ASSIGNMENT: 01

SCHOOL FEE SYSTEM
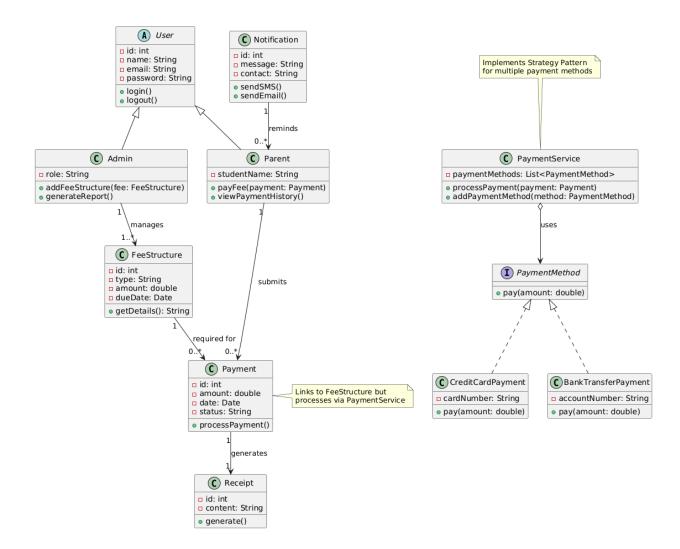
FA22-BSE-043
SUBMITTED BY: TAYYAB ALAM
SUBMITTED TO: MAM HIFZA ALI

Class diagram

User (A)
- □ id: int
- □ name: String
- □ email: String
- □ password: String
- • login()
- • logout()

Notification (C)
- □ id: int
- □ message: String
- □ contact: String
- • sendSMS()
- • sendEmail()

Implements Strategy Pattern for multiple payment methods

Admin (C)
- □ role: String
- • addFeeStructure(fee: FeeStructure)
- • generateReport()

Parent (C)
- □ studentName: String
- • payFee(payment: Payment)
- • viewPaymentHistory()

PaymentService (C)
- □ paymentMethods: List<PaymentMethod>
- • processPayment(payment: Payment)
- • addPaymentMethod(method: PaymentMethod)

reminds 1 / 0..*

manages 1 / 1..*

FeeStructure (C)
- □ id: int
- □ type: String
- □ amount: double
- □ dueDate: Date
- • getDetails(): String

submits

uses

PaymentMethod (I)
- • pay(amount: double)

required for 1 / 0..* / 0..*

Payment (C)
- □ id: int
- □ amount: double
- □ date: Date
- □ status: String
- • processPayment()

Links to FeeStructure but processes via PaymentService

CreditCardPayment (C)
- □ cardNumber: String
- • pay(amount: double)

BankTransferPayment (C)
- □ accountNumber: String
- • pay(amount: double)

generates 1 / 1

Receipt (C)
- □ id: int
- □ content: String
- • generate()

GitHub Code Link

https://github.com/Tayyab-Alam/SchoolFeeSystem

Design Principles

Project: School Fee Submission System

1. SOLID Principles

a. Single Responsibility Principle (SRP)

```java
// PaymentService.java - Handles only payment processing
public class PaymentService {
    public void processPayment(PaymentMethod method, double amount) {
        method.pay(amount);  // Only payment logic
    }
}
```

b. Open/Closed Principle (OCP)

```java
// PaymentMethod.java - Open for extension
public interface PaymentMethod {
    void pay(double amount);
}

// CreditCardPayment.java - Closed for modification
public class CreditCardPayment implements PaymentMethod {
    @Override
    public void pay(double amount) {
        System.out.println("Credit Card Payment: $" + amount);
    }
}
```

c. Liskov Substitution Principle (LSP)

```java
// User.java - Base class
public abstract class User {
    public abstract boolean login(String email, String password);
}

// Admin.java - Substitutable for User
public class Admin extends User {
    @Override
    public boolean login(String email, String password) {
        // Admin-specific login
    }
}
```

2. DRY Principle

```java
// Payment.java - Reusable receipt generation
public Receipt generateReceipt() {
    String content = "Payment of $" + amount;  // Logic used everywhere
    return new Receipt(id, content);
}
```

3. KISS Principle

```java
public void sendSMS(String message) {
    System.out.println("SMS: " + message);
}
```

4. Strategy Pattern

```java
// PaymentService.java - Interchangeable payment methods
public void processPayment(PaymentMethod method, double amount) {
    method.pay(amount);  // Can switch between CreditCard/BankTransfer
}
```

5. Composition Over Inheritance

```java
// Parent.java - Contains Payment instead of inheriting
public void payFee(Payment payment) {
    payment.processPayment();  // Delegates to Payment object
}
```

THE END