Week 1 - SOLID Principles

# 1. SRP - Single Responsibility Principle

✅ Code:

class Employee {  
 String name;  
 public Employee(String name) {  
 this.name = name;  
 }  
}  
  
class SalaryCalculator {  
 public void calculateSalary(Employee e) {  
 System.out.println("Salary calculated for: " + e.name);  
 }  
}  
  
public class SRPExample {  
 public static void main(String[] args) {  
 Employee e = new Employee("Karan");  
 SalaryCalculator sc = new SalaryCalculator();  
 sc.calculateSalary(e);  
 }  
📸 Output: A screenshot of a computer program

AI-generated content may be incorrect.

💡 Explanation: This code separates employee data from salary logic. It follows SRP by giving each class a single responsibility.

# 2. OCP - Open/Closed Principle

✅ Code:

abstract class Shape {  
 public abstract double area();  
}  
  
class Rectangle extends Shape {  
 double length, width;  
 public Rectangle(double l, double w) {  
 this.length = l;  
 this.width = w;  
 }  
 @Override  
 public double area() {  
 return length \* width;  
 }  
}  
  
class Circle extends Shape {  
 double radius;  
 public Circle(double r) {  
 this.radius = r;  
 }  
 @Override  
 public double area() {  
 return Math.PI \* radius \* radius;  
 }  
}  
  
public class OCPExample {  
 public static void main(String[] args) {  
 Shape s1 = new Rectangle(5, 3);  
 Shape s2 = new Circle(4);  
  
 System.out.println("Rectangle Area: " + s1.area());  
 System.out.println("Circle Area: " + s2.area());  
 }  
}

📸 Output: A screen shot of a computer program

AI-generated content may be incorrect.

💡 Explanation: The Shape class is extended but not modified. It follows OCP by allowing extension through Rectangle and Circle classes.

**📘 Overview: What Are SOLID Principles?**

The SOLID principles are five design principles intended to make object-oriented designs more understandable, flexible, and maintainable. They help developers build scalable, extendable, and testable software systems by encouraging clean coding practices.

**🔑 List of SOLID Principles:**

1. **SRP – Single Responsibility Principle**  
   A class should have one and only one reason to change, meaning it should only have one job.
2. **OCP – Open/Closed Principle**  
   Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.
3. **LSP – Liskov Substitution Principle**  
   Subtypes must be substitutable for their base types without altering the correctness of the program.
4. **ISP – Interface Segregation Principle**  
   Clients should not be forced to depend on interfaces they do not use.
5. **DIP – Dependency Inversion Principle**  
   High-level modules should not depend on low-level modules. Both should depend on abstractions.

**🎯 Learning Objectives from Handbook**

After completing this module, you should be able to:

* Grasp the importance of the Single Responsibility Principle (SRP)
* Implement the Open/Closed Principle (OCP) to create extendable software
* Ensure software components adhere to the Liskov Substitution Principle (LSP)
* Design interfaces using Interface Segregation Principle (ISP)
* Apply Dependency Inversion Principle (DIP) for flexible dependency management
* Apply design principles in hands-on coding and real-world scenarios