



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Assignment-1

**Student Name:** Sidak Singh

**UID:** 23BCS13207

**Branch:** BE-CSE

**Section/Group:** KRG 1-B

**Semester:** 6th

**Subject Code:** 23CSH-314

**Subject Name:** System Design

### **Q1: Explain SRP and OCP in detail with proper examples.**

**Solution:** SRP and OCP are two important principles of the SOLID design principles used in object-oriented software design. These principles help in creating maintainable, scalable, and flexible software systems.

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

The **Single Responsibility Principle (SRP)** states that

*“A class should have only one reason to change.”*

This means a class should perform **only one well-defined responsibility**.

#### **Why SRP is Important:**

- **Improves Maintainability:** Changes in one responsibility do not affect other functionalities of the class.
- **Enhances Readability:** Code becomes easier to understand as each class performs a single well-defined task.
- **Reduces Complexity:** Smaller and focused classes reduce overall system complexity.
- **Simplifies Testing:** Classes with single responsibility are easier to test and debug.
- **Minimizes Bugs:** Fewer responsibilities in a class reduce the chances of unintended errors.

#### **Example CODE:**

```
// Class responsible only for employee data
class Employee {
    String name;
    int id;
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
public Employee(String name, int id) {  
    this.name = name;  
    this.id = id;  
}  
}  
  
// Class responsible only for salary calculation  
class SalaryCalculator {  
    public double calculateSalary(Employee emp) {  
        return 50000;  
    }  
}
```

Here,

- Employee handles employee data
- SalaryCalculator handles salary logic

Each class has **one responsibility**, so SRP is followed.

## 2. Open/Closed Principle (OCP)

The Open/Closed Principle (OCP) states that

*“Software entities should be open for extension but closed for modification.”*

This means new functionality should be added without changing existing code.

### Why OCP is Important?

- **Supports Extensibility:** New functionality can be added without modifying existing code.
- **Prevents Code Breakage:** Existing features remain stable while extending the system.
- **Improves Scalability:** System can grow by adding new classes rather than changing old ones.
- **Encourages Reusability:** Common abstractions can be reused across different modules.
- **Enhances Flexibility:** System adapts easily to changing requirements

### Example Code:

```
interface Shape {  
    double area();  
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
class Rectangle implements Shape {  
    public double area() {  
        return 10 * 5;  
    }  
}
```

```
class Circle implements Shape {  
    public double area() {  
        return 3.14 * 7 * 7;  
    }  
}
```

Here,

New shapes can be added by creating new classes

No existing class needs modification

Hence, OCP is satisfied.

## Q2: Discuss in detail about the violations in SRP and OCP along with their fixes.

**Solution-** Violations of SRP and OCP occur when responsibilities are mixed or when existing code is frequently modified for new features.

### 1. Violation of Single Responsibility Principle (SRP)

The Single Responsibility Principle is violated when a single class performs **more than one responsibility**, such as handling business logic, data storage, and presentation logic together. This results in a class having **multiple reasons to change**.

#### Example of SRP Violation:

```
class Employee {  
    void calculateSalary() {  
        // Salary calculation logic  
    }  
}
```

```
void generateReport() {  
    // Report generation logic  
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
void saveToDatabase() {  
    // Database storage logic  
}  
}
```

## Problems Caused by SRP Violation:

- A change in salary calculation may affect reporting or database logic.
- The class becomes large and difficult to understand.
- Testing becomes complex due to multiple functionalities.
- Maintenance becomes time-consuming.

## Fix for SRP Violation:

To fix SRP violation, **each responsibility is separated into a different class**, ensuring that every class has only one reason to change.

Example Code-

```
class SalaryCalculator {  
    void calculateSalary() {  
        // Salary calculation logic  
    }  
}
```

```
class ReportGenerator {  
    void generateReport() {  
        // Report generation logic  
    }  
}
```

```
class EmployeeRepository {  
    void saveToDatabase() {  
        // Database storage logic  
    }  
}
```

## Benefits After Fix:

- Code becomes modular and well-organized.
- Changes in one module do not affect others.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- Testing and debugging become easier.
- System becomes more maintainable.

## 2. Violation of Open/Closed Principle (OCP)

The Open/Closed Principle is violated when **existing source code is modified** to add new functionality instead of extending it. This often occurs when conditional statements are used to handle different cases.

### Example of OCP Violation:

```
class DiscountCalculator {  
    double calculateDiscount(String customerType) {  
        if (customerType.equals("Regular")) {  
            return 5;  
        } else if (customerType.equals("Premium")) {  
            return 10;  
        }  
        return 0;  
    }  
}
```

### Problems Caused by OCP Violation:

- Adding a new customer type requires modifying the existing class.
- Risk of introducing bugs in already working code.
- Code becomes hard to maintain as conditions increase.
- System becomes less flexible.

### Fix for OCP Violation:

To fix OCP violation, **abstraction is used through interfaces or abstract classes**, allowing new functionality to be added without changing existing code.

Example code-

```
interface Discount {  
    double calculate();  
}
```

```
class RegularCustomer implements Discount {  
    public double calculate() {  
        return 5;  
    }  
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}
```

```
}
```

```
class PremiumCustomer implements Discount {
```

```
    public double calculate() {
```

```
        return 10;
```

```
    }
```

```
}
```

## Benefits After Fix:

- New customer types can be added by creating new classes.
- Existing code remains unchanged and stable.
- System becomes extensible and flexible.
- Code follows clean design principles.