

# 정기수행평가 1회\_강중원

작성자 : 강중원

날짜 : 2024/07/26

---

# 목차

01	Account 클래스
02	BankApp 작성과 findAccount 구현
03	createAccount, accountList구현
04	Deposit과 widthdraw 구현
05	Bank-App 실행 테스트
06	git(형상관리) - 원격 저장소 생성
07	git 폴더 지정
08	git bash
09	git 결과

## 01 Account 클래스

```
1 package bank.app;
2
3 public class Account {
4     String ano;
5     String owner;
6     int balance;
7
8     public Account (String ano, String owner, int balance) {
9         this.ano = ano;
10        this.owner = owner;
11        this.balance = balance;
12    }
13
14    public String getAno() {
15        return ano;
16    }
17    public String getOwner() {
18        return owner;
19    }
20    public int getBalance() {
21        return balance;
22    }
23    public void setBalance(int balance) {
24        this.balance = balance;
25    }
26
27
28 }
```

요구사항의 클래스 정보를 기반으로  
Account 클래스를 작성.  
각각의 속성명과 타입을 지정해 준 후  
생산자와 Getter, Setter를  
구현하였다.

생산자는 클래스의 ano, owner,  
balance를 매개변수의 값으로 초기화  
한다.

## 02 BankApp 작성과 findAccount 구현

```
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class BankApp {
8     private static Scanner scanner = new Scanner(System.in);
9     private static List<Account> accounts = new ArrayList<>();
10    public static void main(String[] args) {
11        boolean run = true;
12        while(run) {
13            System.out.println("-----");
14            System.out.println("1. 계좌생성 | 2. 계좌목록 | 3. 예금 | 4. 출금 | 5. 종료");
15            System.out.println("-----");
16            System.out.println("선택> ");
17
18            int selectNo = Integer.parseInt(scanner.nextLine());
19            if(selectNo == 1) {
20                createAccount();
21            } else if(selectNo == 2) {
22                accountList();
23            } else if(selectNo == 3) {
24                deposit();
25            } else if(selectNo == 4) {
26                withdraw();
27            } else if(selectNo == 5) {
28                run = false;
29            }
30        }
31        System.out.println("프로그램 종료");
32    }
33    private static void createAccount() {
34    }
35    private static void accountList() {
36    }
37    private static void deposit() {
38    }
39    private static void withdraw() {
40    }
41    private static Account findAccount(String ano) {
42    }
43 }
```

### BankAPP의 베이스 코드

제공되는 BankApp의  
주요 코드를 작성

기본적인 코드를 작성후  
먼저 작성할 부분을 탐색

Return 값이 주어지지  
않아 에러가 발생하는  
findAccount를 작성을  
목표로 한다.

```
private static Account findAccount(String ano) {
    for(Account acc : accounts) {
        if(acc.getAno().equals(ano)) {
            return acc;
        }
    }
    System.out.println("계좌가 존재하지 않습니다.");
    return null;
}
```

### findAccount 구현

findAccount의 return값이  
Account이며 매개변수가 문자열 값의  
매개변수 명이 “ano”인것으로 계좌번호를  
매개변수로 Account를 반환, 즉  
계좌번호로 계좌를 찾는 메소드임을 인지 후  
메소드를 구현하였다.

## 03 createAccount, accountList구현

```
private static void createAccount() {
    System.out.println("----- 계좌생성 -----");
    System.out.print("계좌번호: ");
    String tmp_ano = scanner.nextLine();
    System.out.print("계좌주: ");
    String tmp_own = scanner.nextLine();
    System.out.print("초기입금액: ");
    int tmp_bal = Integer.parseInt(scanner.nextLine());

    accounts.add(new Account(tmp_ano, tmp_own, tmp_bal));

    System.out.println("결과: 계좌가 생성되었습니다.");
}
```

### createAccount

출력 결과에 맞춰 출력문을 작성후 Scanner객체의 nextLine() 메서드를 이용하여 임시 함수 tmp\_ano, tmp\_own, tmp\_bal에 값을 넣게된다. 이때 tmp\_bal은 int형 이므로 Integer.parseInt()메서드를 사용하여 자료형을 교체한다.

이후 생성자를 이용하여 Account객체를 생성, accounts 리스트에 삽입한다.

```
private static void accountList() {
    System.out.println("----- 계좌목록 -----");
    for(Account acc : accounts) {
        System.out.println(acc.getAno()+" "+acc.getOwner()+" "+acc.getBalance());
    }
}
```

### accountList

해당 메서드는 호출하면 accounts의 모든 객체의 정보를 출력하는것 을 인지

for(Account acc : accounts)의 형식을 사용하여 리스트 accounts의 각각의 Account를 acc로 접근하여 Getter를 사용하여 출력 하도록 구현하였다.

## 04 Deposit과 withdraw 구현

```
private static void deposit() {
    System.out.println("----- 예금 -----");
    System.out.print("계좌번호: ");
    String tmp_ano = scanner.nextLine();
    System.out.print("예금액: ");
    int tmp_bal = Integer.parseInt(scanner.nextLine());
    findAccount(tmp_ano)
        .setBalance(findAccount(tmp_ano)
            .getBalance() + tmp_bal);

    System.out.println("결과: 예금이 성공되었습니다.");
}

private static void withdraw() {
    System.out.println("----- 출금 -----");
    System.out.print("계좌번호: ");
    String tmp_ano = scanner.nextLine();
    System.out.print("출금액: ");
    int tmp_bal = Integer.parseInt(scanner.nextLine());
    findAccount(tmp_ano)
        .setBalance(findAccount(tmp_ano)
            .getBalance() - tmp_bal);

    System.out.println("결과: 출금이 성공되었습니다.");
}
```

### Deposit과 withdraw

Deposit과 withdraw 메서드는 비슷한 행위를 하는것을 확인후 계좌번호를 scanner객체로 받은 후 예금액을 임시 함수 tmp\_bal 함수로 받는다.

이후 계좌번호로 해당 Account객체를 찾는 findAccount 메서드를 사용

해당 return값은 Account객체임으로 Setter인 setBalance메서드를 tmp\_bal과 기존의 금액(findAccount메서드의 return값의 getter)과 연산의 결과를 넣어 값을 변경하였다.

Deposit과 withdraw는 유사성이 높으므로 연산과 출력문을 변경후 작성

# 05 Bank-App 실행 테스트

```
-----
1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 1
-----
----- 계좌생성 -----
계좌번호: 110-11-1001
계좌주: 김유신
초기입금액: 10000
결과: 계좌가 생성되었습니다.
-----

1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 1
-----
----- 계좌생성 -----
계좌번호: 110-11-1002
계좌주: 김춘주
초기입금액: 20000
결과: 계좌가 생성되었습니다.
-----

1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 2
-----
----- 계좌목록 -----
110-11-1001 김유신 10000
110-11-1002 김춘주 20000
```

```
-----
1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 3
-----
----- 예금 -----
계좌번호: 110-11-1001
예금액: 5000
결과: 예금이 성공되었습니다.
-----

1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 4
-----
----- 출금 -----
계좌번호: 110-11-1002
출금액: 3000
결과: 출금이 성공되었습니다.
-----

1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 2
-----
----- 계좌목록 -----
110-11-1001 김유신 15000
110-11-1002 김춘주 17000
-----

1. 계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----
선택> 5
프로그램 종료
```

## Bank-App의 실행

실행 결과 실행 예시와 일치하였다.



## 06 git(형상관리) - 원격 저장소 생성

### Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \* Hasoro4748 / Repository name \* java-back-app  
✔ java-back-app is available.

Great repository names are short and memorable. Need inspiration? How about [stunning-giggle](#) ?

Description (optional)

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

# GitHub

## Java-bank-app의 원격 저장소를 생성

### ...or create a new repository on the command line

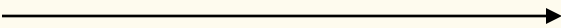
```
echo "# java-back-app" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Hasoro4748/java-back-app.git
git push -u origin main
```

이후 해당 저장소에서 제공하는  
커맨드 라인을 복사하였다



# 07 git 폴더 지정

이름	수정한 날짜	유형
.settings	2024-07-26 오전 9:11	파일 폴더
bin	2024-07-26 오전 9:13	파일 폴더
src	2024-07-26 오전 9:13	파일 폴더
.classpath	2024-07-26 오전 9:11	CLASSPATH 파일
.gitignore	2024-07-26 오전 9:11	Git Ignore 원본 파...
.project	2024-07-26 오전 9:11	PROJECT 파일



이름	수정한 날짜	유형	크기
.git	2024-07-26 오전 10:01	파일 폴더	
.settings	2024-07-26 오전 9:11	파일 폴더	
bin	2024-07-26 오전 9:13	파일 폴더	
src	2024-07-26 오전 9:13	파일 폴더	
.classpath	2024-07-26 오전 9:11	CLASSPATH 파일	
.gitignore	2024-07-26 오전 9:11	Git Ignore 원본 파...	
.project	2024-07-26 오전 9:11	PROJECT 파일	
README.md	2024-07-26 오전 10:01	Markdown 원본 ...	

파일이 있는 폴더에서  
git bash를 연 후  
복사한 커맨드라인을  
붙혀넣어 git폴더를 지정

.git 폴더가 생성되었다.

```
noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (master)
$ echo "# java-back-app" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Hasoro4748/java-back-app.git
git push -u origin main
Initialized empty Git repository in C:/Users/noily/Desktop/workspace/java/java_b
ank_app/.git/
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
[master (root-commit) a30ab69] first commit
 1 file changed, 1 insertion(+)
   create mode 100644 README.md
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 225 bytes | 225.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Hasoro4748/java-back-app.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$ |
```

# 08 git bash

```
noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .classpath
        .gitignore
        .project
        .settings/
        src/

nothing added to commit but untracked files present (use "git add" to track)

noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$ git add .classpath .gitignore .project .settings/ src/
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git
touches it

noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .classpath
        new file:   .gitignore
        new file:   .project
        new file:   .settings/org.eclipse.core.resources.prefs
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   src/bank/app/Account.java
        new file:   src/bank/app/BankApp.java

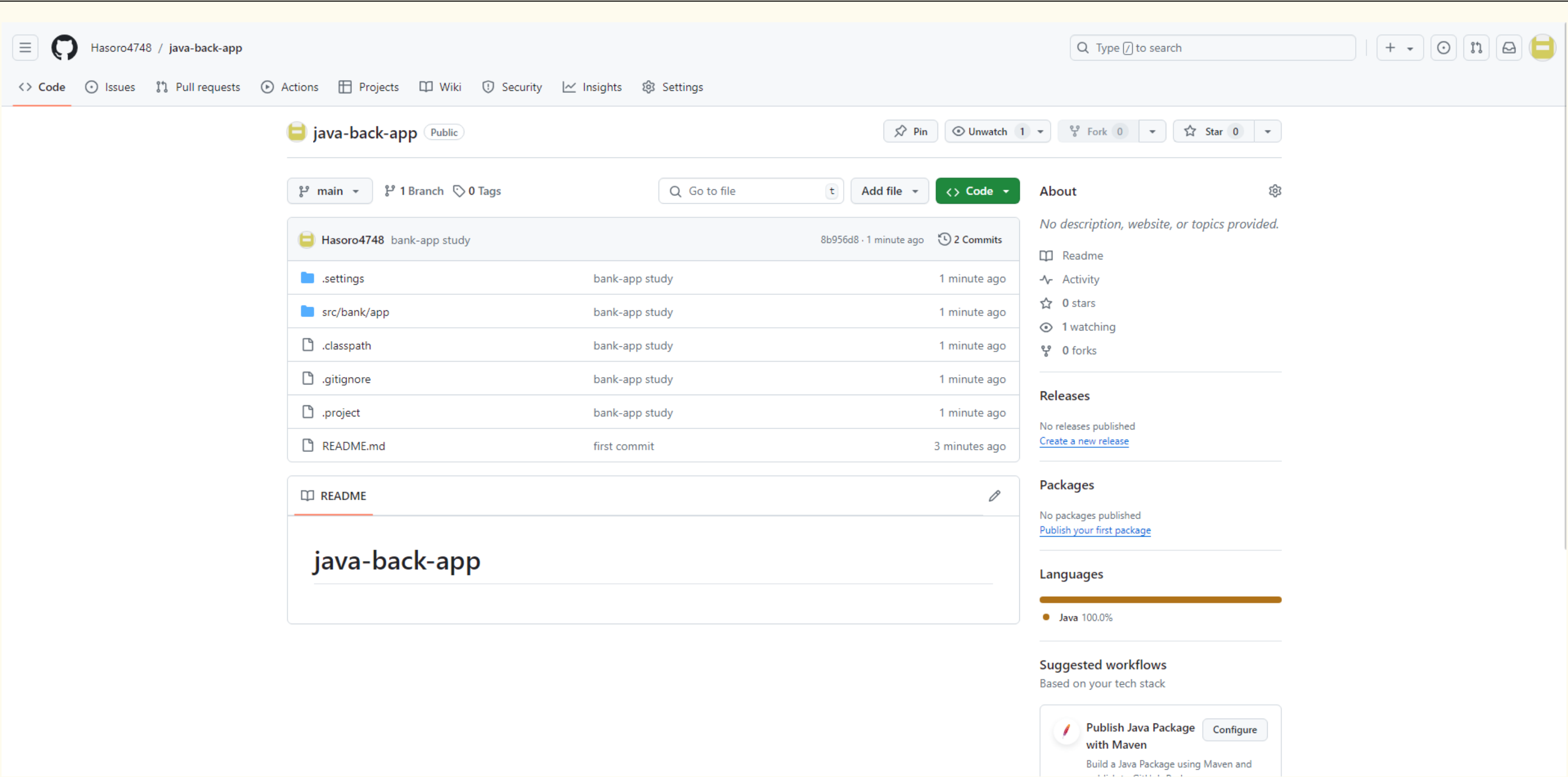
noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$ git commit -m "bank-app study"
[main 8b956d8] bank-app study
 7 files changed, 153 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .gitignore
 create mode 100644 .project
 create mode 100644 .settings/org.eclipse.core.resources.prefs
 create mode 100644 .settings/org.eclipse.jdt.core.prefs
 create mode 100644 src/bank/app/Account.java
 create mode 100644 src/bank/app/BankApp.java

noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (13/13), 2.31 KiB | 2.31 MiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Hasoro4748/java-back-app.git
   a30ab69..8b956d8  main -> main

noily@HASORO MINGW64 ~/Desktop/workspace/java/java_bank_app (main)
$
```

1. Git status 커맨드로 add 가 필요한 파일과 폴더를 확인
2. Git add로 해당 파일을 스테이징한다.
3. 이후 상태를 한번더 확인후 git commit으로 스테이징된 파일을 커밋한다.
4. Git push로 커밋된 파일을 원격 저장소에 등록한다.

# 09 git 결과



이후 원격 저장소에 등록인 된것을 확인할 수 있었다.

**감사합니다**