Open/Closed Principle

يجب أن تكون class مفتوحة
للتوسع لكنها مغلقة للتعديل

Violation

☒

```
class Discount {
    public double calculateDiscount(String type, double price) {
        if (type.equals("Student")) {
            return price * 0.1;
        } else if (type.equals("Senior")) {
            return price * 0.2;
        }
        return 0;
    }
}
```

Compliance

☑

```
interface Discount {
    double calculate(double price);
}

class StudentDiscount implements Discount {
    public double calculate(double price) {
        return price * 0.1;
    }
}

class SeniorDiscount implements Discount {
    public double calculate(double price) {
        return price * 0.2;
    }
}
```

Liskov Substitution Principle

يجب أن تكون class الفرعية
قابلة للاستبدال بـ class
الأساسية دون كسر النظام

Violation

☒

```
class Rectangle {
    private int width;
    private int height;

    public void setWidth(int width) {
        this.width = width;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public int getArea() {
        return width * height;
    }
}

class Square extends Rectangle {
    @Override
    public void setWidth(int width) {
        super.setWidth(width);
        super.setHeight(width);
    }

    @Override
    public void setHeight(int height) {
        super.setWidth(height);
        super.setHeight(height);
    }
}
```

Compliance

☑

```
interface Shape {
    int getArea();
}

abstract class Quadrilateral implements Shape {
    abstract void setWidth(int width);
    abstract void setHeight(int height);
}

class Rectangle extends Quadrilateral {
    private int width;
    private int height;

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

    public void setWidth(int width) {
        this.width = width;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public int getArea() {
        return width * height;
    }
}

class Square extends Quadrilateral {
    private int side;

    public Square(int side) {
        this.side = side;
    }

    public void setWidth(int width) {
        this.side = width;
    }

    public void setHeight(int height) {
        this.side = height;
    }

    public int getArea() {
        return side * side;
    }
}
```

Interface Segregation Principle

يجب ألا تُجبر class على تنفيذ
واجهات لا تحتاجها

Violation

☒

```
interface Worker {
    void work();

    void eat();
}

class Robot implements Worker {
    public void work() {

    }

    public void eat() {

    }
}
```

Compliance

☑

```
interface Workable {
    void work();
}

interface Eatable {
    void eat();
}

class Robot implements Workable {
    public void work() {

    }
}

class Human implements Workable, Eatable {
    public void work() {

    }

    public void eat() {

    }
}
```

Dependency Inversion Principle

يجب أن تعتمد
high-level modules
abstractions على
وليس على
low-level modules

Violation

☒

```
class Keyboard {

}

class Monitor {

}

class Computer {
    private Keyboard keyboard;
    private Monitor monitor;

    public Computer() {
        this.keyboard = new Keyboard();
        this.monitor = new Monitor();
    }
}
```

Compliance

☑

```
interface InputDevice {

}

class Keyboard implements InputDevice {

}


interface DisplayDevice {

}

class Monitor implements DisplayDevice {

}

class Computer {
    private InputDevice inputDevice;
    private DisplayDevice displayDevice;

    public Computer(InputDevice inputDevice, DisplayDevice displayDevice) {
        this.inputDevice = inputDevice;
        this.displayDevice = displayDevice;
    }
}
```

يوتيوب
Youtube
@Nullexia

تيليجرام
Telegram
@Nullexia

Nullexia