OCP and Liskov Principles

SoftUni Team

Technical Trainers

Software University

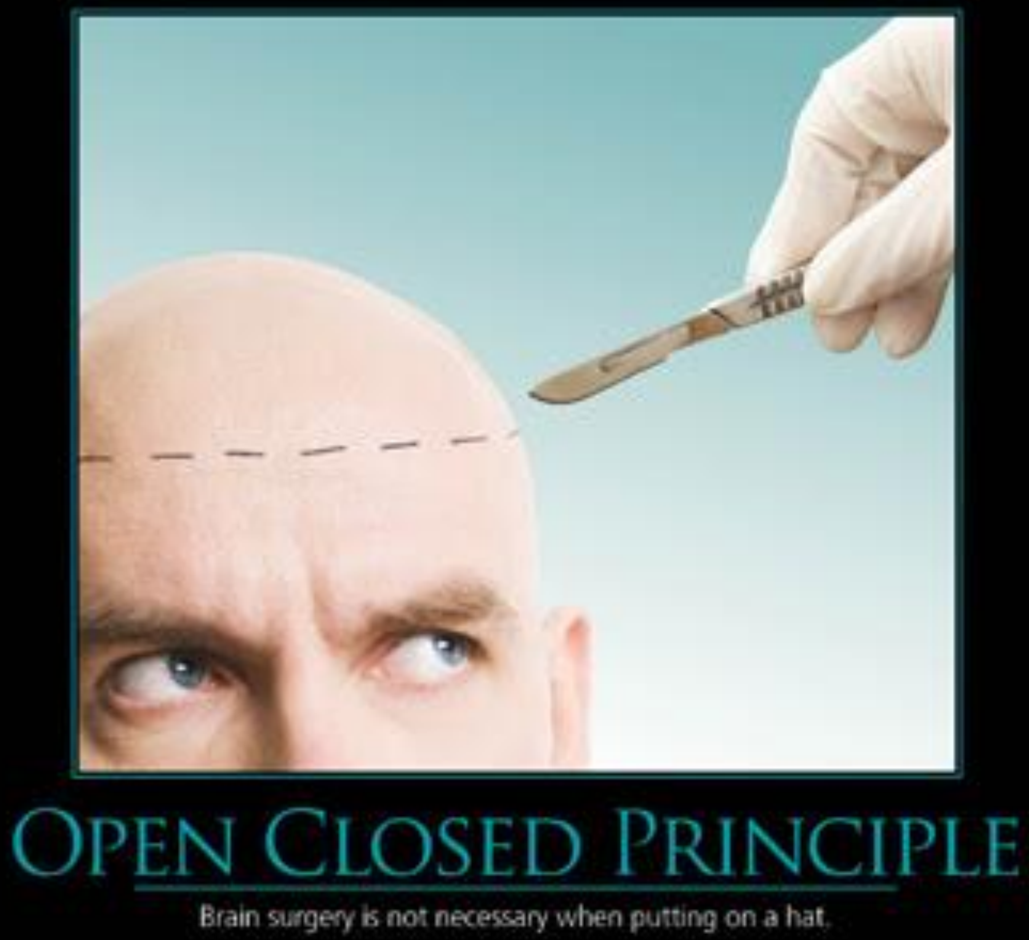
http://softuni.bg

Table of Contents

Questions

sli.do

#JavaOOP-Advanced



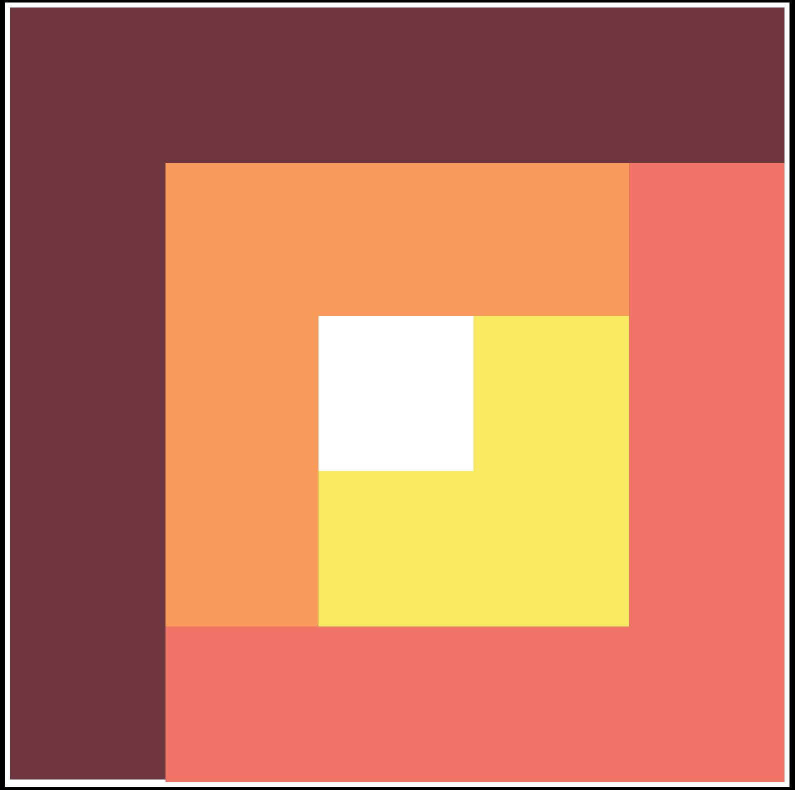
Open/Closed Principle

What is Open/Closed?

```
Software entities like classes, modules
and functions should be open for extension
but closed for modifications
```

Extensibility

- Implementation takes future growth into consideration
- Extensible system is one whose internal structure and data flow are minimally or not affected by new or modified functionality



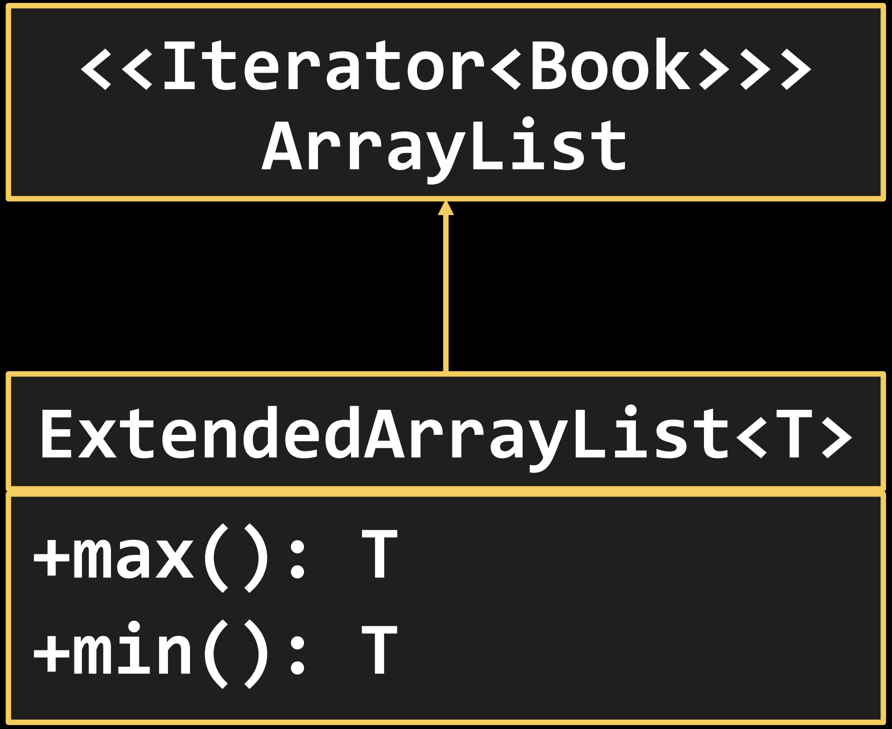
Reusability

- Software reusability more specifically refers to design features of a software element that enhance its suitability for reuse

- Modularity
- Low coupling
- High cohesion

Problem: Extend ArrayList<T>

- Create a class `ExtendedArrayList<T>`, which extends `ArrayList<T>` with two methods:
  - `max()`
  - `min()`
- Try to modify `ArrayList<T>`



Solution: Extend ArrayList<T>

```
public class ExtendedArrayList<T extends Comparable<T>>
extends ArrayList<T> {

  public T max() {
    TODO: Add buisnes logic for finding max element
  }
  public T min() {
    TODO: Add buisnes logic for finding min element
  }
}
```

Open/Closed Principle (OCP)

- Design and writing of the code should be done in a way that new functionality should be added with minimum changes in the existing code

- Changes to source code are not required

Problem: Stream Progress Info

- Refactor skeleton given for this problem:
  - `StreamProgressInfo` class must work with Music class too
  - Be sure if you add new class which provide `getBytesSent()` and `getLength()` will work without touching `StreamProgressInfo`

Solution: Stream Progress Info

```java
public class StreamProgressInfo {
  private Streamable streamable;

  public StreamProgressInfo(Streamable streamResult) {
    this.streamable = streamResult;
  }

  public int calculateStreamProgress() {
    return (this.streamable.getBytesSent() * 100) /
                                     this.streamable.getLength();
  }
}
```

Solution: Stream Progress Info (2)

```java
public interface Streamable {
    int getLength();
    int getBytesSent();
}
```

```java
public class File implements Streamable {
  //TODO: Add business logic
}
public class Music implements Streamable {
 //TODO: Add business logic
}
```

OCP – Violations

- Cascading changes through modules
- Each change requires re-testing
- Logic depends on conditional statements

Problem: Graphic Editor

- Refactor skeleton given for this task so
  - `GraphicEditor` class draw all kind of shapes without asking what kind of shape we pass
  - Be sure if you add new type of shape system will work correctly without touching `GraphicEditor`

Solution: Graphic Editor

```java
public class GraphicEditor {
  void drawShape(Shape shape) {
    shape.draw();
  }
}
```

```java
public interface Shape {
  void draw();
}
```

```java
class Rectangle extends Shape  {
  public void draw() {
    System.out.println("I'm Rectangle");
  }
}
```

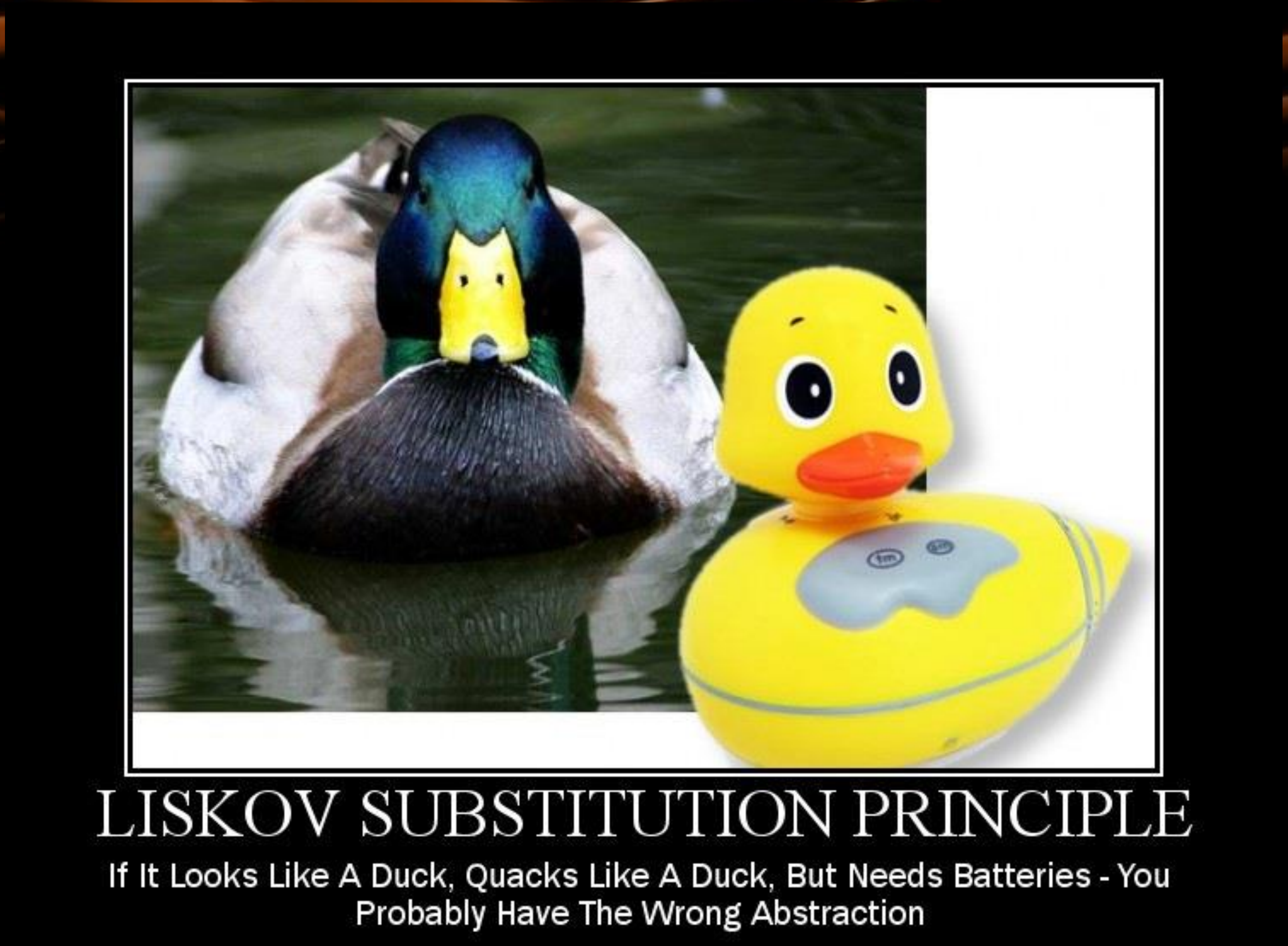
```java
//TODO: Make the same for Circle
```

OCP – Solutions

- Inheritance / Abstraction
- Inheritance / Template Method pattern
- Composition / Strategy patterns

Open/Closed Principle

Live Exercises in Class (Lab)


LISKOV SUBSTITUTION PRINCIPLE
If It Looks Like A Duck, Quacks Like A Duck, But Needs Batteries - You Probably Have The Wrong Abstraction

Liskov Substitution Principle

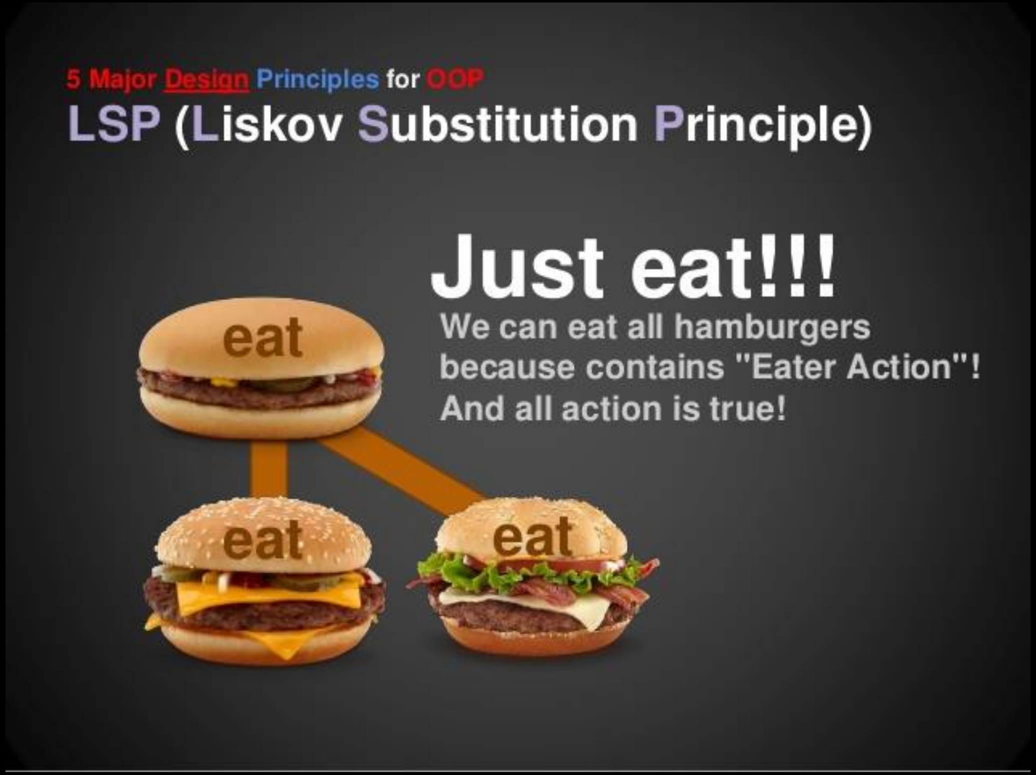
What is Liskov Substitution?

```
Derived types must be completely
substitutable for their base types
```

Liskov Substitution Principle (LSP)

- Reference to the base class can be replaced with a derived class without affecting the functionality of the program module

- Derived classes just extend without replacing the functionality of old classes


5 Major Design Principles for OOP

LSP (Liskov Substitution Principle)

Just eat!!!
We can eat all hamburgers because contains "Eater Action"! And all action is true!
eat
eat
eat

Problem: Detail Printer

- Refactor skeleton given for this task so
  - `DetailPrinter` class print correctly any kind of employee, which is in collection
  - Remove any `instanceof` from your code
  - Be sure if you add new type of employee system will work correctly without touching `DetailPrinter`

Solution: Detail Printer

```
public class Employee {
    private String name;

    public Employee(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Name: " + this.name;
    }
}
```

Solution: Detail Printer (2)

```
public class Manager extends Employee {
  private Iterable<String> docs;

  public Manager(String name, Iterable<String> docs) {
    super(name);
    this.documents = docs;
  }

  @Override
  public String toString() {
    return super.toString() + "Documents: " + this.docs;
  }
}
```

Solution: Detail Printer (3)

```
public class DetailsPrinter {
  private Iterable<Employee> employees;

  public DetailsPrinter(Iterable<Employee> employees) {
    this.employees = employees;
  }

  public void printEmployees() {
    for(Employee e : employees)  {
      System.out.println(e.toString());
    }
  }
}
```

LSP Relationship

- OOP Inheritance

```
Student IS-A Person
```

- Plus LSP

```
Student IS-SUBSTITUTED-FOR Person
```

OCP vs LSP

Liskov Substitution Principle is just an extension of the Open Close Principle and it means that we must make sure that new derived classes are extending the base classes without changing their behavior.

LSP – Violations

- Type Checking
- Overridden methods say "I am not implemented"
- Base class depends on its subtypes

Problem: Square

- We know from Math that square is a rectangle
- Look at skeleton given for this task
- Think how to refactor code so:
  - Square extends rectangle without produce bugs
- Prepare new unit tests for Square after this

LSP – Solutions

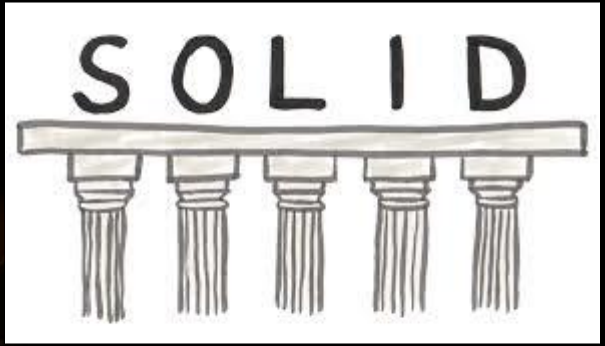
- "Tell, Don't Ask"
- Refactoring to base class

Liskov Substitution Principle

Live Exercises in Class (Lab)

Summary

- OCP – Open / Closed Principle
  - Violations of OCP
- LSP – Liskov Substitution Principle
  - Violations of LSP



OOP Advanced – OCP and LSP

Questions?

https://softuni.bg/java-advanced-oop

License

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg