Come to HUFS Meet the World!

객체지향프로그래밍

24.10.30 실습

- 담당 교수 : 전 병 환 교수님
- 조교 : 윤 종 엄
- 조교 메일 : juyoon@hufs.ac.kr

10월 16일 실습1 정답 코드

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

```java
class Account {
    private String name;
    private int account_num;
    private int balance;

    Account(){
        this("", 0, 0);
    }

    Account(String name, int account_num, int balance) {
        this.name = name;
        this.account_num = account_num;
        this.balance = balance;
    }

    public String getName() {
        return name;
    }

    public int getAccNo() {
        return account_num;
    }

    public int getBalance() {
        return balance;
    }

    public void transaction(int balance) {

        if (this.balance + balance < 0) {
            System.out.println("잔액이 부족합니다.");
        }else {
            this.balance += balance;
        }
    }

    public static void main(String[] args) {
        Account account1 = new Account("A", 0, 100000);
        Account account2 = new Account("B", 0, 100000);

        account1.transaction(-150000);
        account1.transaction(+5000);
        account1.transaction(-2000);
        account1.transaction(-30000);

        account2.transaction(+5000);
        account2.transaction(+5000);
        account2.transaction(+5000);

        System.out.println(account1.getBalance());
        System.out.println(account2.getBalance());

    }
}
```

10월 16일 실습2 정답 코드

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

```java
public class Card {

    static Account account;
    private String cardName;

    Card(Account a){
        this(a, "");
    }

    Card(Account a, String name){
        account = a;//= new Account();
        cardName = name;
    }


    public String getCardName() {
        return cardName;
    }

    public void transaction(int value) {

        account.transaction(value);
    }

    public int inquiry() {
        return account.getBalance();
    }

    public String getAccountName() {

        return account.getName();
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Account accHUFS = new Account("HUFS", 0, 100000);
        Account accYonsei = new Account("Yonsei", 0, 100000);

        //HUFS계좌에 연동된 여러 개 카드
        Card a = new Card(accHUFS, "a");
        Card b = new Card(accHUFS, "b");
        Card c = new Card(accHUFS, "c");

        a.transaction(10000);
        b.transaction(-3000);
        c.transaction(-5000);
        a.transaction(20000);

        System.out.println(a.getCardName()+"님 카드, " + a.getAccountName() + "의 현재 잔액: " + a.inquiry());
        System.out.println(b.getCardName()+"님 카드, " + b.getAccountName() + "의 현재 잔액: " + b.inquiry());
        System.out.println(c.getCardName()+"님 카드, " + c.getAccountName() + "의 현재 잔액: " + c.inquiry());

        //Yonsei 계좌에 연동된 여러 개 카드
        Card d = new Card(accYonsei, "d");
        Card e = new Card(accYonsei, "e");
        Card f = new Card(accYonsei, "f");

        d.transaction(20000);
        e.transaction(-5000);
        f.transaction(-6000);
        d.transaction(10000);

        System.out.println(d.getCardName()+"님 카드, " + d.getAccountName() + "의 현재 잔액: " + d.inquiry());
        System.out.println(e.getCardName()+"님 카드, " + e.getAccountName() + "의 현재 잔액: " + e.inquiry());
        System.out.println(f.getCardName()+"님 카드, " + f.getAccountName() + "의 현재 잔액: " + f.inquiry());


    }
}
```

실습 1: 결제 방식 프로그램 작성

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

문제 설명:

- 아래의 코드는 현재 **CreditCardPayment**와 **PayPalment** 결제 방식만 처리하고 있다. 새 결제 방식인 **BankTransferPayment** 를 사용할 수 있도록 수정하라.

  - **BankTransferPayment** 클래스를 작성하고, **pay** 메서드 구현
  - **PaymentProcessor** 클래스 내에 **BankTransferPayment** 결제방법이 동작하도록 코드 수정
  - 테스트

아래 코드를 수정하여 동작하도록 작성


```java
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.");
    }
}



class PaymentProcessor {
    private CreditCardPayment creditCardPayment;
    private PayPalPayment payPalPayment;
    public PaymentProcessor() {
        this.creditCardPayment = new CreditCardPayment();
        this.payPalPayment = new PayPalPayment();
    }
    public void processPayment(String method, double amount) {
        switch (method) {
        case "CreditCard":
            creditCardPayment.pay(amount);
            break;
        case "PayPal":
            payPalPayment.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);
        // BankTransfer 추가 후 테스트해보기
    }
}
```

```
Paid 100.0 using Credit Card.
Paid 150.0 using PayPal.
Paid 200.0 using Bank Transfer.
```

다형성 – 메소드 오버라이딩 예시 코드

36

```
class Shape { // 슈퍼 클래스
    public Shape next;
    public Shape() { next = null; }

    public void draw() {
        System.out.println("Shape");
    }
}

class Line extends Shape {
    public void draw() { // 메소드 오버라이딩
        System.out.println("Line");
    }
}

class Rect extends Shape {
    public void draw() { // 메소드 오버라이딩
        System.out.println("Rect");
    }
}

class Circle extends Shape {
    public void draw() { // 메소드 오버라이딩
        System.out.println("Circle");
    }
}
```

```
public class MethodOverridingEx {
    static void paint(Shape p) {
        p.draw(); // p가 가리키는 객체 내에 오버라이딩된 draw() 호출.
                  // 동적 바인딩
    }

    public static void main(String[] args) {
        Line line = new Line();
        paint(line);
        paint(new Shape());
        paint(new Line());
        paint(new Rect());
        paint(new Circle());
    }
}
```

```
Line
Shape
Line
Rect
Circle
```

실습 2: 결제 방식 프로그램 작성(다형성 적용)

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

문제 설명:

- 실습 1은 각 결제 방식(**CreditCardPayment**, **PayPalPayment**, **BankTransferPayment**)이 개별 클래스로 구현되어 있고, **PaymentProcessor** 클래스는 각 결제 방식에 의존한다. 새로운 결제방식이 추가될 때 마다 **PaymentProcessor**를 수정해야 한다. 의존성을 낮추기 위하여 다형성을 이용하여 더 유연한 구조를 구현하고자 한다.
  - 개별 결제방식을 추상화하여 **PaymentMethod** 클래스를 작성
  - **CreditCardPayment**, **PayPalPayment**, **BankTransferPayment** 클래스는 **PaymentMethod** 클래스를 상속
  - pay() 메서드를 오버라이딩
  - **PaymentProcessor**가 구체적인 결제 방식에 직접 의존하지 않고, 추상화된 **PaymentMethod** 타입을 매개변수로 넘겨받아 **PaymentProcessor** 메서드 완성

추가 문제

- **실습 2를 통하여 SOLID원칙 중 어떤 원칙이 적용되었고, 개선된 코드가 어떤 면에서 더 유연성을 가지는지 작성하시오.**

아래 코드를 기반으로 동작하도록 작성

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

```java
// PaymentMethod 추상화 클래스 작성

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.");
    }
}

// BankTransferPayment ~~

class PaymentProcessor {

    //

    public PaymentProcessor() {

    //

    }
    public void processPayment(double amount) {


    }
}

public class PaymentDIP {
    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(100.0);

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

        PaymentProcessor paymentProcessor3 = new PaymentProcessor(bankTransferPayment);
        paymentProcessor3.processPayment(200.0);

    }
}
```

```
Paid 100.0 using Credit Card.
Paid 150.0 using PayPal.
Paid 200.0 using Bank Transfer.
```