OPEN-CLOSED PRINCIPLE

Evdokimova Dasha, Steshchenko Nastya
21205

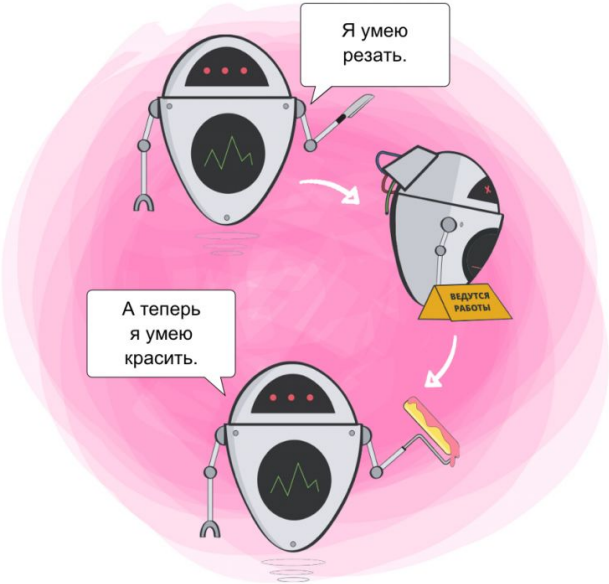
DEFINITION OF PRINCIPLE

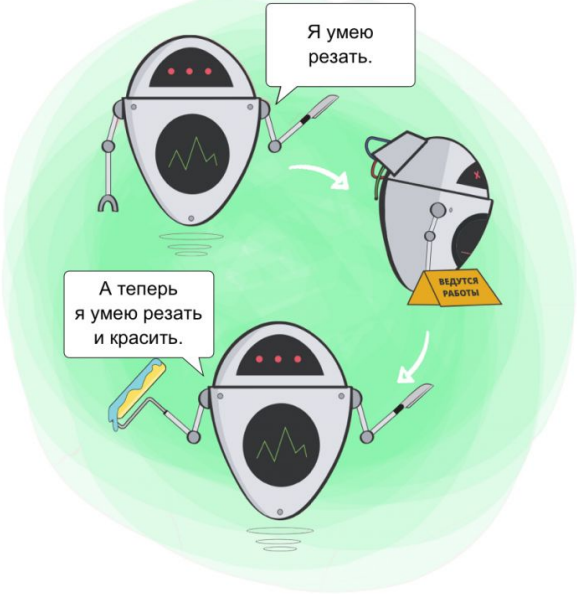
Open-closed principle

The Open Closed principle, first conceptualized by Bertrand Meyer is a Object oriented software design principle. It goes as below:

SOFTWARE ENTITIES (CLASSES, MODULES, FUNCTIONS, ETC.) <u>SHOULD BE OPEN FOR EXTENSION, BUT CLOSED FOR MODIFICATION.</u>

A picture is worth a thousand words

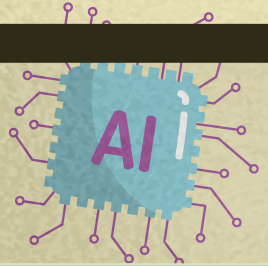




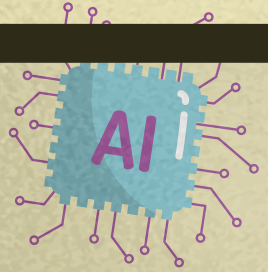
Принцип открытости/закрытости

EXAMPLE OF VIOLATION OF PRINCIPLE

```
class Circle {
    private int radius;
    public Circle(int radius) {
        this.radius = radius;
    }
    public int getRadius() {
        return radius;
    }
}
```



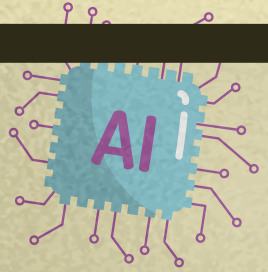

```
class Square {
    private int side;
    public Square(int side){
        this.side = side;
    }
    public int getSide() {
    return side;
    }
}
```




```
class DrawShapes {
    public void draw(Object[] shapes) {
        for (Object shape : shapes) {
            if (shape instanceof Circle) {
                Circle c = (Circle) shape;
                System.out.println("Drawing Circle with radius "
                    + c.getRadius());
            }
            else if (shape instanceof Square) {
                Square s = (Square) shape;
                System.out.println("Drawing Square with side "
                    +   s.getSide());
```

```
            }
        }
    }
}
```

EXAMPLE OF GOOD REALIZATION



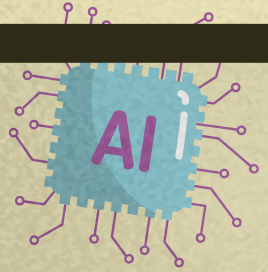

```
abstract class Shape {
    abstract void draw();
}
```

```
class Circle extends Shape {
	private int radius;
	public Circle(int radius) {
		this.radius = radius;
	}
	public int getRadius() {
		return radius;
	}

	@Override
	void draw() {
		System.out.println
		("Drawing Circle with
		radius " + this.getRadius());
	}
}
```

```
class Square extends Shape {
	private int side;
	public Square(int side) {
		this.side = side;
	}
	public int getSide() {
		return side;
	}

	@Override
	void draw() {
	System.out.println
	("Drawing Square with side " +
		this.getSide());
	}
}
```




```
class DrawShapes {
    public void draw(Shape[] shapes) {
        for (Shape shape : shapes) {
            shape.draw();
        }
    }
}
```

Conclusion of the above examples

The Open Closed Principle is dependent on basics of Object Oriented Design - Inheritance, Abstraction and Encapsulation.

Important consideration #1

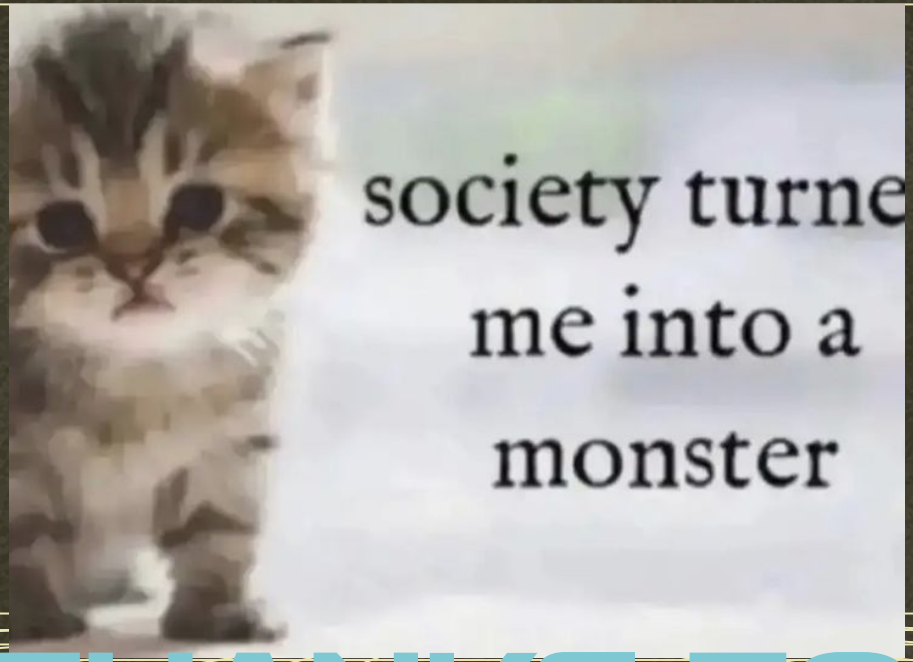
Private member variables When a member variable in a class is public or protected, it can be modified from other classes and sub-classes. Hence, it should be the intention of the developer to make the member variables private.

Important consideration #2

Final member variables Building on Point 1, there might be member variables that never should be modified. These should be declared as final so that they are closed for modification.

Important consideration #3

No global variables No global variables should be left available for modification. All global variables should be declared as static final.



THANKS FOR WATCHING!!!