Sri Lanka Institute of Information Technology

Software Engineering | SE2030

2025 | Year 2 Semester 1

Group ID: 2025-Y2-S1-MLB-B6G2-07

Lab sheet 07 – Design Patterns

Web-based Musical Instrument Selling System

Student ID	**Name**
IT24102374	Wickramaarachchi M. H
IT24101551	Kuyilini. T
IT24101536	Alwis P.G.R.M
IT24101325	Gammanpila J.P
IT24101520	Athukorala A. P. H. B
IT24101376	Nishshanka N.P.S.M.

B.Sc. (Hons) in Information Technology

Strategy Design Pattern

Payment Management Module

Pattern type - Behavioral Pattern

Justification:

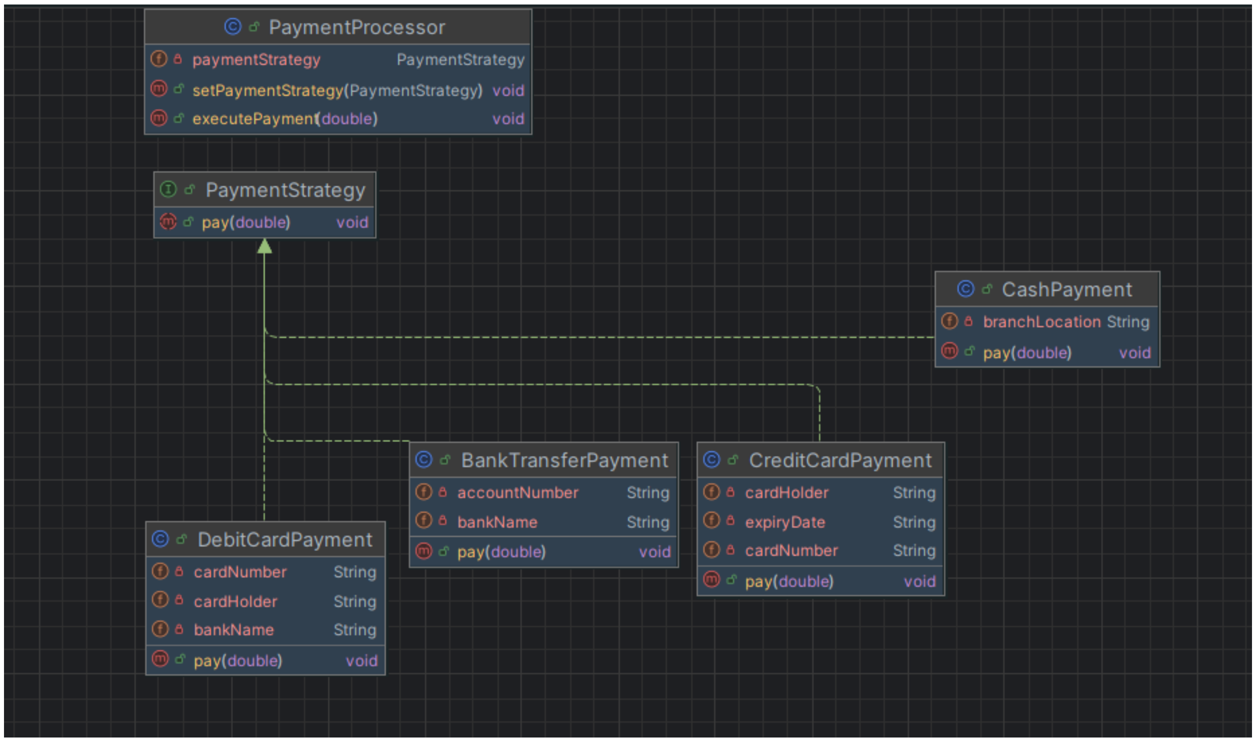
In our system, customers can make payments using multiple payment methods such as Credit Card, Debit Card, Bank Transfer, or Cash. Instead of using multiple if-else statements in one class to handle these different options, the Strategy Pattern allows us to define a common interface (PaymentStrategy) and create separate classes for each payment method.

This approach promotes the Open/Closed Principle – new payment options like PayPal or a mobile wallet can be added easily without modifying existing code.

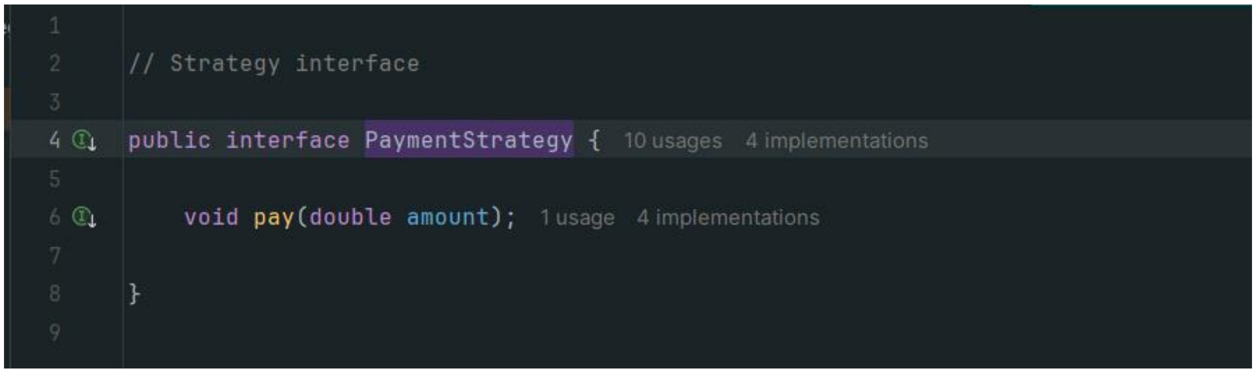
The PaymentProcessor context dynamically selects the appropriate payment strategy at runtime based on the user's selection, which reduces code coupling and improves the system's maintainability.

How it improves the Design:

- Decouples payment logic from the main system flow, making the code cleaner.
- Easily extensible: New payment types can be added by creating a new class without changing the existing PaymentProcessor or other payment classes.
- Cleaner architecture: This encourages a separation of concerns between the details of payment processing and the business logic of the application.
- Better testing: Each payment method can be tested independently, ensuring that each one works correctly on its own.



a) Strategy Interface

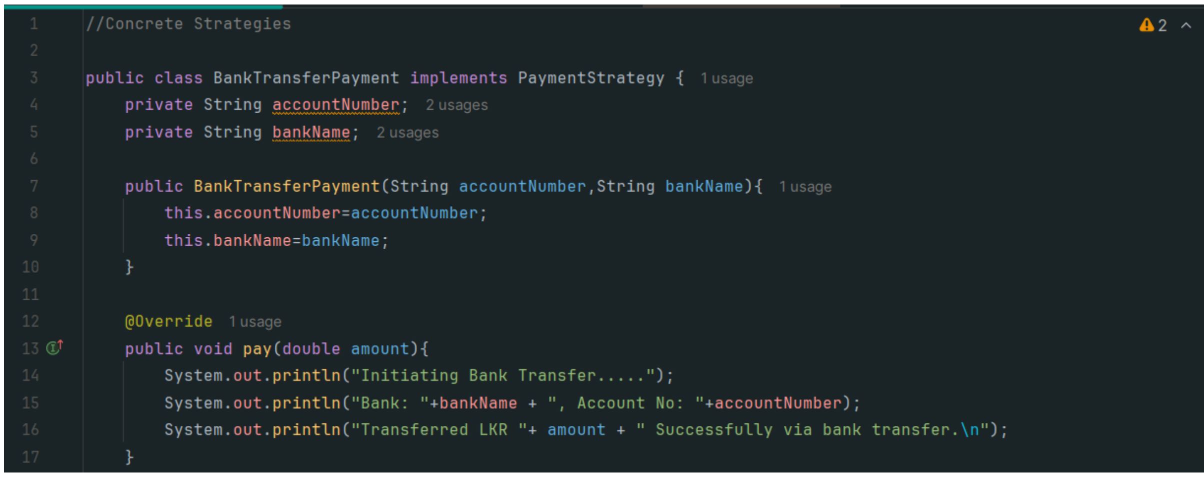


```

// Strategy interface

public interface PaymentStrategy {  10 usages   4 implementations

    void pay(double amount);  1 usage   4 implementations

}

```

Interface declaration with the pay() method.

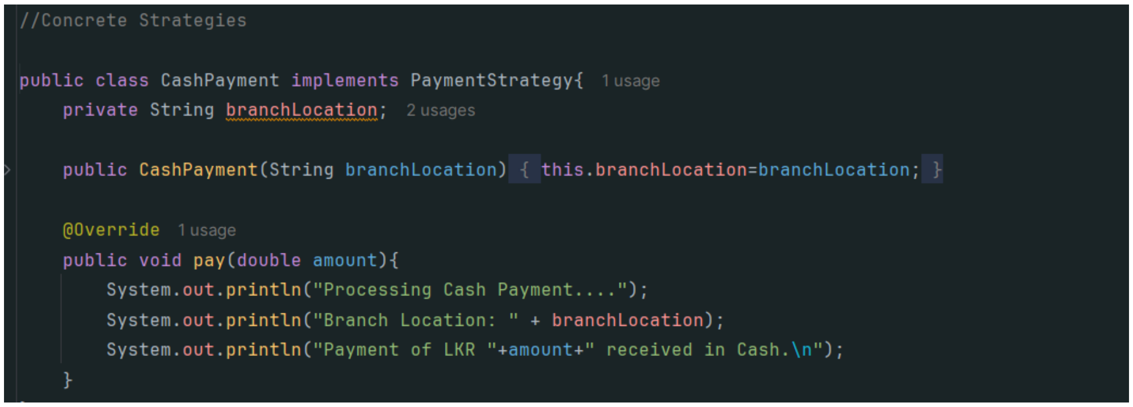
b) Concrete classes

- BankTransferPayment.java



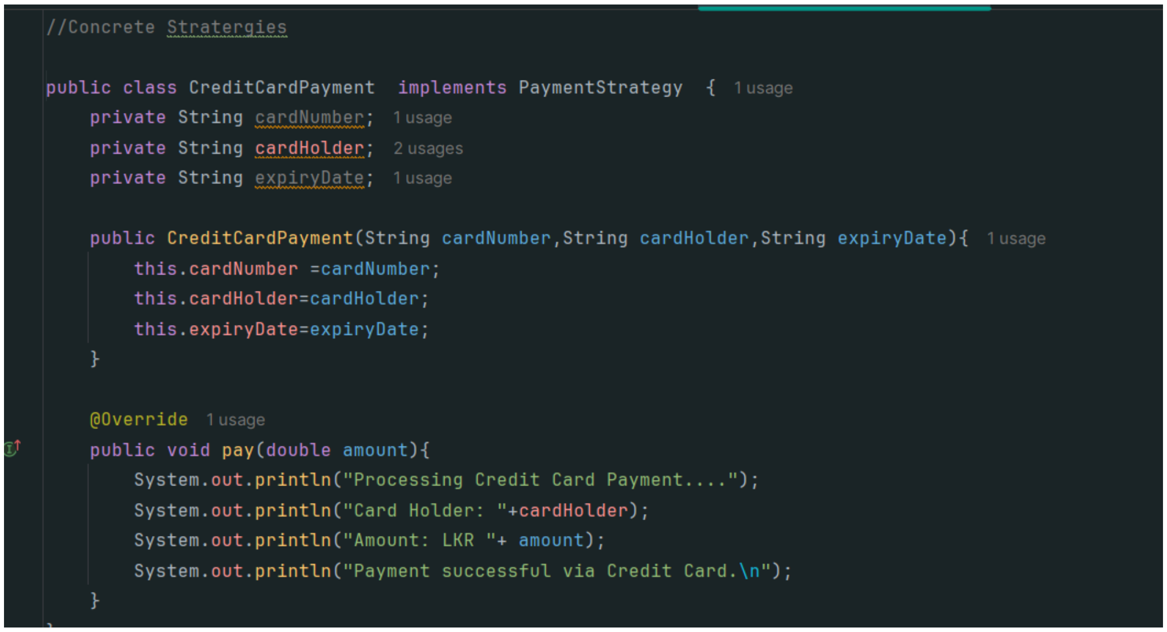
```
//Concrete Strategies

public class BankTransferPayment implements PaymentStrategy {  1 usage
    private String accountNumber;  2 usages
    private String bankName;  2 usages

    public BankTransferPayment(String accountNumber,String bankName){  1 usage
        this.accountNumber=accountNumber;
        this.bankName=bankName;
    }

    @Override  1 usage
    public void pay(double amount){
        System.out.println("Initiating Bank Transfer.....");
        System.out.println("Bank: "+bankName + ", Account No: "+accountNumber);
        System.out.println("Transferred LKR "+ amount + " Successfully via bank transfer.\n");
    }
```

- CashPayment.java



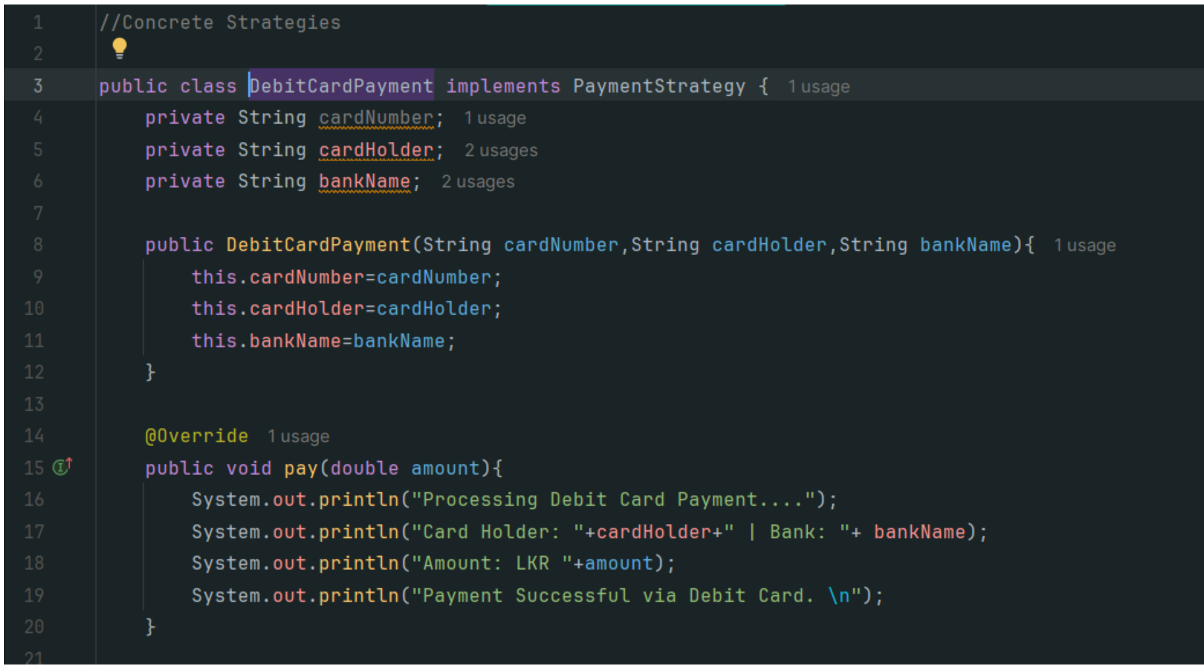
```
//Concrete Strategies

public class CashPayment implements PaymentStrategy{  1 usage
    private String branchLocation;  2 usages

    public CashPayment(String branchLocation) { this.branchLocation=branchLocation; }

    @Override  1 usage
    public void pay(double amount){
        System.out.println("Processing Cash Payment....");
        System.out.println("Branch Location: " + branchLocation);
        System.out.println("Payment of LKR "+amount+" received in Cash.\n");
    }
```

- CreditCardPayment.java



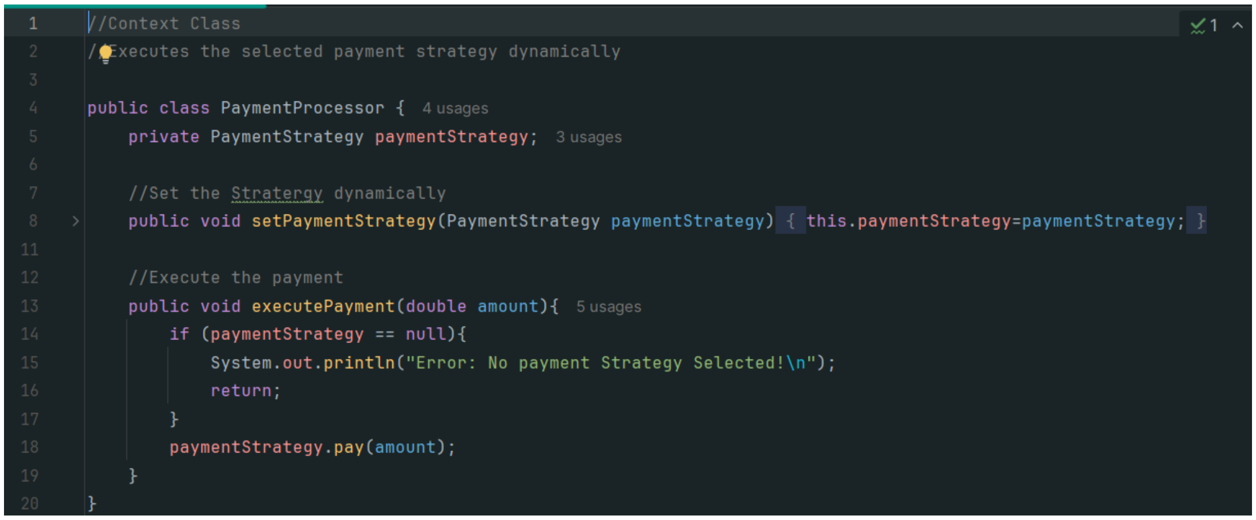
```
//Concrete Stratergies

public class CreditCardPayment  implements PaymentStrategy  {  1 usage
    private String cardNumber;  1 usage
    private String cardHolder;  2 usages
    private String expiryDate;  1 usage

    public CreditCardPayment(String cardNumber,String cardHolder,String expiryDate){  1 usage
        this.cardNumber =cardNumber;
        this.cardHolder=cardHolder;
        this.expiryDate=expiryDate;
    }

    @Override  1 usage
    public void pay(double amount){
        System.out.println("Processing Credit Card Payment....");
        System.out.println("Card Holder: "+cardHolder);
        System.out.println("Amount: LKR "+ amount);
        System.out.println("Payment successful via Credit Card.\n");
    }
```

- DebitCardPayment.java



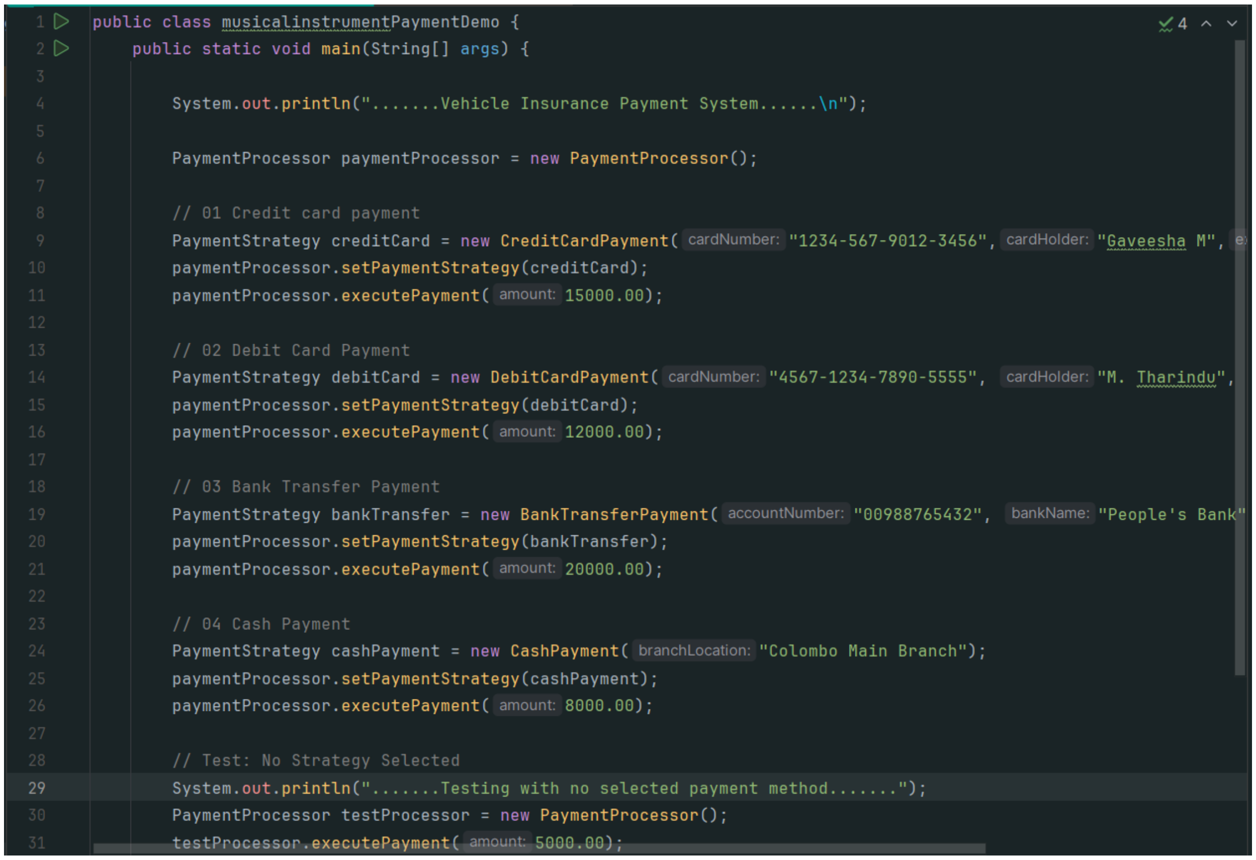
```java
//Concrete Strategies

public class DebitCardPayment implements PaymentStrategy {
    private String cardNumber;
    private String cardHolder;
    private String bankName;

    public DebitCardPayment(String cardNumber,String cardHolder,String bankName){
        this.cardNumber=cardNumber;
        this.cardHolder=cardHolder;
        this.bankName=bankName;
    }

    @Override
    public void pay(double amount){
        System.out.println("Processing Debit Card Payment....");
        System.out.println("Card Holder: "+cardHolder+" | Bank: "+ bankName);
        System.out.println("Amount: LKR "+amount);
        System.out.println("Payment Successful via Debit Card. \n");
    }
```

c) Context class



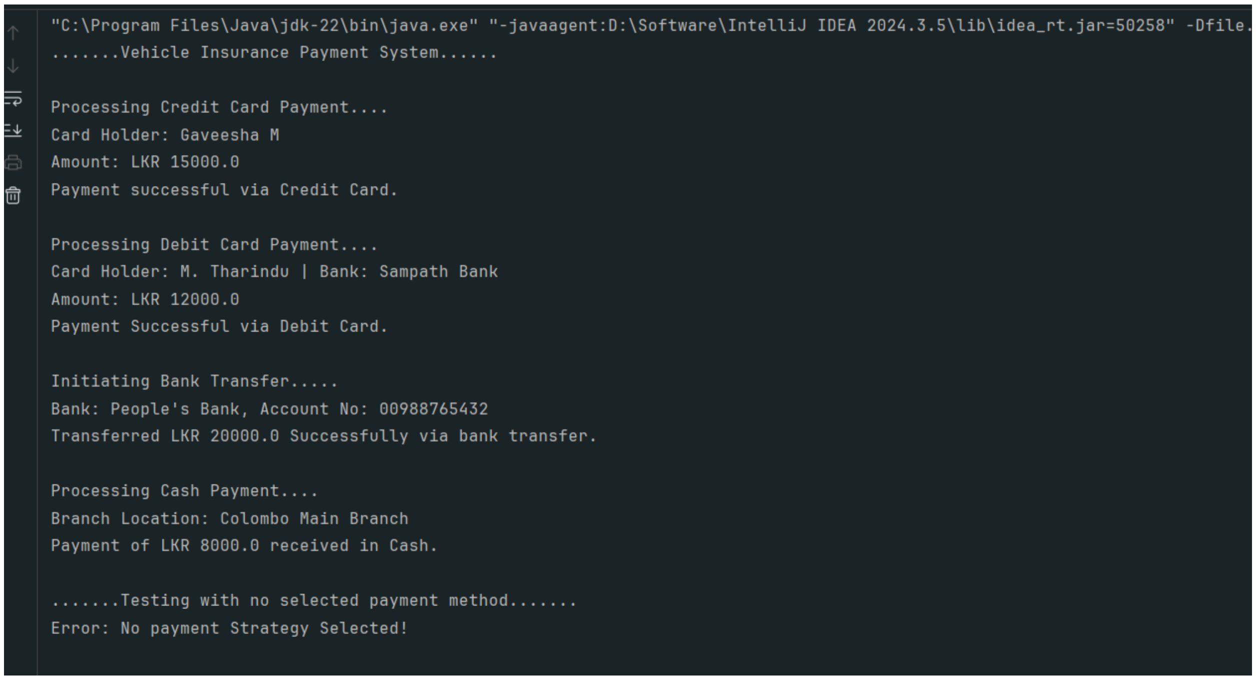
```java
//Context Class
//Executes the selected payment strategy dynamically

public class PaymentProcessor {
    private PaymentStrategy paymentStrategy;

    //Set the Stratergy dynamically
    public void setPaymentStrategy(PaymentStrategy paymentStrategy) { this.paymentStrategy=paymentStrategy; }

    //Execute the payment
    public void executePayment(double amount){
        if (paymentStrategy == null){
            System.out.println("Error: No payment Strategy Selected!\n");
            return;
        }
        paymentStrategy.pay(amount);
    }
}
```

d) Main class



```
public class musicalinstrumentPaymentDemo {
    public static void main(String[] args) {

        System.out.println(".......Vehicle Insurance Payment System......\n");

        PaymentProcessor paymentProcessor = new PaymentProcessor();

        // 01 Credit card payment
        PaymentStrategy creditCard = new CreditCardPayment( cardNumber: "1234-567-9012-3456", cardHolder: "Gaveesha M", e
        paymentProcessor.setPaymentStrategy(creditCard);
        paymentProcessor.executePayment( amount: 15000.00);

        // 02 Debit Card Payment
        PaymentStrategy debitCard = new DebitCardPayment( cardNumber: "4567-1234-7890-5555",  cardHolder: "M. Tharindu",
        paymentProcessor.setPaymentStrategy(debitCard);
        paymentProcessor.executePayment( amount: 12000.00);

        // 03 Bank Transfer Payment
        PaymentStrategy bankTransfer = new BankTransferPayment( accountNumber: "00988765432",  bankName: "People's Bank"
        paymentProcessor.setPaymentStrategy(bankTransfer);
        paymentProcessor.executePayment( amount: 20000.00);

        // 04 Cash Payment
        PaymentStrategy cashPayment = new CashPayment( branchLocation: "Colombo Main Branch");
        paymentProcessor.setPaymentStrategy(cashPayment);
        paymentProcessor.executePayment( amount: 8000.00);

        // Test: No Strategy Selected
        System.out.println(".......Testing with no selected payment method.......");
        PaymentProcessor testProcessor = new PaymentProcessor();
        testProcessor.executePayment( amount: 5000.00);
```

e) Output



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\Software\IntelliJ IDEA 2024.3.5\lib\idea_rt.jar=50258" -Dfile.
.......Vehicle Insurance Payment System......

Processing Credit Card Payment....
Card Holder: Gaveesha M
Amount: LKR 15000.0
Payment successful via Credit Card.

Processing Debit Card Payment....
Card Holder: M. Tharindu | Bank: Sampath Bank
Amount: LKR 12000.0
Payment Successful via Debit Card.

Initiating Bank Transfer.....
Bank: People's Bank, Account No: 00988765432
Transferred LKR 20000.0 Successfully via bank transfer.

Processing Cash Payment....
Branch Location: Colombo Main Branch
Payment of LKR 8000.0 received in Cash.

.......Testing with no selected payment method.......
Error: No payment Strategy Selected!
```