임베디드 시스템 실습 lab7

2016310936 우승민

이번 exercise는 JAVA에서의 exception의 사용과 array list 사용입니다. 첫번째 exercise 코드입니다.

```
class BankAccount{
        double balance;

        public BankAccount (double b){
                try{
                        if(b < 0){
                                IllegalArgumentException exception = new Illegal
ArgumentException("negative balance");
                                throw exception;
                        }

                } catch (IllegalArgumentException ex){
                        System.out.println("contruct failed.");
                        ex.printStackTrace();
                }


                this.balance = b;
        }

        public void deposit (double amount) {
                if(amount < 0){
                        IllegalArgumentException exception = new IllegalArgument
Exception("negative amout is deposited");
                        throw exception;
                }


                balance = balance + amount;
        }

        public void withdraw (double amount) {
                if(amount > balance || amount < 0){
                        IllegalArgumentException exception = new IllegalArgument
Exception("Amount exceeds balance");
                        throw exception;
                }

                balance = balance - amount;
        }

}
```

BankAccount class의 구성입니다. 생성자에서 만약 balance가 음수일 경우에 exception이 발생하도록 하였고, deposit 함수에서 입금 금액이 음수일 경우 exception이 발생하였고, withdraw 함수에서 출금금액이 음수이거나 balance를 초과할 경우에 exception이 발생하도록 하였습니다.

위 class 를 실행하는 코드입니다.

```
public class Week7_1 {
        public static void main(String[] args){
                BankAccount acc = new BankAccount(-100.0);
                try{
                        acc.deposit(-100.0);
                } catch (IllegalArgumentException ex){
                        System.out.println("deposit failed.");
                        ex.printStackTrace();
                }

                try{
                        acc.withdraw(200.0);
                } catch (IllegalArgumentException ex){
                        System.out.println("Withdraw failed.");
                        ex.printStackTrace();
                }


        }
}
```

처음에 계좌를 만들 때 음수를 사용하고, 입금도 음수, 출금도 계좌 금액을 넘도록 만들어 모든 exception 을 발생하도록 하였습니다.

실행 화면입니다.

```
seungmin@seungmin-W65-W67RC:~/java/3$ javac Week7_1.java
seungmin@seungmin-W65-W67RC:~/java/3$ java Week7_1
contruct failed.
java.lang.IllegalArgumentException: negative balance
        at BankAccount.<init>(Week7_1.java:7)
        at Week7_1.main(Week7_1.java:43)
deposit failed.
java.lang.IllegalArgumentException: negative amout is deposited
        at BankAccount.deposit(Week7_1.java:22)
        at Week7_1.main(Week7_1.java:45)
Withdraw failed.
java.lang.IllegalArgumentException: Amount exceeds balance
        at BankAccount.withdraw(Week7_1.java:32)
        at Week7_1.main(Week7_1.java:52)
```

두 번째 exercise 인 array list 를 사용하는 코드입니다. 먼저 array list 로 사용되는 class 인 Bankac class 입니다. balance 만 저장하고 있습니다.

```
import java.util.ArrayList;

class Bankac{
        double balance;

        public Bankac(double balance){
                this.balance = balance;
        }

}

class Bank {

        ArrayList<Bankac> BankAccount = new ArrayList<Bankac>();

        public Bank(){};

        public void addAccount(double initalBalance){
                BankAccount.add(new Bankac(initalBalance));
        }

        public double getBalance(int account){
                return BankAccount.get(account).balance;
        }


        public void deposit(int account, double amount){
                double a = getBalance(account) + amount;
                BankAccount.remove(account);
                BankAccount.add(account, new Bankac(a));

        }

        public void withdraw(int account, double amount){
                double a = getBalance(account) - amount;
                BankAccount.remove(account);
                BankAccount.add(account, new Bankac(a));

        }

}
```

Bank class 에서 Bankac class 를 array list 로 선언하고, 각 필요한 함수에 맞게 구현하였습니다.

1. addAccount 에서는 E.add 로 새로운 arraylist 를 추가하였고

2. getBalacne 에서는 E.get 으로 account 의 index 에 해당하는 계좌의 금액을 return 합니다.

3. deposit, withdraw 에서는 기존의 금액을 getBalance 함수를 사용하여 알아내고 각 상황에 맞게 출금/입금할 금액을 추가하여 기존의 것을 제거하고 새로 추가합니다.

실행 코드입니다.

```java
public class Week7_2 {
	public static void main (String[] args){
		Bank bank = new Bank();
		bank.addAccount(50);
		bank.addAccount(150);
		bank.addAccount(550);
		bank.addAccount(250);
		bank.addAccount(590);

		System.out.println("0 : " + bank.getBalance(0));
		System.out.println("1 : " + bank.getBalance(1));
		System.out.println("2 : " + bank.getBalance(2));
		System.out.println("3 : " + bank.getBalance(3));
		System.out.println("4 : " + bank.getBalance(4) + "\n");

		bank.deposit(0, 13);
		System.out.println("deposit 0, 13");
		bank.deposit(2, 153);
		System.out.println("deposit 2, 153");
		bank.deposit(4, 123);
		System.out.println("deposit 4, 123\n");


		bank.withdraw(0, 1);
		System.out.println("withdraw 0, 1");
		bank.withdraw(1, 100);
		System.out.println("withdraw 1, 100");
		bank.withdraw(3, 3);
		System.out.println("withdraw 3, 3");
		bank.withdraw(4, 90);
		System.out.println("withdraw 4, 90\n");


		System.out.println("0 : " + bank.getBalance(0));
		System.out.println("1 : " + bank.getBalance(1));
		System.out.println("2 : " + bank.getBalance(2));
		System.out.println("3 : " + bank.getBalance(3));
		System.out.println("4 : " + bank.getBalance(4));

	}
}
```

5 개의 계좌를 만들고, 일부는 출금, 일부는 입금을 하며 금액의 변화를 살펴보도록 하였습니다.

실행 화면입니다.

```
seungmin@seungmin-W65-W67RC:~/java/3$ javac Week7_2.java
seungmin@seungmin-W65-W67RC:~/java/3$ java Week7_2
0 : 50.0
1 : 150.0
2 : 550.0
3 : 250.0
4 : 590.0

deposit 0, 13
deposit 2, 153
deposit 4, 123

withdraw 0, 1
withdraw 1, 100
withdraw 3, 3
withdraw 4, 90

0 : 62.0
1 : 50.0
2 : 703.0
3 : 247.0
4 : 623.0
```