

**JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY.**

**M.Sc. SOFTWARE ENGINEERING**

**ICS 3105: - OBJECT ORIENTED SOFTWARE ENGINEERING.**

**ASSIGNMENT 1**

**Dr Eunice Njeri.**

**ALEX KEMBOI**

**SCT 313-0530/2023**

A Research proposal submitted to Jomo Kenyatta University of Agriculture and Technology, School of Computing. Partial fulfillment of the requirement for the award of degree of Masters of Science in Software Engineering.

2025

# ****SOLID Principles****

**SOLID** is an acronym for five design principles that help create maintainable and scalable software.

Single Responsibility Principle (SRP)

● Open/Closed Principle (OCP)

● Liskov Substitution Principle (LSP)

● Interface Segregation Principle (ISP)

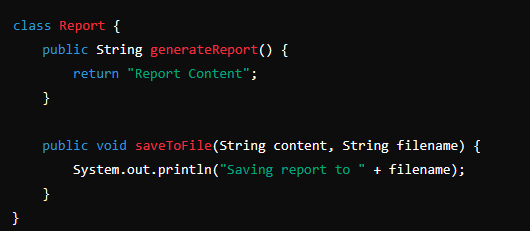
● Dependency Inversion Principle (DIP)

**SOLID** principles improve software **maintainability**, **scalability**, and **flexibility** by reducing tight coupling and enforcing best practices.

## ****1. Single Responsibility Principle (SRP)****

A class should have only one reason to change.  
Each class should have only one responsibility.

### ****Violation of SRP****



**Problem**: The Report class handles both report generation and file saving.

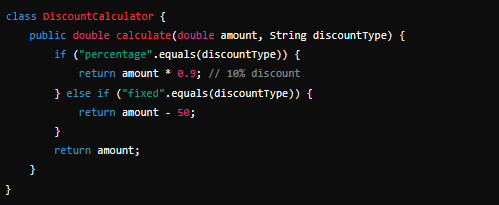
**Refactored Code (SRP Applied)**



## ****2. Open/Closed Principle (OCP)****

Software entities should be open for extension but closed for modification.

### ****Violation of OCP****



**Problem**: Adding a new discount type requires modifying the class.

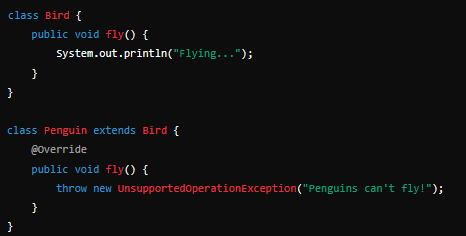
### ****Refactored Code (OCP Applied)****



## ****3. Liskov Substitution Principle (LSP)****

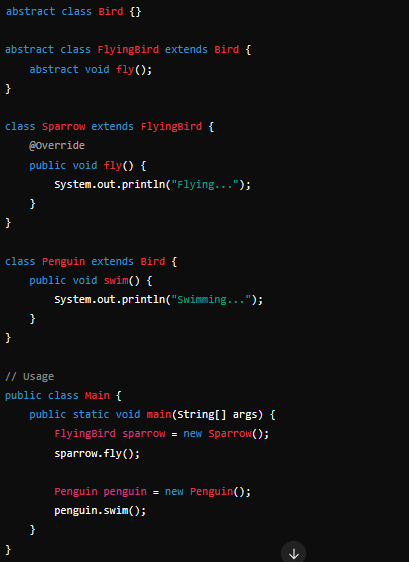
Subtypes must be substitutable for their base types without breaking functionality.

### ****Violation of LSP****



**Problem**: Penguin extends Bird but cannot fly, violating the expected behavior.

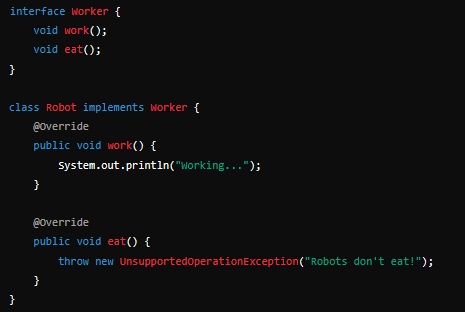
### ****Refactored Code (LSP Applied)****



## ****4. Interface Segregation Principle (ISP)****

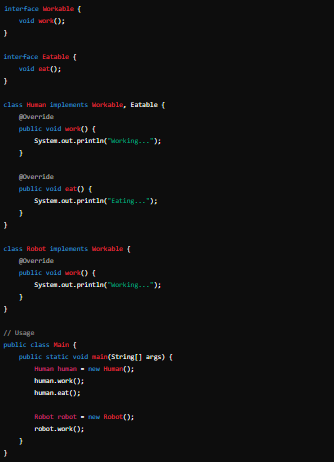
A class should not be forced to implement interfaces it does not use.

### ****Violation of ISP****



**Problem**: Robot is forced to implement eat(), which it does not need.

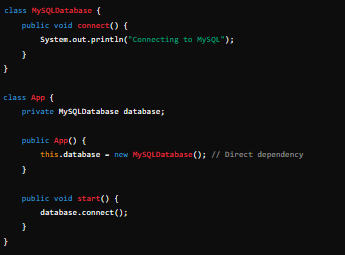
### ****Refactored Code (ISP Applied)****



## ****5. Dependency Inversion Principle (DIP)****

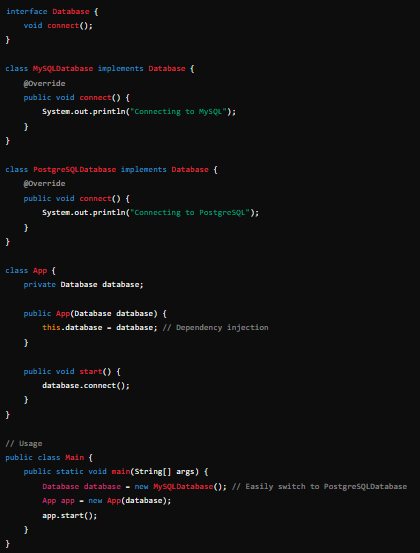
High-level modules should not depend on low-level modules. Both should depend on abstractions.

### ****Violation of DIP****



**Problem**: App is tightly coupled to MySQLDatabase.

### ****Refactored Code (DIP Applied)****



**GIT HUB LINK**

<https://github.com/alexkemboi/Solid-Principles.git>