

# ATM 프로그램

2015341004 김민영

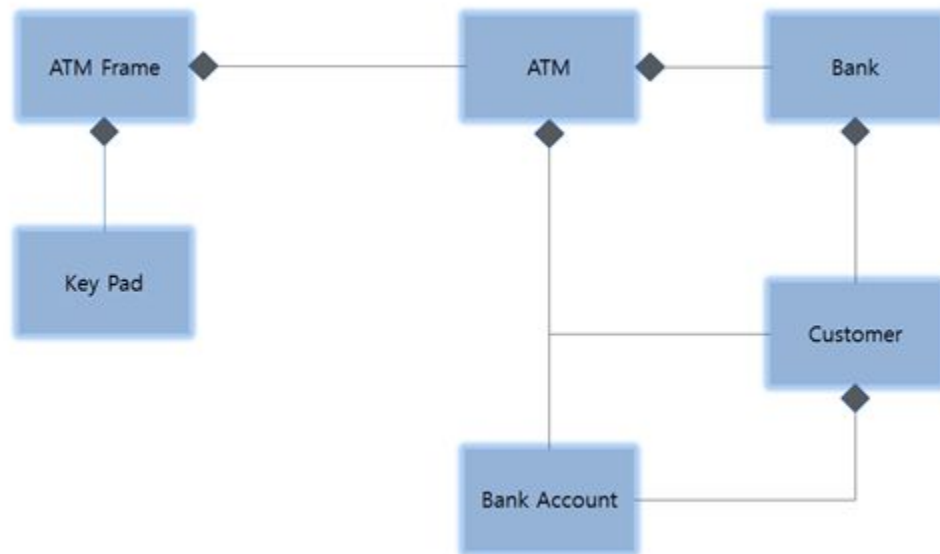
2015341005 김선경



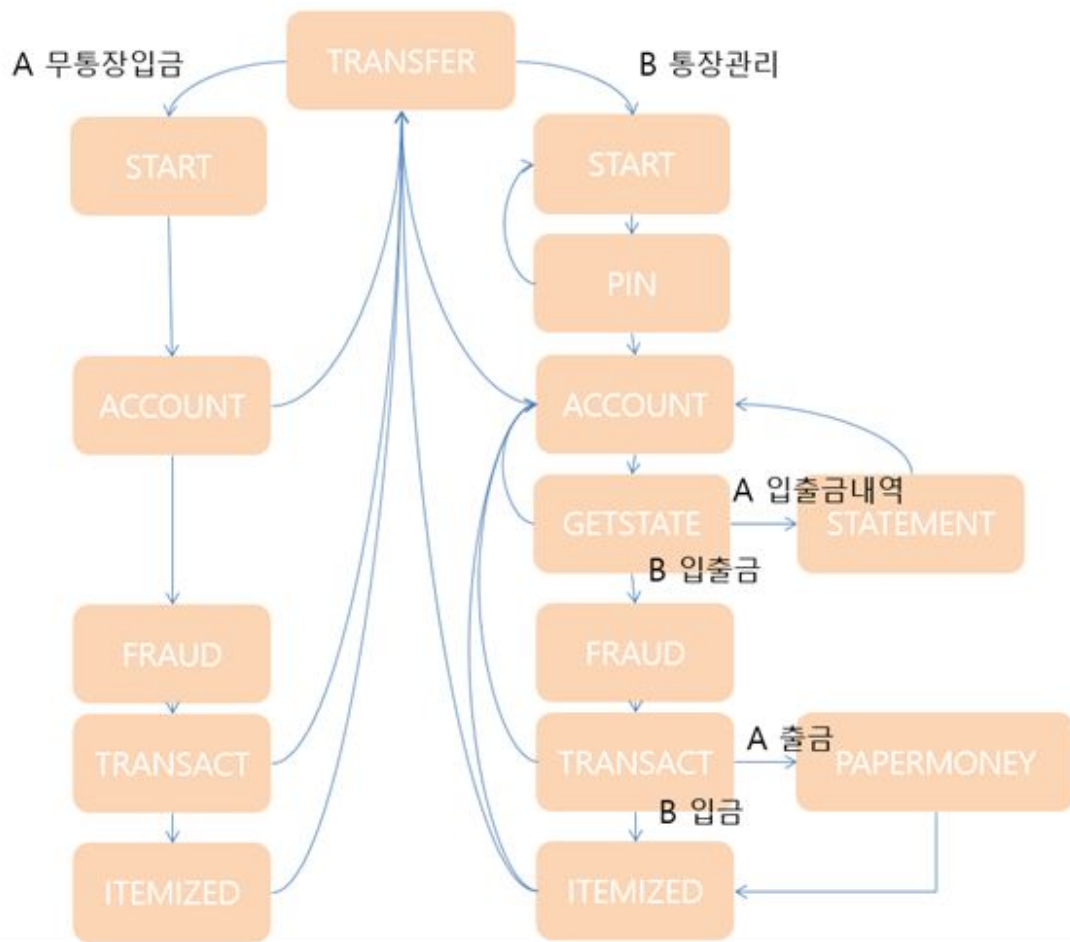
# Agenda

- ATM 프로그램 구조 설명
- 새로 추가한 코드
- 프로그램 소스
- 프로그램 실행 결과

# ATM 프로그램 구조



# State 순서도



# 새로 추가한 상수 및 변수

상수 선언 >>

```
private int state;
private int transfer;    //1: transfer선택, 2: transact선택
private int customerNumber;
private Customer currentCustomer;
private BankAccount currentAccount;
private Bank theBank;

public static final int START =1;
public static final int PIN = 2;
public static final int ACCOUNT = 3;
public static final int TRANSACT = 4;
public static final int ITEMIZED = 5;    //명세표
public static final int FRAUD = 6;       //사기거래
public static final int PAPERMONEY = 7;  //출금시 지폐권선택단계
public static final int STATEMENT = 8;   //거래내역출력
public static final int GETSTATE = 9;    //거래내역 or 일반거래 선택단계
public static final int TRANSFER = 10;   //무통장입금 or 일반입출금 선택단계
```

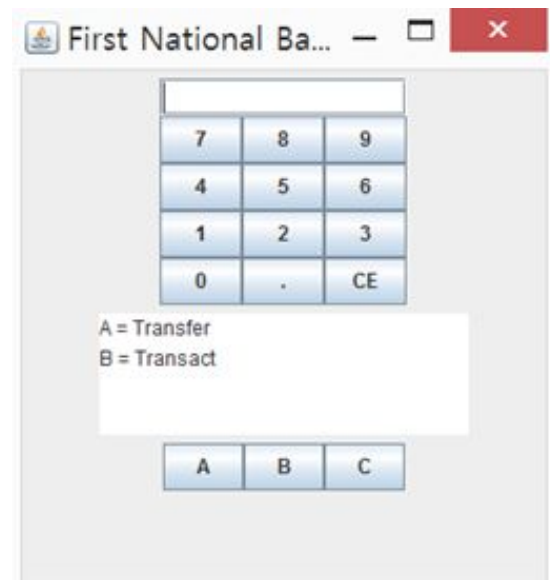
# 새로 추가한 상수 및 변수

변수 선언 >>

```
private ATM theATM;  
private Customer customer; // 고객정보를 불러오기 위해 선언  
private double amount;    // 거래한 금액  
private double finalamount;  
private double[] statement; //거래내역에서 거래한 금액 저장  
private String[] sta;      //거래내역에서 입/출금 을 저장  
private int count;  
private double limit = 1000000; //출금한도  
  
private Date d;            // 날짜를 불러옴
```

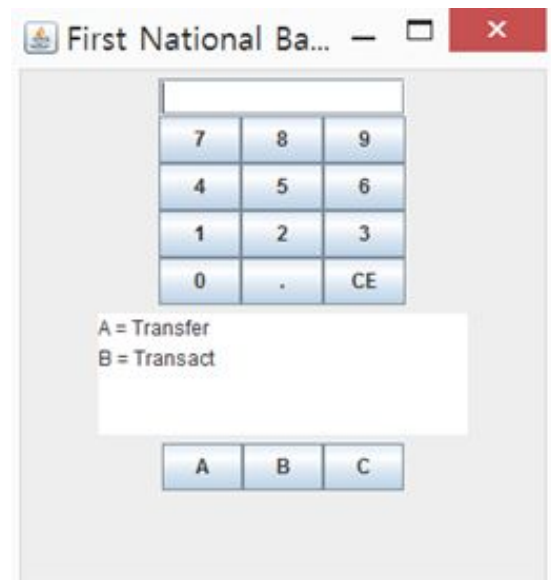
# 무통장 입금 & 본인 통장

```
public void transfer() {           //transfer메소드
    assert state == TRANSFER; //transfer상태일때
    transfer =1;                 //transfer를 1로 바꿈
    state =START;                //상태를 start로 바꿈
}
```



# 무통장 입금 & 본인 통장 showstate

```
else if (state == ATM.TRANSFER)
    display.setText("A = Transfer\nB = Transact");
```





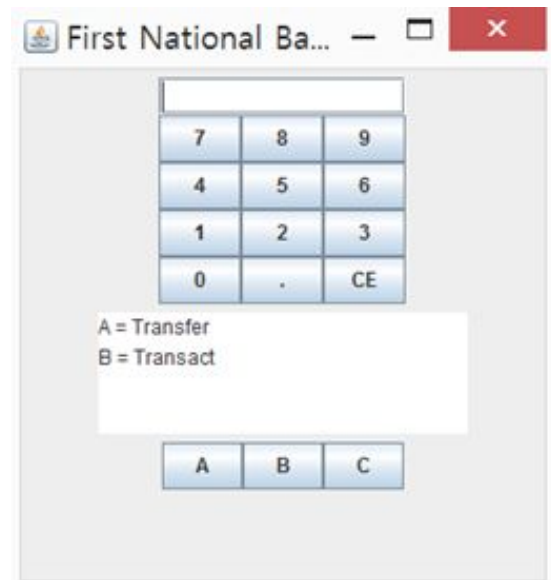
# 무통장 입금 & 본인 통장 이벤트

Abutton

```
} else if (state == ATM.TRANSFER) { //무통장거래 선택장면  
    theATM.transfer(); //무통장거래 선택
```

Bbutton

```
else if (state == ATM.TRANSFER) //무통장입금이 아닌 일반거래를 선택할 경우  
    theATM.start(); //start상태로 변환
```



# 1. 무통장 입금(START)

```
public void transferCustomerNumber(int number) { //transfer가 1일때 호출됨 (손님번호를 받아옴)
    assert state == START; //start상태일때
    customerNumber = number; //customerNumber를 받아온 값으로 저장
    currentCustomer = theBank.findCustomer(customerNumber); //bank에서 손님번호에 맞는 손님을 불러옴
    state = ACCOUNT; //상태를 account로 바꿈
}
```

The screenshot shows a Java Swing window titled "First National...". Inside the window, there is a numeric keypad with buttons for digits 0-9, a decimal point, and "CE". Above the keypad is a small text field containing the number "1". Below the keypad is a text input area with the prompt "Enter customer number" and the text "A = OK". At the bottom of the window are three buttons labeled "A", "B", and "C".

# 무통장 입금(TRANSACT)

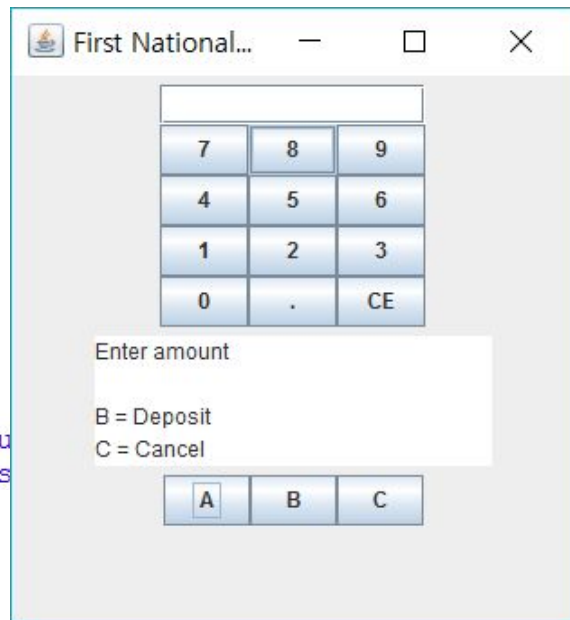
```
else if (state == ATM.TRANSACT) {  
    if (theATM.gettransfer() == 1) //transfer=1 즉 무통장입금상태일때  
        display.setText("Enter amount\n" + "\nB = Deposit\nC = Cancel");  
    else  
        display.setText("Balance = " + theATM.getBalance() + "\nEnter amount\n" +  
            "withdraw limit = " + limit + "\nA = Withdraw\nB = Deposit\nC = Cancel");  
}
```

## Bbutton

```
else if (state == ATM.TRANSACT) {  
    theATM.deposit(pad.getValue());  
    amount = pad.getValue(); // 거래한 금액을 amount에 저장 (후에 명세표 출력)  
    finalamount = pad.getValue();  
    statement[count] = amount; // 더블타입의 배열 statement에 받아온값 저장  
    sta[count] = "Deposit"; //스트링타입의 배열 sta에 deposit저장  
    if (count == 99)  
        count = 0;  
    else  
        count++;  
    theATM.itemized(); // transact단계에서 명세표단계로 진행  
}
```

## Cbutton

```
else if (state == ATM.TRANSACT) {  
    if (theATM.gettransfer() == 1) //무통장입금상태일 때  
        theATM.reset(); //리셋  
    else  
        theATM.back(); //아닌경우  
    //account단계로 돌아가감  
}
```



# 무통장 입금(ITEMIZED)

```
} else if (state == ATM.ITEMIZED) { // 명세표 출력
    d = new Date(); //날짜를 받아옴
    customer = theATM.getCustomer();
    String transfer; //transfer상태를 저장할 string
    if (theATM.gettransfer() == 1) //무통장입금상태일때
        transfer = "Transfer"; //Transfer저장
    else //일반입출금상태일때
        transfer = "Transact"; //Transact저장
    display.setText("  An Itemized Account\n" + "Transaction\n"
        + transfer + "\nAccountNumber: " + customer.getCustomerNumber() + "\n"
        + finalamount + "\nB = Continue\nC = Exit");
```

First National...

7	8	9
4	5	6
1	2	3
0	.	CE

An Itemized Account

Transaction date: Thu Dec 08 00:54:16 GMT+09:00 2016

kind of transaction: Transfer

AccountNumber: 1

transaction amount: 5000.0

B = Continue

C = Exit

A	B	C
---	---	---

transaction: "

## 2. 명세표

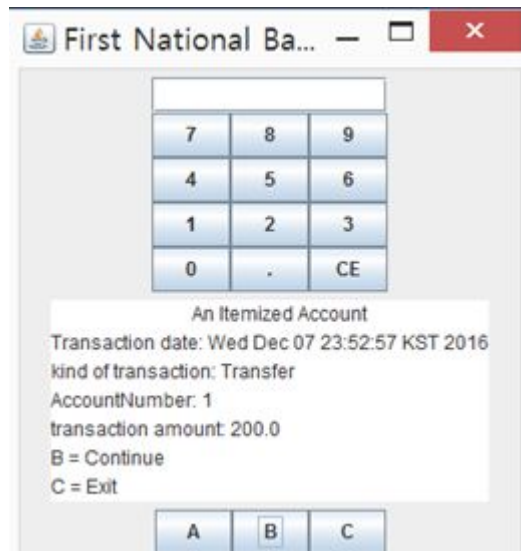
```
public void itemized(){ //명세표단계로 넘어가는 메소드
    assert state == TRANSACT || state == PAPERMONEY; //상태가 transact나 papermoeny일 경우
    state= ITEMIZED;
}
```

Bbutton

```
} else if (state == ATM.ITEMIZED) { //명세표출력단계일 경우
    theATM.back(); //account단계로 돌아감
}
```

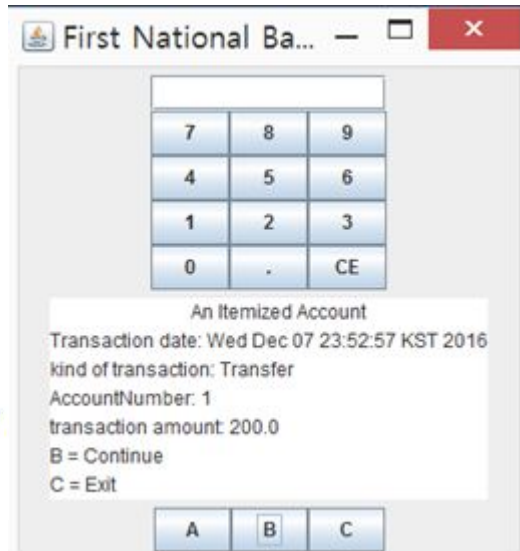
Cbutton

```
else if (state == ATM.ITEMIZED) { // 명세표 단계일경우 리셋
    theATM.reset();
}
```



# 명세표 showstate

```
} else if (state == ATM.ITEMIZED) { // 명세표 출력
    d = new Date(); //날짜를 받아옴
    customer = theATM.getCustomer();
    String transfer; //transfer상태를 저장할 string타입
    if (theATM.gettransfer() == 1) //무통장입금상태일때
        transfer = "Transfer"; //Transfer저장
    else //일반입출금상태일때
        transfer = "Transact"; //Transact저장
    display.setText("  An Itemized Account\n" + "Transaction date: " + d.toString() + "\nkind of transaction: "
        + transfer + "\nAccountNumber: " + customer.getCustomerNumber() + "\ntransaction amount: "
        + finalamount + "\nB = Continue\nC = Exit");
```



### 3. 거래내역

showstate

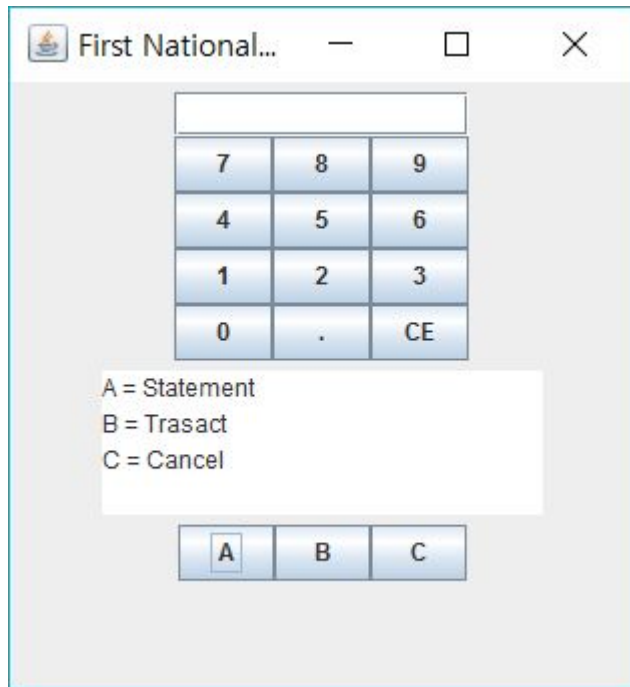
```
} else if (state == ATM.GETSTATE) { //거래내역 일반입출금 선택상태  
    display.setText("A = Statement\nB = Trasact\nC = Cancel");
```

Abutton

```
else if (state == ATM.GETSTATE) //a클릭시 거래내역상태로  
    theATM.statement();
```

Bbutton

```
else if (state == ATM.GETSTATE) //거래내역 선택화면에서 일반입출금 선택시  
    theATM.fraud(); //사기거래화면 출력
```

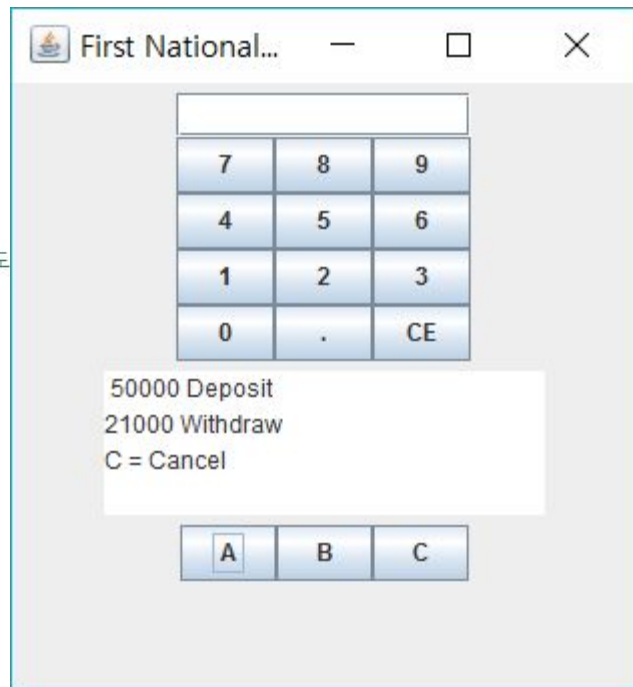




# 거래내역

```
else if (state == ATM.TRANSACTION) { //거래하면
    if (theATM.gettransfer() == 1) { //무통장거래인 경우 (아무것도 실행 안함)

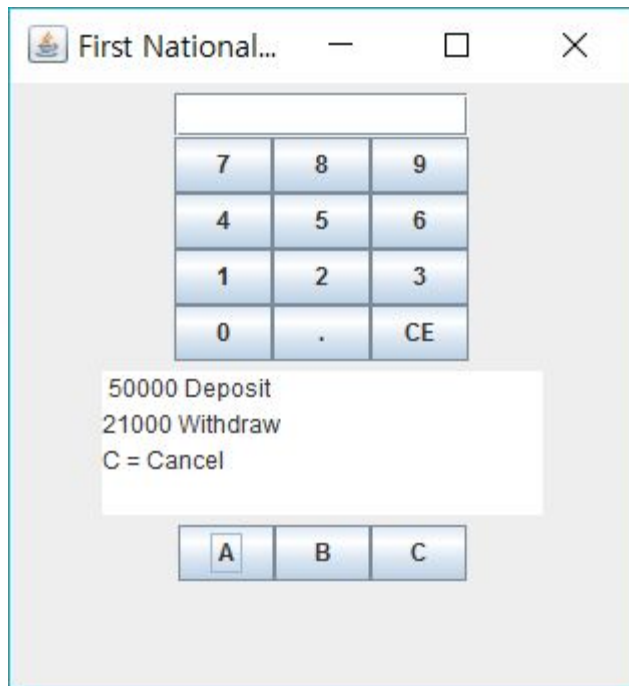
    } else { //무통장이 아닌 경우
        if (pad.getValue() > 1000000) { //입력값이 백만보다 클 경우
            JOptionPane.showMessageDialog(null, "withdraw limit = 1000000"); //출금한도
        }
        else {
            theATM.withdraw(pad.getValue()); //텍스트필드에 받아온 값을 출금함
            amount = pad.getValue(); // 거래한 금액을 amount에 저장 (후에 명세표출력)
            finalamount = pad.getValue();
            statement[count] = amount; //더블타입배열 statement에 받아온 값 저장
            sta[count] = "Withdraw"; //스트링타입배열 sta에 withdraw저장
            if (count == 99) //카운트가 99일때 0으로 초기화
                count = 0;
            else //카운트가 99가 아닐때
                count++; //카운트값을 1증가
            theATM.papermoney(); // transact단계에서 지폐선택단계로 진행
        }
    }
}
```





# 거래내역

```
else if (state == ATM.TRANSFER) //무통장입금이 아닌 일반거래를 선택할 경우
    theATM.start(); //start상태로 변환
else if (state == ATM.GETSTATE) //거래내역 선택화면에서 일반입출금 선택시
    theATM.fraud(); //사기거래화면 출력
else if (state == ATM.TRANSACTION) {
    theATM.deposit(pad.getValue());
    amount = pad.getValue(); // 거래한 금액을 amount에 저장 (후에 명세표 출력)
    finalamount = pad.getValue();
    statement[count] = amount; //더블타입의 배열 statement에 받아온값 저장
    sta[count] = "Deposit"; //스트링타입의 배열 sta에 deposit저장
    if (count == 99)
        count = 0;
    else
        count++;
    theATM.itemized(); // transact단계에서 명세표단계로 진행
```



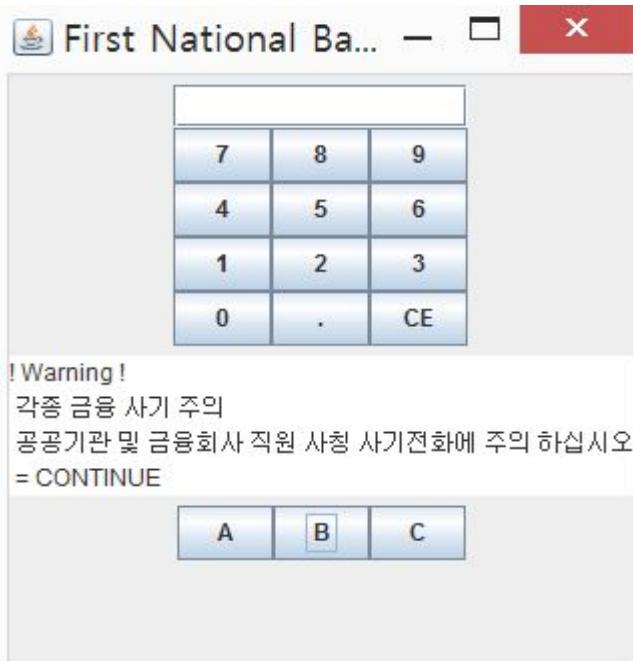
# 거래내역

```
} else if (state == ATM.STATEMENT) { //거래내역상태일때
    String s = " "; //스트링 타입 s초기값 null
    for (int i = 0; i < count; i++) { //count (입출력시 1씩 증가하는 int타입)
        String[] statement = new String[this.statement.length]; //string타입의 statement배열
        statement[i] = Integer.toString((int) this.statement[i]); //statement필드 (더블타입) 을 스트링타입으로 변환
        s = s + statement[i] + " " + sta[i] + "\n"; //s에 기존 s와 스트링으로 변환한 숫자값과 deposit/withdraw(sta[i])를 저장
    }
    display.setText(s + "C = Cancel\n"); //s 출력
}
```

## 4. 금융사기 주의

```
else if (state == ATM.FRAUD)
```

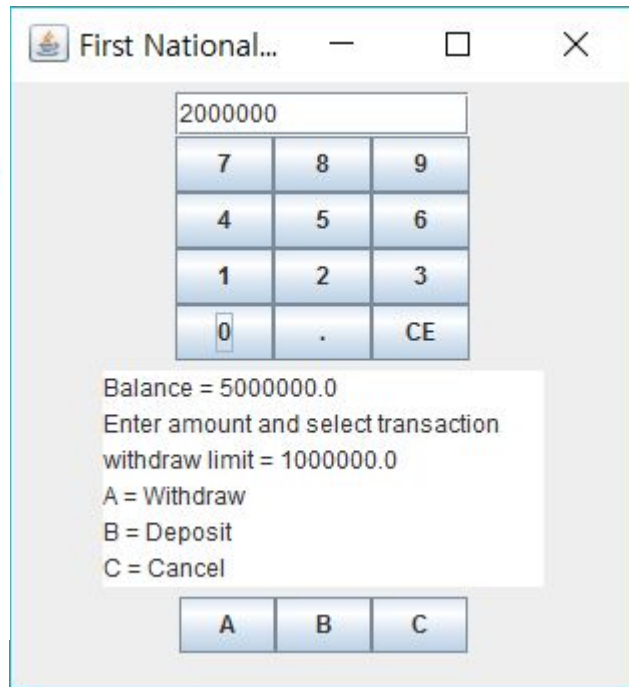
```
display.setText("! Warning !\nA 각종 금융사기 주의\n 공공기관 및 금융회사 직원사칭 사기전화에 주의 하십시오. = CONTINUE"); // 금융사기 문구 출력
```



## 5. 출금 한도

```
else if (state == ATM.TRANSACTION) { //거래하면
    if (theATM.gettransfer() == 1) { //무통장거래인 경우 (아무것도 실행 안함)

    } else { //무통장이 아닌 경우
        if (pad.getValue() > 1000000) { //입력값이 백만보다 클 경우
            JOptionPane.showMessageDialog(null, "withdraw limit = 1000000"); //출금한도 = 1000000 출력
        }
        else {
            theATM.withdraw(pad.getValue()); //텍스트필드에 받은 값을 출금할
            amount = pad.getValue(); // 거래한 금액을 amount에 저장 (후에 명세표출력)
            finalamount = pad.getValue();
            statement[count] = amount; //더블타입배열 statement에 받은 값 저장
            sta[count] = "Withdraw"; //스트링타입배열 sta에 withdraw 저장
            if (count == 99) //카운트가 99일때 0으로 초기화
                count = 0;
            else //카운트가 99가 아닐때
                count++; //카운트값을 1증가
            theATM.papermoney(); // transact 단계에서 지폐선택단계로 진행
        }
    }
}
```



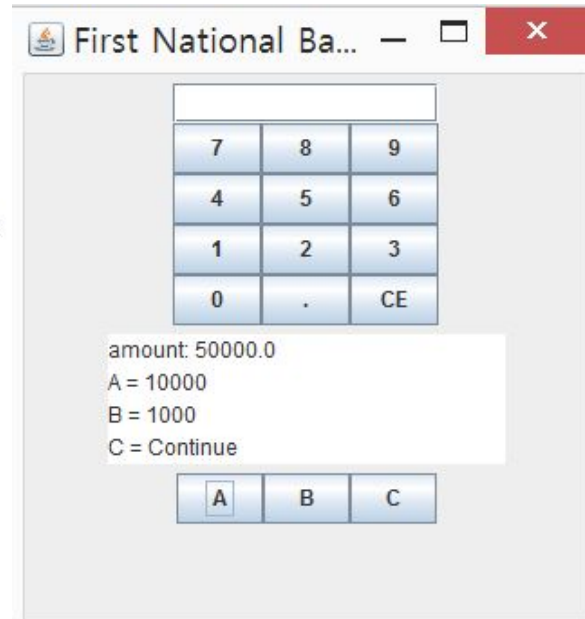
## 6. 출금시 지폐권 선택

### ATM 클래스

```
} else if (state == ATM.PAPERMONEY) { //출금시 지폐권선택단계  
    display.setText("amount: " + amount + "\nA = 10000\nB = 1000\nC = Continue");
```

### Abutton

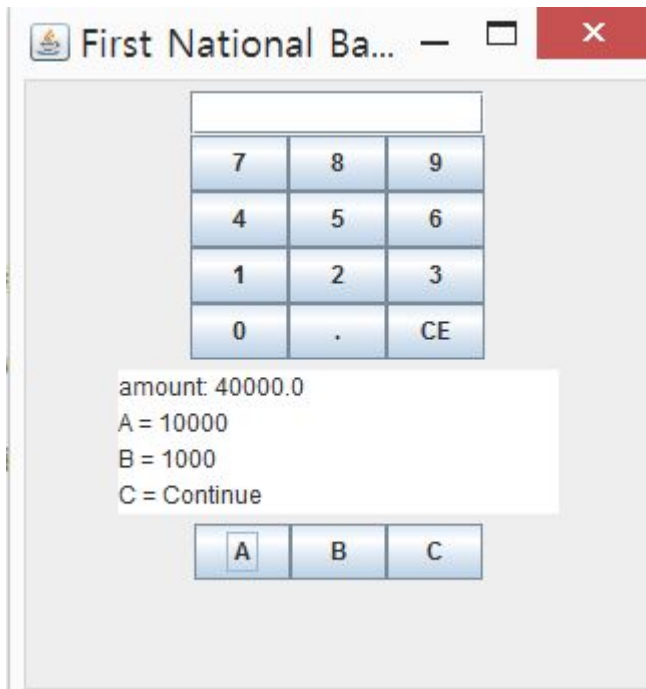
```
    } else if (state == ATM.PAPERMONEY) { //지폐권 선택 단계일때  
        if (amount >= 10000) { //값이 10000보다 클경우  
            amount -= 10000; //값에서 10000을 뺌  
        } else if (amount == 0) { //값이 0일때  
            theATM.itemized(); //명세표 출력  
        }  
    }
```



# 출금시 지폐권 선택

Bbutton

```
} else if (state == ATM.PAPERMONEY) { //지폐권선택단계에서 천원권
    if (amount >= 1000) { //출금할 값이 1000보다 클 경우
        amount -= 1000; //값에서 1000을 빼줌
    } else if (amount == 0) { //값이 0일경우
        theATM.itemized(); //명세표 출력
    }
}
} else if (state == ATM.ITEMIZED) { //명세표출력단계일 경우
    theATM.back(); //account단계로 돌아감
}
```



A를 누른 실행 결과 화면

## 7. Checking, Saving 클래스 연동

```
public void selectionAccount(int account){//계좌선택하는 메소드
    assert state == ACCOUNT || state == TRANSACT;//검증
    if(account == CHECKING)//받아온 계좌가 checking일경우
        currentAccount = currentCustomer.getCheckingAccount();//bankaccount타입의
currentaccount에 checkingaccount를 저장.
    else//받아온 계좌가 saving일경우
        currentAccount = currentCustomer.getSavingsAccount();//bankaccount타입의
currentaccount에 savingsaccount를 저장.
    state = TRANSACT;//다음단계
```

Thank you