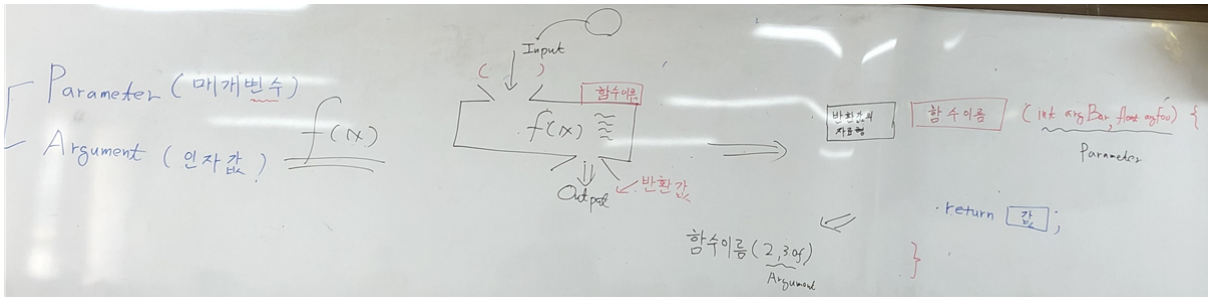
12.23 金 _ 객체지향 언어 4

Parameter (매개변수)

: 함수에 선언된 **변수**

Argument (인자 값)

: 함수를 호출할 때 넣어주는 **값**



Overloading of Constructor

⭐ 매개 변수의 자료형과 개수에 따라서 메소드를 호출 할 때 해당 매개 변수 인자 값의 자료형과 인자 값의 개수에 따라서 알맞은 메소드를 호출해 준다

❗ 동일한 기능을 수행하는데(다 생성자) 들어가는 인자 값의 자료형이나, 인자 값의 개수에 따라서만 다르게 알고리즘이 적용되면 , 각각 매소드 이름을 다 다르게 선언해야 한다. → 편의 제공

default constructor

```
package trial;

class Scv {
```

```
  String name;
  int    hp;

  //default constructor  = 기본 생성자 (매개 변수가 없는 생성자)
  Scv() {

  }

  Scv(String argName){
    name = argName;
  }

  Scv(String argName , int argHp){
    name = argName;
    hp = argHp;
                // Q. Scv 객체를 만들면 생성자 3개가 동시에 호출되나?
                // A. NO! => 원하는 것 1개만호출을 한다.
                //    호출을 하는 대상 자체가 생성자 or 매소드
                //    그 때 인자 값이 뭐가 들어가는가에 따라
                //    해당 유형에 맞는 생성자 or 매서드를 호출한다
                //   => 1개만 호출된다 (무엇을 호출시킬지는 사용자의 선택)
  }
}

public class test1 {

  public static void main(String[] args) {

    Scv s1 = new Scv();
    s1.name = "Scv 1";
    s1.hp = 20;
                    // 각각의 생성자를 호출할 때 각각의 argument (인자 값) 의 유형에 따라서
                    //  알맞은 생성자를 찾아서 호출을 한다
    Scv s2 = new Scv("Scv 2");
    Scv s3 = new Scv("Scv 2",20);

    Scv s4 = new Scv (2.0f);

  }
}
```

default constructor

JAVA에서는 class안에 constructor가 없으면 자바 컴파일러가 컴파일 하기 전에 `전 처리 과정` 을 거친다

`전 처리 과정`을 거치면서 class를 훑어보고 프로그래머가 class안에 생성자를 하나도 생성 안 해 놓았을 경우 자동적으로 `default constructor` 을 생성한다

> `default constructor` 를 생성하는 조건 : class 안에 생성자가 0개일 時!!!!

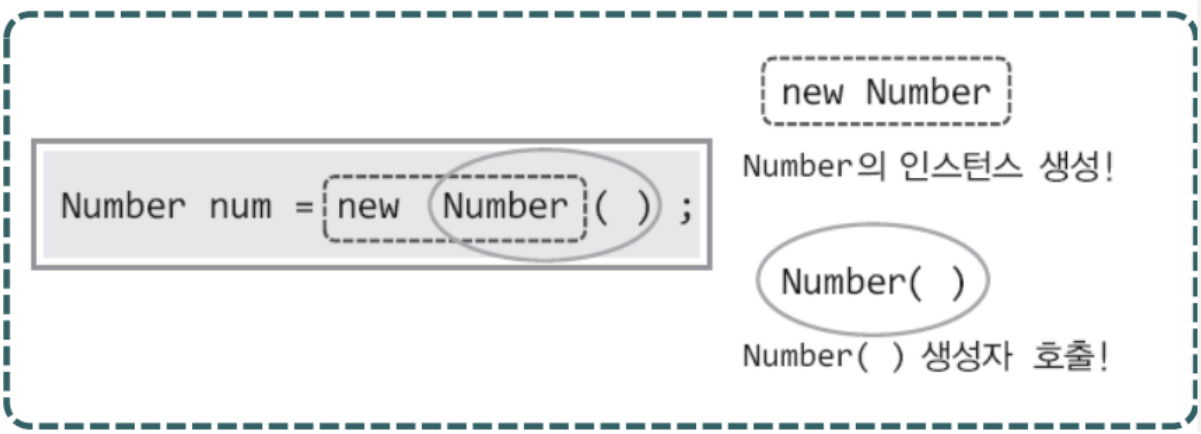
```
package trial;

class Scv {
  String name;
  int    hp;

  //default constructor


public class test1 {

  public static void main(String[] args) {

    Scv s1 = new Scv();    // 자바는 전처리 과정을 거쳐 생성자가 없으면 디폴트 생성자를 만들어 넣는다

  }
}
```

default constructor の誤謬!

```
package trial;

class Scv {
  String name;
  int    hp;

  //default constructor

  Scv(String argName){  name = argName; }  // 생성자


public class test1 {

  public static void main(String[] args) {

  //  default constructor를 호출
    Scv s1 = new Scv();  // 디폴트 생성자 조건은 생성자가 0개일 때 작동!
                         // 위에는 생성자가 1개 있기 때문에 디폴트 생성자가 작동 불가능!!!

  }

}
```

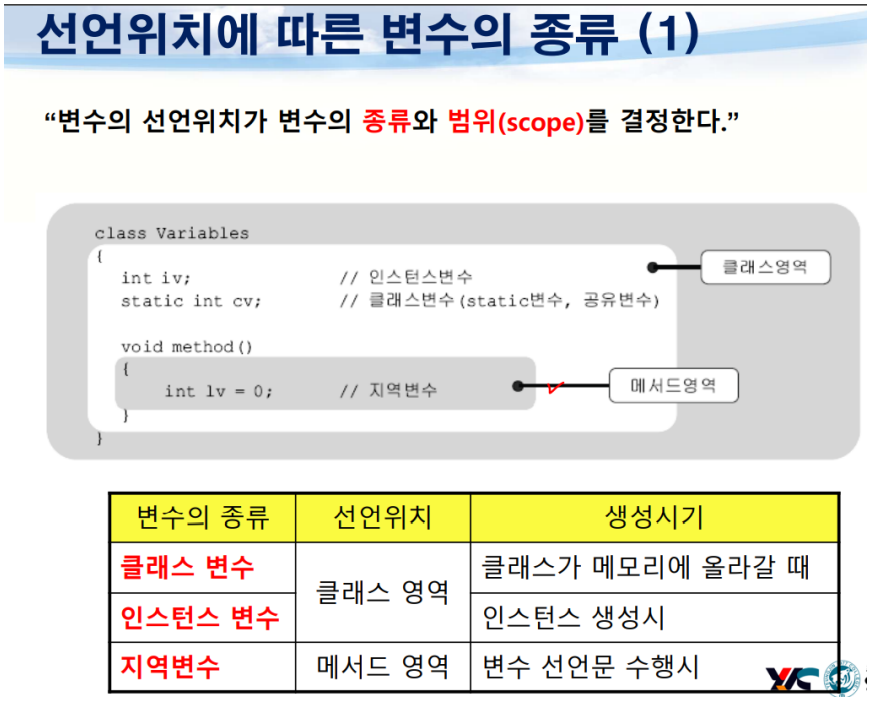
⭐ **생성자**는 어떻게 만들어 내는가?

1. class 內에 class 이름과 동일하게 선언을 한다
2. 그 안에 객체를 생성할 때 초기화 작업에 관련된 알고리즘이 생성자 內에 들어 간다
3. 생성자의 특징은 method 와 달리 반환 형이 없다 ❗
   ⇒ 생성자는 new 연산자를 이용해 객체를 생성한 다음에 호출이 되는데 이 때 넘어 오는 값은
   현재 생성된 객체의 주소가 넘어와야 하기 때문에 반환 형이 있을 수 없다 ❗
4. 생성자는 method 와 같이 `Overloading` 개념을 적용한다. 때문에 class 內 에 다수 개의 생성자가 들어올 수 있다. `Overloading` 을 적용 유무는 해당 class를 설계하는 사람에 따라서 정해진다.
5. JAVA에서 생성자를 만들 때 규칙이 하나 있는데 class 內 에 생성자가 0 개일 경우에는 JAVA가 컴파일 하기 전에 `default constructor` 를 추가한다 → 만약 사용자가 생성자를 1개 이상 선언을 해버리면 `default constructor` 를 자동으로 추가하지 않는다.



변수

- ★ **접근 범위** (Scope) - 해당 함수를 어디까지 사용할 수 있나?
- ★ **생명 주기** (Life cycle) - 해당 함수가 언제까지 메모리에 남아있나?



JAVA 에서 사용되는 변수

1. 멤버 변수

> class 로 객체를 찍어내면 생성 / 객체가 사라질 때 사라진다 (가비지 컬렉터에 잡힐 때)

- **클래스 中 에 선언된 변수** (메소드 내 선언된 변수는 제외)
- 클래스 내에서는 다 접근 가능하다
  - 인스턴스 멤버 변수
  - 클래스 멤버 변수

2. 지역 변수

> 메소드가 호출 될 때 생성된다 / 메소드가 종료되면 죽는다

: **메소드 or 생성자 中 에 선언된 변수** (매개변수도 지역변수)

지역변수가 선언되고 종료 블레이스를 만날 때까지 실행 - 블레이스 만나면 지역변수 회수함

3. 매개 변수

: 메소드 or 생성자의 Argument(인자 값)을 받는 변수

「 Life cycle 」

	生	死
`멤버 변수`	class 가 생성되면 (class 로 객체를 찍어내면)	객체가 사라질 때 (가비지 컬렉터에 잡힐 때)
`지역 변수`	메소드가 호출 될 때	선언된 범위 부터 블레이스 (`}`)를 만날 때

Q. 몇 개의 변수가 있는가? A.
3 개

```
package trial;

class Scv {
  String name;
  int    hp; // 맴버변수

  void prtSomething (int argA /*매개변수*/) {

    int sum /*지역변수*/= argA + hp;

    System.out.println(sum);
  }

  void foo() {
    int bar = hp+2;
  }
}
public class test2 {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    Scv s1 = new Scv();
  }

}
```

변수의 생명 주기 (Life cycle)

```
package trial;

class Scv {
  String name;
  int    hp = 1; // 맴버변수
```

```
  void prtSomething (int argA /*매개변수*/) {

    int sum /*지역변수*/= argA + hp;

    System.out.println(sum);
  }

  void foo() {
    int bar = sum + 2;   // 위에서 sum은 메서드 종료가 되어 죽은 변수이다
  }

}
public class test2 {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    Scv s1 = new Scv();
  }

}
```

지역변수 생명 주기 (Life cycle)

자바는 닫히는 블레이스 지역변수 메모리를 회수 해 간다

```
package trial;

class Scv {
  String name;
  int    hp; // 맴버변수

  void prtSomething (int argA /*매개변수*/) {

    int sum /*지역변수*/= argA + hp;

    System.out.println(sum);
  }

  void foo() {
    int temp = 2;

    if (temp > 3) {
      int k = 20;
    }

    System.out.println(k);

  }
}
```

```
public class test2 {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    Scv s1 = new Scv();
  }

}
```

멤버 변수 생명 주기

→ 자바는 **가비지 컬렉터**가 있어서 **소멸자**가 없다

소멸자 - 객체가 메모리 상에서 해야 할 일이 있으면 소멸되기 전에 그 알고리즘을 집어 넣는다

```
package trial;

class Scv {
  String name;
  int    hp; // 맴버변수

  void prtSomething (int argA /*매개변수*/) {

    int sum /*지역변수*/= argA + hp;

    System.out.println(sum);
  }


}


public class test2 {

  public static void main(String[] args) {


    Scv s1 = new Scv();
    s1.hp = 100;   // 인스턴스 멤버 변수는 객체를 찍어야 사용가능!!

    Scv s2 = new Scv();
    s2.hp = 200;

    System.out.println(s1.hp + " : " + s2.hp);

    s1 = null;   // 참조하는 변수가 없으면 가비지 컬렉터가 잡아간다
//  System.out.println(s1.hp + " : " + s2.hp);
```

```
  }
}
```