Chapter 07. 클래스와 인스턴스

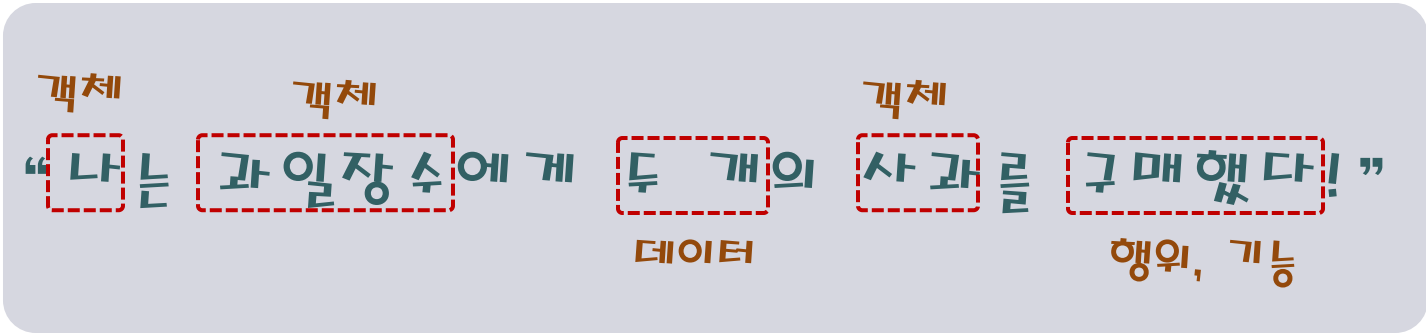
나는 정말 JAVA를 공부한 적이 없다고

07-1.
클래스의 정의와 인스턴스의 생성

객체지향 프로그래밍과 객체

나 정말 JAVA를 공부한 적 없는데요

- 객체(Object)
  - 사전적 의미 : 물건 또는 대상
  - 객체지향 프로그래밍: 객체 중심의 프로그래밍



객체지향 프로그래밍에서는 나, 과일장수, 사과라는 객체를 등장 시켜서 두 개의 사과 구매라는 행위를 실체화 한다.

■ 객체를 이루는 것은 데이터와 기능입니다.

나 정말 JAVA를 공부한 적 없는데요

과일장수 객체의 표현

과일장수는 과일을 팝니다. 행위

과일장수는 사과 20개, 오렌지 10개를 보유하고 있습니다. 상태

과일장수의 과일판매 수익은 50,000원입니다. 상태

과일장수의 데이터 표현

보유하고 있는 사과의 수: int numOfApple;

판매 수익: int myMoney;

과일장수의 행위 표현

```
int saleApple(int money)      // 사과 구매액이 메소드의 인자로 전달
{
    int num = money/1000;     // 사과가 개당 1000원이라 가정
    numOfApple -= num;    // 사과의 수가 줄어들고,
    myMoney += money;     // 판매 수익이 발생한다.
    return num;           // 실제 구매가 발생한 사과의 수를 반환
}
```

■ 클래스(class)라는 틀 기반의 객체 생성

객체 생성에 앞서 선행되어야 하는 클래스의 정의

```
class FruitSeller
{
    int numOfApple=20;
    int myMoney=0;

    public int SaleApple(int money)
    {
        int num = money/1000;
        numOfApple -= num;
        myMoney += money;
        return num;
    }
}
```

변수 선언

메소드 정의

FruitSeller라는 이름의 틀 정의

■ '과일장수' 클래스 정의와 키워드 final


```
class FruitSeller
{
    final int APPLE_PRICE=1000;
    int numOfApple=20;
    int myMoney=0;

    public int saleApple(int money)
    {
        int num=money/APPLE_PRICE;
        numOfApple-=num;
        myMoney+=money;
        return num;
    }
    public void showSaleResult()
    {
        System.out.println("남은 사과 : " + numOfApple);
        System.out.println("판매 수익 : " + myMoney);
    }
}
```

변수 APPLE_PRICE를 상수화한다.

과일 판매 기능의 표현

오늘 좀 많이 파셨어요? 에 대한 대답

‘나(me)’ 클래스 정의

과일 구매자 관점에서의 나!

```
class FruitBuyer
{
    int myMoney=5000;
    int numOfApple=0;

    public void buyApple(FruitSeller seller, int money)
    {
        numOfApple+=seller.saleApple(money);
        myMoney-=money;
    }
    public void showBuyResult()
    {
        System.out.println("현재 잔액 : " + myMoney);
        System.out.println("사과 개수 : " + numOfApple);
    }
}
```

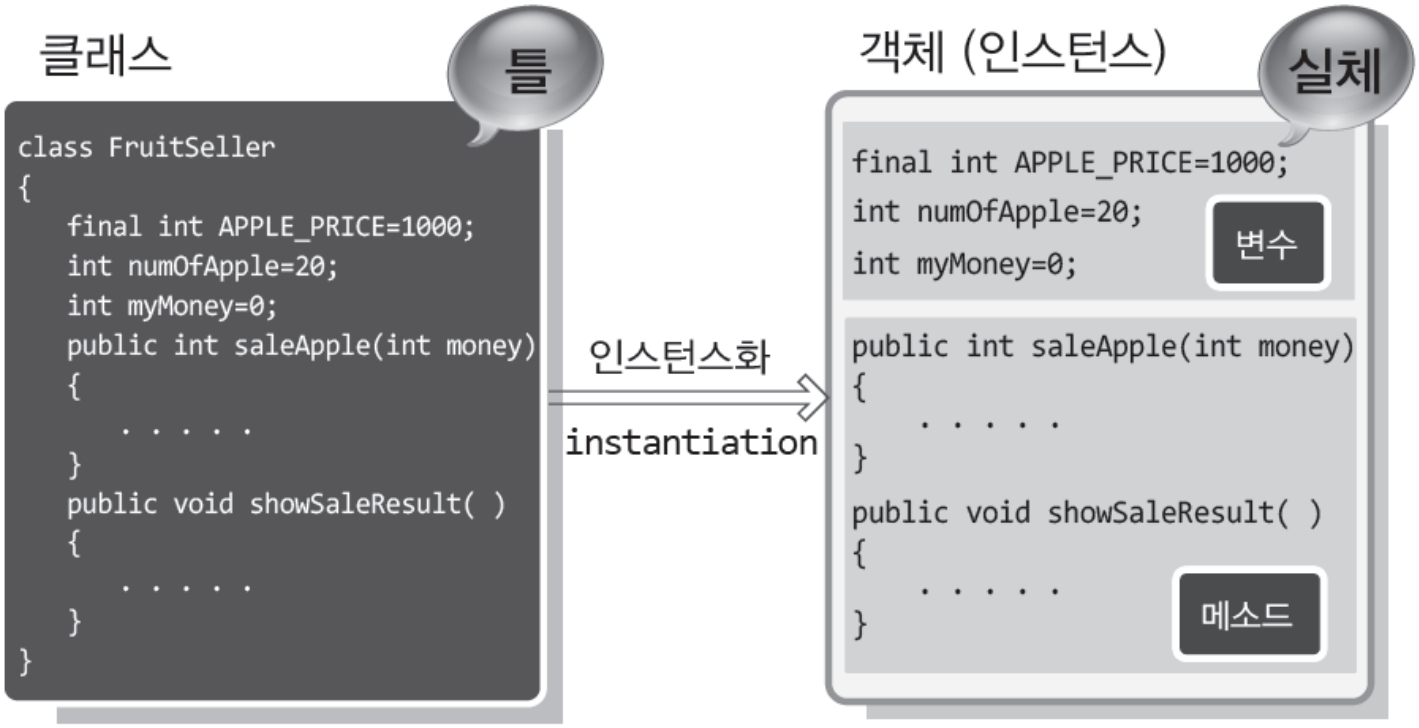
소유 현금

소유 과일

과일 구매행위의 표현

네가 가지고 있는 돈과 과일의 수는?

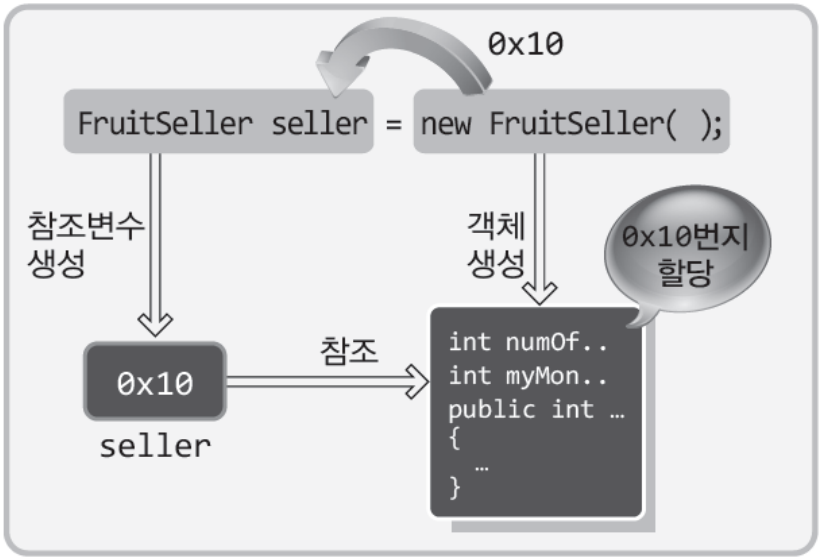
■ 클래스를 기반으로 객체 생성하기



참조변수의 선언 인스턴스의 생성

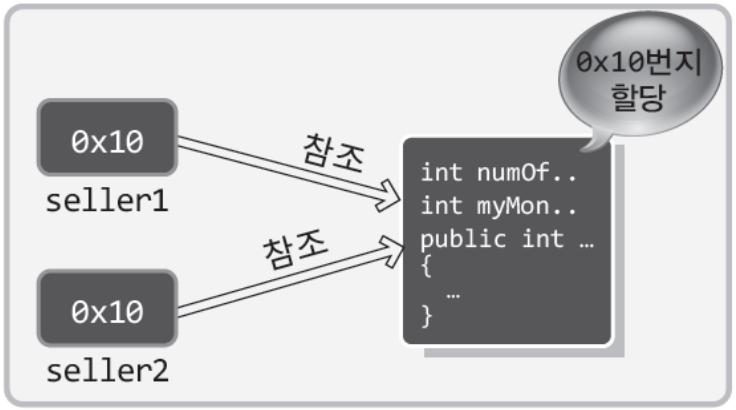
```
FruitSeller seller = new FruitSeller();
FruitBuyer buyer = new FruitBuyer();
```

객체생성과 참조의 관계를 정확히 말하라!



참조변수의 역할

```
FruitSeller seller1 = new FruitSeller();
FruitSeller seller2 = seller1;
```



■ 생성된 객체의 접근방법

객체 내에 존재하는 변수의 접근

```
Fruitseller seller=ner FruitSeller();
seller.numOfApple=20;
```

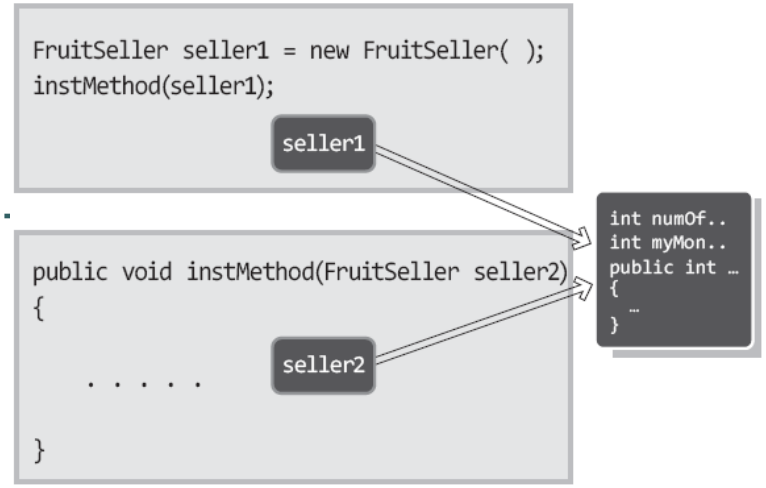
객체 내에 존재하는 메소드의 접근(호출)

```
Fruitseller seller =nerFruitSeller();
seller.saleApple(10);
```

참조변수와 메소드의 관계

```
public void myMethod()
{
    FruitSeller seller1 = new FruitSeller();
    instMethod(seller1);
    . . . .
}
```

```
public void instMethod(FruitSeller seller2)
{
    . . . . .
}
```



■ 참조변수와 메소드의 관계확인을 위한 예제


```
class Number
{
    int num=0;
    public void addNum(int n)
    {
        num+=n;
    }
    public int getNumber()
    {
        return num;
    }
}
```

인스턴스 변수라 한다.

인스턴스 메소드라 한다.

```
class PassInstance
{
    public static void main(String[] args)
    {
        Number nInst=new Number();
        System.out.println("메소드 호출 전 : "+nInst.getNumber());
        simpleMethod(nInst);
        System.out.println("메소드 호출 후 : "+nInst.getNumber());
    }
    public static void simpleMethod(Number numb)
    {
        numb.addNum(12);
    }
}
```

실행결과

```
메소드 호출 전 : 0
메소드 호출 후 : 12
```

■ 참조변수의 null 초기화


```
MyInst my=null;    // MyInst라는 클래스의 참조변수 my의 선언

if(my==null)
    System.out.println("참조변수 my는 현재 참조하는 객체가 없습니다.");
```

null은 아무것도 참조하지 않음을 의미하는 키워드

■ 사과장수 시뮬레이션 완료!

```
class FruitSalesMain1
{
    public static void main(String[] args)
    {
        FruitSeller seller = new FruitSeller();
        FruitBuyer buyer = new FruitBuyer();
        buyer.buyApple(seller, 2000);   // seller에게 2000원어치 사과 구매

        System.out.println("과일 판매자의 현재 상황");
        seller.showSaleResult();

        System.out.println("과일 구매자의 현재 상황");
        buyer.showBuyResult();
    }
}
```

메소드 호출 → 메시지 전달(passing)

```
public void buyApple(FruitSeller seller, int money)
{
    numOfApple+=seller.saleApple(money);
    myMoney-=money;
}
```

메시지 전달은 두 객체간의 대화방법이다. 위 예제에서의 buyApple 메소드가 의미하는 바는 다음과 같다.

"아저씨 사과 2000원어치 주세요!"

07-2. 생성자(Constructor)

■ 두 명의 과일장수와 한 명의 구매자

서로 다른 인스턴스의 생성은, 인스턴스 변수의 초기화라는 문제를 고민하게 한다.

- 과일장수1 : 보유하고 있는 사과의 수는 30개이고, 개당 가격은 1,500원
- 과일장수2 : 보유하고 있는 사과의 수는 20개이고, 개당 가격은 1,000원

```
class FruitSeller
{
    int numOfApple;
    int myMoney;
    int APPLE_PRICE;        // 이전 예제에서는 final로 선언되었다!
    . . . . .
    public void initMembers(int money, int appleNum, int price)
    {
        myMoney=money;
        numOfApple=appleNum;
        APPLE_PRICE=price;
    }
}
```

APPLE_PRICE의 값이 변경되어야 하므로 final이 빠진다.

바람직하지 않다!

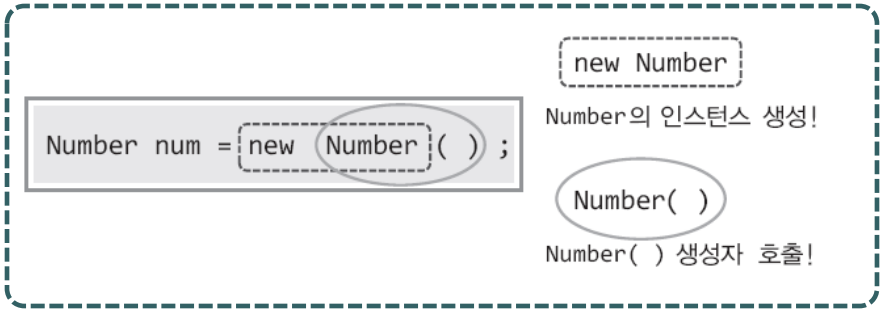
- final이 빠지므로..
- 초기화 과정의 불편함..

```
public static void main(String[] args)
{
    FruitSeller seller1 = new FruitSeller();
    seller1.initMembers(0, 30, 1500);
    FruitSeller seller2 = new FruitSeller();
    seller2.initMembers(0, 20, 1000);
    FruitBuyer buyer = new FruitBuyer();
    buyer.buyApple(seller1, 4500);
    buyer.buyApple(seller2, 2000);
    . . . . . .
}
```

■ 딱 한번만 호출되는 메소드! 생성자!

나 정말 JAVA를 공부한 적이 없다구요

```
class Number
{
    int num;
    public Number()     // 생성자!
    {
        num=10;
        System.out.println("생성자 호출!");
    }
    public int getNumber()
    {
        return num;
    }
}
```




```
public static void main(String[] args)
{
    Number num1=new Number();
    System.out.println(num1.getNumber());
    Number num2=new Number();
    System.out.println(num2.getNumber());
}
```

실행결과

```
생성자 호출!
10
생성자 호출!
10
```

생성자의 조건

- 클래스의 이름과 동일한 이름의 메소드
- 반환형이 선언되어 있지 않으면서, 반환하지 않는 메소드

자바 인스턴스 생성시 생성자는 반드시 호출된다!

값을 전달받는 생성자


```
Number num = new Number(10);
Number num = new Number(30);
```

호출되는 생성자

```
public Number(int n)
{
    . . . .
}
```

```
class FruitSeller
{
    int numOfApple;
    int myMoney;
    final int APPLE_PRICE;
    public FruitSeller(int money, int appleNum, int price)
    {
        myMoney=money;
        numOfApple=appleNum;
        APPLE_PRICE=price;
    }
    . . . . . .
}
```

```
public static void main(String[] args)
{
    FruitSeller seller1 = new FruitSeller(0, 30, 1500);
    FruitSeller seller2 = new FruitSeller(0, 20, 1000);
    . . . . . .
}
```

생성자 내에서는 final 멤버 변수의 초기화가 가능하다!

■ 디폴트 생성자(Default Constructor)


```
FruitSeller seller = new FruitSeller();
FruitBuyer buyer = new FruitBuyer();
```

호출되는 디폴트 생성자

```
public FruitSeller()
{
    // 텅 비어있다!
}
```

생성자를 정의하지 않았을 때에만 삽입!

생성자가 없어도 인스턴스 생성이 가능한 이유는 자동으로 삽입되는 디폴트 생성자에 있다.

나는 정말 JAVA를 공부한 적이 없다고

07-3.
자바의 이름 규칙(Naming Rule)

■ 클래스, 메소드, 상수의 이름 규칙

나 정말 JAVA를 공부한 적 없는데요

클래스 이름

- class MyClass
- class ProgrammingBook

Camel Case

- 대문자로 시작
- 둘 이상의 단어가 묶여서 하나의 이름 구성 시, 새 단어는 대문자로 시작

인스턴스 변수, 메소드 이름

- int addYourMoney(int money)
- int yourAge

변형된 Camel Case

- 소문자로 시작
- 둘 이상의 단어가 묶여서 하나의 이름 구성 시, 새 단어는 대문자로 시작

상수 이름

- final int COLOR=7
- final int COLOR_RAINBOW=7

- 전부 대문자로 표현
- 둘 이상의 단어가 묶여서 하나의 이름 구성 시, 두 단어 사이에 _ 삽입