

실습 1:

-실습코드

```
package 실습과제1030;

class CreditCardPayment {
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using Credit Card.");
    }
}

class PayPalPayment {
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using PayPal.");
    }
}

// 새로 추가된 BankTransferPayment 클래스
class BankTransferPayment {
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using Bank Transfer.");
    }
}
```

```
class PaymentProcessor {
    private CreditCardPayment creditCardPayment;
    private PayPalPayment payPalPayment;
    private BankTransferPayment bankTransferPayment; // BankTransferPayment 추가

    public PaymentProcessor() {
        this.creditCardPayment = new CreditCardPayment();
        this.payPalPayment = new PayPalPayment();
        this.bankTransferPayment = new BankTransferPayment(); // BankTransferPayment 초기화
    }

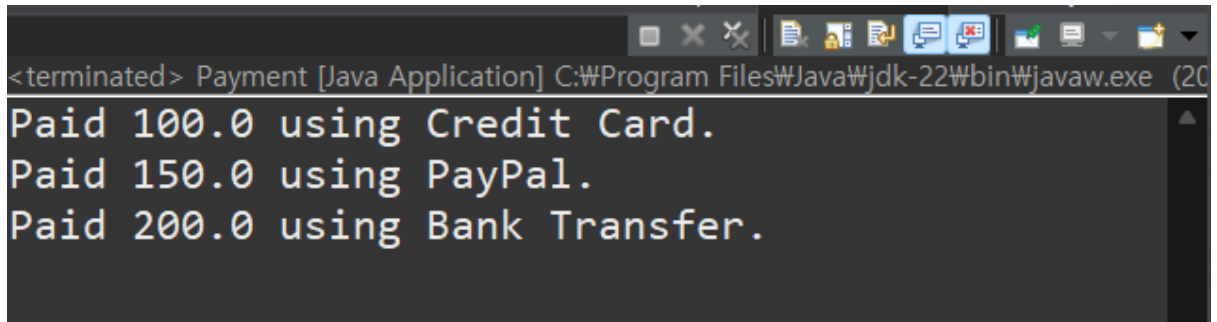
    public void processPayment(String method, double amount) {
        switch (method) {
            case "CreditCard":
                creditCardPayment.pay(amount);
                break;
            case "PayPal":
                payPalPayment.pay(amount);
                break;
            case "BankTransfer": // BankTransfer 결제 방식 추가
                bankTransferPayment.pay(amount);
                break;
            default:
                System.out.println("Unsupported payment method.");
                break;
        }
    }
}
```

```

public class Payment {
    public static void main(String[] args) {
        PaymentProcessor paymentProcessor = new PaymentProcessor();
        paymentProcessor.processPayment("CreditCard", 100.0);
        paymentProcessor.processPayment("PayPal", 150.0);
        paymentProcessor.processPayment("BankTransfer", 200.0); // BankTransfer 테스트
    }
}

```

-결과출력



```

<terminated> Payment [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (20
Paid 100.0 using Credit Card.
Paid 150.0 using PayPal.
Paid 200.0 using Bank Transfer.

```

-코드관련 설명

# BankTransferPayment 클래스를 추가하여, pay 메서드를 포함하여 BankTransferPayment 클래스를 작성했습니다.

# PaymentProcessor 클래스 수정합니다. BankTransferPayment 필드를 추가하고, 생성자에서 이를 초기화했습니다. 또한 processPayment 메서드에 "BankTransfer" 옵션을 추가하여 BankTransfer 결제 방식을 처리할 수 있도록 했습니다.

# main 메서드에서 processPayment 메서드를 사용하여 "BankTransfer" 방식으로 결제를 테스트했습니다.

실습 2:

-실습 코드

```
// PaymentMethod 추상화 클래스 정의
abstract class PaymentMethod {
    public abstract void pay(double amount);
}

// CreditCardPayment 클래스
class CreditCardPayment extends PaymentMethod {
    @Override
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using Credit Card.");
    }
}

// PayPalPayment 클래스
class PayPalPayment extends PaymentMethod {
    @Override
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using PayPal.");
    }
}

// BankTransferPayment 클래스
class BankTransferPayment extends PaymentMethod {
    @Override
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using Bank Transfer.");
    }
}
```

```
// PaymentProcessor 클래스
class PaymentProcessor {
    private PaymentMethod paymentMethod;

    // 생성자에서 PaymentMethod 타입을 받아옴
    public PaymentProcessor(PaymentMethod paymentMethod) {
        this.paymentMethod = paymentMethod;
    }

    // processPayment 메서드
    public void processPayment(double amount) {
        paymentMethod.pay(amount);
    }
}

// 테스트용 메인 클래스
public class PaymentDIP {
    Run | Debug
    public static void main(String[] args) {
        PaymentMethod creditCardPayment = new CreditCardPayment();
        PaymentMethod payPalPayment = new PayPalPayment();
        PaymentMethod bankTransferPayment = new BankTransferPayment();

        PaymentProcessor paymentProcessor1 = new PaymentProcessor(creditCardPayment);
        paymentProcessor1.processPayment(amount:100.0);

        PaymentProcessor paymentProcessor2 = new PaymentProcessor(payPalPayment);
        paymentProcessor2.processPayment(amount:150.0);

        PaymentProcessor paymentProcessor3 = new PaymentProcessor(bankTransferPayment);
        paymentProcessor3.processPayment(amount:200.0);
    }
}
```

-결과 출력

```
[Running] cd "d:\고흥규\2024 한국외대 수학과
2학년\객체지향프로그래밍\1030 실습과제\" && javac PaymentDIP.java &&
java PaymentDIP
Paid 100.0 using Credit Card.
Paid 150.0 using PayPal.
Paid 200.0 using Bank Transfer.
```

-코드관련 설명

# PaymentMethod 추상 클래스는 모든 결제 방식 클래스가 구현해야 하는 pay 메서드를 정의한 추상 클래스입니다.

# 개별 결제 방식 클래스인 CreditCardPayment, PayPalPayment, BankTransferPayment는

모두 PaymentMethod 클래스를 상속받아 pay 메서드를 오버라이딩하여 각 결제 방식에 맞는 로직을 구현합니다.

# PaymentProcessor 클래스는 PaymentMethod 타입을 생성자로 받아오며, processPayment 메서드를 통해 결제를 수행합니다. PaymentProcessor는 특정 결제 방식에 의존하지 않으므로 결제 방식이 추가되어도 코드 수정이 필요하지 않습니다.

#SOLID 원칙:

1. Open-Closed Principle): 코드가 확장에는 열려 있고 수정에는 닫혀 있습니다. 새로운 결제 방식이 추가될 때 PaymentProcessor 코드를 수정할 필요 없이 새로운 결제 클래스를 작성해 추가할 수 있습니다.
2. Dependency Inversion Principle : PaymentProcessor는 구체적인 결제 방식이 아닌 추상화된 PaymentMethod에 의존하여 유연성이 높아졌습니다.

이렇게 수정하면, 새로운 결제 방식이 추가될 때도 PaymentProcessor를 수정하지 않아도 되기 때문에 코드의 유연성과 유지보수성이 높아집니다.