기말프로젝트

1910896 천옥희(통계학과)

1.코드에 대한 설명

1.1 사용자 정보가 담긴 엑셀 파일로부터 사용자 리스트를 받아오기

```
import classBankAccount    #classBankAccount모듈 부르기
import openpyxl            #openpyxl모듈 부르기
import time                #time 모듈 부르기

filename = 'UserList.xlsx'                                  #엑셀 파일에 filename이라는 변수명 지정
# 사용자 리스트 받고, 등록자 판별
wb_obj = openpyxl.load_workbook(filename)                   #엑셀 모듈 여는 함수에 xb_obj라는 변수명 지정
sheet_obj = wb_obj.active                                   #시트 객체 지정
a = {}                                                      #a 딕셔너리로 지정
for i in range(2, sheet_obj.max_row + 1):                   #행에 대한 반복문
    result = []                                             #result는 리스트임을 지정
    for j in range(1, sheet_obj.max_column + 1):            #열에 대한 반복문
        currentCell_obj = sheet_obj.cell(row=i, column=j)   #각 셀의 데이터를 currentCell_obj으로 지정
        result.append(currentCell_obj.value)               #각 셀들의 값을 리스트에 추가
    a[result[0]] = (result[1], result[2])                   #각 행에 대한 {name:(pw,balane)} 값 딕셔너리에 추가
```

-import함수를 이용해 classBankAccount와 openpyxl, time모듈을 불러왔다.
openpyxl를 불러온 이유는 사용자 정보가 담긴 엑셀파일을 불러오기 위해서다. 엑셀파일은 filename이라는 변수명으로 지정해주었다.
-wb_obj라는 변수명으로 openpyxl의 모듈의 load_workbook이라는 함수를 써 엑셀 파일을 열어 주었다.
sheet_obj라는 변수명에는 시트 각각의 객체를 지정해주었으며, a라는 변수 명에는 후에 딕셔너리를 추가해주기 위해서 빈 딕셔너리를 지정해주었다.
-각각의 셀 값을 읽기 위해서 for문을 활용했다. for에 i를 넣고, range를 이용하여 2번째 행부터 현재시트의 행수+1까지 지정해 행(2~현재시트의 수)범위까지 읽어지도록 했다. 행에 대한 for문 밑에 열에 대한 for 문을 넣어주었다. 행에 대한 각각의 열의 값을 받기위해서다. 열 for문에는 j를 넣어주고, range를 이용해 1번째 행부터 현재시트의 열수+1까지 지정해 열(1~현재시트의 수)범위를 읽어질 수 있도록 했다.
-읽어진 각각의 셀의 값을 각각의 행에 대해 리스트로 만들기 위해서 행 for문 밑에 열 for문 들어가기 전 result=[]로 빈 리스트를 지정해주고, 열for문 밑에 append함수를 써 각 행별로 리스트가 만들어 질 수 있도록 하였다.
-이것을 name:(pw,balance)형태의 딕셔너리로 바꿔주고자 하였다. 여기서 name은 각각의 행에 대한 리스트의 첫 번째 수이고, pw는 리스트에 2번째, balance값은 리스트의 3번째 값이다. 따라서 a[result[0]]=(result[1], result[2])로 써줘 name이라는 key값에 대한 value값인 pw와 balance가 행이 바뀌면 추가될 수 있도록 행 for문 밑에가 써주었다.

1.2 사용자가 리스트에 등록되어있는지 판별, 등록된 사용자라면, 비밀번호 입력받기

```
class Atm(classBankAccount.BankAccount):                    #은행계좌클래스를 상속받은 Atm클래스함수
    def __init__(self):                                     #init함수 오버라이딩
        name = input('Enter the username: ')                #사용자로부터 사용자 이름을 받음
        self.name=name
        if self.name in a.keys():                           #사용자이름이 리스트에 등록되어 있을 때의 조건문
            print('%s 님 환영합니다' %self.name)
            while True:                                                          #반복문
                user_pass = int(input('%s님의 패스워드를 입력하세요 ' % self.name))  #사용자에게 비밀번호를 입력을 받음
                if user_pass == a[self.name][0]:                                 #사용자의 비밀번호가 일치했을 때 조건문
                    print('사용자 정보가 확인되었습니다')
                    self.balance = a[self.name][1]                               #balance변수값 지정
                    break                                                        #반복수행 정지
                else:                                                            #비밀번호 일치하지 않았을 경우
                    print('비밀번호가 틀렸습니다.')
```

-클래스 Atm에 은행계좌클래스를 상속받아오기 위해서 1.1번에서 import classBankAccount의 모듈을 써주었다. 이 클래스에서 은행계좌클래스를 상속방기위해 classBankAccount.BankAccount를 Atm()의 ()안에 넣어주었다.

```
a=BankAccount('Ok',100)
print(a)
```

```
Ok 님 환영합니다.
초기 금액  100 으로 계좌가 만들어졌습니다.
```

사진1.2(1)　　사진1.2(2)

- 사진1.2(1)처럼 BankAccount에 초깃값을 넣어주고, 출력하게 되면, 사진1.2(2)와 같은 결과를 얻게 된다. 이것은 Atm에서는 필요 없는 출력 값이기 때문에 이를 없애주고자 했다. 이를 위해 __init__함수가 다시 구현되도록 클래스 Atm에서 __init__함수를 재입력하였다. __init__(self)함수에서 사용자가 리스트에 등록이 된 사용자 인지 확인하고자 했다.
- 사용자로부터 사용자의 이름을 받고, 이를 name이라고 변수명을 지정해주었다. name을 self.name=name으로 다시 지정해 self.name이 name의 값으로 정의 될 수 있도록 하였다. self.name이 리스트 안에 있는 사용자인지 확인하기 위해 if문을 이용해 if self.name in a.keys()를 써줬다.

- a.keys()는 a딕셔너리의 key값들을 리스트로 만들어주는 함수이기 때문에 self.name이 a딕셔너리의 key값들의 리스트 안에 있다면 이라는 조건문이다. 이 조건문이 성립이 된다면, 출력문을 통해 사용자의 환영메세지가 뜨고, 반복문을 실행하도록 하였다.
- 반복문(while) 안에서는 사용자의 패스워드를 입력하도록 int(input('%s님의 패스워드를 입력하세요.'%self.name))을 써주고, int를 붙여줘 문자열이 아닌 정수로 받을 수 있도록 하였으며, 이를 user_pass라는 변수명으로 지정해주었다. 입력한 패스워드가 사용자의 패스워드와 같아야한다.
- 사용자의 패스워드는 a딕셔너리의 value에서 튜플에서 첫 번째의 값이므로, a[self.name][0]로 나타낼 수 있다. 따라서 조건문에서 if user_pass==a[self.name][0]으로 써주었고, 이게 성립이 된다면, '사용자 정보가 확인 되었습니다'라는 출력문과 패스워드를 입력하는 반복문을 끝내게 되도록 하였다. 만약, 패스워드와 사용자의 패스워드가 다르다면, 다시 비밀번호를 입력할 수 있게 반복수행을 하도록 패스워드 입력을 받는 user_pass의 위에 while 반복문을 써줬다. (추가한 내용)

1.3 등록되지 않은 사용자 등록하기

```
#등록되지 않았을 경우, 사용자 정보 받아 새롭게 등록
    else:
        tell = input('%s 님은 등록되지 않았습니다. 추가하시겠습니까?(yes or no)' % self.name)  # 사용자에게 등록여부받기
        if tell == 'yes':                                                                      #등록하겠다고 답했을 때 조건문
            user_pass=int(input('%s 님의 비밀번호를 입력하세요: ' % self.name))                  #비밀번호 설정
            self.balance = int(input('%s 님의 초기 잔액을 입력하세요: ' % self.name))           #초기잔액 설정
            a[self.name] = (user_pass, self.balance)                                           #딕셔너리 a에 {name:(pw,balane)} 값 추가
            print('등록이 되었습니다.')
        else:                                                                                  # 등록하지 않았을 경우
            print('서비스를 종료합니다.')
```

- 1.2에서 사용한 사용자이름이 등록되었는지에 대한 if문과 반대로 등록이 되지 않았을 경우에 대한 조건문이기 때문에 else문을 써준다. 사용자이름이 등록되어있지 않았을 경우, 등록여부에 대해 사용자로부터 받아와야함으로, input문을 써 등록여부를 사용자로부터 받아온다. 이를 tell이라는 변수명으로 지정해준다.

- 사용자가 'yes'를 입력했을 경우, 사용자의 정보를 새롭게 받아 등록한다. 따라서 if tell=='yes':라는 조건문을 써준다.
- 등록할 경우 비밀번호, 초기잔액을 사용자로부터 받아야한다. int와 input문장을 이용해 비밀번호와 초기잔액을 사용자로부터 받고, 비밀번호를 user_pass, 초기잔액을 self.balance로 변수명을 지정해준다. 초기잔액을 self.balance로 지정한 이유는 클래스함수에서 함수에 대해 balance의 값이 필요하기 때문에 self.balance로 지정해주어야 한다.
- 사용자에 대한 정보를 얻었으면, a딕셔너리에 추가해주고, '등록되었습니다'라는 출력 값이 나오도록 한다.

- 'yes'라고 입력하지 않았을 때는 등록하지 않겠다는 뜻으로 if tell=='yes':문과 반대이므로, else라는 조건문을 사용해 '서비스를 종료합니다'라는 출력값만 나오도록 하였다.

1.4 메뉴 출력

```
# 입출금 메뉴 반복 수행, 명세표 출력
    if self.name in a.keys():
        while True:
            print('=' * 30)  # 메뉴 출력
            print('원하시는 메뉴를 선택하세요')
            print('1. Deposit \n'
                  '2. Withdraw \n'
                  '3. Check Balance \n'
                  '4. Quit')
            answer = int(input('>>>'))
```

사진1.4(1)

```
==============================
원하시는 메뉴를 선택하세요
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
```

사진1.4(2)

-입출금 메뉴는 사용자가 확인 되었을 때, 사용할 수 있다. 사용자 확인은 1.2와 1.3에서 다 했기 때문에, 사용자 이름이 a딕셔너리의 key값에 들어있으면 입출금메뉴를 실행하도록 하면 된다. 따라서 if 조건문을 이용해 if self.name in a.keys():를 써주었다.
-입출금 메뉴가 사용자가 끝내겠다는 메뉴를 누를 때까지 반복수행을 하도록 만들었다. while 반복문을 이용해 메뉴판이 사진1.4(2)와 같이 나올 수 있도록 print함수 써주었다. 또한, 메뉴판을 사용자가 숫자로 고를 수 있도록, int와 input문장을 사용하였고, answer이라는 변수명을 붙어주었다.

1.5 메뉴선택과 명세표출력

(1) 1번을 선택했을 경우

```
if answer == 1:                                                    #1번인 입금을 선택했을 경우
    amount = int(input('입금하실 금액을 입력하세요 : '))              #입금할 금액 사용자로부터 받기
    self.deposit(amount)                                           #deposit 함수 불러오기
    paper = input('명세표를 출력하시겠습니까?(y or n)')               #명세표 출력여부 사용자로부터 받기
    if paper == 'y':                                               #명세표 출력한다고 했을 때 조건문
        print('*' * 35)
        print('             명세표             ')
        print('거래 시간:', end=' ')                                 #거래시간 출력
        print(time.strftime('%a %b %d %X %Y', time.localtime(time.time())))
        print('이름: %s' % self.name)                                #거래자명 출력
        print('입금액: %d' % amount)                                 #입금액 출력
        print('남은 잔액: %d' % self.balance)                        #남은 잔액 출력
        print('거래해주셔서 감사합니다. -Sookmyung Bank')
        print('*' * 35)
```

- 1.4 answer에서 사용자가 1번을 입력한 것은 입금을 하겠다는 것이다. 따라서, if answer==1:이라는 조건문 밑에 amount라는 변수명을 지정해주고, 사용자로부터 입금할 금액을 입력 받을 수 있도록 int와 input함수를 써주었다. 입금에 대한 함수는 classBankAccount에서 deposit(amount)로 만들어 놓았다. 우리는 1.2에서 Atm클래스를 만들 때 classBankAccount를 상속받아온 것이기 때문에, self. deposit(amount)를 이용해 입금 함수를 사용할 수 있다. 따라서 self.deposit(amount)로 입금에 대한 함수는 불러온다.
- 입금을 다했으면, 명세표출력에 대해 사용자로부터 받아 명세표를 출력할 것인지 말 것인지 결정을 한다. 사용자로부터 명세표를 출력할 것인가를 input함수로 물어보고, 이를 paper이라는 변수명을 써주었다. 사용자가 'y'라고 답했을 경우, 명세표를 출력하기 위해 if paper=='y':라는 조건문을 써준다.
- 조건문 밑에 출력할 명세표를 print함수를 통해 입력해준다. 여기서 거래시간은 time함수를 사용해야하기 때문에 1.1에서 time모듈을 불러 온 것이다. 거래시간은 요일줄임말, 달 줄임말, 날, 현재 설정된 로케일에 기반 한 시간, 년도출력 형태로 출력하기 위해서 time.strftime함수와 포맷코드를 이용해 출력했다. 이름은 사용자가 입력한 이름, 입금액, 남은잔액을 출력하기 위해 포맷코드를 이용해 self.name, amout, self.balance를 넣어주었다.

(2) 2번을 선택했을 경우

```
elif answer == 2:                                                  #2번인 출금을 선택했을 경우
    amount = int(input('출금하실 금액을 입력하세요 : '))              #출금할 금액 사용자로부터 받기
    self.withdraw(amount)                                          #withdraw 함수 불러오기
    paper = input('명세표를 출력하시겠습니까?(y or n)')               #명세표 출력여부 사용자로부터 받기
    if paper == 'y':                                               #명세표 출력한다고 했을 때 조건문
        print('*' * 35)
        print('             명세표             ')
        print('거래 시간:', end=' ')                                 #거래시간 출력
        print(time.strftime('%a %b %d %X %Y', time.localtime(time.time())))
        print('이름: %s' % self.name)                                #거래자명 출력
        print('출금액: %d' % amount)                                 #출금액 출력
        print('남은 잔액: %d' % self.balance)                        #남은 잔액 출력
        print('거래해주셔서 감사합니다. -Sookmyung Bank')
        print('*' * 35)
```

- 1.4 answer에서 사용자가 2번을 입력한 것은 출금을 하겠다는 것이다. 이는 1번과는 다른 조건문이기 때문에 elif answer==2:이라는 조건문 밑에 amount라는 변수명을 지정해주고, 사용자로부터 출금할 금액을 입력 받을 수 있도록 int와 input함수를 써주었다. 출금에 대한 함수는 classBankAccount에서 withdraw(amount)로 만들어 놓았다. 우리는 1.2에서 Atm클래스를 만들 때 classBankAccount를 상속받아온 것이기 때문에, self.withdraw (amount)를 이용해 출금 함수를 사용할 수 있다.
따라서 self.withdraw(amount)로 출금에 대한 함수는 불러온다.
- 출금을 다했으면, 명세표출력에 대해 사용자로부터 받아 명세표를 출력할 것인지 말 것인지 결정을 한다. 이는 위에 1.5(1)과 같이 조건문을 실행해주고, 명세표의 출력형태도 같지만, 입금액이 아닌 출금액을 출력 해줘야한다.

(3) 3번을 선택했을 경우와 4번을 선택했을 경우

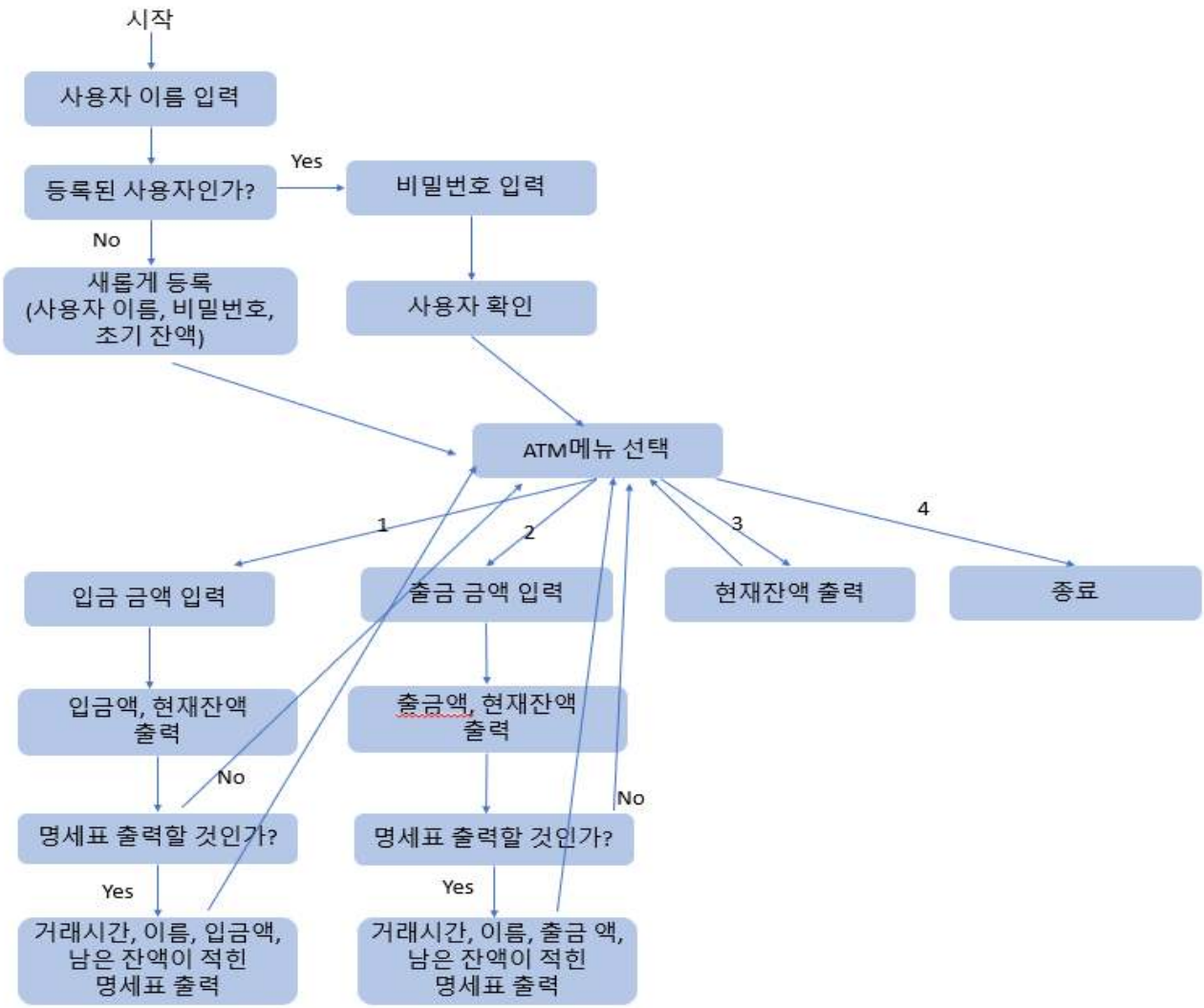
```
elif answer == 3:                                    #3번 선택했을 경우
    print('현재 잔액은 %d입니다.' % self.balance)      #잔액 알려줌
else:                                                #4번 선택했을 경우
    print('서비스를 종료합니다.')
    break                                            #서비스 종료(반복문 종료)
```

- 3번을 선택했을 경우 잔액에 대해 알려주는 것이기 때문에 elif answer==3이라는 조건문을 써주고, 현재잔액을 출력해준다. 현재 잔액은 self.balance이기 때문에 포맷함수를 이용해 출력해준다.

- 4번을 선택했을 경우는 1번 2번 3번이 아닌 수이기 때문에 else라는 조건문을 이용했고, '서비스를 종료합니다'라는 문장이 출력되도록 하였다. 이는 메뉴를 종료하겠다는 의미이므로, break를 사용해 메뉴선택의 반복문을 종료 시켜준다.

2. 진행방향

프로그램을 실행시켰을 때의 진행방향을 Flowchart를 이용해 나타내보았다.



3. 실행 case에 대한 결과

3-1.등록된 사용자일 경우

(1)비밀번호를 맞혔을 경우

```
>>> user1=classAtm.Atm()
Enter the username: >? Susan
Susan 님 환영합니다
Susan님의 패스워드를 입력하세요 >? 1234
사용자 정보가 확인되었습니다
==============================
원하시는 메뉴를 선택하세요
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
```

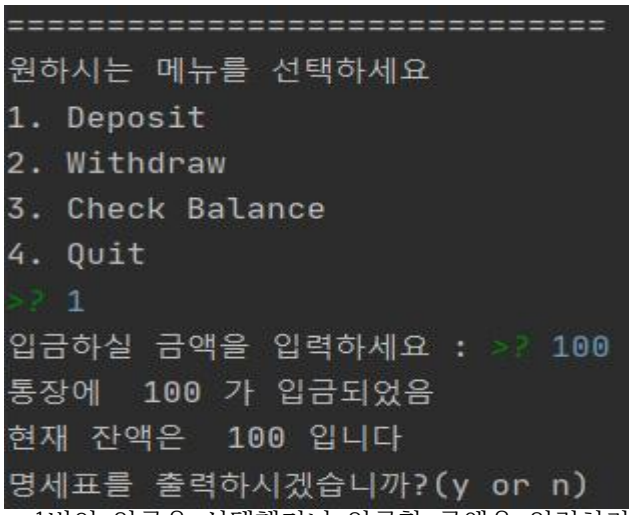
-user1이라는 개체를 만들어 Atm 클래스가 출력되도록 하였다. Susan이라는 등록된 사용자를 입력했을 경우, 'Susan님 환영합니다'라는 출력결과가 나온다. 이후, Susan이라는 사용자 이름을 넣어 패스워드를 입력하라고 뜬다. 패스워드가 맞았을 경우 '사용자 정보가 확인되었습니다'라는 문장이 출력되고, 또 다시 입출금 메뉴가 출력되는 것을 확인할 수 있다.

(2)비밀번호를 틀렸을 경우

```
>>> user1=classAtm.Atm()
Enter the username: >? Susan
Susan 님 환영합니다
Susan님의 패스워드를 입력하세요 >? 3456
비밀번호가 틀렸습니다.
Susan님의 패스워드를 입력하세요 >? 4567
비밀번호가 틀렸습니다.
Susan님의 패스워드를 입력하세요 >? 1234
사용자 정보가 확인되었습니다
==============================
원하시는 메뉴를 선택하세요
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
```

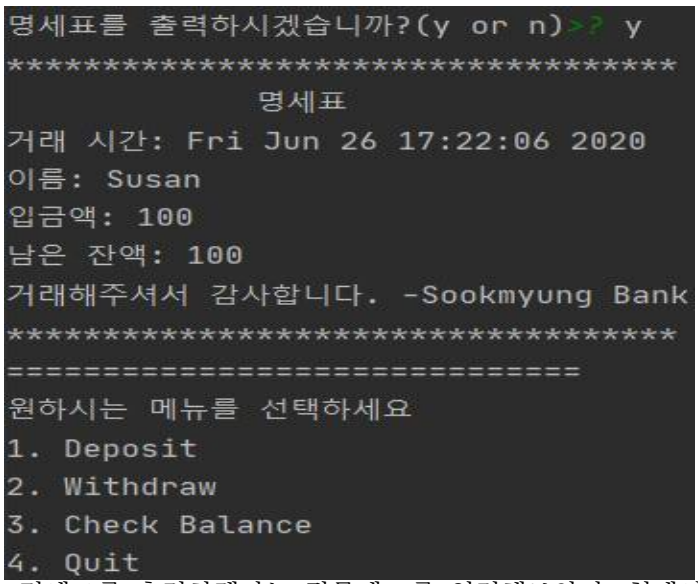
-user1이라는 개체를 만들어 Atm 클래스가 출력되도록 하였다. (1)에서와 같이 'Susan'은 등록된 사용자이지만, 비밀번호를 틀렸을 경우, 비밀번호를 맞출 때까지 비밀번호를 입력하도록 한다. 비밀번호가 맞았을 때, (1)와 같이 '사용자 정보가 확인되었습니다'가 출력되고, 입출금메뉴가 출력됨을 확인할 수 있다.

(3) 메뉴1를 선택했을 경우



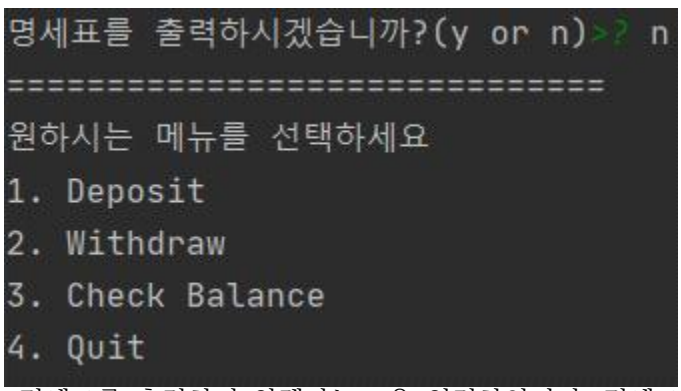
- 1번인 입금을 선택했더니 입금할 금액을 입력하라고 떴다. 100을 입금하겠다고 입력해보았다. 입금한 금액과 잔액에 대해 출력되었다. Susan의 초기값이 0이었기 때문에 100을 입금하면 잔액이 100이 남는 게 맞다. 이는 올바르게 출력되었음을 확인할 수 있다. 이후, 명세표를 출력하겠냐는 질문이 나왔다.

(3-1)명세표를 출력할 경우



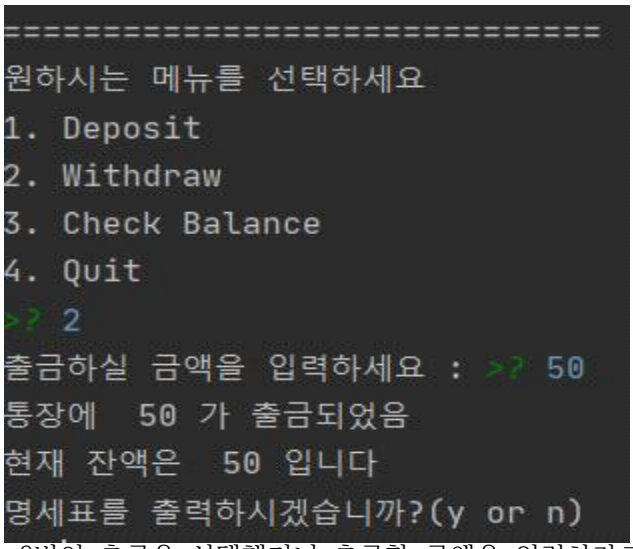
-명세표를 출력하겠냐는 질문에 y를 입력해보았다. 현재시각인 2020/6/26 17:22:06 인 지금의 시간이 요일줄임말, 달 줄임말, 날, 현재 설정된 로케일에 기반 한 시간, 년도출력의 순서와 형태로 출력되었음을 확인할 수 있다. 이름은 사용자의 이름인 Susan으로 나오고, 입금액 100, 잔액 100으로 올바르게 출력되었음을 확인할 수 있다. 또한, 명세표가 출력된 후에 입출금 메뉴가 다시 나오는 것을 확인할 수 있다.

(3-2)명세표를 출력하지 않을 경우



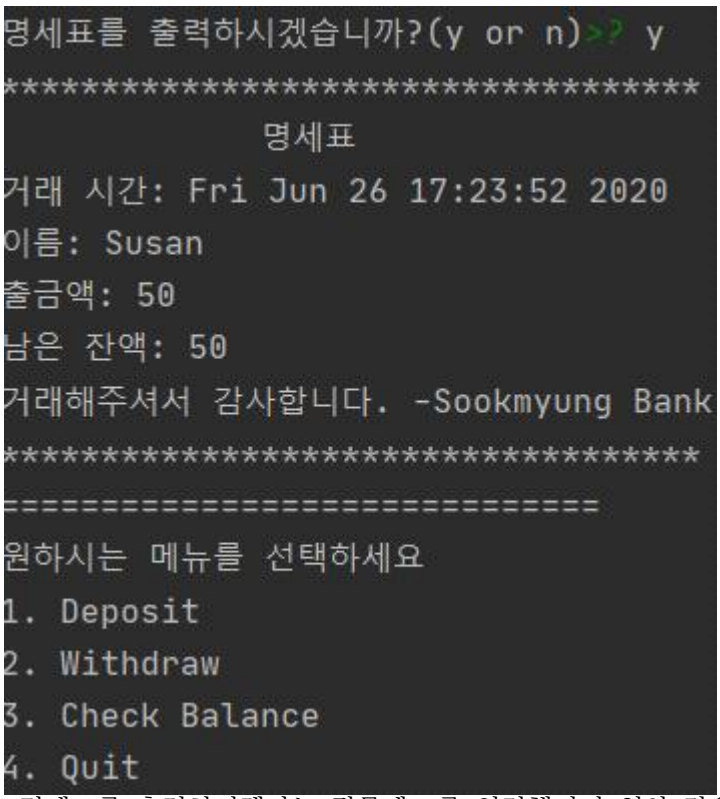
-명세표를 출력하지 않겠다는 n을 입력하였더니, 명세표 출력은 없고 입출금 메뉴판이 나왔음을 확인할 수 있다.

(4)메뉴2를 선택했을 경우



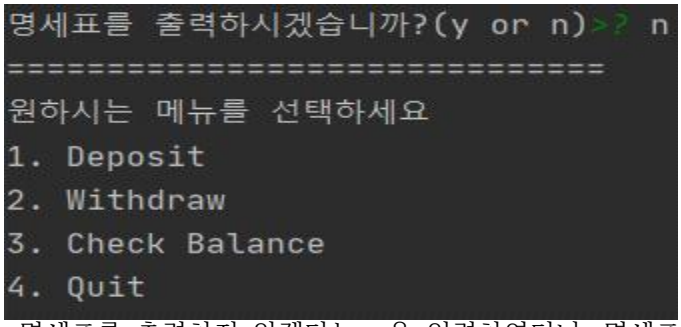
-2번인 출금을 선택했더니 출금할 금액을 입력하라고 떴다. 50을 출금하겠다고 입력해보았다. 출금한 금액과 잔액에 대해 출력되었다. Susan은 초깃값은 0이었지만 (3)에서 100을 입금했기 때문에 100이 남아있다. 여기서 50을 출금을 하니 잔액이 50이 있어야한다. 출력결과 출금할 금액 50과 잔액 50이 올바르게 출력되었음을 확인할 수 있으며, 명세표를 출력여부를 묻는다.

(4-1)명세표를 출력할 경우



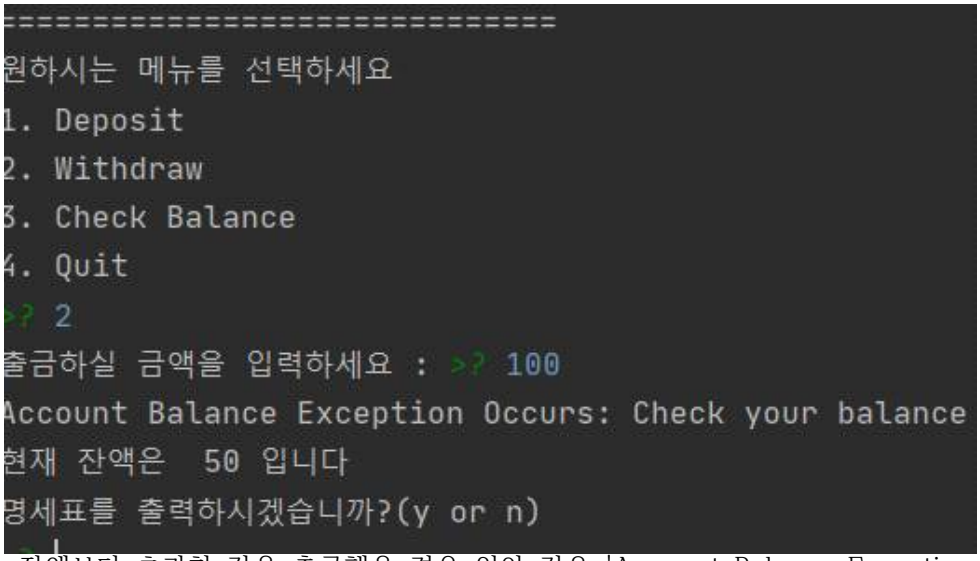
-명세표를 출력하시겠냐는 질문에 y를 입력했더니 위와 같은 명세표가 나왔다. 입금에서와 같이 현재시각인 2020/6/26 17:23:52 인 지금의 시간이 요일줄임말, 달 줄임말, 날, 현재 설정된 로케일에 기반 한 시간, 년도출력 형태로 출력되었음을 확인할 수 있다. 또한, 이름은 사용자의 이름인 Susan으로 나오고, 출금액은 50, 잔액50으로 올바른 형태로 출력되었음을 확인할 수 있다. 명세표가 출력된 이후에 또 다시 입출금메뉴가 나오는 것을 볼 수 있다.

(4-2)명세표를 출력하지 않을 경우



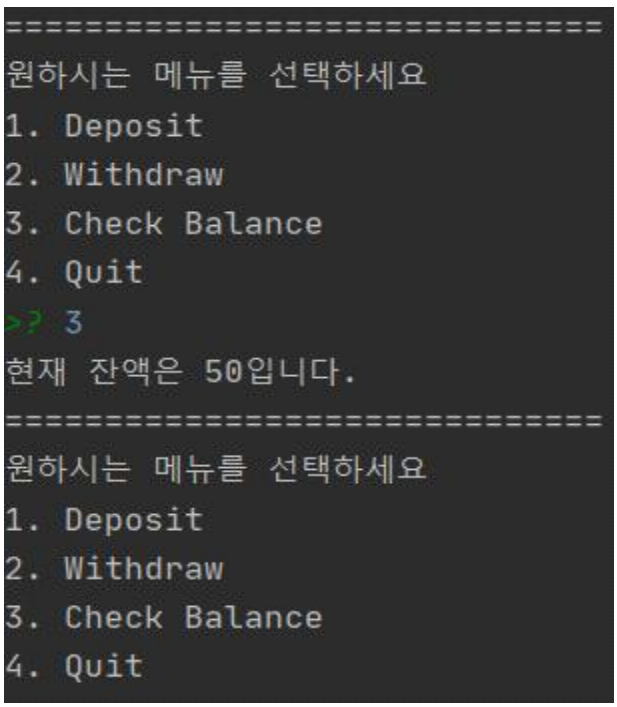
-명세표를 출력하지 않겠다는 n을 입력하였더니, 명세표 출력은 없고, 입출금 메뉴판이 나왔음을 확인할 수 있다.

(4-3)잔액보다 초과한 값을 출금했을 경우



-잔액보다 초과한 값을 출금했을 경우 위와 같은 'Account Balance Exception Occurs: Check your balance' 문장이 출력되고, 잔액을 초과 값이기 때문에 출금은 되지 않으며 잔액은 그대로 남아있다. 이는 classBankAccount에서 지정해준 withdraw함수에서 잔액보다 초과한 값을 출금했을 경우 출력되고, 출금이 되지 않게 하는 것이다. 따라서 classBankAccount 클래스가 Atm 클래스에 상속이 잘되었고, withdraw함수가 잘 작동되었음을 확인할 수 있다.

(5)메뉴3를 선택했을 경우



-4번에서 잔액이 50이 남았음을 확인했다. 메뉴3은 현재 잔액에 대해 알려주는 메뉴이다. 따라서 '현재잔액은 50입니다'라는 문장이 올바르게 출력되었음을 확인할 수 있다. 이후 또다시 입출력 메뉴가 출력되는 것을 확인할 수 있다.

(6)메뉴4를 선택했을 경우

```
원하시는 메뉴를 선택하세요
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
>? 4
서비스를 종료합니다.

>>>
```

-1,2,3의 메뉴와 달리 4번은 종료하는 것이기 때문에 '서비스를 종료합니다.'라는 메시지를 끝으로 입출금메뉴가 나오지 않고, 메뉴가 종료됨을 볼 수 있다.

3-2.등록되지 않은 사용자일 경우

(1)사용자 등록을 할 경우

```
>>> user2=classAtm.Atm()
Enter the username: >? Ok
Ok 님은 등록되지 않았습니다. 추가하시겠습니까?(yes or no)>? yes
Ok 님의 비밀번호를 입력하세요: >? 1234
Ok 님의 초기 잔액을 입력하세요: >? 100
등록이 되었습니다.
==============================
원하시는 메뉴를 선택하세요
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
```

-user2라는 객체를 만들어 Atm클래스가 출력이 되도록하였다. 사용자 등록이 되지 않은 'Ok'라는 이름을 넣었더니 'Ok'라는 사용자 이름을 넣어 등록되지 않았음을 알려주고, 추가하겠냐고 묻는 질문이 나온다. 추가하기 위해 'yes'라 입력했더니 'Ok'의 비밀번호를 설정하라고 뜬다. 1234로 설정을 해보았다. 'Ok'의 초기 잔액을 설정하라고 떴다. 100으로 초기 잔액을 설정했더니 등록이 되었다고 뜨고, 입출금메뉴가 나타나는 것을 확인할 수 있다.

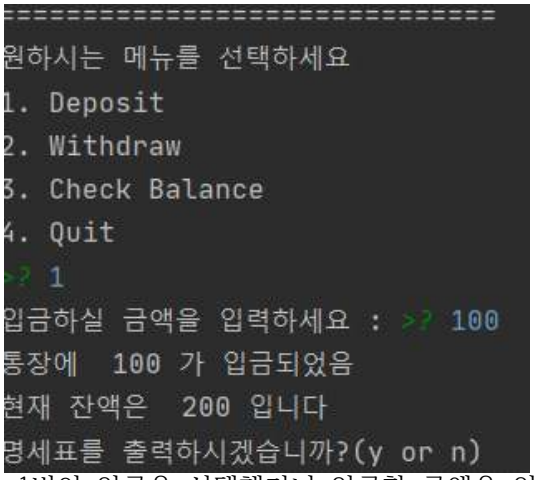
(2)등록하지 않을 경우

```
>>> user2=classAtm.Atm()
Enter the username: >? Ok
Ok 님은 등록되지 않았습니다. 추가하시겠습니까?(yes or no)>? no
서비스를 종료합니다.
```

-user2라는 객체를 만들어 Atm클래스를 출력 되도록 하였다. 사용자 등록이 되지않은 'Ok'라는 이름을 넣었더니 'Ok'라는 사용자 이름을 넣어 등록되지 않았음을 알려주고, 추가하겠냐고 묻는 질문이 나온다. 여기서 추가하지 않기

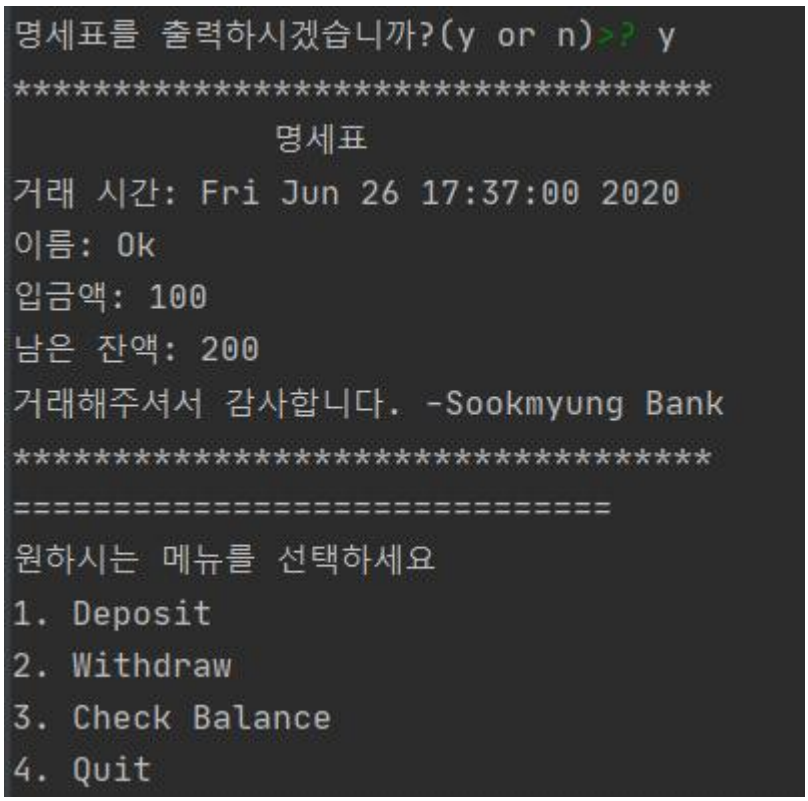
위해, no를 입력했다. '서비스를 종료합니다'라는 문자가 출력되었고, 등록했을 때처럼 입출금 메뉴가 나오지 않았으며 프로그램이 끝났음을 확인할 수 있다.

(3) 메뉴 1번을 선택할 경우



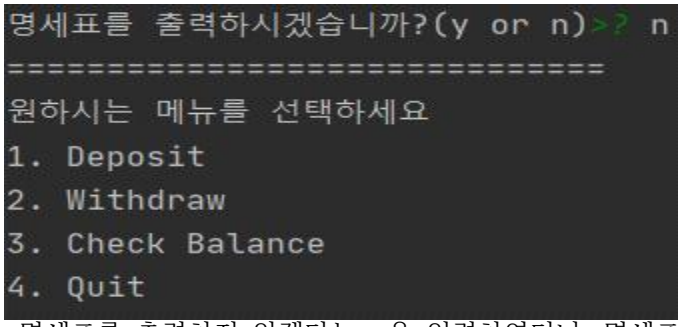
- 1번인 입금을 선택했더니 입금할 금액을 입력하라고 떴다. 100을 입금하겠다고 입력해보았다. 입금한 금액과 잔액에 대해 출력되었다. Ok의 초기값을 100으로 설정했기 때문에 100을 입금하면 잔액이 200이 남는게 맞다. 올바르게 출력되었음을 확인할 수 있다. 이후, 명세표를 출력하겠냐는 질문이 나왔다.

(3-1)명세표를 출력할 경우



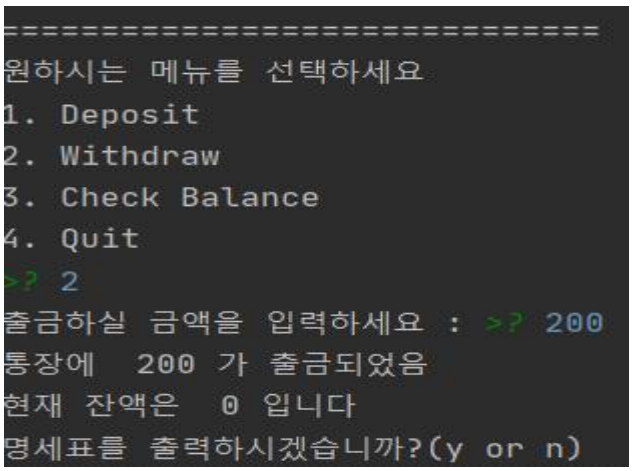
-명세표를 출력하겠냐는 질문에 y를 입력해보았다. 현재시각인 2020/6/26 17:37:00 인 지금의 시간이 요일줄임말, 달 줄임말, 날, 현재 설정된 로케일에 기반 한 시간, 년도출력 형태로 출력되었음을 확인할 수 있다. 새로 등록한 사용자의 이름인 Ok으로 나오고, 입금액은 100, 잔액200으로 올바른 형태로 출력되었음을 확인할 수 있다. 또한 명세표가 출력된 후에 입출금 메뉴가 다시 나오는 것을 확인할 수 있다.

(3-2)명세표를 출력하지 않을 경우



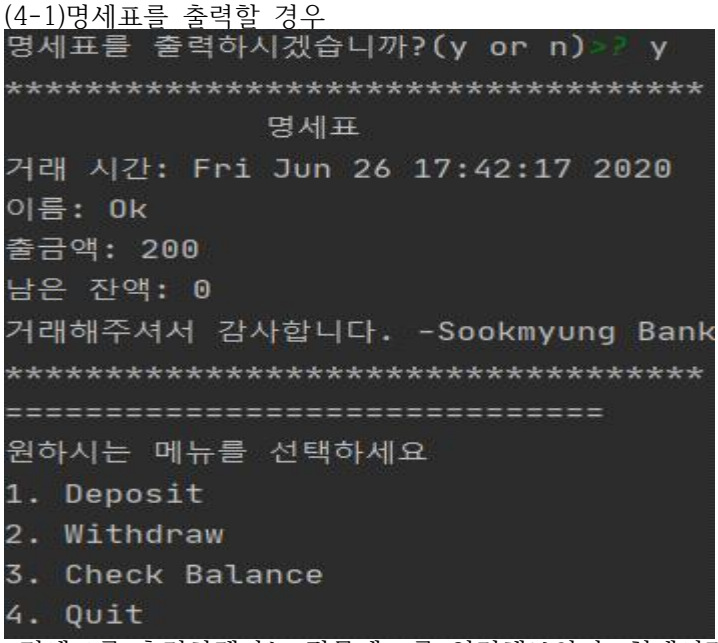
-명세표를 출력하지 않겠다는 n을 입력하였더니, 명세표 출력은 없고 입출금 메뉴판이 나왔음을 확인할 수 있다.

(4) 메뉴 2번을 선택했을 경우



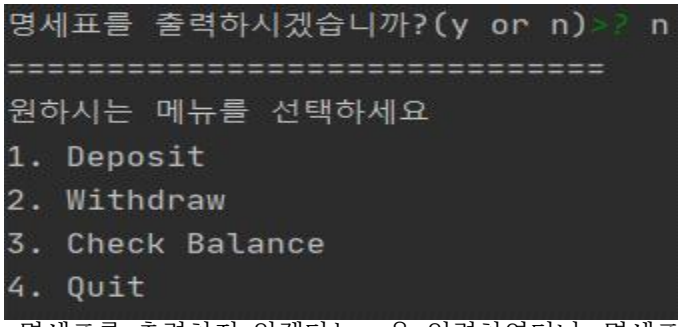
-2번인 출금을 선택했더니 출금할 금액을 입력하라고 떴다. 200을 출금하겠다고 입력해보았다. 출금한 금액과 잔액에 대해 출력되었다. Ok는 초깃값은 100으로 등록하였지만, (2)에서 100을 입금했기 때문에 200이 남아있다. 여기서 200을 출금을 했으니 잔액이 0이 있어야한다. 출금할 금액 200과 잔액 0이 올바르게 출력되었음을 확인할 수 있으며, 명세표를 출력할 것인지에 여부를 묻는다.

(4-1)명세표를 출력할 경우



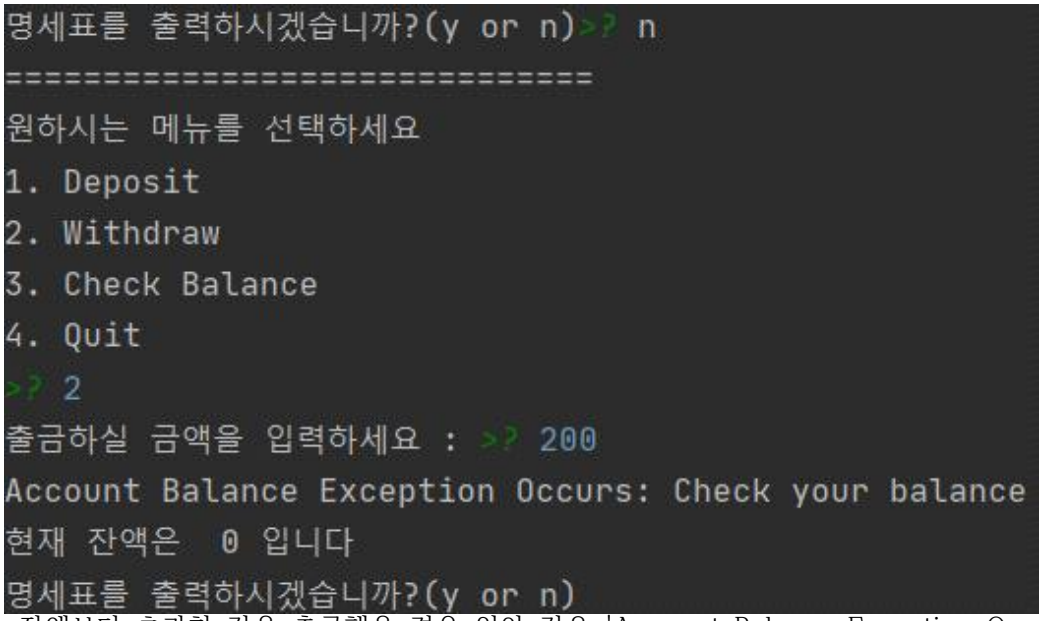
-명세표를 출력하겠냐는 질문에 y를 입력해보았다. 현재시각인 2020/6/26 17:42:17 인 지금의 시간이 요일줄임말, 달 줄임말, 날, 현재 설정된 로케일에 기반 한 시간, 년도출력 형태로 출력되었음을 확인할 수 있다. 새로 등록한 사용자의 이름인 Ok으로 나오고, 출금액은 200, 잔액0으로 올바른 형태로 출력되었음을 확인할 수 있다. 또한 명세표가 출력된 후에 입출금 메뉴가 다시 나오는 것을 확인할 수 있다.

(4-2)명세표를 출력하지 않았을 경우



-명세표를 출력하지 않겠다는 n을 입력하였더니, 명세표 출력은 없고 입출금 메뉴판이 나왔음을 확인할 수 있다.

(4-3)잔액보다 초과한 값을 출금했을 경우

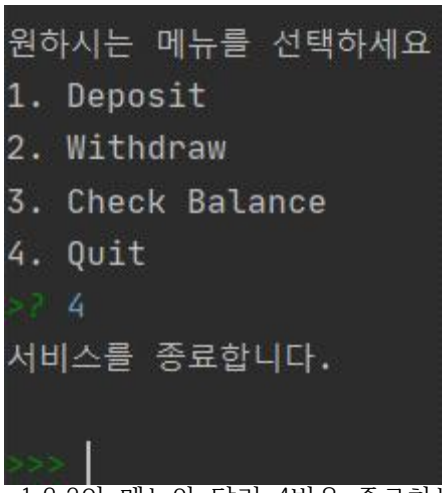


-잔액보다 초과한 값을 출금했을 경우 위와 같은 'Account Balance Exception Occurs: Check your balance' 문장이 출력되고, 잔액을 초과 값이기 때문에 출금은 되지 않으며 잔액은 그대로 남아있다. 이는 classBankAccount에서 지정해준 withdraw함수에서 잔액보다 초과한 값을 출금했을 경우 출력되고, 출금이 되지 않게 하는 것이다. 따라서 classBankAccount 클래스가 Atm 클래스에 상속이 잘되었고, withdraw함수가 잘 작동되었음을 확인할 수 있다.

(5)메뉴 3번을 선택했을 경우

-3에서 잔액이 0이 남았음을 확인했다. 메뉴3은 현재 잔액에 대해 알려주는 메뉴이다. 따라서 '현재잔액은 0입니다'라는 문장이 올바르게 출력되었음을 확인할 수 있다. 또 다시 입출력 메뉴가 출력되는 것을 확인할 수 있다.

(6)메뉴 4번을 선택했을 경우



-1,2,3의 메뉴와 달리 4번은 종료하는 것이기 때문에 '서비스를 종료합니다.'라는 메시지를 끝으로 입출금메뉴가 나오지 않고, 메뉴가 종료됨을 볼 수 있다.

사용자가 등록되어 있을 경우 엑셀에서 리스트를 불러와 사용자에 대해 확인하는 과정이 올바르게 이루어진 것을 확인하였다. 사용자가 등록되어 있지 않아 사용자를 추가해 실행했을 경우 사용자를 추가시키는 과정이 오류 없이 잘 구현되었음을 확인할 수 있었다. 그러나 사용자 추가를 하지 않으면 입출금을 할 수없이 입출금메뉴가 나오지 않음을 확인할 수 있었다. 또한, 사용자가 등록되어 있을 경우와 등록되어 있지 않아 추가했을 경우 모두 입출력메뉴에서 입력과 출력의 함수, 사용자 이름과 잔액 등 모두 올바르게 실행, 출력되었음을 확인하였다. 따라서 클래스 Atm은 사용자를 확인하고, 등록된 사용자가 아니면 추가해 ATM메뉴를 사용할 수 있도록 잘 구현되었음을 확인할 수 있다.