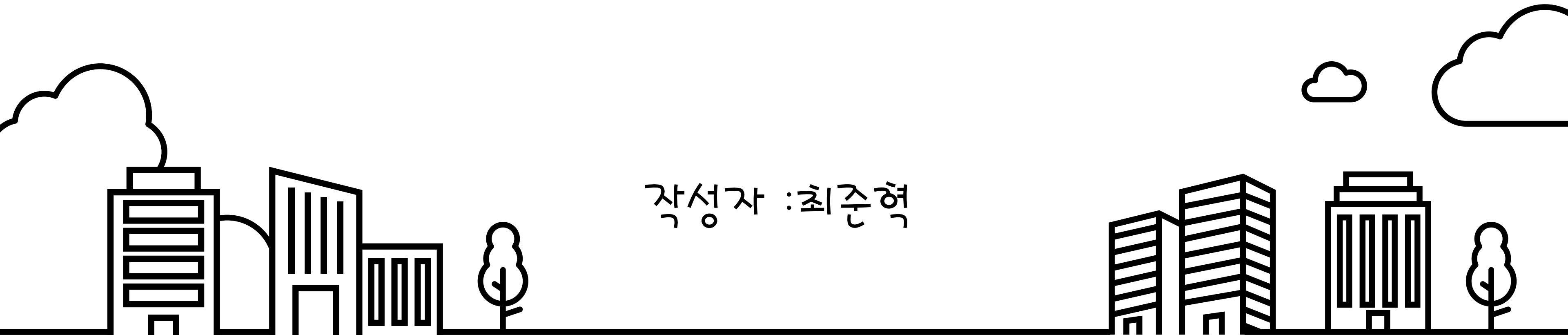


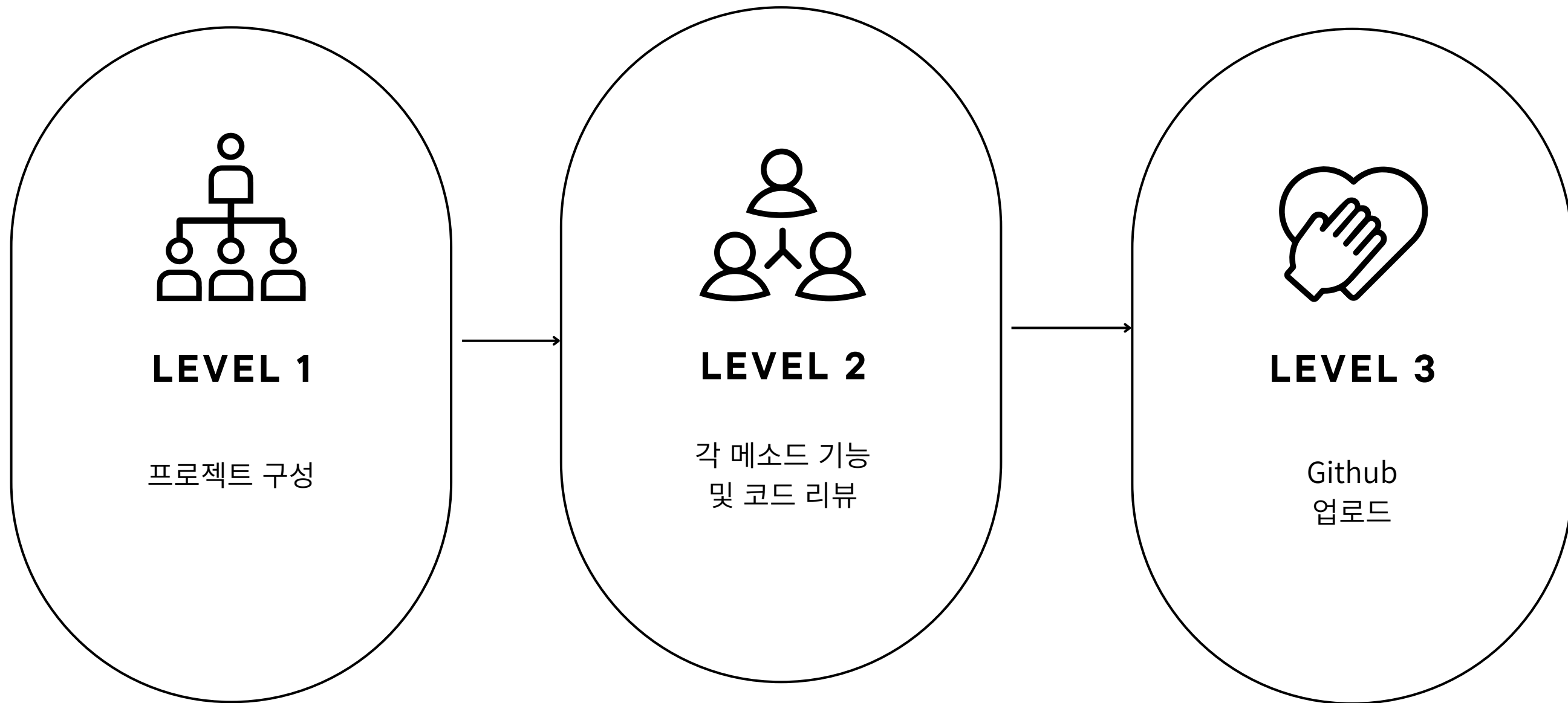
Java mini project

정기수행평가 1회

작성자 : 최준형

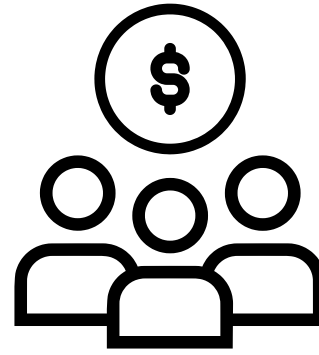


— 진행 순서



Level 1

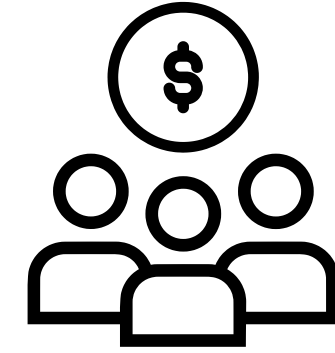
프로젝트 구성



Account.java

```
1 package bank.app;
2
3 public class Account {
4     private String ano;
5     private String owner;
6     private 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
18    public String getOwner() {
19        return owner;
20    }
21
22    public int getBalance() {
23        return balance;
24    }
25
26    public void setBalance(int balance) {
27        this.balance = balance;
28    }
29
30 }
```

계좌를 관리하는 역할을 수행하는 클래스



BankApp.java

```
package bank.app;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

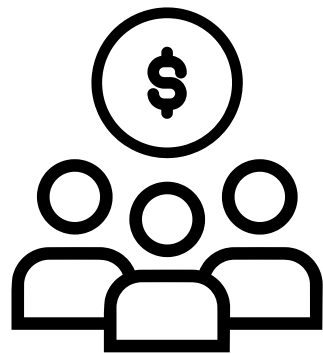
public class BankApp {
    private static Scanner scanner = new Scanner(System.in);
    private static List<Account> accounts = new ArrayList<>();

    public static void main(String[] args){
        boolean run = true;
        while(run) {
            System.out.println("-----");
            System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료");
            System.out.println("-----");
            System.out.print("선택>");

            try {
                int selectNo = Integer.parseInt(scanner.nextLine());

                if(selectNo == 1) {
                    createAccount();
                } else if (selectNo == 2) {
                    accountList();
                } else if (selectNo == 3) {
                    deposit();
                } else if (selectNo == 4) {
                    withdraw();
                } else if (selectNo == 5) {
                    run = false;
                } catch (NumberFormatException e) {
                    System.out.println("번호형식에 맞게 선택값을 입력하세요!");
                }
            }
        }
        System.out.println("프로그램 종료");
    }
}
```

Bank의 여러 기능을 수행하는 클래스



Account.java

속성 정보

String ano	계좌 번호
String owner	예금주
int balance	잔고

메서드 정보

getAno()	ano(계좌번호) Getter
getOwner()	owner(입금주) Getter
getBalance()	balance(잔고) Getter
setBalance()	balance(잔고) Setter

프로젝트 구성



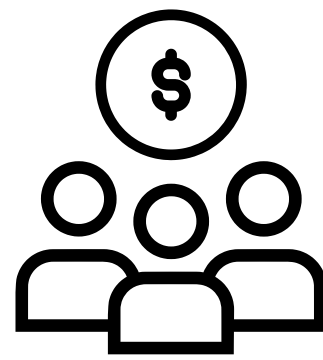
Level 1

```
1 package bank.app;
2
3 public class Account {
4     private String ano;
5     private String owner;
6     private 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
18    public String getOwner() {
19        return owner;
20    }
21
22    public int getBalance() {
23        return balance;
24    }
25
26    public void setBalance(int balance) {
27        this.balance = balance;
28    }
29
30 }
```



프로젝트 구성

Level 1



BankApp.java

```
package bank.app;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class BankApp {
    private static Scanner scanner = new Scanner(System.in);
    private static List<Account> accounts = new ArrayList<>();

    public static void main(String[] args){

        boolean run = true;
        while(run) {
            System.out.println("-----");
            System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료");
            System.out.println("-----");
            System.out.print("선택>");

            try {
                int selectNo = Integer.parseInt(scanner.nextLine());

                if(selectNo == 1) {
                    createAccount();
                } else if (selectNo == 2) {
                    accountList();
                } else if (selectNo == 3) {
                    deposit();
                } else if (selectNo == 4) {
                    withdraw();
                } else if (selectNo == 5) {
                    run = false;
                } catch (NumberFormatException e) {
                    System.out.println("번호형식에 맞게 선택값을 입력하세요!");
                }
            }

            System.out.println("프로그램 종료");
        }

        private static void createAccount() {
            System.out.println("----- 계좌생성 -----");
            System.out.print("계좌번호: ");
            String ano = scanner.nextLine();
            System.out.print("계좌주: ");
            String owner = scanner.nextLine();
            System.out.print("초기입금액: ");
            ...
        }
    }
}
```

속성 정보

Scanner scanner	입력 스캐너 객체
List<Account> account	계좌 리스트

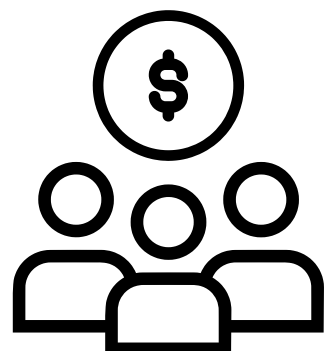
메서드 정보

static main()	프로그램 메인메서드
static createAccount()	계좌생성 정적메서드
static accountList()	계좌목록 정적메서드
static deposit()	입금 정적메서드
static withdraw()	출금 정적메서드
static findAccount(String)	계좌검색 정적메서드

Level 2

각 메소드 기능

main() 메서드



```
public static void main(String[] args){
```

```
    boolean run = true;
```

```
    while(run) {
```

```
        System.out.println("-----");
```

```
        System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료");
```

```
        System.out.println("-----");
```

```
        System.out.print("선택>");
```

```
    try {
```

```
        int selectNo = Integer.parseInt(scanner.nextLine());
```

```
        if(selectNo == 1) {
```

```
            createAccount();
```

```
        } else if (selectNo == 2) {
```

```
            accountList();
```

```
        } else if (selectNo == 3) {
```

```
            deposit();
```

```
        } else if (selectNo == 4) {
```

```
            withdraw();
```

```
        } else if (selectNo == 5) {
```

```
            run = false;
```

```
        }
```

```
    } catch (NumberFormatException e) {
```

```
        System.out.println("번호형식에 맞게 선택값을 입력하세요!");
```

```
    }
```

```
}
```

```
System.out.println("프로그램 종료");
```

```
}
```

while문을 통하여 run값을 매개변수로 은행 어플 실행기능

입력값을 int형으로 받아 각 기능을 하는 메서드를 구분

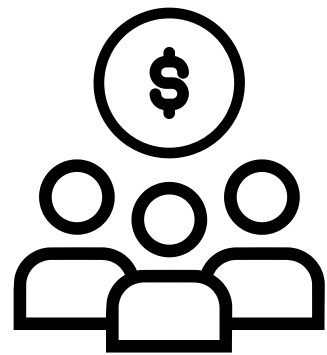
종료(5)를 입력값으로 받는다면
false로 바꾸어 반복문 탈출함으로써
프로그램 종료

입력을 int형으로 안할시 발생오류를 예외처리를
통해 메시지 출력

Level 2

각 메소드 기능

createAccount() 메서드



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

    Account account = new Account(ano, owner, balance);
    accounts.add(account);

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

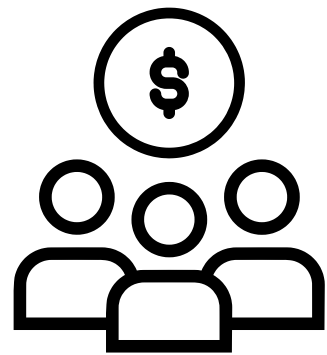
계좌번호, 계좌주, 초기입금액 입력받기

해당 입력값을 통해 account 객체 생성후 account 관리하는 리스트인 accounts에 삽입

```
-----
1. 계좌생성 | 2. 계좌목록 | 3. 예금 | 4. 출금 | 5. 종료
-----
선택>1
-----
계좌생성
계좌번호: 110-11-1001
계좌주: 김유신
초기입금액: 10000
계좌가 생성되었습니다.
-----
1. 계좌생성 | 2. 계좌목록 | 3. 예금 | 4. 출금 | 5. 종료
-----
선택>1
-----
계좌생성
계좌번호: 110-11-1002
계좌주: 김춘추
초기입금액: 20000
계좌가 생성되었습니다.
-----
```

Level 2 각 메소드 기능

accountList() 메서드



accounts 리스트에 객체가 있다면 해당 객체 출력
반복문을 통해 accounts 접근 후 getter를 통해 해당 객체 정보 출력

```
private static void accountList() {  
  
    System.out.println("----- 계좌목록 -----");  
    if(accounts.size() != 0) {  
        for(int i=0; i<accounts.size();i++) {  
            Account account = accounts.get(i);  
            System.out.print(account.getAno()+" "+account.getOwner()+" "+account.getBalance());  
            System.out.println();  
        }  
    } else {  
        System.out.println("생성된 계좌가 없습니다!");  
    }  
}
```

계좌목록에 아무것도 없다면 생성계좌 없다 출력

1. 계좌생성 | 2. 계좌목록 | 3. 예금 | 4. 출금 | 5. 종료

선택>2

----- 계좌목록 -----

110-11-1001 김유신 10000

110-11-1002 김춘추 20000

1. 계좌생성 | 2. 계좌목록 | 3. 예금 | 4. 출금 | 5. 종료

선택>2

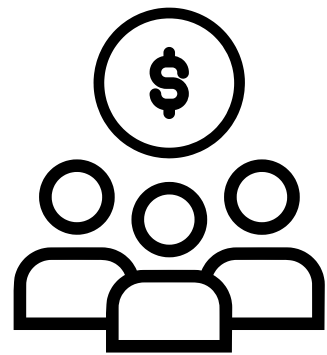
----- 계좌목록 -----

생성된 계좌가 없습니다!

1. 계좌생성 | 2. 계좌목록 | 3. 예금 | 4. 출금 | 5. 종료

Level 2 각 메소드 기능

deposit() 메서드



계좌번호를 입력받아 findAccount()메서드를 통
하여 해당 계좌 찾기

입금 후 계좌 잔액 보여주기

계좌번호 잘못 입력시 잘못되었다 메시지 출력

```
private static void deposit() {  
    System.out.println("-----예금-----");  
    System.out.print("계좌번호: ");  
    String ano = scanner.nextLine();  
    System.out.print("예금액: ");  
    int deposit_balance = Integer.parseInt(scanner.nextLine());  
  
    if(findAccount(ano) != null) {  
        findAccount(ano).setBalance(findAccount(ano).getBalance()+deposit_balance);  
        System.out.println("남은 잔액: " + findAccount(ano).getBalance());  
  
        System.out.println("예금이 성공되었습니다.");  
    } else {  
        System.out.println("계좌번호가 잘못되었습니다.");  
    }  
}
```

1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료

선택>3

예금

계좌번호: 110-11-1001

예금액: 5000

남은 잔액: 15000

예금이 성공되었습니다.

1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료

선택>3

예금

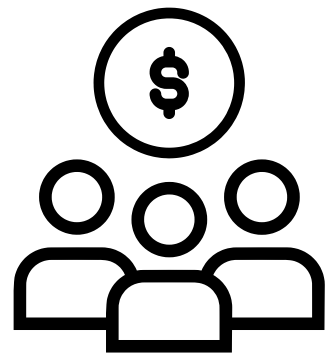
계좌번호: 110-12-11

예금액: 5000

계좌번호가 잘못되었습니다.

Level 2 각 메소드 기능

withdraw() 메서드



계좌번호를 입력받아 findAccount()메서드를 통
하여 해당 계좌 찾기

계좌번호 잘못 입력시 잘못되었다 메시지 출력

남은 잔액보다 출금액이 많을시 해당 메시지 출력

```
private static void withdraw() {  
    System.out.println("----- 출금 -----");  
    System.out.print("계좌번호: ");  
    String ano = scanner.nextLine();  
    System.out.print("출금액: ");  
    int withdraw_balance = Integer.parseInt(scanner.nextLine());  
  
    if(findAccount(ano).getBalance() > withdraw_balance && findAccount(ano) != null) {  
        findAccount(ano).setBalance(findAccount(ano).getBalance()-withdraw_balance);  
        System.out.println("남은 잔액: " + findAccount(ano).getBalance());  
        System.out.println(" 출금이 성공되었습니다.");  
    } else if (findAccount(ano).getBalance() < withdraw_balance){  
        System.out.println("출금액이 계좌 잔액보다 많습니다!");  
    } else if (findAccount(ano) != null) {  
        System.out.println("계좌번호가 잘못되었습니다.");  
    }  
}
```

1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료

선택>4

출금

계좌번호: 111-11-1001

출금액: 5000

남은 잔액: 5000

출금이 성공되었습니다.

1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료

선택>4

출금

계좌번호: 110-11-1001

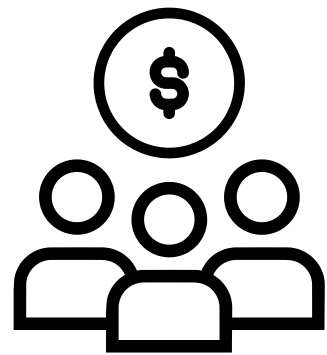
출금액: 50000

출금액이 계좌 잔액보다 많습니다!

Level 2

각 메소드 기능

findAccount() 메서드



계좌번호를 매개변수로 받아 getAno()메서드를 통해 해당 계좌번호를 갖는 account 객체를 받아와 retrun

```
private static Account findAccount(String ano) {  
    Account account = null;  
    for(int i=0; i<accounts.size();i++) {  
        if(accounts.get(i).getAno().equals(ano)) {  
            account = accounts.get(i);  
        }  
    }  
    return account;  
}
```

Level 3

Github 업로드




Github에서 새로운 로컬저장소 생성

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *	Repository name *
 junhyeokkk ▾	/ java-bank-app
<div>✔ Your new repository will be created as -java-bank-app. The repository name can only contain ASCII letters, digits, and the character -.</div>	
Great repository names are short and memorable. Need inspiration? How about jubilant-train ?	
Description (optional)	
<input type="text"/>	

새로운 로컬저장소 연결

```
lotte4@DESKTOP-8N3GG40 MINGW64 ~/Desktop/workspace/java (main)
$ echo "# java-bank-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/junhyeokkk/java-bank-app.git
git push -u origin main
Reinitialized existing Git repository in C:/Users/lotte4.DESKTOP-8N3GG40/Desktop/workspace/java/.git/
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
[main f10511c] first commit
1 file changed, 1 insertion(+)
```

업로드 할 파일 add -> commit -> push 순으로 로컬저장소에 파일 업로드

```
lotte4@DESKTOP-8N3GG40 MINGW64 ~/Desktop/workspace/java (main)
$ git commit 'add Test'
error: pathspec 'add Test' did not match any file(s) known to git

lotte4@DESKTOP-8N3GG40 MINGW64 ~/Desktop/workspace/java (main)
$ git commit -m 'add Test'
[main 21696b7] add Test
7 files changed, 52 insertions(+)
create mode 100644 Test/.classpath
create mode 100644 Test/.gitignore
create mode 100644 Test/.project
create mode 100644 Test/.settings/org.eclipse.core.resources.prefs
create mode 100644 Test/.settings/org.eclipse.jdt.core.prefs
create mode 100644 Test/README.md
create mode 100644 Test/src/module-info.java

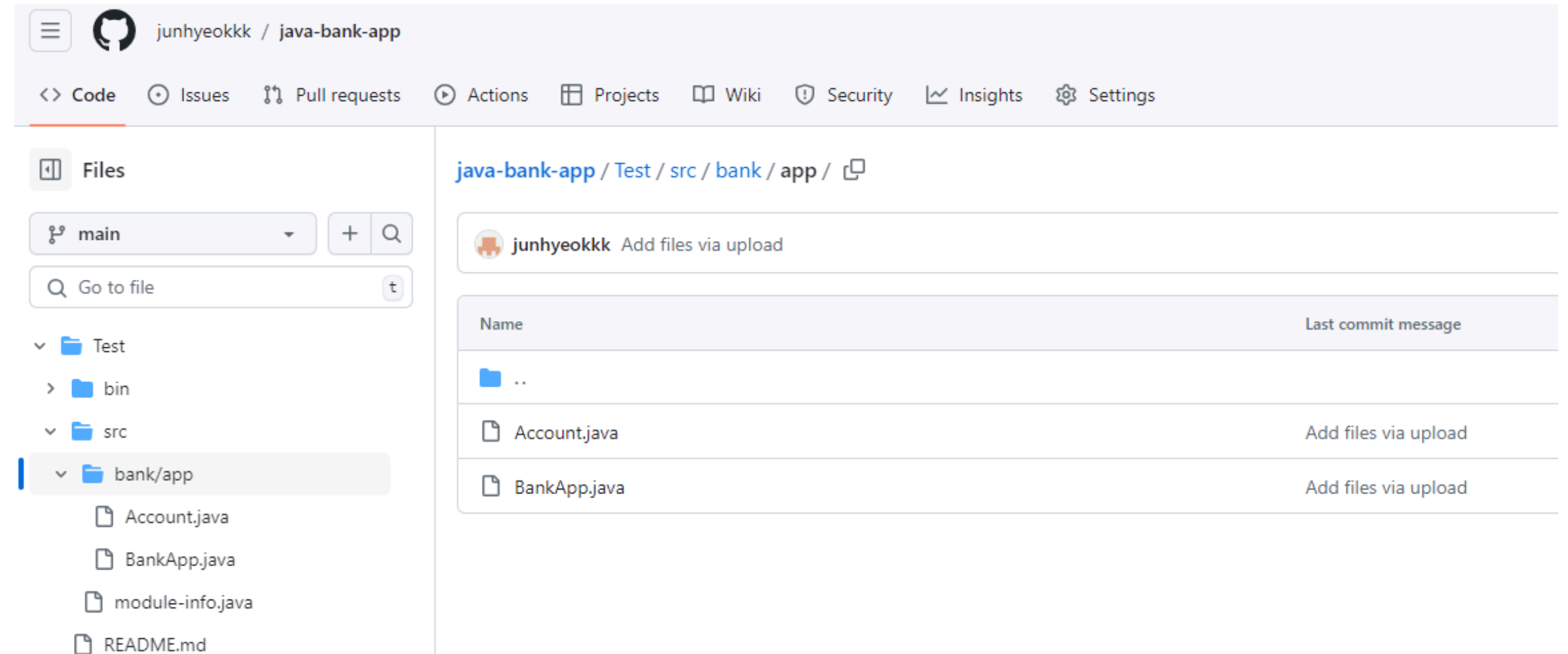
lotte4@DESKTOP-8N3GG40 MINGW64 ~/Desktop/workspace/java (main)
$ git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
```

Level 3

Github 업로드



업로드 성공 캡처화면



감사합니다