Assignment: 01

Student Name: Himanshi Kaushal
UID: 23BCS12735
Branch: BE-CSE
Section/Group: KRG-2A
Semester: 6th
Date of Performance: 03/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

Q1. Explain the role of Interfaces and Enums in software design with proper examples?

Ans:

1. Interfaces in Software Design:

An interface defines a contract that a class must follow. It specifies what a class should do, but not how it should do it. Interfaces help in achieving abstraction, loose coupling, and multiple inheritance.

Role of Interfaces:

- Provide **standard behavior** across different classes
- Support **abstraction** by hiding implementation details
- Enable **loose coupling** between components
- Allow **multiple inheritance**
- Improve **maintainability and scalability**

Example of Interface:

```
interface Payment {
   void pay(double amount);
}
class CreditCardPayment implements Payment {
   public void pay(double amount) {
      System.out.println("Paid " + amount + " using Credit Card");
   }
}
class UpiPayment implements Payment {
   public void pay(double amount) {
      System.out.println("Paid " + amount + " using UPI");}
}
```

2. Enums in Software Design:

An **enum (enumeration)** is a special data type used to define a **fixed set of constant values**. Enums make the code **more readable, safer, and error-free**.

Role of Enums:

- Represent **fixed choices or states**
- Improve **type safety**
- Avoid use of hard-coded constants
- Make code **more readable and maintainable**
- Reduce logical errors

Example of Enum:

```
enum OrderStatus {
    PLACED,
    SHIPPED,
    DELIVERED,
    CANCELLED
}
class Order {
    OrderStatus status;
}
```

Q2. Discuss how interfaces enable loose coupling with example?

Ans:

Loose coupling means that different components of a software system have **minimal dependency** on each other.

A loosely coupled system allows changes in one component **without affecting** other components.

Interfaces play a crucial role in achieving loose coupling by separating **what a class does** from **how it does it**.

Role of Interfaces in Loose Coupling:

Interfaces enable loose coupling in the following ways:

- Classes depend on interfaces, not concrete implementations
- Implementation details can be changed without modifying dependent classes
- Promotes flexibility, reusability, and scalability
- Makes the system easier to test and maintain

Example: Loose Coupling Using Interface

Step 1: Define an Interface:

```
interface MessageService {
    void sendMessage(String message);
}
```

Step 2: Implement the Interface:

```
class EmailService implements MessageService {
    public void sendMessage(String message) {
        System.out.println("Email sent: " + message);
    }
}
class SMSService implements MessageService {
    public void sendMessage(String message) {
        System.out.println("SMS sent: " + message);
    }
```

```
}
```

Step 3: Use Interface in Client Class

```
class Notification {
    MessageService service;
    Notification(MessageService service) {
        this.service = service;
    }
    void notifyUser(String message) {
        service.sendMessage(message);
    }
}
```