JackPot ATM
OOPT Stage 2050 & 2060

2018.05.23

Team 5

201112052 방민석

201312259 백만일

201211383 조영래

목차

1. Implement Class & Method Definitions

Type	Class
Name	System_Control
Purpose	사용자(User 혹은 Manager)로부터 데이터를 입력받아 저장하고 실제 데이터의 처리가 동작하는 Bank, Account 등 다양한 클래스와 데이터를 주고받는 클래스
Overview	N/A
Cross Reference	jackPot, withdraw, deposit,remittance, view_account_detail, manage_ATM
Exceptional Course of Event	N/A

Type	Method
Name	input_menu
Purpose	사용자로부터 진입하고자하는 ATM 의 기능을 입력받아 inputMenu 어트리뷰트에 저장하는 메소드
Overview	N/A
Cross Reference	withdraw, deposit,remittance, view_account_detail, manage_ATM
Input	void
Output	void
Abstract Operation	사용자가 입력한 기능을 inputMenu 어트리뷰트에 저장한다
Exceptional Course of Event	N/A

1. Implement Class & Method Definitions

Type	Method	Type	Method
Name	input_ID	Name	input_amount
Purpose	사용자로부터 ID를 입력받아 inputID 어트리뷰트에 저장하는 메소드	Purpose	사용자로부터 금액을 입력받아 inputAmount 어트리뷰트에 저장하는 메소드
Overview	N/A	Overview	N/A
Cross Reference	withdraw, deposit,remittance, view_account_detail	Cross Reference	withdraw, deposit, remittance
Input	void	Input	void
Output	void	Output	void
Abstract Operation		Abstract Operation	사용자로부터 입력받은 값을 inputAmount 에 저장한다.
Exceptional Course of Event	N/A	Exceptional Course of Event	N/A

1. Implement Class & Method Definitions

Type	Method
Name	input_PW
Purpose	사용자로부터 비밀번호를 입력받아 inputPW 어트리뷰트에 저장하는 메소드
Overview	N/A
Cross Reference	withdraw, deposit,remittance, view_account_detail
Input	void
Output	void
Abstract Operation	사용자로부터 입력받은 값을 inputPW에 저장한다.
Exceptional Course of Event	N/A

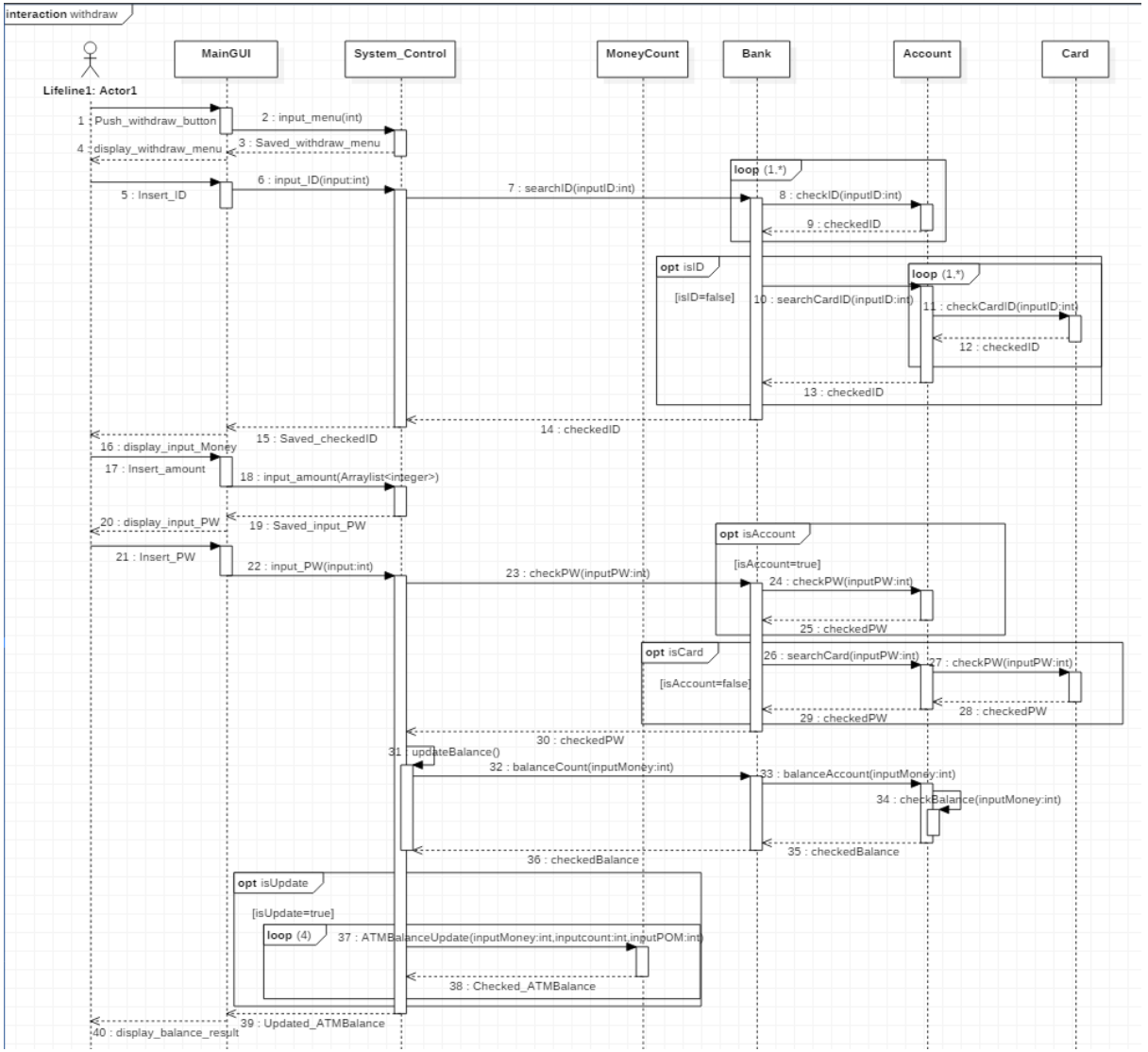
Type	Method
Name	updateBalance
Purpose	사용자의 거래에 따른 계좌의 잔고를 변화시켜주는 메소드
Overview	N/A
Cross Reference	withdraw, deposit,remittance
Input	void
Output	void
Abstract Operation	사용자의 거래에 따라 Account.Balance를 변화시켜준다.
Exceptional Course of Event	N/A

1. Implement Class & Method Definitions

Type	Method	Type	Method
Name	checkBalance	Name	Card.checkPW
Purpose	사용자로부터 입력받은 금액이 잔고보다 적은지 체크하는 메소드	Purpose	사용자로부터 입력받은 password가 Card의 password와 일치하는지 확인하는 method
Overview	N/A	Overview	N/A
Cross Reference	withdraw,remittance	Cross Reference	withdraw,remittance, view_account_detail
Input	inputMoney: int	Input	inputPW: int
Output	int	Output	int
Abstract Operation	inputMoney가 계좌의 잔액보다 적은지 비교한다	Abstract Operation	inputPW와 Card에 저장된 password가 같은지 비교한다
Exceptional Course of Event	N/A	Exceptional Course of Event	N/A

2. Implement Windows

1) Withdraw



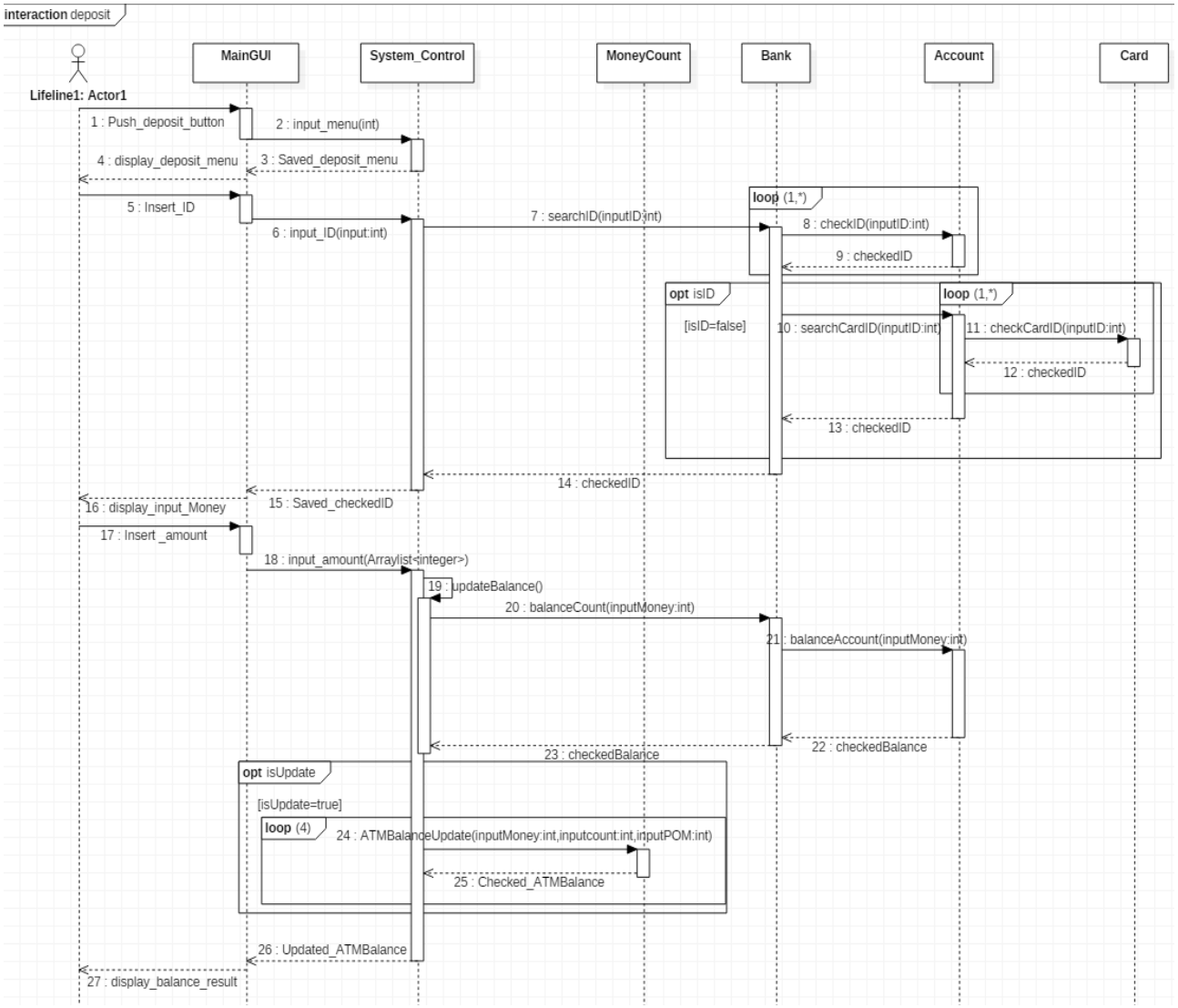
2. Implement Windows

Name	Push_withdraw_button	Name	Insert_amount
Responsibilities	MainGUI의 withdraw 버튼을 누른다	Responsibilities	MainGUI에 금액을 입력한다
Type	MainGUI	Type	MainGUI
Cross Reference	R1.2	Cross Reference	R1.2
Notes	MainGUI의 withdraw 버튼을 누른다	Notes	MainGUI에 금액을 입력한다
Pre-Conditions	N/A	Pre-Conditions	ID가 확인되어 있어야 한다
Post-Conditions	출금 창을 화면에 출력한다	Post-Conditions	비밀번호 입력 창을 화면에 출력한다

Name	Insert_ID	Name	Insert_PW
Responsibilities	MainGUI에 ID를 입력한다	Responsibilities	MainGUI에 비밀번호를 입력한다
Type	MainGUI	Type	MainGUI
Cross Reference	R1.2	Cross Reference	R1.2
Notes	MainGUI에 ID를 입력한다	Notes	MainGUI에 비밀번호를 입력한다
Pre-Conditions	input_menu에 withdraw값이 들어있어야 한다	Pre-Conditions	금액이 입력되어 있어야 한다
		Post-Conditions	거래 내역을 출력한다
Post-Conditions	금액 입력 창을 화면에 출력한다		

2. Implement Windows

2) Deposit



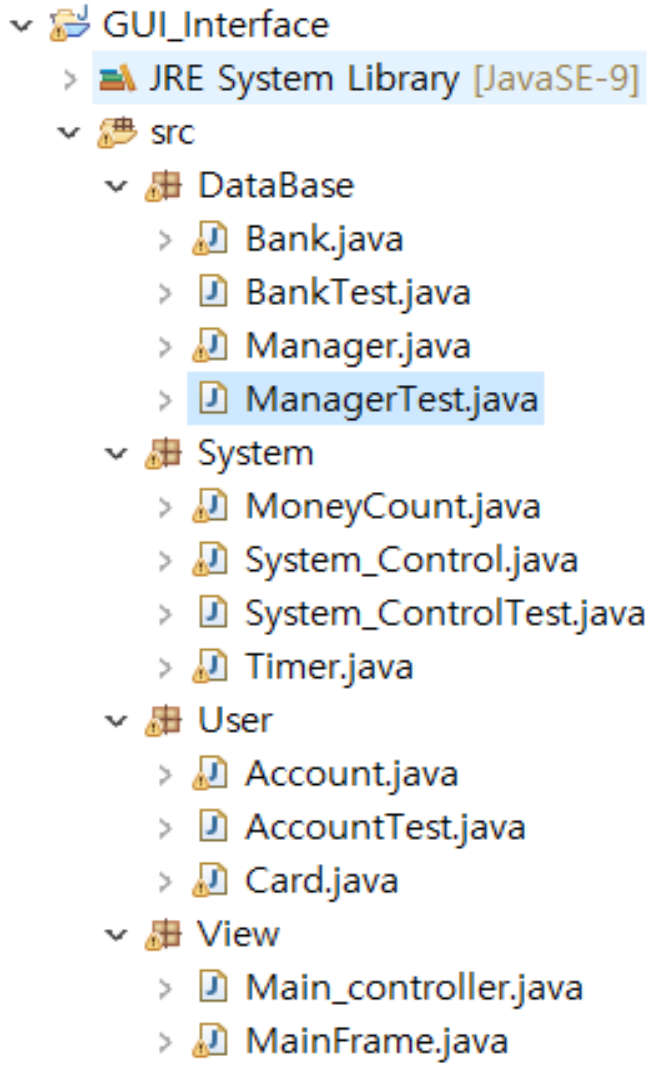
2. Implement Windows

Name	Push_deposit_button
Responsibilities	MainGUI의 deposit 버튼을 누른다
Type	MainGUI
Cross Reference	R.2
Notes	MainGUI의 deposit 버튼을 누른다
Pre-Conditions	N/A
Post-Conditions	입금 창을 화면에 출력한다

Name	Insert_ID
Responsibilities	MainGUI에 ID를 입력한다
Type	MainGUI
Cross Reference	R.2
Notes	MainGUI에 ID를 입력한다
Pre-Conditions	input_menu에 deposit값이 들어있어야 한다
Post-Conditions	금액 입력 창을 화면에 출력한다

Name	Insert_amount
Responsibilities	MainGUI에 금액을 입력한다
Type	MainGUI
Cross Reference	R.2
Notes	MainGUI에 금액을 입력한다
Pre-Conditions	ID가 확인되어 있어야 한다
Post-Conditions	비밀번호 입력 창을 화면에 출력한다

3. Write Unit Test Code




```java
package DataBase;

import static org.junit.Assert.*;

public class BankTest {

    @Test //아이디 제대로 입력
       public void searchIDtest1() {
          //fail("Not yet implemented");
          Bank test = new Bank(1);
          assertEquals(1,test.searchID(325,1));
       }

       @Test //아이디 다르게 입력
       public void searchIDtest2() {
          Bank test = new Bank(1);
          assertEquals(0,test.searchID(324, 1));
       }

}
```

3. Write Unit Test Code

```java
package DataBase;

import static org.junit.Assert.*;

public class ManagerTest {

    @Test
        public void input_MID_TrueTest() {
            Manager test = new Manager(123,235);
            boolean output = test.checkID(123);
            assertEquals(true,output);
        }
        @Test
        public void input_MID_FalseTest() {
            Manager test = new Manager(123,235);
            boolean output = test.checkID(11111);
            assertEquals(false,output);
        }
        @Test
        public void input_MPW_TrueTest() {
            Manager test = new Manager(123,235);
            boolean output = test.checkPW(235);
            assertEquals(true,output);
        }
        @Test
        public void input_MPW_FalseTest() {
            Manager test = new Manager(123,235);
            boolean output = test.checkPW(111111);
            assertEquals(false,output);
        }
}
```

3. Write Unit Test Code

```
1  package System;
2
3  import static org.junit.Assert.*;
6
7  public class System_ControlTest {
8
9      @Test
10     public void input_ID_TrueTest() {//시스템에 등록되어 있는 계좌를 확인
11         System_Control test = new System_Control();
12         boolean output = test.input_ID(325);
13         assertEquals(true,output);
14     }
15     @Test
16     public void input_ID_FalseTest() {
17         System_Control test = new System_Control();
18         boolean output = test.input_ID(111111);
19         assertEquals(false,output);
20
```

3. Write Unit Test Code

```
22      @Test
23      public void input_RID_TrueTest() {
24          System_Control test = new System_Control();
25          boolean output = test.input_RID(325);
26          assertEquals(true,output);
27
28      }
29      @Test
30      public void input_RID_FalseTest() {
31          System_Control test = new System_Control();
32          boolean output = test.input_RID(11111);
33          assertEquals(false,output);
34
35      }
36      @Test
37      public void input_MID_TrueTest() {
38          System_Control test = new System_Control();
39          boolean output = test.input_MID(123);
40          assertEquals(true,output);
41      }
```

```
    }
    @Test
    public void input_MID_FalseTest() {
        System_Control test = new System_Control();
        boolean output = test.input_MID(11111);
        assertEquals(false,output);
    }

    @Test
    public void input_MPW_TrueTest() {
        System_Control test = new System_Control();
        boolean output = test.input_MPW(235);
        assertEquals(true,output);
    }
    @Test
    public void input_MPW_FalseTest() {
        System_Control test = new System_Control();
        boolean output = test.input_MPW(111111);
        assertEquals(false,output);
    }


}
```

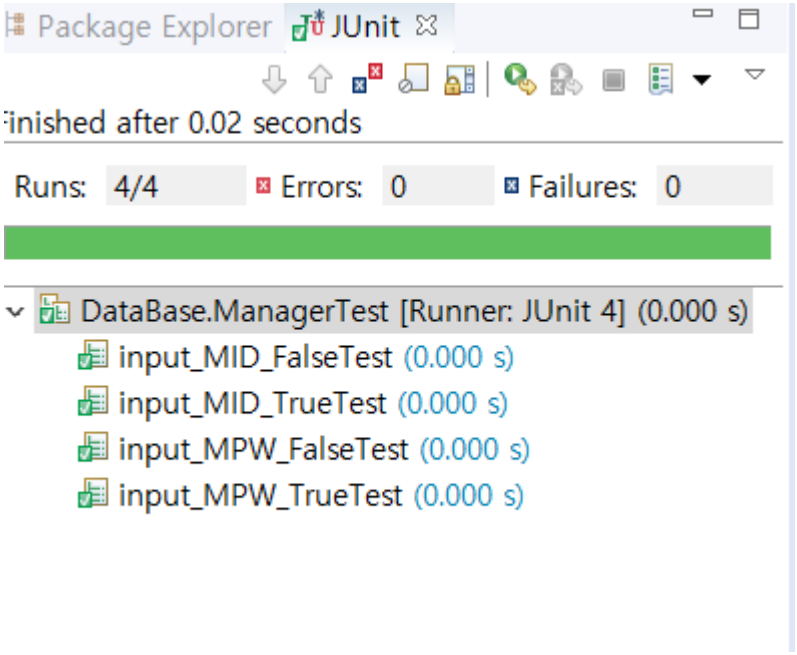
3. Write Unit Test Code

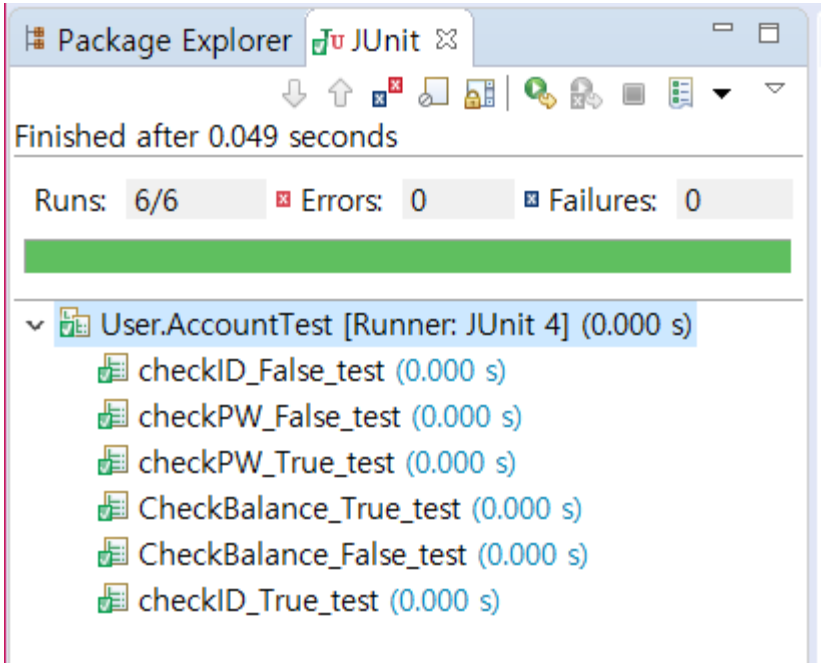
```java
package User;

import static org.junit.Assert.*;

public class AccountTest {

    @Test
    public void checkID_True_test() {
        Account test = new Account(123,456,"신한",10000,1);
        int output = test.checkID(123);
        assertEquals(1,output);
    }

    @Test
    public void checkID_False_test() {
        Account test = new Account(123,456,"신한",10000,1);
        int output = test.checkID(11111);
        assertEquals(0,output);
    }

    @Test
    public void checkPW_True_test() {
        Account test = new Account(123,456,"신한",10000,1);
        int output = test.checkPW(456);
        assertEquals(1,output);
    }
```

3. Write Unit Test Code

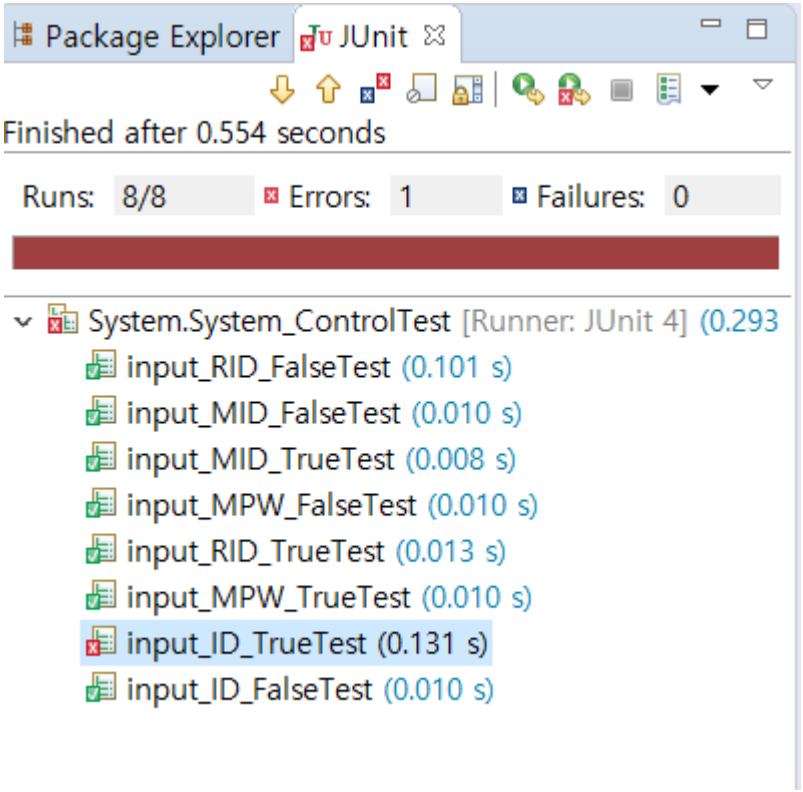
```java
@Test
public void checkPW_False_test() {
    Account test = new Account(123,456,"신한",10000,1);
    int output = test.checkPW(11111);
    assertEquals(0,output);
}

@Test
public void CheckBalance_True_test() {
    Account test = new Account(123,456,"신한",10000,1);
    int output = test.checkBalance(5000);
    assertEquals(5000,output);
}
@Test
public void CheckBalance_False_test() {
    Account test = new Account(123,456,"신한",10000,1);
    int output = test.checkBalance(50000);
    assertEquals(-1,output);
}
```

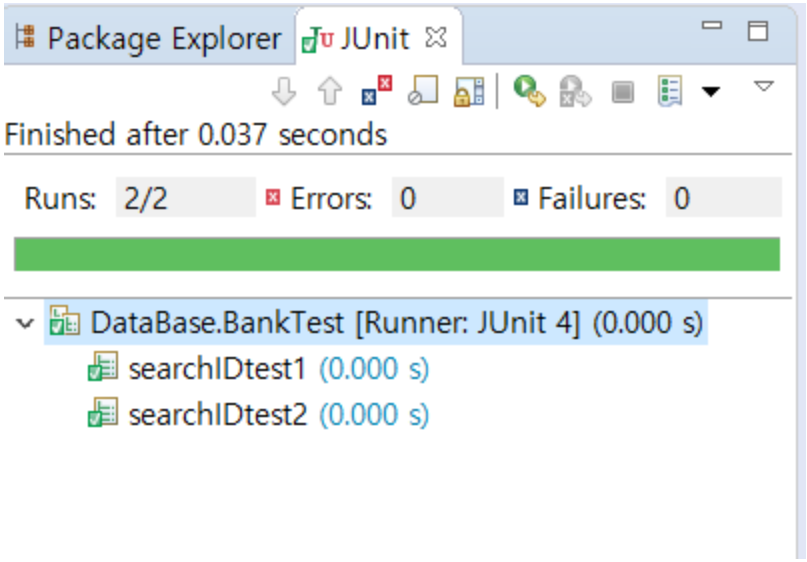
4. Unit Testing





4. Unit Testing





5. System Testing

Test Number	Test 항목	Description	Use Case	Pass / Fail
1-1	input_menu Test	사용자가 입력한 해당 메뉴 화면으로 올바르게 진입하는지 확인한다.	R1.2 ,R.2, R.3, R.4, R.5	Pass
2-1	input_ID	사용자가 입력한 계좌 혹은 카드번호가 유효하지 않으면 오류메시지를 출력하는지 확인한다.	R1.2, R.2,R.3,R.4	Pass
2-2	input_ID	사용자가 입력한 계좌 혹은 카드번호가 유효하면 다음 단계로 넘어가는지 확인한다.	R1.2, R.2,R.3,R.4	Pass
3-1	input_PW	사용자가 입력한 비밀번호가 계좌번호의 비밀번호가 아니면 오류메시지를 출력하는지 확인한다	R1.2, R.3, R.4	Pass
3-2	input_PW	사용자가 입력한 비밀번호가 계좌번호의 비밀번호가 맞으면 다음단계로 넘어가는지 확인한다.	R1.2, R.3, R.4	Pass
4-1	input_amount	출금의 경우 사용자가 입력한 금액이 계좌 잔고보다 많으면 오류메시지를 출력하는지 확인한다.	R1.2	Pass
4-2	input_amount	출금의 경우 사용자가 입력한 금액이 ATM 잔고보다 많으면 오류메시지를 출력하는지 확인한다.	R1.2	Pass
4-3	input_amount	송금의 경우 사용자가 입력한 금액이 계좌 잔고보다 많으면 오류메시지를 출력하는지 확인한다.	R.3	Pass
4-4	input_amount	사용자의 입력금액이 계좌 잔고보다 작고 ATM 잔고보다 작을때 다음 단계로 넘어가는지 확인한다.	R1.2, R.2, R.3	Pass

Q & A