Solid

Solid Principles (Bag. 1)

Pemrograman Berorientasi Objek

Riza Satria Perdana, S.T., M.T.

Teknik Informatika - STEI ITB

SOLID

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov's Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Bad Designs

Rigidity
It is hard to change because every change affects too many other parts of the system

Fragility
When you make a change, unexpected parts of the system break

Immobility
It is hard to reuse in another application because it cannot be disentangled from the current application

Single Responsibility Principle

a **class** should be having one and only one responsibility

a **class** should have one and only one reason to change

Single Responsibility Principle

```
1  public class Employee {
2      public calculatePay(Money m) {
3          // business logic for payment
4      }
5
6      public Employee save(Employee e) {
7          // store employee
8      }
9  }
```

Single Responsibility Principle

```java
public class Employee {
    public calculatePay(Money m) {
        // business logic for payment
    }
}
```

```java
public class EmployeeRepo {
    public Employee save(Employee e) {
        // store employee
    }
}
```

Example (Cont.)

```java
public class AreaCalculator {
    public int sum(List<Object> shapes) {
        int sum = 0;
        for (int i = 0; i < shapes.size(); i++) {
            Object shape = shapes.get(i);
            if (shape instanceof Circle) {
                sum += Math.PI * Math.pow(((Circle) shape).getRadious(), b:2);

            } else if (shape instanceof Square) {
                sum += Math.pow(((Square) shape).getLength(), b:2);
            }
        }
        return sum;
    }

    public String json(List<Object> shapes) {
        return "{sum: %s}".formatted(sum(shapes));
    }

    public String csv(List<Object> shapes) {
        return "sum,%s".formatted(sum(shapes));
    }
}
```

src > com > solidtutorial > Main.java > Main > main(String[])

```java
package com.solidtutorial;

import java.util.List;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        AreaCalculator areaCalculator = new AreaCalculator();
        Circle circle = new Circle(radius:5);
        Square rectangle = new Square(length:5);

        List<Object> shapes = List.of(circle, rectangle);
        int sum = areaCalculator.sum(shapes);
        System.out.println(areaCalculator.json(shapes));
        System.out.println(areaCalculator.csv(shapes));
    }
}

```

```
>
>  cd /Users/galuhdipabharata/Developer/OOP/SOLID ; /usr/bin/env /Users/galuhdipabharata/.sdkman/candidates/java/17.0.9-amzn/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/galuhdipabharata/Library/Application\ Support/Code/User/workspaceStorage/1569b9166e28afe13ee1e403fad766e3/redhat.java/jdt_ws/SOLID_d6c6c0ea/bin com.solidtutorial.Main

{sum: 103}
sum,103

 ~/Developer/OOP/SOLID ..... 08:51:37
>
```

```java
package com.solidtutorial;
import java.util.List;

public class AreaCalculator {
    public int sum(List<Object> shapes) {
        int sum = 0;
        for (int i = 0; i < shapes.size(); i++) {
            Object shape = shapes.get(i);
            if (shape instanceof Circle) {
                sum += Math.PI * Math.pow(((Circle) shape).getRadious(),
                b:2);

            } else if (shape instanceof Square) {
                sum += Math.pow(((Square) shape).getLength(), b:2);
            }
        }
        return sum;
    }

}
```

src > com > solidtutorial > ShapesPrinter.java > ...

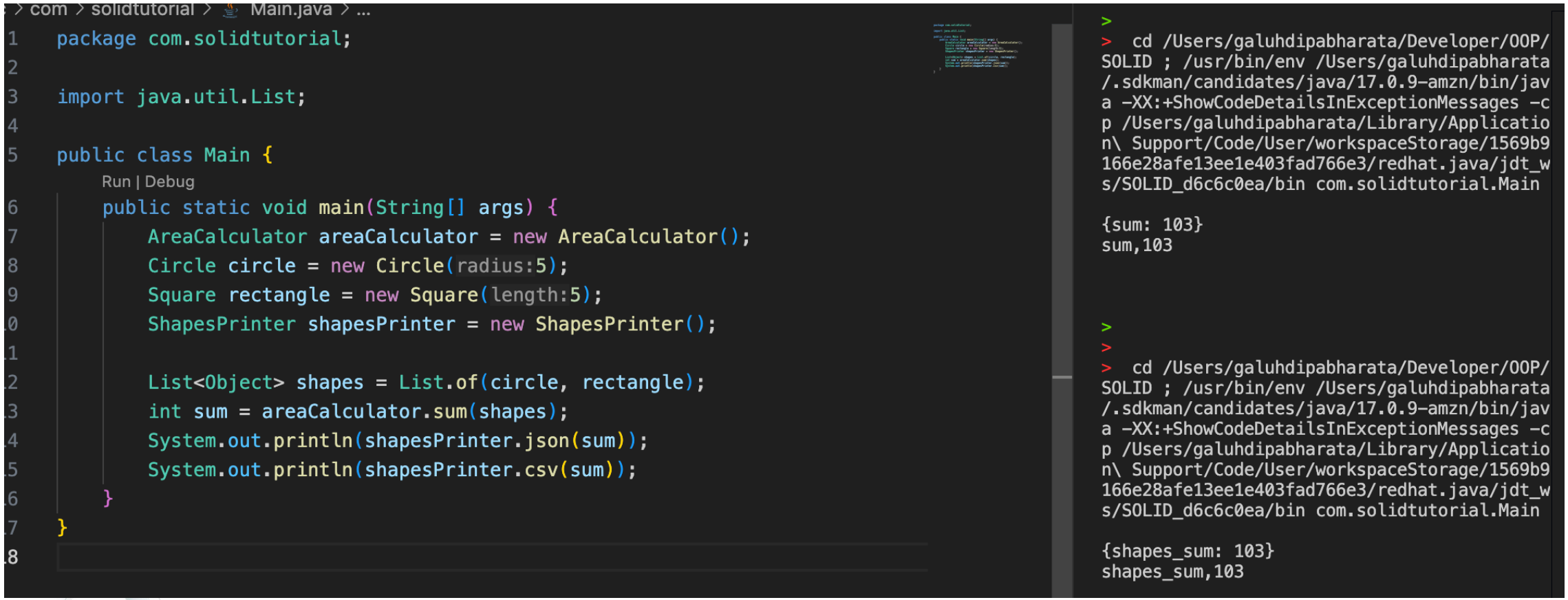
```java
package com.solidtutorial;

public class ShapesPrinter {

    public String json(int sum) {
        return "{shapes_sum: %s}".formatted(sum);
    }

    public String csv(int sum) {
        return "shapes_sum,%s".formatted(sum);
    }
}

```


com > solidtutorial > Main.java > ...
1 package com.solidtutorial;
2
3 import java.util.List;
4
5 public class Main {
Run | Debug
6 public static void main(String[] args) {
7 AreaCalculator areaCalculator = new AreaCalculator();
8 Circle circle = new Circle(radius:5);
9 Square rectangle = new Square(length:5);
10 ShapesPrinter shapesPrinter = new ShapesPrinter();
11
12 List<Object> shapes = List.of(circle, rectangle);
13 int sum = areaCalculator.sum(shapes);
14 System.out.println(shapesPrinter.json(sum));
15 System.out.println(shapesPrinter.csv(sum));
16 }
17 }
18
> cd /Users/galuhdipabharata/Developer/OOP/SOLID ; /usr/bin/env /Users/galuhdipabharata/.sdkman/candidates/java/17.0.9-amzn/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/galuhdipabharata/Library/Application\ Support/Code/User/workspaceStorage/1569b9166e28afe13ee1e403fad766e3/redhat.java/jdt_ws/SOLID_d6c6c0ea/bin com.solidtutorial.Main
{sum: 103}
sum,103
> cd /Users/galuhdipabharata/Developer/OOP/SOLID ; /usr/bin/env /Users/galuhdipabharata/.sdkman/candidates/java/17.0.9-amzn/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/galuhdipabharata/Library/Application\ Support/Code/User/workspaceStorage/1569b9166e28afe13ee1e403fad766e3/redhat.java/jdt_ws/SOLID_d6c6c0ea/bin com.solidtutorial.Main
{shapes_sum: 103}
shapes_sum,103

Open/Closed Principle

Software entities (classes, modules, functions) should be open for extension, but closed for modification

- **Open**: add data members and operations through inheritance
- **Close**: available for use by other components but may not, itself, be changed

Open/Closed Principle

```
1  public class PaymentManager {
2      private PaymentType ptype;
3
4      public void pay(Money m) {
5          if (ptype==PaymentType.Cash) {
6              // pay with cash
7          }
8          else {
9              // pay with credit card
10         }
11     }
12 }
```

Open/Closed Principle

```java
public abstract class Payment {
    public abstract void pay(Money m);
}

public class CashPayment extends Payment {
    public void pay(Money m) {
        // pay with cash
    }
}

public class CreditCardPayment extends Payment {
    public void pay(Money m) {
        // pay with credit dard
    }
}
```

src > com > solidtutorial > Cube.java > ...

```java
package com.solidtutorial;

public class Cube {
    private final int length;

    public Cube(int length) {
        this.length = length;
    }

    public int getLength() {
        return length;
    }
}
```

```
public class AreaCalculator {
    public int sum(List<Object> shapes) {
        int sum = 0;
        for (int i = 0; i < shapes.size(); i++) {
            Object shape = shapes.get(i);
            if (shape instanceof Circle) {
                sum += Math.PI * Math.pow(((Circle) shape).getRadious(),
                b:2);

            }
            if (shape instanceof Square) {
                sum += Math.pow(((Square) shape).getLength(), b:2);
            }
            if (shape instanceof Cube)  {
                sum += Math.pow(((Cube) shape).getLength(), b:3);
            }
        }
        return sum;
    }

}
```

src > com > solidtutorial > Shape.java > ...

```java
package com.solidtutorial;

public interface Shape {
    double area();
}

```

```java
public class Circle implements Shape {
    private final int radius;

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

    public int getRadious() {
        return radius;
    }

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

}
```

```java
package com.solidtutorial;

public class Square implements Shape {
    private final int length;

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

    public int getLength() {
        return length;
    }

    @Override
    public double area() {
        return length * length;
    }
}
```

```java
package com.solidtutorial;

public class Cube implements Shape{
    private final int length;

    public Cube(int length) {
        this.length = length;
    }

    public int getLength() {
        return length;
    }

    @Override
    public double area() {
        return 6 * length * length;
    }
}
```

```java
public class AreaCalculator {
    public int sum(List<Object> shapes) {
        int sum = 0;
        for (int i = 0; i < shapes.size(); i++) {
            Object shape = shapes.get(i);
            if (shape instanceof Circle) {
                sum += Math.PI * Math.pow(((Circle) shape).getRadious(),
                b:2);

            }
            if (shape instanceof Square) {
                sum += Math.pow(((Square) shape).getLength(), b:2);
            }
            if (shape instanceof Cube)  {
                sum += Math.pow(((Cube) shape).getLength(), b:3);
            }
        }
        return sum;
    }

}
```

```java
package com.solidtutorial;
import java.util.List;

public class AreaCalculator {
    public int sum(List<Shape> shapes) {
        int sum = 0;
        for (int i = 0; i < shapes.size(); i++) {
            sum += shapes.get(i).area();
        }
        return sum;
    }
}
```

```java
package com.solidtutorial;

import java.util.List;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        AreaCalculator areaCalculator = new AreaCalculator();
        Circle circle = new Circle(radius:5);
        Square rectangle = new Square(length:5);
        ShapesPrinter shapesPrinter = new ShapesPrinter();
        Cube cube = new Cube(length:5);

        List<Shape> shapes = List.of(circle, rectangle, cube);

        int sum = areaCalculator.sum(shapes);
        System.out.println(shapesPrinter.json(sum));
        System.out.println(shapesPrinter.csv(sum));
    }
}
```

```
>  cd /Users/galuhdipabharata/Developer/OOP/SOLID ; /usr/bin/env /Users/galuhdipabharata/.sdkman/candidates/java/17.0.9-amzn/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/galuhdipabharata/Library/Application\ Support/Code/User/workspaceStorage/1569b9166e28afe13ee1e403fad766e3/redhat.java/jdt_ws/SOLID_d6c6c0ea/bin com.solidtutorial.Main

{shapes_sum: 253}
shapes_sum,253

 ~/Developer/OOP/SOLID ..... 09:43:50
>
```

Terima Kasih