9.접근제한자와 static

1)접근제한자

제어자	같은 클래스	같은 패키지	자손클래스	전 체
public	○	○	○	○
protected	○	○	○	
(default)	○	○		
private	○			

- 접근제한이란 클래스의 데이터나 메소드에 대해서 다른 클래스로부터 필요에 따라 접근을 제한하는 것을 말한다

public : 외부에서의 접근을 제한하지 않는다. 다른 모든 클래스에서 사용 가능

protected : 해당 클래스와 동일한 패키지에 있거나 상속받은 자손클래스일 경우에 사용 가능

default : 접근제어자를 명시하지 않은 경우의 디폴드 접근제어자.

같은 패키지내의 클래스들은 public 권한을 갖고 접근가능

private : 해당 클래스만 사용가능. 외부객체에서의 접근은 제한된다..

Ex)

```java
public class AccessTest {
    private int privateMember;// 같은 클래스 내에서만
    int defaultMember;// 디폴트값 : 같은 패키지에서만
    protected int protectedMember;// 같은 패키지나 상속받은 하위클래스에서만
    public int publicMember;// 제한없음

    private void privateMethod() {
        System.out.println("private 메소드");
    }
    void defaultMethod() {
        System.out.println("디폴트 메소드");
    }
    protected void protectedMethod() {
        System.out.println("protected 메소드");
    }
    public void publicMethod() {
        System.out.println("public 메소드");
    }
}
```

Ex)접근 제한자 예제 이어서

```java
package com.lec.ex3_access;

public class AccessTestMain {

    public static void main(String[] args) {
        AccessTest obj = new AccessTest();
//      System.out.println(obj.privateMember);  //클래스가 다르기 때문에 출력 x
        System.out.println(obj.defaultMember);
        System.out.println(obj.protectedMember);
        System.out.println(obj.publicMember);

//      obj.privateMethod;  //클래스가 다르기 때문에 출력 x
        obj.publicMethod();
        obj.protectedMethod();
        obj.defaultMethod();
```

Ex)접근 제한자 예제 이어서(다른 패키지에서)

```java
package com.lec.ex4_access;

import com.lec.ex3_access.AccessTest;

public class AccessTestMain {
    public static void main(String[] args) {
        AccessTest obj = new AccessTest();
//      System.out.println(obj.privateMember);//같은 클래스가 아니기때문에 출력x
//      System.out.println(obj.defaultMember);//같은 패키지가 아니기때문에 출력x
//      System.out.println(obj.protectedMember);//상속관계가 아니기때문에 출력x
        System.out.println(obj.publicMember);

//      obj.privateMethod();//같은 클래스가 아니기때문에 출력x
//      obj.defaultMethod();//같은 패키지가 아니기때문에 출력x
//      obj.protectedMethod();//상속관계가 아니기때문에 출력x
        obj.publicMethod()
```

2)static

Static변수(클래스 변수)나 매소드는 각각 생성된 객체가 공유한다.

객체변수(객체속성)는 객체가 생성될 때마다 각 객체 안의 속성 변수들이 생성되지만, 클래스 변수는 클래스로부터 생성된 객체들의 수와 상관없이 하나만 생성된다.

Static은 다음 같은 특징이 있다.

- 클래스에 따른 객체를 생성하지 않고 "클래스명.변수(함수)명"으로 바로 사용할 수 있다.
- Static이 붙은 변수나 메소드는 클래스 생성과 동시에 데이터영역에 생성되있기 때문
- Static 메소드 안에서는 객체변수는 접근 불가하고 같은 static변수만 접근가능하다.

3) 자료형 타입

1. primitive type (원시타입)

총 8가지의 기본형 타입(Primitive type)을 미리 정의하여 제공한다.

기본값이 있기 때문에 Null값이 없으며 .실제 값을 저장하는 공간으로 스택(Stack) 메모리에 저장된다.

2. Reference type (참조형 타입)

기본형 타입을 제외한 타입들이 모두 참조형 타입(Reference type)이며 빈 객체를 의미하는 Null이 존재한다.

타입명은 대문자로 시작하며 값이 저장되어 있는 곳의 주소값을 저장하고 힙(Heap) 메모리에 저장된다.

4) java의 메모리 관리 stack영역 과 heap 영역

1.stack영역(stack memory)

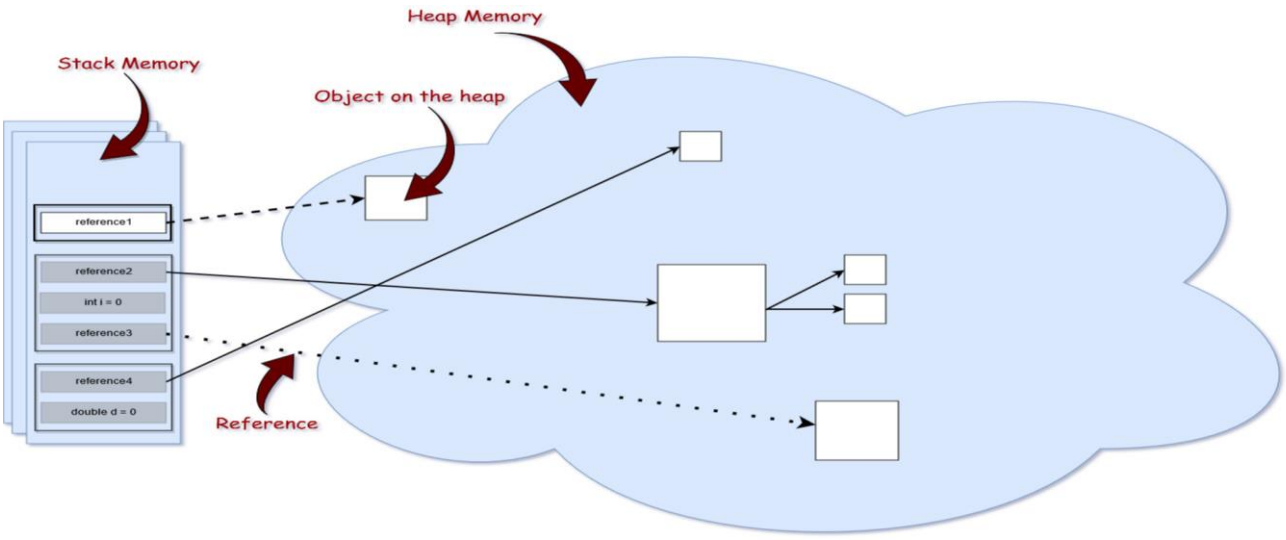
- Heap 영역에 생성된 Object 타입의 데이터의 참조값이 할당되며 원시타입의 데이터도 여기서 값과 함께 할당된다.

  이때 원시타입의 데이터들은 참조값을 저장하는 것이 아니라 실제 값을 stack에 직접 저장한다.

- 휘발성 데이터 있는 곳이다.

2.heap영역(heap memory)

- 모든 Object 타입(Integer, String, ArrayList, ...)은 heap 영역에 생성된다
- 클래스(설계도)들이 모여 있는 공간이다.



5)final

변수나 메서드명 앞에 final을 붙이면 외부에선 해당 변수와 메서드의 접근과 변경을 제한한다.

변수에는 final형식의 상수변수를 많이 사용하나 메소드에선 활용빈도가 적다.

Final이 붙은 메소드는 상속이 불가능하다.

실무에서는 보통 여러 클래스에서 사용하는 고정적인 상수 변수를 따로 모아서 관리하는 편이다.

Ex)

```
1 public class Constant {
2     public static final int PRICE=500;
3 
4     public static final double PI=3.141592653589793;
5   // 상수변수에 public으로 외부에서 접근 사용가능하게하고
6   // final 상수변수 설정은 보통 대문자로 하는편이다.
7   //static형식을 이용해 해당 변수를 인스턴스 없이 바로 사용하게하고
8   //상수화 하기위해 final형식으로 외부에서 변수 변경을 막는다.
```