DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

ASSIGNMENT

Subject Name – System Design

Subject Code – 23CST- 314

Submitted To:

Er. Alok Kumar

(E17196)

Submitted By:

Name: Shishank

UID: 23BCS13048

Section: KRG_2-B

ASSIGNMENT-1

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

=> 1. Single Responsibility Principle (SRP)

Definition:
Single Responsibility Principle states that **a class should have only one reason to change**, i.e., it should perform **only one specific responsibility**.

Explanation:
If a class handles multiple responsibilities (like business logic, data storage, and printing), then a change in one responsibility may affect the others. This makes the system hard to maintain and error-prone.

Example (Violation):

```
class Report {
   void calculateReport() { }
   void saveToFile() { }
   void printReport() { }
}
```

Here, the class is doing **calculation, storage, and printing**, which violates SRP.

SRP Applied (Correct Design):

```
class Report {
   void calculateReport() { }
}
class ReportSaver {
   void saveToFile() { }
}
class ReportPrinter {
   void printReport() { }
}
```

Benefit:
Improves **maintainability, readability, and testability** of the code.

2. Open–Closed Principle (OCP)

Definition:
Open–Closed Principle states that **software entities should be open for extension but closed for modification**.

Explanation:
This means we should be able to **add new functionality without changing existing code**, reducing the risk of introducing bugs.

Example (Violation):

```
class Discount {
   double calculate(String type) {
      if(type.equals("Student"))
         return 10;
      else if(type.equals("Senior"))
         return 20;
      return 0;
   }
}
```

If a new discount type is added, the class must be modified → **OCP violated**.

OCP Applied (Correct Design):

```
interface Discount {
   double calculate();
}
class StudentDiscount implements Discount {
   public double calculate() { return 10; }
}
class SeniorDiscount implements Discount {
   public double calculate() { return 20; }
}
```

Benefit:
Makes the system **flexible, scalable, and easy to extend**.

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

=> 1. Violation of Single Responsibility Principle (SRP)

Violation:
SRP is violated when **a single class performs multiple responsibilities**. If a class handles more than one task, it will have **multiple reasons to change**, which increases complexity and maintenance cost.

Example (SRP Violation):

```
class Invoice {
   void calculateTotal() { }
   void printInvoice() { }
   void saveToDatabase() { }
}
```

This class handles **calculation, printing, and database storage**, violating SRP.

Fix:
Separate each responsibility into different classes.

Correct Design (SRP Fix):

```
class Invoice {
   void calculateTotal() { }
}
class InvoicePrinter {
   void printInvoice() { }
}
class InvoiceRepository {
   void saveToDatabase() { }
}
```

Result:
Each class has **one responsibility**, making the system easier to maintain and test.

2. Violation of Open–Closed Principle (OCP)

Violation:
OCP is violated when **existing code must be modified** to add new functionality. This can introduce bugs in already working code.

Example (OCP Violation):

```
class Payment {
   void pay(String method) {
      if(method.equals("CreditCard")) { }
      else if(method.equals("UPI")) { }
   }
}
```

Adding a new payment method requires modifying the class.

Fix:
Use **abstraction (interface or inheritance)** to extend behavior without changing existing code.

Correct Design (OCP Fix):

```
interface Payment {
   void pay();
}
class CreditCardPayment implements Payment {
   public void pay() { }
}
class UPIPayment implements Payment {
   public void pay() { }
}
```

Result:
New payment methods can be added by creating new classes **without modifying existing code**.

Q3. Design an HLD for an Online Examination System applying these principles.

=>

Designed the required HLD in attached .drawio file.