

능동적 사고 방식의

java

강사 박주병



## Part07 메모리



01 파일분리

02 지역변수, 멤버변수

03 메모리 영역

04 실습 문제



ParkJuByeong

# 01

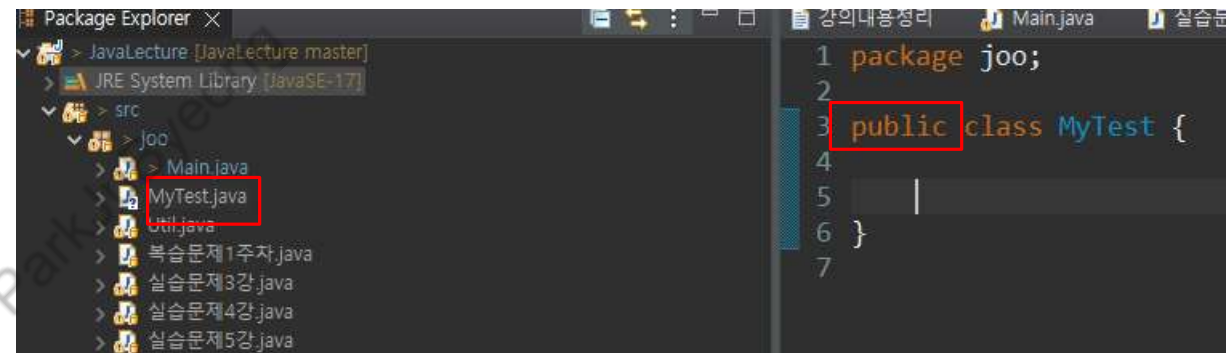
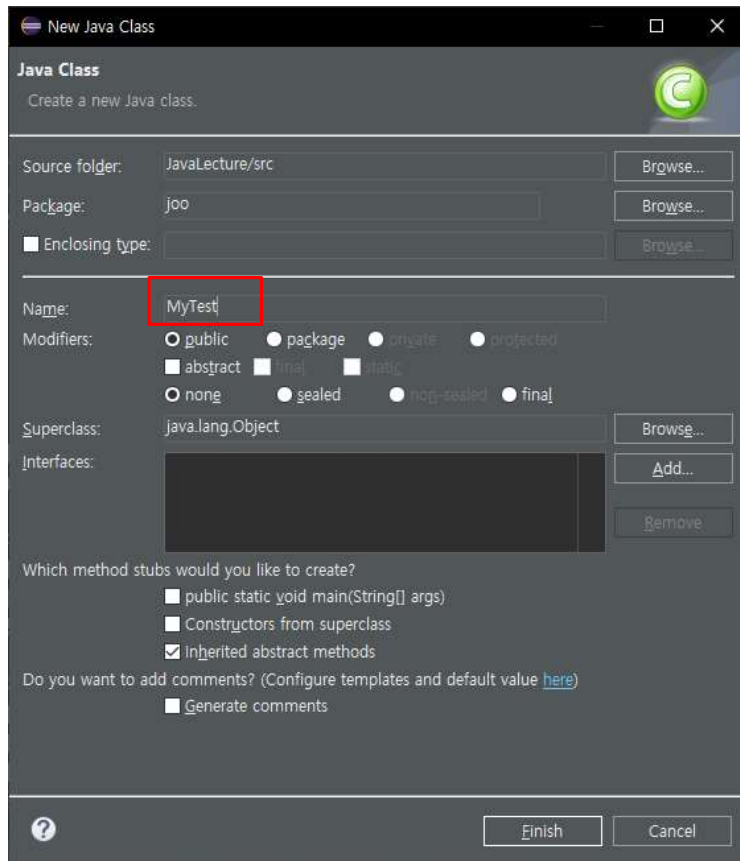
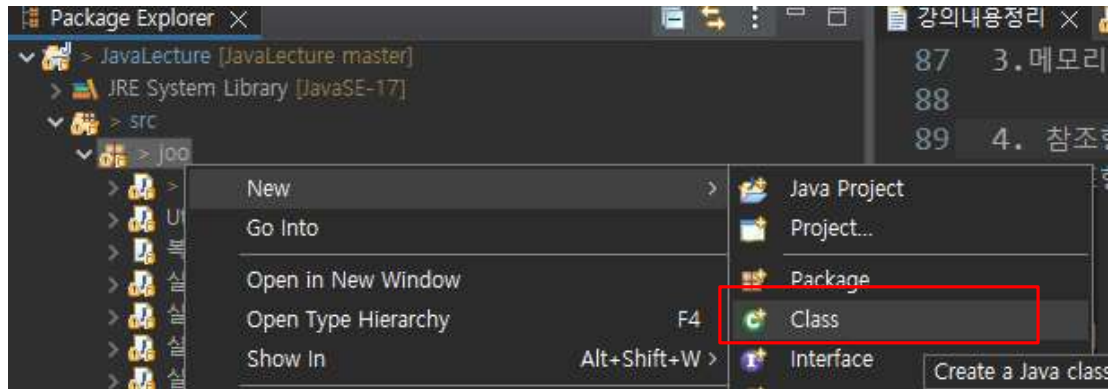
## 파일 분리

메인 클래스와 같은 파일에 있던 클래스를 분리하자



ParkJuByeong

모든 클래스는 패키지 하위에 들어간다.



# — 02

지역변수, 멤버변수

ParkJuByeong

변수

지역변수

인스턴스 변수

클래스 변수



# 지역변수

특정 범위에서만 유지된다.

```
for(int i=0;i<5;i++)  
{  
    System.out.println(i);  
}
```

```
System.out.println(i);
```

지역변수 범위를 벗어나 소멸되었다.

```
public class Main {  
    public static void main(String[] args) {  
        int a=10;  
        for(int i=0;i<5;i++)  
        {  
            System.out.println(i);  
        }  
    }  
    static void test()  
    {  
        System.out.println(a);  
    }  
}
```

main 메서드 내에서 선언된  
지역변수

```
class test{  
    int iv =10;  
  
    void a(int param)  
    {  
        param=5;  
        System.out.println(param);  
    }  
}
```

ParkJuByeong

## 인스턴스 변수(멤버변수)

```
public class Main {  
    int iv=10;  
    public static void main(String[] args) {  
        int a=10;  
        for(int i=0;i<5;i++)  
        {  
            System.out.println(i);  
        }  
    }  
  
    void test()  
    {  
        iv=11;  
    }  
  
    void test1()  
    {  
        System.out.println(iv);  
    }  
}
```

→ 클래스 내 어디서든 사용가능하다

## 인스턴스 변수 와 지역변수 이름이 같다면?

```
class test{  
    int iv= 10;  —————> 인스턴스  
    void a()      변수  
    {  
        int iv =5; —————> 지역변수  
        System.out.println(iv);  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
  
        test t1= new test();  
  
        t1.a();  
    }  
}
```

```
<terminate  
5
```

## 클래스 변수(static 변수)

```
01 public class Main {  
02  
03     int iv=10;  
04  
05     static int cv;  
06  
07     public static void main(String[] args) {  
08  
09         int a=10;  
10  
11         for(int i=0;i<5;i++)  
12         {  
13             System.out.println(i);  
14         }  
15  
16         cv=10;  
17     }  
18 }
```

ParkJuByeong

인스턴스 변수 vs 클래스 변수

ParkJuByeong

03

메모리 영역

## JVM의 메모리 구조

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



```

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

    void attack(Monster target)
    {
        if(shootingRange < target.distance)
            move();
    }

    void move()
    {
    }

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

```

```

Marine m1 = new Marine();
Marine m2 = new Marine();
Marine m3 = new Marine();

m1.hp = 40;
m2.hp = 30;
m3.hp = 50;

m1.shootingRange = 10;
m2.shootingRange = 50;
m3.shootingRange = 1;

m1.showState();
m2.showState();
m3.showState();

```

메서드	shootingRange
스택	
힙	m1, m2, m3

```

<terminated> Main [Java Application]
체력: 40    사정거리: 1
체력: 30    사정거리: 1
체력: 50    사정거리: 1

```

## 스코프와 라이프사이클

```
1 class test{
2     int iv= 10;
3     static int cv =1;
4     void a()
5     {
6         int i =5;
7         System.out.println(i);
8         System.out.println(cv);
9         System.out.println(iv);
10    }
11
12    void b()
13    {
14
15        System.out.println(i);
16        System.out.println(cv);
17        System.out.println(iv);
18    }
19 }
```

변수종류	스코프	라이프사이클
지역변수	해당지역	생성 : 해당 영역이 수행될때 소멸: 해당 영역 수행이 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

```

122 public class Main {
123
124     public static void main(String[] args) {
125
126
127         System.out.println(Marine.shootingRange);
128     }
129
130 }
131
132
133
134 }
135

```

Problems Javadoc Declaration Search Console X Git Staging History  
 <terminated> main [Java Application] C:\Users\Wzest1\p2\pool\plugins\org.eclipse.justj.openjdk.hc  
 6

클래스변수는 객체생성 없이 쓸수 있다.

```

Marine m1 = new Marine();
System.out.println(Marine.shootingRange);

System.out.println(m1.shootingRange);


```

## 클래스 메서드

```
08 class Marine
09 {
10     int hp;
11     static int shootingRange=6;
12
13     void attack(Monster target)
14     {
15         if(shootingRange < target.distance)
16             move();
17     }
18
19     void move()
20     {
21     }
22
23     void showState()
24     {
25         System.out.println("체력: "+hp+"\t 사정거리:"+shootingRange);
26     }
27
28     static void test()
29     {
30         System.out.println("클래스 메서드 호출!");
31     }
32 }
```

```
Marine.test();
```

```
Marine m1 = new Marine();
m1.test();
```



클래스 메서드 VS 인스턴스 메서드



## 클래스 메서드

- 객체 생성없이 바로 사용
- 인스턴스 변수 사용 불가
- 인스턴스 매서드 사용불가

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

## 인스턴스 메서드

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

```
class test
{
    int a = 10;
    static int b = 20;
```

```
test.b = 30; // 객체 생성 없이 사용가능한 static
test.a = 10; // 객체 생성을 안했기에 사용할수 없는 instance 변수
```

클래스 매서드는 객체 생성 없이 사용 가능하다.  
따라서 실행 시 아직 객체가 없을수도 있다.  
변수 or 인스턴스 메서드가 아직 생성 안되었을수 있기에 사용할수 없다.

## 스택 영역의 메서드 호출 순서

```
class test{  
    void first()  
    {  
        second();  
    }  
    void second()  
    {  
        System.out.println("second 호출!");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        test t1= new test();  
        t1.first();  
    }  
}
```

println
second
first
main



```

class test
{
    int a = 10;
    static int b = 20;
    int c = b;
    static int d = a;
    static void staticMethod()
    {
        System.out.println(a);
        System.out.println(b);
    }

    void instanceMethod()
    {
        System.out.println(a);
        System.out.println(b);
    }

    static void staticMethod2()
    {
        staticMethod();
        instanceMethod();
    }

    void instanceMethod2()
    {
        staticMethod();
        instanceMethod();
    }
}

```

가능!

인스턴스 변수는 아직 생성 안됐을수도 있다 ERROR!

ERROR!

OK!

OK!

OK!

OK!

ERROR!

OK!

OK!

ParkJuByeong

## 참조형 리턴의 라이프사이클?

```
class Point
{
    int x;
    int y;
}

class MyTest {
    Point getPoint()
    {
        Point pt = new Point();
        System.out.println(pt);
        return pt;
    }
}
```

```
MyTest t1= new MyTest();

Point pt = t1.getPoint();

System.out.println(pt);
```

Problems Javadoc Declarations

<terminated> main [Java Application] C:\

joo.Point@5c18298f

joo.Point@5c18298f

객체가 생성된 위치가 아니라 해당 객체를 가리키는 참조변수의 생성위치가 라이프사이클을 결정한다

# 실습문제1

1. Marine 클래스의 powerUp, armorUp 메서드를 만들자.(모든 객체가 같은 공격력과 방어력을 가지며 업그레이드시 모든 객체가 다같이 올라가야 한다.)

```
Marine marine1 = new Marine();
Marine marine2 = new Marine();

marine1.powerUp();
marine1.armorUp();

marine1.showState();
marine2.showState();
```

```
<terminated> Main [Java Application] C:\Users\WU
체력: 40    공격력: 5    방어력: 1
체력: 40    공격력: 5    방어력: 1
```

클래스명	Marine	
멤버변수	int hp	40
	int power	0
	int armor	0
메서드	powerUp()	매개변수:없음 내용: 모든 유닛의 power를 1증가 시킨다. 리턴:없음
	armorUp()	매개변수:없음 내용: 모든 유닛의 armor를 1증가 시킨다. 리턴:없음

2. Marine 클래스에 attack 메서드를 구현하자  
매개변수로 받은 Marine 객체의 체력을 감소시킨다.  
공격력은 상대의 방어력 만큼 감소된다.

```
Marine marine1 = new Marine();
Marine marine2 = new Marine();

marine1.powerUp();
marine1.armorUp();

marine1.attack(marine2);

marine1.showState();
marine2.showState();
```

```
<terminated> Main [Java Application] C:\Users\WU
체력: 40    공격력: 5    방어력: 1
체력: 36    공격력: 5    방어력: 1
```

클래스명	Marine	
메서드	attack()	매개변수: Marine 리턴:없음



# THANK YOU

ParkJuByeong



강사 박주병