



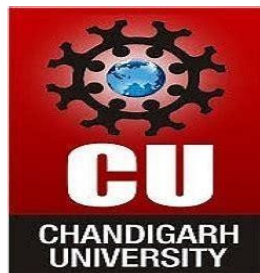
**CHANDIGARH  
UNIVERSITY**  
Discover. Learn. Empower.

**NAAC  
GRADE A+**  
Accredited University

# UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE - 6<sup>th</sup> Sem)



Subject Name: System Design

Subject Code: 23CSH-314

SUBMITTED BY:

Name: Anmolpreet Singh

UID: 23BCS11616

Section: KRG-1A

SUBMITTED TO:

Er. Alok Kumar (E15012)

Q1. Explain SRP and OCP in detail with proper examples.

Ans.

### Single Responsibility Principle (SRP):

- A class should have one, and only one, reason to change.
- This means that a class must have only one responsibility.
- When a class performs just one task, it contains a small number of methods and member variables making them more usable and easier to maintain.
- If a class has multiple responsibilities, it becomes harder to understand, maintain, and modify and increases the potential for bugs because changes to one responsibility could affect the others.
- Code:

```
class ReportGenerator {  
    void generateReport() {  
        System.out.println("Generating report");  
    }  
}  
  
class ReportSaver {  
    void saveToFile() {  
        System.out.println("Saving report to file");  
    }  
}
```

### Open-closed principle (OCP):

- Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.
- This means the design of a software entity should be such that you can introduce new functionality or behavior without modifying the existing code since changing the existing code might introduce bugs.
- If a class has multiple responsibilities, it becomes harder to understand, maintain, and modify and increases the potential for bugs because changes to one responsibility could affect the others.

- Code:

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

```
class Circle implements Shape {  
    public double area() {  
        return 3.14 * 5 * 5;  
    }  
}
```

```
class Square implements Shape {  
    public double area() {  
        return 4 * 4;  
    }  
}
```

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

### Code violating SRP:

```
public class Vehicle {  
    public void printDetails() {}  
    public double calculateValue() {}  
    public void addVehicleToDB() {}  
}
```

### Code that fulfills the SRP:

```
public class Vehicle {  
    private String make;  
    private String model;  
    public Vehicle (String make, String model) {  
        this.make = make;  
        this.model = model;  
    }  
}
```

```

    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public void printDetails() {
        System.out.println("Make: " + make);
        System.out.println("Model: " + model);
    }

    public static void main(String[] args) {
        Vehicle firstCar = new Vehicle("Toyota", "Camry");
        firstCar.printDetails();
    }
}

```

### Code violating OCP:

```

public class VehicleCalculations {
    public double calculateValue(Vehicle v) {
        if (v instanceof Car) {
            return v.getValue() * 0.8;
        }
        if (v instanceof Bike) {
            return v.getValue() * 0.5;
        }
    }
}

```

```
}  
}
```

### Code that fulfills the OCP:

```
class Vehicle {  
    private double value;  
  
    public Vehicle(double value) {  
        this.value = value;  
    }  
  
    public double getValue() {  
        return value;  
    }  
  
    public double calculateValue() {  
        return value; // No depreciation by default  
    }  
}  
  
class Car extends Vehicle {  
    public Car(double value) {  
        super(value);  
    }  
  
    @Override  
    public double calculateValue() {  
        return super.calculateValue() * 0.8; // Apply 80% depreciation  
    }  
}  
  
class Truck extends Vehicle {  
    public Truck(double value) {  
        super(value);  
    }  
}
```

```
@Override  
public double calculateValue() {  
    return super.calculateValue() * 0.9; // Apply 90% depreciation  
}  
}
```