

JAVA 7강 멤버변수와 메모리

강사 박주병

1. 변수의 종류

java에서 변수를 선언하는 위치와 특정키워드에 따라 다른 특성을 가진다.

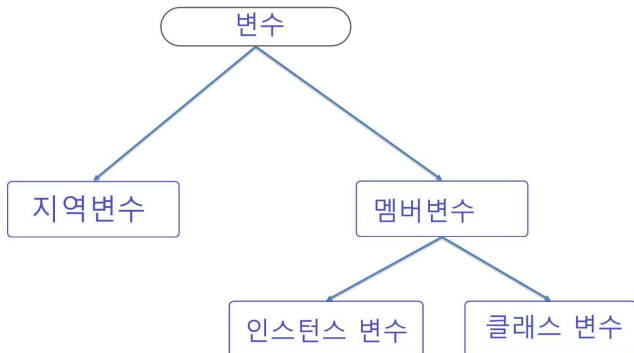


그림 1

```
class test{  
    static int classValue;  → 멤버변수 -클래스 변수  
    int memberValue;  → 멤버변수,인스턴스 변수  
    void method1(int parameter) → 지역변수  
    {  
        int localValue;  → 지역변수  
        for(int i=0; i <10;i++)  
        {  
            → 지역변수  
        }  
    }  
}
```

그림 2

변수는 선언하는 위치에 따라 지역변수, 멤버변수로 나눌수 있으며 멤버변수는 다시 static이라는 키워드를 사용 하느냐 안하느냐에 따라 인스턴스 변수와 클래스변수로 나뉘어진다.

가. 지역변수

1) 지역변수의 특징

- 메서드 범위 내에서 생성된 변수
- 초기화 하지 않으면 사용 할 수 없다.
- 메서드 실행 시 생성되며 메서드 영역이 끝나면 삭제된다.
- 메모리의 스택영역에 저장된다.

```
public class Test  
{  
    void test()  
    {  
        int age = 30;  
    }  
    void print()  
    {  
        System.out.println(age);  
    }  
}
```

age변수는 test()메서드 내에서만 사용가능

그림 3

그림3에서 test메서드 내부에서 age라는 변수가 선언되었다. age변수는 메서드내에서 선언되었기에 지역변수이며 해당 메서드 내부에서만 사용가능 하므로 print()메서드에서는 사용 할 수 없다.

```
public class Test  
{  
    void test()  
    {  
        for(int i=0;i<5;i++)  
        {  
            System.out.println(i);  
        }  
        System.out.println(i);  
    }  
    void print()  
    {  
    }  
}
```

for문 내에서 생성된 지역변수
이므로 for 영역을 벗어나면
사라진다.

그림 4

ex)if문 내부에서 만들어졌으면 if문을 벗어나면 변수는 삭제된다.

그림4 에서는 for 반복문에서 I 변수를 선언한 것을 볼 수 있다.

I변수역시 지역변수이며 반복문 내에서 생성되었기 때문에 반복문 내부에서만 사용할수 있다. 따라서 반복문의 블록을 벗어나면 해당 변수는 삭제되므로 사용 할 수 없다.

이렇듯 메서드 내부에서도 또다시 블록 안에서 만들어진 변수들은 해당 블록에서만 사용가능하다.

```

public class MyMath {

    int add(int a ,int b)
    {
        int result= a+b;

        return result;
    }

    void test()
    {
        System.out.println(a);
    }
}

```

매개변수 역시 지역변수이므로
범위를 벗어나서 사용 불가

그림 5

메서드의 매개변수 역시 지역변수 이므로 그림5 의 add메서드의 매개변수인 a,b변수는 add메서드 내부에서만 사용가능하다 test() 메서드 내부에서 사용할수 없는 것을 볼수 있다.

나. 멤버변수-인스턴스변수

1) 인스턴스변수의 특징

- 클래스 범위 내에서 생성된 변수
- 클래스가 객체화 될 때 객체마다 별도로 생성된다.
- 메모리의 힙 영역에 생성된다.

```

public class Test
{
    int iv=50;

    void test()
    {
        System.out.println(iv);
    }

    void print()
    {
        System.out.println(iv);
    }
}

```

클래스 내 어디서든 사용가능하다

그림 6

그림6에서 Test클래스 내부에서 iv변수가 만들어졌다. 클래스내부이면서 메서드의 밖 영역에 선언된 변수를 멤버변수라고 부른다.

인스턴스변수는 지역변수와 다르게 메서드의 밖에서 선언되었으므로 클래스 내부 어디서든 사용가능하다.

iv 변수는 test() print() 둘 다 자유롭게 사용가능한 것을 볼 수 있다.

```

public class Person
{
    String name;
    String RRN;
}

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Person p1 = new Person();
        Person p2 = new Person();
        Person p3 = new Person();

        p1.name = "홍길동";
        p2.name = "김길동";
        p3.name = "박길동";

        System.out.println(p1.name);
        System.out.println(p2.name);
        System.out.println(p3.name);
    }
}

```

객체마다 별도의 공간을 사용한다.

인스턴스변수는 클래스 밖에서도 사용가능하다. 다만 클래스 밖에서는 클래스를 이용해 객체를 생성후 사용 해야 한다.

그림7을 보면 Person 객체를 3개 만든 후 각각의 참조변수를 이용해 name 변수를 초기화 하고 있다. 인스턴스변수는 객체마다 각자 생성되어 값을 따로 가지고 있다.

p1,p2,p3 객체 모두 name의 값을 따로 가지고 있는 것을 볼수 있다.

그림 7

2) 인스턴스 변수와 지역변수의 이름이 같다면?

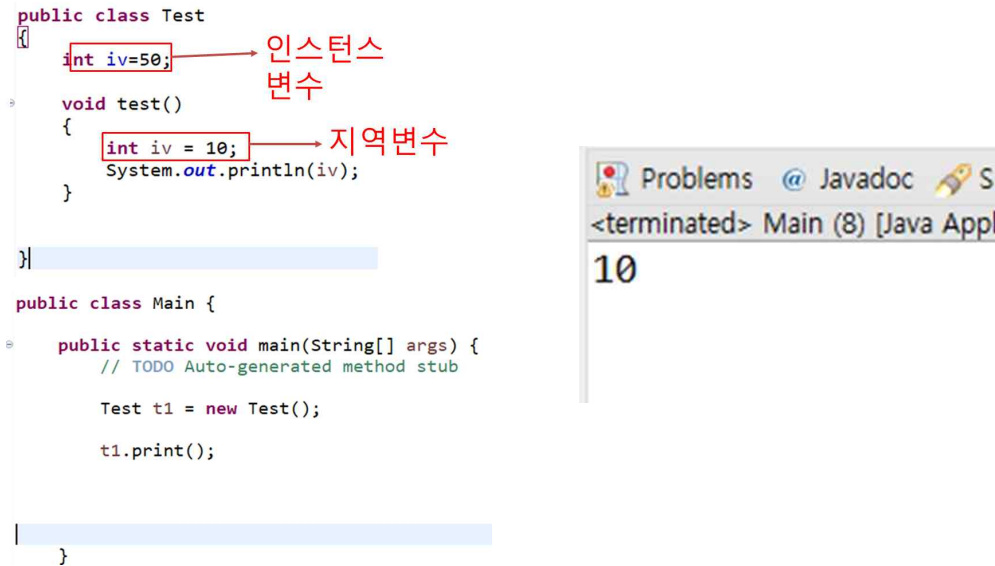


그림 8

그림8의 Test 클래스를 보면 iv로 인스턴스 변수를 선언하고 test() 메서드 내부에서 지역변수로 iv변수를 선언하였다. 이렇게 인스턴스변수와 지역변수의 이름이 동일하게 생성이 가능하다.

그렇다면 test()메서드 내에서 iv변수의 값을 출력해보면 어느값이 출력이 될까?

정답은 10이 출력된다. 인스턴스변수와 지역변수의 이름이 같을 경우 지역변수가 우선시 되어 진다. 물론 그림8에서 test()메서드 이외에 다른 메서드에서는 iv 지역변수가 없기 때문에 멤버변수의 값이 나올 것이다.

3) 인스턴스변수의 초기화

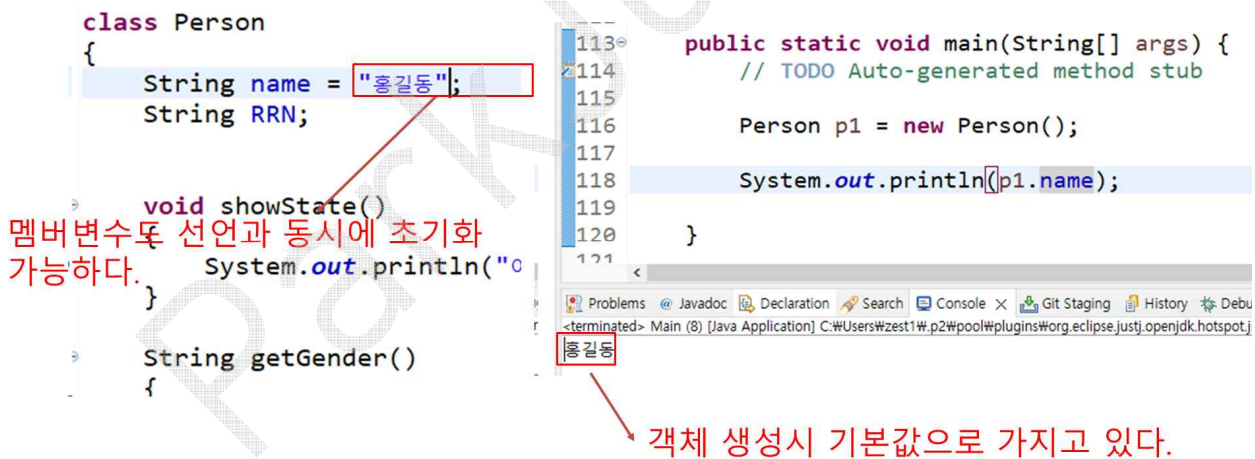


그림 10

인스턴스변수 역시 일반적인 변수와 마찬가지로 변수 선언과 동시에 초기화를 할 수 있다. 그림10의 Person 클래스의 name 변수는 변수 선언과 동시에 홍길동 이라는 문자열을 저장하고 있다.

이렇게 될 경우 인스턴스변수가 생성되는 시기는 new Person()을 통해 객체를 생성할 때 이다. 그리고 이와 동시에 name의 값은 홍길동이 된다. 따라서 Person 객체를 생성하면 무조건 기본값으로 모든 객체들이 name은 홍길동으로 초기화가 되는 것이다.

4) 인스턴스변수와 멤버변수의 초기화 차이점

```
class test
{
    int v;
    Person p;

    void fnc()
    {
        int localV;
        Person localP;

        System.out.println(v);
        System.out.println(p);

        System.out.println(localP);
        System.out.println(localV);
    }
}
```

멤버 변수가 기본형이라면 기본값이 들어가 있다.

멤버 변수가 참조형이라면 null이 기본이다.

지역변수는 초기화하지 않으면 사용할 수 없다.

<terminated>
0
null

그림 11

인스턴스변수의 경우 별도의 초기화를 하지 않고 사용하면 해당변수의 타입에 알맞은 적절한 기본값이 들어가 있다. 참고로 변수p의 데이터타입은 Person이다. 이는 참조타입이며 참조형변수의 경우 객체의 주소를 가지고 있다. 그러나 객체의 주소값을 넣어주지 않으면 기본은 NULL이 들어가 있다.

반면에 localV, localP와 같은 지역변수의 경우 초기화를 하지 않고 사용하려면 컴파일 에러가 발생한다. 지역변수는 반드시 초기화를 하고 사용해야 한다.

이러한 특징은 서로 저장되는 메모리영역이 달라서 발생하는 문제로 이에 대해서는 자바교재에 적절하지 않으므로 다루지 않도록 한다.

다. String 클래스

이제 3가지 변수 종류중 2가지를 보았다. 이제 남은건 클래스변수(static변수)인데 이것을 다루기전에 문자열을 위한 클래스인 String클래스에 대해 먼저 배우고 넘어가도록 하자.

원래 프로그래밍에서 문자열을 다루는 것은 char배열을 이용해야 한다. 그러나 이러한것들을 직접 구현하기에는 난이도가 높고 아주 기본적인 기능들이라서 웬만하면 언어자체적으로 지원을 해준다. 그것이 바로 String 클래스 이다. 지금까지 문자열을 사용할 때 String str = "안녕하세요"; 와 같이 작성 하였을 것이다. 이는 사실 int, char 와 같은 기본형 과는 다르게 String역시도 클래스 이며 str변수는 String객체의 주소를 가지는 참조변수이다.

그래서 원래 정석적인 String 사용방법은 아래와 같다.

```
String str = new String();
str = "안녕하세요";
```

하지만 위와 같이 사용하기에는 사용빈도가 다른 객체에 비해 높아 기본형 변수처럼 쓸수 있듯이 기능을 제공해주는 것이다. ex) String str = "안녕하세요";

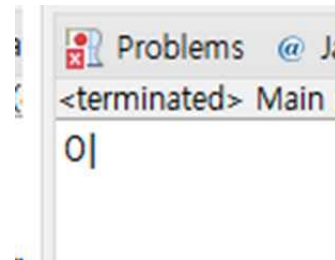
String 역시 클래스 이고 str은 String 객체의 주소를 가지고 있다고 하였다. 따라서 str변수 역시 마침표를 통해 객체 내부에 있는 변수와 메서드를 사용할수 있다. 지금부터 문자열을 다루기 위한 유용한 String 클래스의 메서드들을 보도록 하자.

1) charAt()

문자열 중에 특정위치의 문자 하나를 char 타입으로 반환하는 메서드이다.

```
String str = "문자열이 들어갈수 있습니다.";
char ch = str.charAt(3);
System.out.println(ch);
```

0부터 시작하므로 4번째



매개변수로 int 타입의 정수를 하나 받으며 추출하고자 하는 문자의 위치를 나타낸다. 배열처럼 0부터 시작을 하며 위의 예시처럼 3을 입력하면 4번째 문자를 반환한다. 스페이스 또한 문자 이므로 사용시 스페이스의 개수까지 고려해서 위치를 지정해야 한다.

2) substring()

특정 범위만큼 문자열을 잘라서 String 타입으로 반환한다.

```
String str = "문자열이 들어갈수 있습니다.";
String str1 = str.substring(0,3);
String str2 = str.substring(5,7);
System.out.println(str1);
System.out.println(str2);
```

0번째 위치 부터 2번째 위치까지

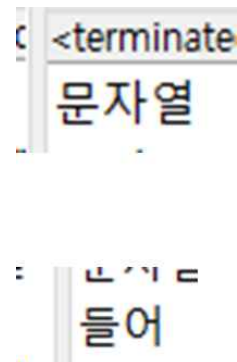


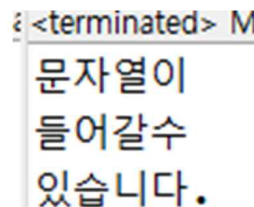
그림 14

0,3을 매개변수로 넘겨주면 0번째 위치부터 2번째 위치까지의 문자열을 잘라서 반환한다. 이때 str변수가 가지고 있는 문자열 원본은 그대로 이고 해당 범위의 문자열을 반환한다.

3) split()

문자열을 특정 구분자 단위로 잘라서 String배열로 반환한다.이 역시 원본은 훼손하지 않은채 반환한다.

```
String str = "문자열이 들어갈수 있습니다.";
String[] temp = str.split(" ");
System.out.println(temp[0]);
System.out.println(temp[1]);
System.out.println(temp[2]);
```



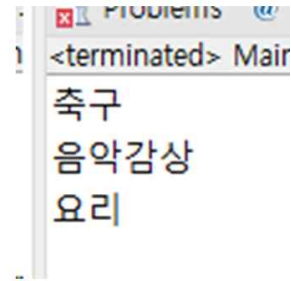
split() 의 매개변수로 공백 하나를 넘겨주면 문자열에서 공백을 기준으로 잘라내어 길이3의 String 배열을 반환한다.

매개변수로는 문자열을 넘겨 줄 수 있기에 2글자 이상의 구분자를 지정하는것도 가능하다.


```
String str = "축구,음악감상,요리,캠핑";

String[] hobbies = str.split(",");

System.out.println(hobbies[0]);
System.out.println(hobbies[1]);
System.out.println(hobbies[2]);
```



4) equals()

문자열을 비교 할 때 사용하며 매개변수로 비교하고싶은 문자열을 넘겨주면된다. 매개변수로 넘겨받은 문자와 일치한 다면 true를 반환하고 일치하지 않으면 false를 반환한다.

```
String temp = "안녕";

if(temp.equals("안녕"))
    System.out.println("일치합니다");
```

그림 18

참고로 문자열은 == 연산자를 이용해 비교하지 말아야 한다. String타입은 기본형이 아니므로 엄밀히 따지면 String타입의 변수는 문자열의 주소값을 가지고 있다. ==연산자를 이용해 문자열을 비교 한다면 사실 변수에 들어가 있는 주소 값을 비교하여 주소가 일치해야지만 true가 나오게 된다. 직접 해봤을때 정상적으로 true가 나올때도 있고 아닐 경우도 있을 것이다. 이에 대해서는 자바2에서 자세히 다루도록 하고 지금은 두 개의 문자열이 같은지 비교 할때는 반드시 equals()를 활용해야 한다는것만 기억하자.

라. 멤버변수-클래스변수

지금까지 변수를 선언하는 위치에 따라 메서드내부에서 선언하면 지역변수 클래스 내부에서 선언하면 멤버변수라고 부르며 그중에 특별한 키워드를 사용하지 않고 그냥 선언한 변수를 인스턴스변수 라고 배웠다.

클래스 변수는 바로 클래스 내부에 선언하여 멤버변수이기는 하지만 static이라는 특별한 키워드를 사용하여 만드는 특별한 변수이다.

1) 클래스 변수의 특징

- static 키워드를 사용하여 멤버변수 영역에 선언한다.
- 프로그램이 실행될 때 메모리에 올라간다(객체 생성 이전단계) 그러므로 객체 생성없이 사용가능하다.
- 객체들끼리 값을 공유하는 변수이다.

우선은 클래스 변수가 가지는 특징들을 먼저 살펴보고 이러한 특징이 어떻게 해서 발생되는지 원인에 대해서 보도록 하자.

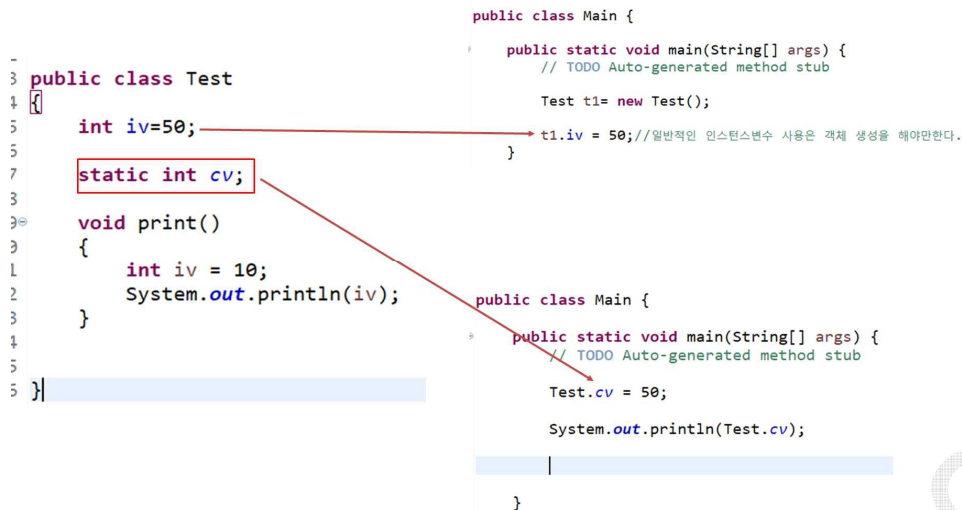


그림 19

클래스 변수는 static 키워드를 사용하여 생성한다. 그리고 인스턴스변수와는 다르게 객체를 생성하지 않아도 바로 사용할 수 있다. 위의 예제에서 cv변수를 사용 할 때 클래스명.cv 로 접근하여 값을 초기화 하여 출력하는 것을 볼 수 있다. 객체를 생성하지 않고 쓰기 때문에 클래스이름으로 접근하여 사용하는 것이다.

```
public class Main {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Test.cv = 50;
        System.out.println(Test.cv);
        Test t1 = new Test();
        System.out.println(t1.cv);
    }
}
```

객체를 통해 접근해도 된다.(권장안함)

그림 20

그림20에서처럼 객체를 생성한후 참조변수를 통해 cv변수를 사용해도 가능한 하다. 하지만 클래스변수를 사용한다는 것은 객체 생성없이 객체와는 무관하게 사용한다는 의미이므로 의미상 적절하지 못하다. 따라서 위와 같은 방식은 권장하지 않는다.

```

public class Test
{
    int iv;
    static int cv;
}

Test t1 = new Test();
Test t2 = new Test();

t1.iv = 10;
t2.iv = 20;
System.out.println(t1.iv);
System.out.println(t2.iv);

t1.cv = 10;
t2.cv = 20;
System.out.println(t1.cv);
System.out.println(t2.cv);

```

그림 21

그림21을 보면 cv변수를 사용 할 때 t1 참조변수를 통해 사용하고 있다. 앞서 이러한 방식은 권장하지 않는다고 했지만 클래스변수가 객체들끼리 공유하는 변수라는걸 명확히 보여주기 위해 사용하고 있다. t1,t2 변수를 만들어 두 개의 객체를 생성하였다. iv 인스턴스변수의 경우 각각 10과 20을 초기화 하고 값을 출력하니 t1.iv 와 t2.iv의 값이 다르게 나오는걸 확인 할수 있다.

그러나 t1.cv 와 t2.cv를 보면 분명 다른값을 넣었음에도 불구하고 둘다 20이 출력 되는 것을 볼수 있다. 이는 cv 변수는 클래스 변수이고 클래스변수는 객체와 상관없이 값을 공유하기 때문이다. 정확히 말하면 공유하는 것이 아닌 클래스 단위로 단 하나의 객체만 존재 하는것이고 t1.cv와 t2.cv는 완전히 같은 변수인 것이다. 그래서 t2.cv를 통해 값을 20으로 초기화 한다는 것은 t1.cv의 값을 20으로 변경하는것과 똑같은 것이다.

이렇게 클래스변수는 인스턴스변수와 다르게 객체 생성없이 그리고 객체들끼리 공유 할 수 있는 변수이다. 이러한 특징이 왜 발생하는지에 대해 알기 위해서는 메모리 구조를 알아야 한다.

마. 자바의 메모리 구조

자바의 메모리 구조는 크게 3가지로 분류 된다.

메서드	클래스 정보, static
스택	지역변수, 연산 중간결과
힙	객체, 인스턴스 변수

클래스의 정보들이 각메모리 영역에 나누어서 관리된다.

1) 메서드

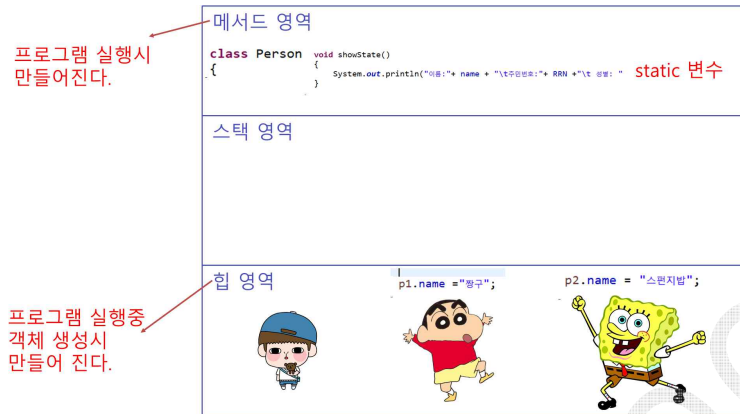
클래스에 대한 정보(소스코드)와 static 키워드로 생성된 변수나 메서드 등이 저장된다. 프로그램 실행시 단하나의 정보만 기록이 된다. 예를 들어 객체는 여러개이지만 객체를 만들기 위한 클래스에 대한 정보는 1개면 된다는 것이다. 1개의 설계도만 있어도 그 설계도로 여러개의물건을 만들 수 있지 않은가?

2) 스택

메서드의 실행과 밀접한 관련이 있다. 메서드가 실행될 때 메서드에 대한 정보가 스택에 올라가며 지역변수역시 메서드 실행시 생성되므로 스택에 저장된다. 그리고 계산에 필요한 임시데이터들도 스택에 저장된다.

3) 힙

메모리를 소스코드에서 동적으로 할당할수 있는 공간이다. new 키워드가 바로 힙 영역을 할당하는 역할을 한다. 객체를 생성할 때 new 클래스명();을 호출하는데 이때 힙 영역에 해당 객체에 대한 정보가 올라간다.



4) 스코프와 라이프사이클

가) 스코프

스코프란 접근 할 수 있는 범위가 어디까지인가를 나타내는 말이다. 지역변수의 경우 해당 지역 내에서만 접근 할수 있다. 다른 메서드에서는 사용이 불가능하다. 물론 애초에 변수자체가 삭제되기 때문에 사용이 불가능한것도 맞다. 멤버변수의 경우에는 클래스 내부일 경우 어디서든 접근이 가능하다.

나) 라이프사이클

라이프사이클이란 데이터가 언제 생성되고 언제 소멸되는지를 나타내는 용어이다. 지역변수의 경우 해당 메서드가 실행될 때 생성되며 메서드 수행이 끝나면 같이 사라진다.

멤버변수 중에 인스턴스변수의 경우 객체가 생성될 때 변수도 같이 만들어지며 해당 객체를 가리키는 참조변수가 하나도 없어 접근이 불가능할 때 가비지컬렉션이 객체를 정리하게 되는데 이때 같이 소멸된다. 즉 객체의 라이프사이클과 동일하다.

```
class test{
    int iv= 10;
    static int cv =1;
    void a()
    {
        int i =5;
        System.out.println(i);
        System.out.println(cv);
        System.out.println(iv);
    }
    void b()
    {
        System.out.println(i);
        System.out.println(cv);
        System.out.println(iv);
    }
}
```

변수종류	스코프	라이프사이클
지역변수	해당지역	생성: 해당 영역이 수행될때 소멸: 해당 영역수행이 끝나고 즉시
인스턴스변수	접근제어에 따라 다름 최소 클래스 내부	생성: 객체 생성시 소멸: 객체 소멸시
클래스변수	접근제어에 따라 다름 최소 클래스 내부	생성: 프로그램 실행시 소멸: 프로그램 종료시

클래스변수의 경우 프로그램이 실행 되는시점에 이미 메모리에 올라가서 사용이 가능하다. 앞에서 클래스 변수는 객체 생성없이 사용가능한 이유가 이 때문이다. 그리고 소멸시점은 프로그램이 종료되는 시점에 같이 사라진다.

즉 인스턴스변수보다 먼저 생성되고 프로그램이 실행되는동안에는 계속 남아있게 된다.

5) 클래스 메서드

static 키워드를 변수를 선언할 때 사용하여 클래스변수를 만들었다. static은 변수뿐만 아니라 메서드에도 사용 할 수 있는데 이러한 메서드를 클래스메서드(static메서드) 라고 부른다. 클래스메서드의 특징은 클래스변수와 같다.

바로 객체 생성없이 사용할수 있다는 특징이다.

```
class Marine {
    int hp=40;
    static int shootingRange=6;

    void showState()
    {
        System.out.println("체력: "+hp+"\t 공격력: "+shootingRange);
    }

    static void test()
    {
        System.out.println("클래스 메서드 호출!");
    }
}
```

그림 25

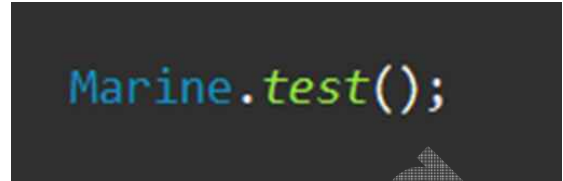


그림 26

그림25의 marine 클래스의 test() 메서드는 클래스 메서드이다.

이 메서드를 사용 할 때는 그림26처럼 클래스이름.메서드이름() 으로 접근하여 사용 할수 있다. 즉 객체 생성없이 바로 호출이 가능하다.

```
class Marine {
    int hp=40;
    static int shootingRange=6;

    void showState()
    {
        System.out.println("체력: "+hp+"\t 공격력: "+shootingRange);
    }

    static void test()
    {
        System.out.println("체력: "+ hp);
    }
}
```

그림 27

그림27에서 hp 부분이 왜 오류가 발생하는지 이유를 생각해보자. hp변수는 인스턴스 변수이다. 그리고 test() 메서드는 클래스메서드이다. test()메서드는 객체 생성없이 사용할 수가 있다. 만약 marine.test() 로 호출을 하였을 때 hp라는 인스턴스변수가 생성되었을거라고 보장이 되는가? 만약 객체생성을 한적이 없다면 hp변수는 아직 존재하지 않는다. 물론 객체 생성이후에 test()를 호출한다면 문제가 없겠지만 변수가 없을 가능성을 내포하고 있다. 따라서 이런 경우 컴파일러를 허용하지 않아 에러가 발생할 가능성을 아예 없앤다.

따라서 그림28처럼 클래스 메서드와 인스턴스메서드 내부에서 사용할수 있는 메서드나 변수들의 종류가 나뉘게 된다. 이는 모두 생성시점이 다름에 따라 발생하는 현상이므로 무작정 외우기 보단 해당 메서드가 실행될 때 필요한 데이터가 생성이 되었는가를 따져보며 생각해보자.

클래스 메서드

- 객체 생성없이 바로 사용
- 인스턴스 멤버변수 사용 불가
- 인스턴스 멤버메서드 사용불가
-

```
Math.random();
```

그림 28

인스턴스 메서드

- 객체를 생성해야 사용가능
- 클래스, 인스턴스, 지역 변수 다 가능
- 클래스, 인스턴스 매서드 둘다 사용 가능

바. 참조형 타입의 매개변수

메서드를 사용 할 때 매개변수로 지금까지는 기본형 타입의 변수들을 받아서 사용해왔다. 하지만 매개변수로 참조형 타입을 받을 수도 있다. 즉 객체의 주소를 넘겨받아 객체를 이용할수도 있다는 것이다.

```
public class Person
{
    String name;
    String RRN;

    int money;

    void send(Person p1)
    {
        //매개변수로 넘어온 객체의 금액을 내가 가진 금액만큼 증가 시킨다.
        p1.money+=money;
        //나의 돈은 0원으로 만든다.
        money=0;
    }
}

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Person p1 = new Person();
        Person p2 = new Person();

        p1.money = 5000;
        p2.money = 10000;

        //p1이 p2에게 돈을 준다.
        p1.send(p2);

        System.out.println(p1.money);
        System.out.println(p2.money);
    }
}
```

Problems @ JavaC
<terminated> Main (8)
0
15000

그림 29

그림29의 Person 클래스를 보면 send() 메서드의 매개변수로 Person 객체를 받도록되어 있다.

main() 메서드 내에서 실행하는 코드를 살펴보면 p1객체의 send를 호출하면서 매개변수로 p2 객체를 넘겨주고 있다. 이러면 send()메서드 내부에서 p2 객체에 대한 멤버변수와 메서드에 접근을 할 수 있다.

send 메서드는 매개변수로 넘겨받은 Person 객체에게 자신이 가진 돈을 모두 주는 기능을 하고 있다.