



열혈 Java 프로그래밍

Chapter 07. 클래스와 인스턴스

07-1.

클래스의 정의와 인스턴스 생성

프로그램의 기본 구성

데이터

프로그램상에서 유지하고 관리해야 할 데이터

기능

데이터를 처리하고 조작하는 기능

```
class BankAccountPO {  
    static int balance = 0;    // 예금 잔액  
  
    public static void main(String[] args) {  
        deposit(10000);    // 입금 진행  
        checkMyBalance();    // 잔액 확인  
        withdraw(3000);    // 출금 진행  
        checkMyBalance();    // 잔액 확인  
    }  
  
    public static int deposit(int amount) {    // 입금  
        balance += amount;  
        return balance;  
    }  
  
    public static int withdraw(int amount) {    // 출금  
        balance -= amount;  
        return balance;  
    }  
  
    public static int checkMyBalance() {    // 예금 조회  
        System.out.println("잔액 : " + balance);  
        return balance;  
    }  
}
```

클래스 = 데이터 + 기능

인스턴스 변수

클래스 내에 선언된 변수

인스턴스 메소드

클래스 내에 정의된 메소드

```
class BankAccount {  
    // 인스턴스 변수  
    int balance = 0;  
  
    // 인스턴스 메소드  
    public int deposit(int amount) {...}  
    public int withdraw(int amount) {...}  
    public int checkMyBalance() {...}  
}
```



```
new BankAccount();    // BankAccount 인스턴스 1  
new BankAccount();    // BankAccount 인스턴스 2
```

인스턴스와 참조변수

```
BankAccount myAcnt1;    // 참조변수 myAcnt1 선언
```

```
BankAccount myAcnt2;    // 참조변수 myAcnt2 선언
```

```
myAcnt1 = new BankAccount();    // myAcnt1이 새로 생성되는 인스턴스를 가리킴
```

```
myAcnt2 = new BankAccount();    // myAcnt2가 새로 생성되는 인스턴스를 가리킴
```

```
myAcnt1.deposit(1000);    // myAcnt1이 참조하는 인스턴스의 deposit 호출
```

```
myAcnt2.deposit(2000);    // myAcnt2가 참조하는 인스턴스의 deposit 호출
```

```

class BankAccount {
    int balance = 0;    // 예금 잔액

    public int deposit(int amount) {
        balance += amount;
        return balance;
    }
    public int withdraw(int amount) {
        balance -= amount;
        return balance;
    }
    public int checkMyBalance() {
        System.out.println("잔액 : " + balance);
        return balance;
    }
}

```

클래스, 인스턴스 관련 예제

```

class BankAccount00 {
    public static void main(String[] args) {
        // 두 개의 인스턴스 생성
        BankAccount yoon = new BankAccount();
        BankAccount park = new BankAccount();

        // 각 인스턴스를 대상으로 예금을 진행
        yoon.deposit(5000);
        park.deposit(3000);

        // 각 인스턴스를 대상으로 출금을 진행
        yoon.withdraw(2000);
        park.withdraw(2000);

        // 각 인스턴스를 대상으로 잔액을 조회
        yoon.checkMyBalance();
        park.checkMyBalance();
    }
}

```

참조변수의 특성

1. `BankAccount yoon = new BankAccount();`
2.
3. `yoon = new BankAccount();` `// yoon이 새 인스턴스를 참조한다.`
4.

1. `BankAccount ref1 = new BankAccount();`
2. `BankAccount ref2 = ref1;` `// 같은 인스턴스 참조`
3.

```
class BankAccount {  
    int balance = 0;  
  
    public int deposit(int amount) {  
        balance += amount;  
        return balance;  
    }  
    public int withdraw(int amount) {  
        balance -= amount;  
        return balance;  
    }  
    public int checkMyBalance() {  
        System.out.println("잔액 : " + balance);  
        return balance;  
    }  
}
```

```
class DupRef {  
    public static void main(String[] args) {  
        BankAccount ref1 = new BankAccount();  
        BankAccount ref2 = ref1;  
  
        ref1.deposit(3000);  
        ref2.deposit(2000);  
        ref1.withdraw(400);  
        ref2.withdraw(300);  
        ref1.checkMyBalance();  
        ref2.checkMyBalance();  
    }  
}
```

참조변수 관련 예제

```
class BankAccount { . . . }

class PassingRef {
    public static void main(String[] args) {
        BankAccount ref = new BankAccount();
        ref.deposit(3000);
        ref.withdraw(300);
        check(ref);    // '참조 값'의 전달
    }

    public static void check(BankAccount acc) {
        acc.checkMyBalance();
    }
}
```

참조변수의 매개변수 선언

참조변수에 null 대입

1. `BankAccount ref = new BankAccount();`
2.
3. `ref = null;` `// ref가 참조하는 인스턴스와의 관계를 끊음`

1. `BankAccount ref = null;`
2.
3. `if(ref == null)` `// ref가 참조하는 인스턴스가 없다면`
4. `null 저장 유무에 대한 비교 연산 가능!`

07-2.

생성자와 String 클래스의 소개

String 클래스에 대한 첫 소개

```
public static void main(String[] args) {  
    String str1 = "Happy";  
    String str2 = "Birthday";  
    System.out.println(str1 + " " + str2);  
  
    printString(str1);  
    printString(str2);  
}
```

코드상에서의 문자열 표현은
String 인스턴스의 생성으로 이어진다.

```
public static void printString(String str) {  
    System.out.print(str);  
}
```

문자열을 메소드의 인자로 전달할 수 있다.

매개변수로 String형 참조변수를 선언하여 문자열을 인자로 전달받을 수 있다.

클래스 정의 모델: 인스턴스 구분에 필요한 정보를 갖게 하자.

```
class BankAccount {  
    int balance = 0;    // 예금 잔액  
  
    public int deposit(int amount) {...}  
    public int withdraw(int amount) {...}  
    public int checkMyBalance() {...}  
}
```

문제 있는 클래스 정의

```
class BankAccount {  
    String accNumber;    // 계좌번호  
    String ssNumber;    // 주민번호  
    int balance = 0;    // 예금 잔액  
  
    public int deposit(int amount) {...}  
    public int withdraw(int amount) {...}  
    public int checkMyBalance() {...}  
}
```

좋은 클래스 정의 후보!

좋은 클래스 정의 후보를 위한 초기화 메소드!

```
class BankAccount {  
    String accNumber;    // 계좌번호  
    String ssNumber;     // 주민번호  
    int balance = 0;     // 예금 잔액
```

초기화를 위한 메소드

```
    public void initAccount(String acc, String ss, int bal) {  
        accNumber = acc;  
        ssNumber = ss;  
        balance = bal; // 계좌 개설 시 예금액으로 초기화  
    }
```

```
    . . .
```

```
}
```

```
    public static void main(String[] args) {  
        BankAccount yoon = new BankAccount();    // 계좌 생성  
        yoon.initAccount("12-34-89", "990990-9090990", 10000);    // 초기화  
        . . .  
    }
```

초기화 메소드를 대신하는 생성자

```
class BankAccount {  
    String accNumber; // 계좌번호  
    String ssNumber; // 주민번호  
    int balance; // 예금 잔액
```

생성자의 이름은 클래스의 이름과 동일해야 한다.

생성자는 값을 반환하지 않고 반환형도 표시하지 않는다.

```
    public BankAccount(String acc, String ss, int bal) { // 생성자  
        accNumber = acc;  
        ssNumber = ss;  
        balance = bal;  
    }
```

초기화를 위한 생성자

```
    . . .  
}
```

```
    public static void main(String[] args) {  
        BankAccount yoon = new BankAccount("12-34-89", "990990-9090990", 10000);  
        . . .  
    }
```

디폴트 생성자

```
class BankAccount {  
    int balance;  
    public BankAccount() { // 컴파일러에 의해 자동 삽입되는 '디폴트 생성자'  
        // empty  
    }  
  
    public int deposit(int amount) {...}  
    public int withdraw(int amount) {...}  
    public int checkMyBalance() {...}  
}
```

이렇듯 모든 클래스의 인스턴스 생성은 생성자 호출을 동반한다.

07-3. 자바의 이름 규칙

클래스의 이름 규칙

클래스 이름의 첫 문자는 대문자로 시작한다.

둘 이상의 단어가 묶여서 하나의 이름을 이룰 때, 새로 시작하는 단어는 대문자로 한다.

ex)

Circle + Point = CirclePoint

Camel Case 모델

메소드와 변수의 이름 규칙

메소드 및 변수 이름의 첫 문자는 **소문자**로 시작한다.

둘 이상의 단어가 묶여서 하나의 이름을 이룰 때, 새로 시작하는 단어는 대문자로 한다.

ex)

Add + Your + Money = addYourMoney

Your + Age = yourAge

변형된 Camel Case 모델

상수의 이름 규칙

상수의 이름은 모든 문자를 대문자로 구성한다.

둘 이상의 단어가 묶여서 하나의 이름을 이룰 때 단어 사이를 언더바로 연결한다.

ex)

```
final int COLOR_RAINBOW = 7;
```

The End

Chapter 07의 강의를 마칩니다.