[과제]온라인 뱅킹 시스템 구현

501_C189031_서창현

1. 프로그램 개요

온라인 뱅킹 시스템을 구현한 코드

일반 계좌(General Account),

CMA 계좌(Cash Management Account),

증권 예탁 계좌(Securities Account),

정기예금 계좌(Fixed Deposit Account)

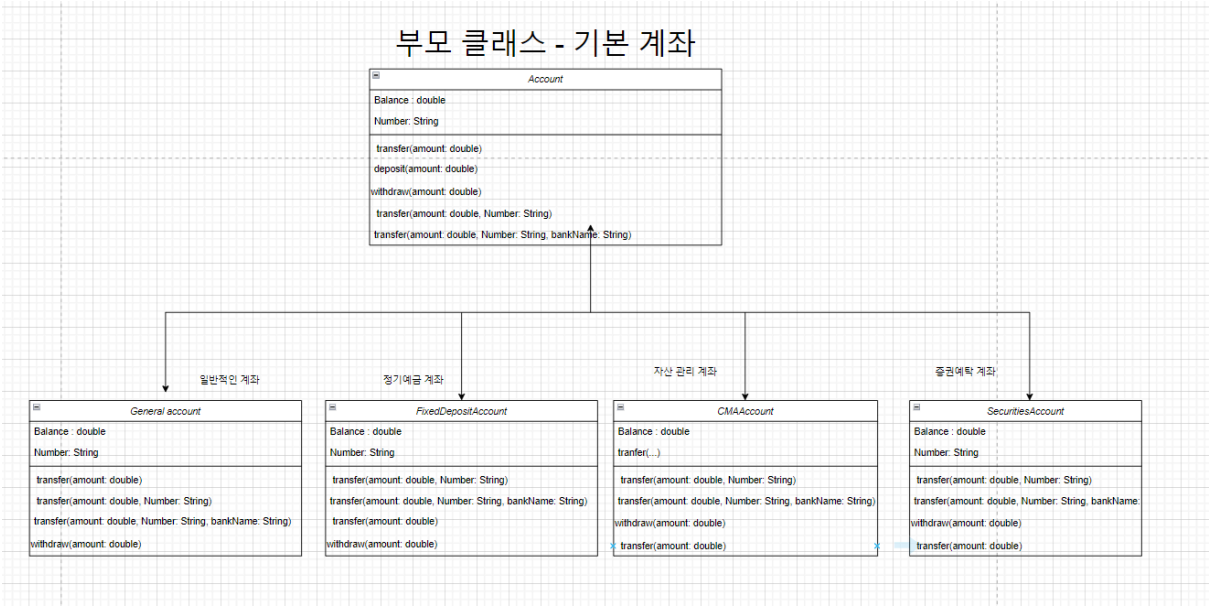
2. 계좌 유형

시스템은 여러 계좌 유형을 지원하며, 각 계좌는 `Account` 클래스를 상속받음. 공통 인터페이스를 따르되, 각 계좌는 필요에 따라 메서드를 **오버라이딩**하여 특화된 동작을 구현.

1. **Account (부모 클래스)**
   - **속성**
     - `Balance: double` : 계좌 잔액
     - `Number: String` : 계좌 번호
   - **메서드**
     - `transfer(amount: double)` : 기본 이체
     - `transfer(amount: double, Number: String)` : 특정 계좌로 이체
     - `transfer(amount: double, Number: String, bankName: String)` : 은행 이름을 포함한 이체
     - `withdraw(amount: double)` : 출금
2. **GeneralAccount (일반 계좌)**
   - `transfer(amount: double)` : 일반 계좌에서 금액 이체
   - `withdraw(amount: double)` : 일반 계좌에서 출금

3. **FixedDepositAccount (정기예금 계좌)**
    - `transfer(amount: double)` : 정기예금에서 이체 제한이 있을 수 있음
    - `withdraw(amount: double)` : 정기예금에서 출금 시 제한이 있을 수 있음
4. **CMAAccount (자산 관리 계좌)**
    - `transfer(amount: double)` : 자산 관리 계좌에서 이체
    - `withdraw(amount: double)` : 자산 관리 계좌에서 출금
5. **SecuritiesAccount (증권예탁 계좌)**
    - `transfer(amount: double)` : 증권예탁 계좌에서 이체
    - `withdraw(amount: double)` : 증권예탁 계좌에서 출금

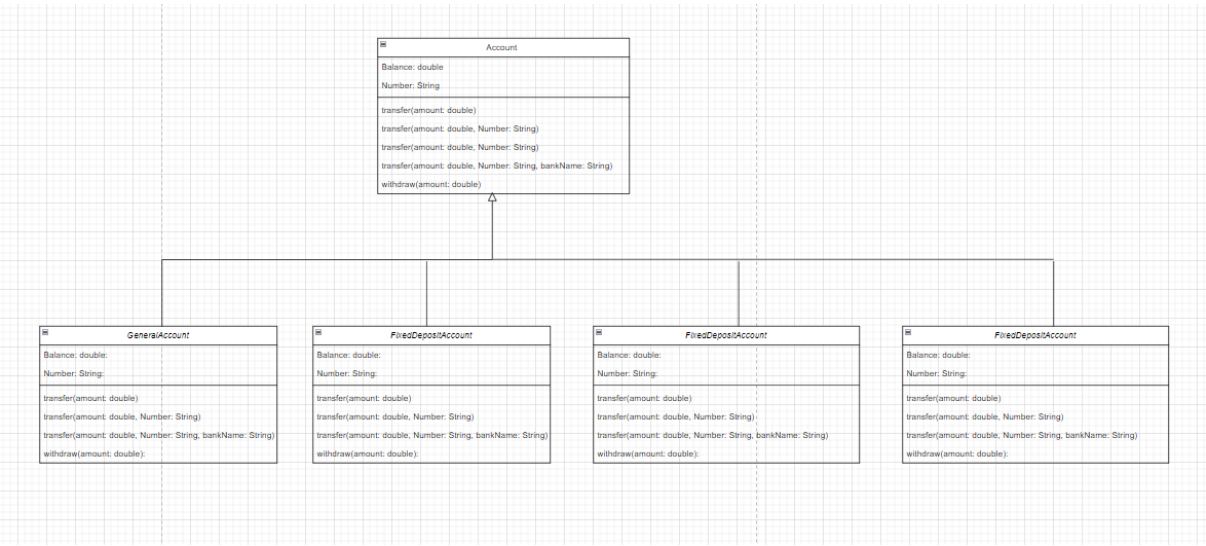
기존 다이어 그램



문제점

`Account` 클래스를 상속받아, 기본적인 `transfer` , `withdraw` 매서드를 오버라이딩하려고 하는데 이때, 속성은
`Balance` 와 `Number` 만 동일하게 유지되며, 각 클래스에서 필요한 방식으로 매서드를 구현하려는 구조.

수정한 다이어그램



- **처음 다이어그램**은 각 자식 클래스가 부모 클래스 `Account`에서 상속받은 동일한 메서드를 가진다고만 보여준다. 각 메서드의 **구체적인 동작 방식**은 설명되지 않았다.
- 각 계좌 유형에 맞게 **다르게 동작**하도록 구현된다는 점을 강조

---

3. 오버라이딩 및 다형성

각 자식 클래스는 부모 클래스에서 상속받은 메서드를 **오버라이딩**하여, 계좌 종류에 맞게 다르게 동작함.

4.코드

```
import java.util.Scanner;

class account {
    protected double balance;
    protected String number;

    public account(String number, double balance) {
        this.number = number;
```

```
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println(number + " 계좌에 " + amount + "원이 입금되었습니다.");
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println(number + " 계좌에서 " + amount + "원이 출금되었습니
        } else {
            System.out.println("잔액이 부족합니다.");
        }
    }

    public void transfer(double amount, String targetAccount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println(targetAccount + " 계좌로 " + amount + "원이 이체되었
        } else {
            System.out.println("잔액이 부족하여 이체할 수 없습니다.");
        }
    }
}

class generalAccount extends account {
    public generalAccount(String number, double balance) {
        super(number, balance);
    }

    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("일반 계좌: " + amount + "원 출금 완료!");
        } else {
```

```
            System.out.println("잔액 부족!");
        }
    }
}

class fixedDepositAccount extends account {
    public fixedDepositAccount(String number, double balance) {
        super(number, balance);
    }
}

class cmaAccount extends account {
    public cmaAccount(String number, double balance) {
        super(number, balance);
    }

    @Override
    public void transfer(double amount, String targetAccount) {
        System.out.println("CMA 계좌에서 " + targetAccount + " 계좌로 " + amount
    }
}

class securitiesAccount extends account {
    public securitiesAccount(String number, double balance) {
        super(number, balance);
    }

    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("증권 계좌: " + amount + "원 출금 (주식 매도 과정 포함)"
        } else {
            System.out.println("잔액 부족!");
        }
    }
}
```

```
public class OnlineBankingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        generalAccount general = new generalAccount("111-111", 10000);
        cmaAccount cma = new cmaAccount("222-222", 20000);
        securitiesAccount securities = new securitiesAccount("333-333", 15000

        while (true) {
            System.out.println("\n 온라인 뱅킹 시스템");
            System.out.println("1. 입금");
            System.out.println("2. 출금");
            System.out.println("3. 이체");
            System.out.println("4. 종료");
            System.out.print("선택: ");
            int choice = scanner.nextInt();

            if (choice == 4) {
                System.out.println("시스템을 종료합니다.");
                break;
            }

            System.out.print("계좌 번호 입력 (111-111, 222-222, 333-333): ");
            String accNum = scanner.next();

            account selectedAccount = null;
            if (accNum.equals("111-111")) {
                selectedAccount = general;
            } else if (accNum.equals("222-222")) {
                selectedAccount = cma;
            } else if (accNum.equals("333-333")) {
                selectedAccount = securities;
            } else {
                System.out.println("잘못된 계좌 번호입니다.");
                continue;
            }

            if (choice == 1) {
```

```
                System.out.print("입금할 금액: ");
                double amount = scanner.nextDouble();
                selectedAccount.deposit(amount);
            } else if (choice == 2) {
                System.out.print("출금할 금액: ");
                double amount = scanner.nextDouble();
                selectedAccount.withdraw(amount);
            } else if (choice == 3) {
                System.out.print("이체할 금액: ");
                double amount = scanner.nextDouble();
                System.out.print("이체할 계좌 번호 입력: ");
                String targetAcc = scanner.next();
                selectedAccount.transfer(amount, targetAcc);
            } else {
                System.out.println("잘못된 선택입니다.");
            }
        }

        scanner.close();
    }
}
```

- `account` 클래스:
  - 계좌 정보를 담고 있으며, 입금(`deposit`), 출금(`withdraw`), 이체(`transfer`) 기능을 가진다.
  - `deposit` 메서드는 금액을 입금하고, `withdraw` 메서드는 금액을 출금한다. 출금 시 잔액을 확인하고, `transfer`는 다른 계좌로 금액을 이체한다.
- `generalAccount`, `fixedDepositAccount`, `cmaAccount`, `securitiesAccount` 클래스는 모두 `account` 클래스를 상속받는다.
  - `generalAccount`는 출금 시 메시지를 "일반 계좌"로 변경한다.
  - `cmaAccount`는 이체 기능을 오버라이딩하여 메시지를 다르게 출력한다.
  - `securitiesAccount`는 출금 시 "주식 매도 과정 포함" 메시지를 출력한다.
- `OnlineBankingSystem` 클래스:

    - `main` 메서드에서 사용자 입력을 받아, 계좌를 선택하고, 입금, 출금, 이체 기능을 실행한다.
    - 무한 루프를 통해 사용자가 종료를 선택할 때까지 계속 실행된다.
- 각 계좌는 번호(예: "111-111", "222-222", "333-333")로 구분되며, 사용자 입력에 따라 해당 계좌에서 입금, 출금, 이체 작업을 수행한다.

```
온라인 뱅킹 시스템
1. 입금
2. 출금
3. 이체
4. 종료
선택: 1
계좌 번호 입력 (111-111, 222-222, 333-333): 111-111
입금할 금액: 25000
111-111 계좌에 25000.0원이 입금되었습니다.

온라인 뱅킹 시스템
1. 입금
2. 출금
3. 이체
4. 종료
선택: 2
계좌 번호 입력 (111-111, 222-222, 333-333): 111-111
출금할 금액: 30000
일반 계좌: 30000.0원 출금 완료!

온라인 뱅킹 시스템
1. 입금
2. 출금
3. 이체
4. 종료
선택: 3
계좌 번호 입력 (111-111, 222-222, 333-333): 111-111
이체할 금액: 20000
이체할 계좌 번호 입력: 222-222
잔액이 부족하여 이체할 수 없습니다.

온라인 뱅킹 시스템
1. 입금
2. 출금
3. 이체
4. 종료
선택: 4
시스템을 종료합니다.
```

5. 개선 할점

아마 현재 기본 잔액은 설정되어있으나 잔액 확인이 안되는 상황이다. 따라서

잔액 확인 기능을 balance를 통해서 확인 할 수 있으면 더욱 좋을 것 같다.