Chapter 10. 클래스 변수와 클래스 메소드

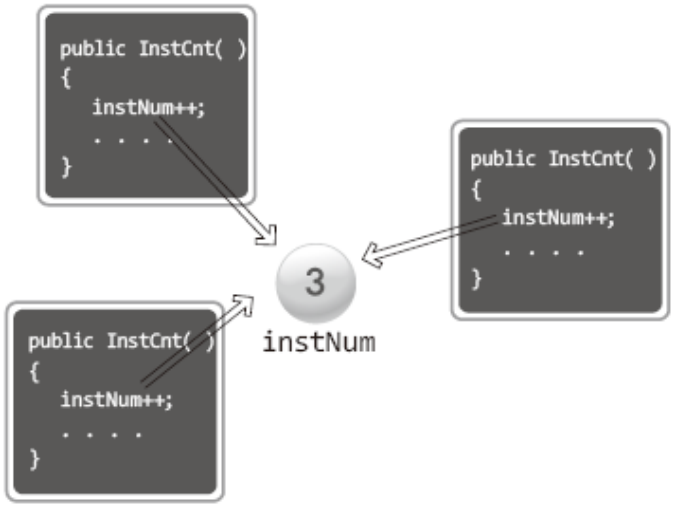
나는 정말 JAVA를 공부한 적이 없다고

10-1. static 변수(클래스 변수)

static 변수

- 인스턴스의 생성과 상관없이 초기화되는 변수
- 하나만 선언되는 변수
- public으로 선언되면 누구나 어디서든 접근 가능!

```
class InstCnt
{
    static int instNum=0;
    public InstCnt()
    {
        instNum++;
        System.out.println("인스턴스 생성 : "+instNum);
    }
}
class ClassVar
{
    public static void main(String[] args)
    {
        InstCnt cnt1=new InstCnt();
        InstCnt cnt2=new InstCnt();
        InstCnt cnt3=new InstCnt();
    }
}
```



실행결과

```
인스턴스 생성 : 1
인스턴스 생성 : 2
인스턴스 생성 : 3
```

실행결과를 통해서 변수가 공유되고 있음을 확인할 수 있다.

static 변수의 접근방법

어떠한 형태로 접근을 하건, 접근의 내용에는 차이가 없다. 다만 접근하는 위치에 따라서 접근의 형태가 달라질 수 있다.

```
class AccessWay
{
    static int num=0;

    AccessWay()
    {
        incrCnt();
    }
    public void incrCnt() { num++; }
}

class ClassVarAccess
{
    public static void main(String[] args)
    {
        AccessWay way=new AccessWay();
        way.num++;
        AccessWay.num++;
        System.out.println("num="+AccessWay.num);
    }
}
```

클래스 내부 접근방법

인스턴스의 이름을 이용한 접근방법

클래스의 이름을 이용한 접근방법

static 변수의 초기화 시점

나는 정말 JAVA를 공부한 적이 없다

- JVM은 실행과정에서 필요한 클래스의 정보를 메모리에 로딩한다.
- 바로 이 Loading 시점에서 static 변수가 초기화 된다.

```
class InstCnt
{
    static int instNum=100;

    public InstCnt()
    {
        instNum++;
        System.out.println("인스턴스 생성 : "+instNum);
    }
}
class StaticValNoInst
{
    public static void main(String[] args)
    {
        InstCnt.instNum-=15;
        System.out.println(InstCnt.instNum);
    }
}
```

이 예제에서는 한번의 인스턴스 생성이 진행되지 않았다. 즉, 인스턴스 생성과 static 변수와는 아무런 상관이 없다.

■ static 변수의 활용의 예

나는 정말 JAVA를 공부한 적이 없다

- 동일한 클래스의 인스턴스 사이에서의 데이터 공유가 필요할 때 static 변수는 유용하게 활용된다.
- 클래스 내부, 또는 외부에서 참조의 목적으로 선언된 변수는 static final로 선언한다.

```
class Circle
{
    static final double PI=3.1415;
    private double radius;

    public Circle(double rad){ radius=rad; }
    public void showPerimeter()    // 둘레 출력
    {
        double peri=(radius*2)*PI;
        System.out.println("둘레 : "+peri);
    }
    public void showArea()     // 넓이 출력
    {
        double area=(radius*radius)*PI;
        System.out.println("넓이 : "+area);
    }
}
```

PI 값은 인스턴스 별로 독립적으로 유지할 필요가 없다. 그리고 그 값의 변경도 불필요하다는 특성이 있다.

```
class ClassVarUse
{
    public static void main(String[] args)
    {
        Circle cl=new Circle(1.2);
        cl.showPerimeter();
        cl.showArea();
    }
}
```

나는 정말 JAVA를 공부한 적이 없다고

10-2. static 메소드(클래스 메소드)

static 메소드의 정의와 호출

- static 메소드의 기본적인 특성과 접근방법은 static 변수와 동일하다.

```
class NumberPrinter
{
    public static void showInt(int n){ System.out.println(n); }
    public static void showDouble(double n) { System.out.println(n); }
}

class ClassMethod
{
    public static void main(String[] args)
    {
        NumberPrinter.showInt(20);
        NumberPrinter np=new NumberPrinter();
        np.showDouble(3.15);
    }
}
```

클래스의 이름을 통한 호출

인스턴스의 이름을 통한 호출

이 예제에서 보이듯이 인스턴스를 생성하지 않아도 static 메소드는 호출 가능하다.

static 메소드의 활용의 예


```
class SimpleMath
{
    public static final double PI=3.1415;
    public static double add(double n1, double n2) { return n1+n2; }
    public static double min(double n1, double n2) { return n1-n2; }
    public static double mul(double n1, double n2) { return n1*n2; }
}
```

```
class AreaMath
{
    public static double calCircleArea(double rad)
    {
        double result=SimpleMath.mul(rad, rad);
        result=SimpleMath.mul(result, SimpleMath.PI);
        return result;
    }
    public static double calRectangleArea(double width, double height)
    {
        return SimpleMath.mul(width, height);
    }
}
```

위 클래스들의 특징은 인스턴스의 생성이 의미가 없다는 것이다!

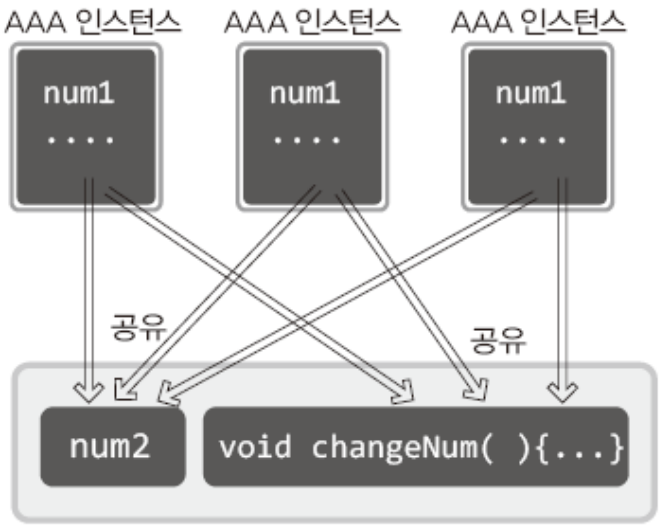
달리 말하면 위의 두 클래스에는 인스턴스 변수가 포함되어 있지 않다!

■ static 메소드의 인스턴스 접근? 말이 안 된다!

나 정말 JAVA를 공부한 적 없는데

• static 메소드는 인스턴스에 속하지 않기 때문에 인스턴스 멤버에 접근이 불가능하다!

```
class AAA
{
    int num1;
    static int num2;
    static void changeNum()
    {
        num1++;      // 문제 됨!
        num2++;      // 문제 안됨!
    }
    . . . .
}
```



AAA 인스턴스에 의해 공유되는 static멤버

static 변수 num2의 증가를 어떤 인스턴스를 대상으로 진행해야 하겠는가?

때문에 이는 논리적으로 인식될 수 없는 코드이다.

나는 정말 JAVA를 공부한 적이 없다고

10-3. System.out.println & public static void main

System하고 out이 무엇이냐?

- System : java.lang 패키지에 묶여있는 클래스의 이름
  - import java.lang.*; 자동 삽입되므로 System이란 이름을 직접쓸 수 있음.
- out : staitc 변수이되 인스턴스를 참조하는 참조변수
  - PrintStream이라는 클래스의 참조변수

```
public class System
{
    public static final PrintStream out;
    . . . .
}
```

static final로 선언되었으니, 인스턴스의 생성 없이 system.out 이라는 이름으로 접근 가능하다!

System.out.println()은 Sytem 클래스의 멤버 out이 참조하는 인스턴스의 println 메소드를 호출하는 문장이다!

public static void main

```
class Employer     /* 고용주 */
{
    private int myMoney;
    public Employer(int money)
    {
        myMoney=money;
    }
    public void payForWork(Employee emp, int money)
    {
        if(myMoney<money)
            return;
        emp.earnMoney(money);
        myMoney-=money;
    }
    public void showMyMoney()
    {
        System.out.println(myMoney);
    }
}
```

```
class Employee     /* 고용인 */
{
    private int myMoney;
    public Employee(int money)
    {
        myMoney=money;
    }
    public void earnMoney(int money)
    {
        myMoney+=money;
    }
    public void showMyMoney()
    {
        System.out.println(myMoney);
    }
}
```

```
Employer emr=new Employer(3000);
Employee eme=new Employee(0);

emr.payForWork(eme, 1000);
emr.showMyMoney();
eme.showMyMoney();
```

이 문장들을 main 메소드로 묶어서 어디에 넣겠는가?

main 메소드는 static의 형태로 정의하기로 약속했으므로, 어디에 존재하든 상관없다!

다만 실행하는 방식에만 차이가 있을 뿐이다!

■ main 메소드의 위치와 인스턴스의 관계

나 정말 JAVA를 공부한 적이 없는데요

```
C:\JavaStudy>java Employer
```

Employer 클래스에 main 메소드가 정의되었을 때 실행방법

```
C:\JavaStudy>java Employee
```

Employee 클래스에 main 메소드가 정의되었을 때 실행방법

```
class AAA
{
    public static void makeAAA()
    {
        AAA a1 = new AAA();
        . . . .
    }
    . . . .
}
```

이렇듯 모든 메소드는 자신이 속한 클래스의 인스턴스 생성이 가능! 이는 main 메소드도 마찬가지!

따라서 main 메소드는 어디든 존재할 수 있다.