12.26 月 _ stack , heap

 new를 해서 만들어 지는 건 전부 **heap 영역**에 올라간다

서버 사이드 스크립트 언어

웹에서 사용되는 스크립트 언어 중 서버 사이드에서 실행되는 스크립트 언어

Overloading 적용 범위 (딱 맞춰서 사용해라)

- 매개변수의 개수
- 매개변수의 자료형

⇒ **인자 값과 일치!**

최대한 형을 맞춰서 집어 넣는다 🔻

(jvm 버전 마다 다름 → 가능한 제대로 맞춰라)

```
package test;

class Scv {

  Scv() {

  }

  Scv(double a) {
    System.out.println(a);
  }

}

public class test3 {

  public static void main(String[] args) {

    // 12.26 1교시
    new Scv(3.0f);
```

형이 맞으면 맞는 걸 찾아낸다 🔻

```
package test;

class Scv {

  Scv() {

  }

  Scv(double a) {
    System.out.println(a + "1");
  }

  Scv(float a) {
    System.out.println(a + "2");
  }

}

public class test3 {

  public static void main(String[] args) {
```

```
    }

}
```

```
    // 12.26 1교시
    new Scv(3.0f);
  }

}
```

운영체제 메모리 영역

> **stack** • heap 의 공통점

- **Memory 영역**

> **stack** • heap 의 차이점

- **stack - 보통 깔끔하게 정돈된**

  : 메소드를 호출할 때 선언되는

  ⭐ ⭐ ⭐ 변수 (지역변수 , 매개변수) 들을 저장하는 곳 ⭐ ⭐ ⭐ , **LIFO 형식**

- **heap - 아무렇게 쌓아 놓은**

  : ⭐**new 연산자**를 이용해서 동적(프로그램이 돌면서 메모리가 만들어짐)으로 할당한 메모리

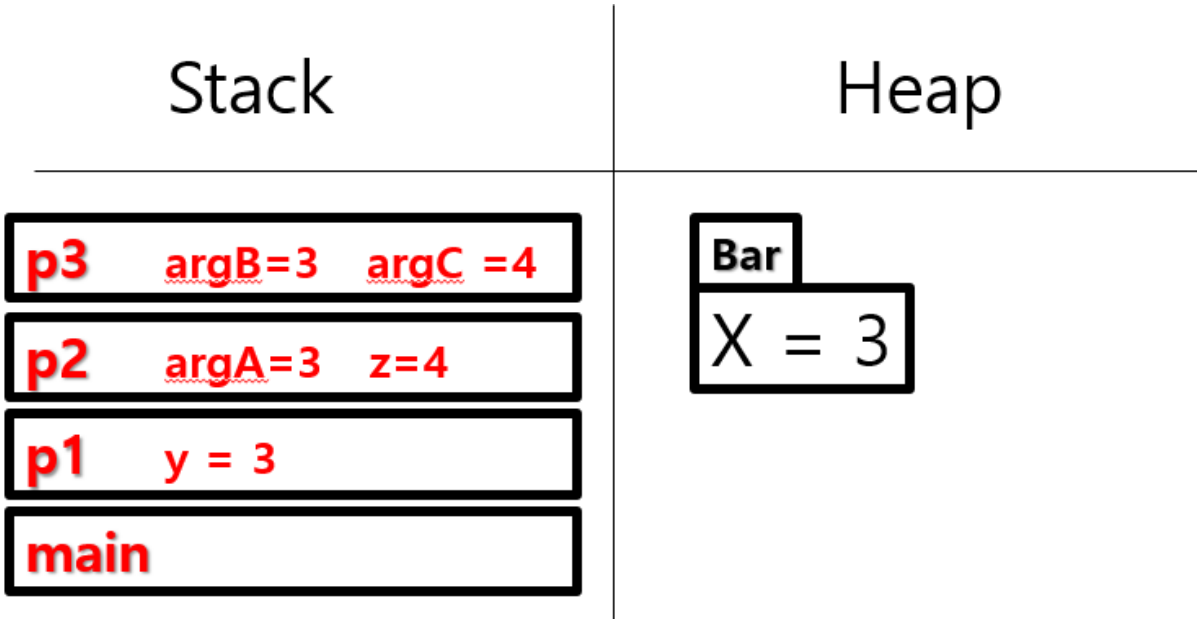
  ⇒ 객체(+멤버 변수) 를 저장 하는 곳

「 ***life cycle*** 」

	生まれ	死に
지역변수	메소드가 호출 되고 메소드 or 생성자 내에서 선언 시	선언되 있는 블록에서 블레이스가 닫히면
매개변수	호출 될 시	메서드가 종료 되면
맴버변수	객체가 생성될 때	객체가 종료될 때

연습문제 1

```
package test;

class Bar{
  int x = 3;
  void p1() {
    int y =3 ;
    p2(y);
  }

  void p2(int argA) {
    int z =4 ;
    p3(argA, z);
  }
  void p3 (int argB , int argC) {
    System.out.println(argB + argC);
  }

}

public class test1 {

  public static void main(String[] args) {
    領域
      (new Bar()).p1();

  }

}
```



- 메소드들은 heap stack 이외의 영역에 있다 → Code 이기 때문
- 메소드는 호출될 때 실행된다 → 실행이 다 끝나면 호출한 곳으로 돌아 간다
- 해당 부분이 종료되면 메모리가 사라진다

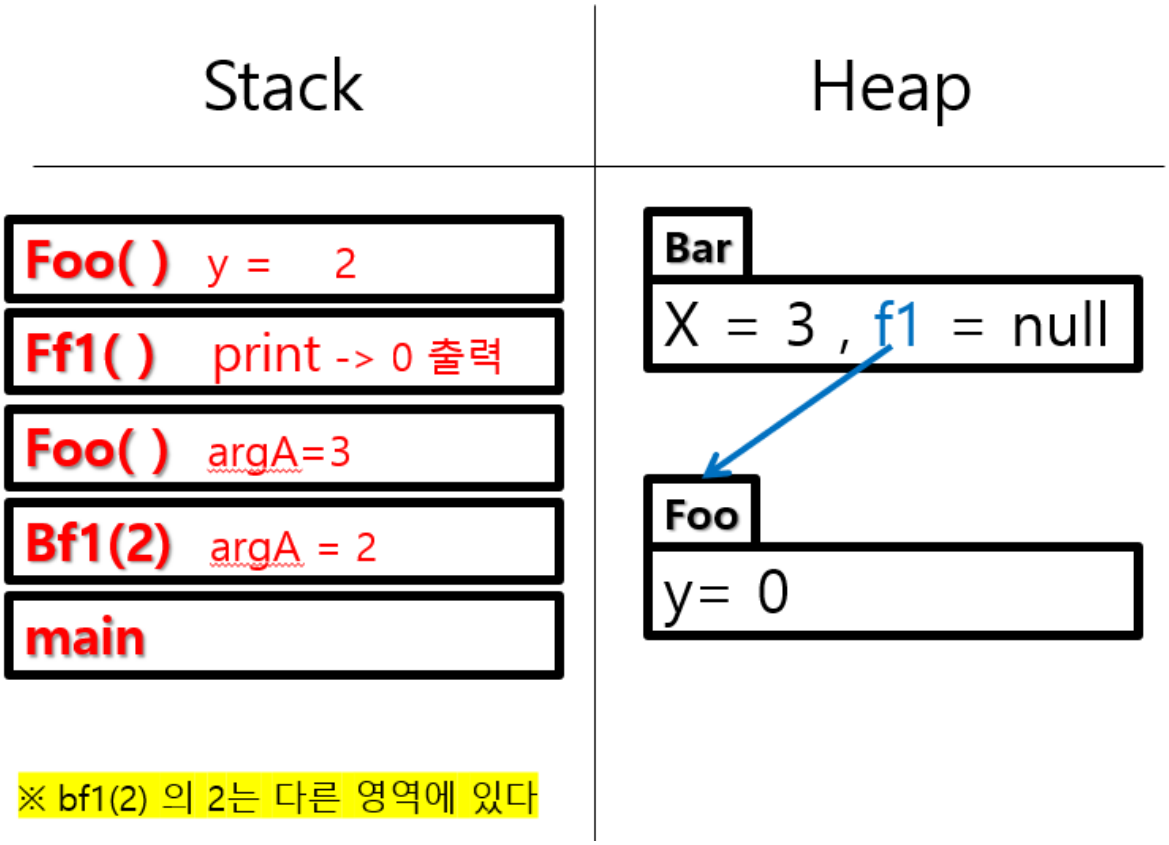
- 멤버 변수는 객체 안에 있다
  → ⭐⭐⭐⭐⭐ heap에 올라가는 것**이 객체에 대한 여러가지 정보들이 들어가는데 그중에 하나가** `멤버 변수` ! ⭐⭐⭐⭐⭐⭐⭐

❓ stack 필요한 이유

메서드를 호출하면 그 메서드 안에서 또 다른 메서드를 호출한다 →한 개의 메소드를 실행하는 도중에 다른 메소드를 호출 → 코드에 남아 있다 → 위에서 선언한 것이 다른데 갔다 오더라도 유지되어야 한다 ⇒ 쌓아 올렸다가 종료되면 다시 하나씩 지우면서 내려옴

연습문제 2

```
package test;

class Bar {
  int x = 3;
  Foo f1;  => f1 참조변수!

  void bf1(int argA) {
    f1 = new Foo(argA);  //new 로 object를 할당하려면 해당 클래스에 가서 멤버들을 처리를 해줘야한다
  }

}
-----------------------------------------------------------------
      class Foo {
        int y;

        Foo(int argA) {
          ff1();
          y = argA;
        }

        void ff1() {
          System.out.println(y);
        }
      }

public class test3 {

  public static void main(String[] args) {

    (new Bar()).bf1(2);

  }

}
```



인스턴스 멤버 변수 - 현재 생성된 객체에 포함 붙어있는 멤버