

실습 1:

-실습코드

```
package 실습과제1016;

class Account {
    // 멤버 변수 선언
    private String name;
    private int account_num;
    private int balance;

    // 기본 생성자 작성
    public Account(){
        this.name="";
        this.account_num=0;
        this.balance=0;
    }
    // 인자가 있는 생성자 작성
    public Account(String name, int account_num,int balance) {
        this.name=name;
        this.account_num=account_num;
        this.balance=balance;
    }

    // 메서드
    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;
        }
    }
}
```

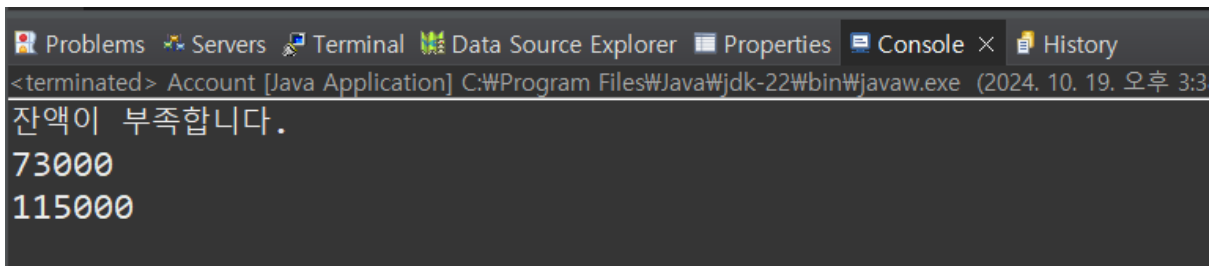
```
// 클라이언트 코드
public static void main(String[] args) {
    Account account1 = new Account("A", 0, 100000);
    Account account2 = new Account("B", 0, 100000);

    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());
}
}
```

-결과출력



```
<terminated> Account [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (2024. 10. 19. 오후 3:30)
잔액이 부족합니다.
73000
115000
```

-코드관련 설명

#String name,int account_num,int balance 멤버의 접근지정자를 private로 선언해 보호합니다.

#기본생성자를 통해 name을 ""으로, account_num을 0으로, balance를 0으로 초기화합니다. This를 씁니다.

#인자가 있는 생성자를 통해 name, account_num, balance매개변수로 받아 초기화 합니다

다. This를 씁니다.

#getName() 메소드를 통해 Account클래스의 name 을 반환하도록 메소드를 만듭니다.

#getAccNo() 메소드를 통해 account_num, getBalance()를 통해 balance를 반환합니다.

#transaction(int balance) 메소드를 통해 Account의 balance의 값을 수정하도록 합니다. 파라미터로 받은 값을 Account의 balance에 더하고, 그게 0보다 작으면 "잔액이 부족합니다" 라고 출력합니다. 0보다 크면 더한값을 Account의 balance에 저장합니다.

해당 메소드의 코드는 이렇습니다.

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

나머지 코드는 과제pdf의 코드 캡처된 부분을 코딩하면, Account 클래스가 완성됩니다.

실습 2:

-실습 코드

```
package 실습과제1016;

public class Card {
    static Account account;
    private String cardName;

    Card(Account a) {
        this(a, "");
    }
    Card(Account a, String name) {
        account=a;
        cardName=name;
    }
    public String getCardName() {
        return cardName;
    }
    public void transaction(int value) {
        account.transaction(value);
        // account의 함수를 활용
    }
    public int inquiry() {
        return (account).getBalance();
        // account의 함수를 활용
    }
    public String getAccountName() {
        return account.getName();
        // account의 함수를 활용
    }
}
```

```

public static void main(String[] args) {
    // 변수 선언
    Account accHUFS = new Account("HUFS", 0, 100000);
    Account accYonsei = new Account("Yonsei", 0, 100000);

    // HUFS 계좌에 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());
}
}

```

-결과 출력

```

<terminated> Card [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (2024. 10. 19. 오후 3:38:48 - 오후 3:38:48) [pid: 40116]
a님 카드, HUFS의 현재 잔액: 122000
b님 카드, HUFS의 현재 잔액: 122000
c님 카드, HUFS의 현재 잔액: 122000
d님 카드, Yonsei의 현재 잔액: 119000
e님 카드, Yonsei의 현재 잔액: 119000
f님 카드, Yonsei의 현재 잔액: 119000

```

-코드관련 설명

#위에서 만든 Account 클래스를 씁니다. Account 클래스는 접근지정자가 디폴트이므로 같은 패키지 내에서 쓸 수 있습니다. 따라서 Card 클래스로 컴포지션 합니다.

#static으로 Account클래스의 account라는 멤버를 선언합니다. String cardName도 선언합니다. 이때 보호를 위해 접근지정자는 private로 합니다.

#생성자를 만듭니다. 두개의 파라미터를 받는데, 하나는 Account 클래스의 객체를 받아 account에 저장하고, 하나는 문자열 name를 받아 cardName 에 저장합니다.

#getCardName() 메소드를 통해 cardName을 반환합니다.

#transaction(int value) 메소드를 통해 위에서 저장한 객체 account의 함수들을 활용합니다. 객체 account는 클래스 Account에서 만들어졌으므로 클래스 Account안에 있는 함수 transaction(),getBalance(),getName() 등을 쓸 수 있습니다. 따라서 Card의 메소드 transaction이 int value를 파라미터로 받기에 *account.transaction(value)*; 이렇게 코딩하면 됩니다. 그러면 account의 balance 값을 바꿀 수 있습니다.

#마찬가지로 Card 클래스의 inquiry 함수를 통해 잔액을 확인하고자 한다면, (*account*).getBalance(); 라고 하고, Card 클래스의 getAccountName() 메소드를 만들고자 한다면 *account.getName()*; 값을 반환하면 됩니다.

#이렇게 추가로 입력해야 하는 코드들은 끝났고, 나머지는 과제의 캡처된 코드를 따라 작성하고 run 하면 원하는 출력값을 얻을 수 있습니다.

#출력되는 과정을 간략히 설명하자면, 먼저 Account 클래스의 accHUFs,accYonsei 객체를 만듭니다. Account는 static하므로 accHUFs,accYonsei

Card 클래스의 객체 a,b,c,를 만듭니다. 해당 객체들은 파라미터로 accHUFs, name 을 받습니다. Account는 static하므로 그에 따른 객체들 accHUFs,accYonsei 역시 각각 static하게, accHUFs,accYonsei 각각의 객체들에 관계없이 유지되는 값들이 존재합니다.

따라서, Card 클래스의 객체 a,b,c는 각각 name이 다르지만, accHUFs라는 객체 속 서로 공유하는 값들(account_num,balance) 등이 있습니다. 이를 이용해서 공유하는 은행 계좌의 입출금과 조회를 하면 됩니다. 그건 위에서 만든 transaction, getCardName, getAccountName(),inquiry() 등의 메소드를 쓰면됩니다.

#Card의 객체 d,e,f, 도 마찬가지로 방법으로 만들고, 이후 메소드를 사용한 값들을 출력하면 됩니다.