01.02 月_ 초기화 블록

초기화 블록

초기화 : 어떤 상태를 처음으로 설정 하는 것

초기화 블록	생성자
함수형태	함수 형태
매개변수가 없다	매개변수를 가지고 있다
new 연산 다음 바로 , 생성자 뒤에 생성된다	초기화 블록 생성 후 생성자가 실행 된다

초기화 블록 ?

⇒ `Overloading`

모든 생성자에서 우선적으로 일어나야 할 일들이 있다.

모든 생성자에 중복되는 부분이 있어야 한다

```
package test;

public class test1 {

  public static void main(String[] args) {

    Bara b1 = new Bara();

  }

}

class Bara {
    int a = 3;
    static int b = 3;

    // static 블록
    // 클래스가 처음으로 사용될 때 실행되는 블록
```

```
    // 주로 클래스 변수를 초기화 시키는 코드를 둔다.
    static {
        System.out.println("static");
//        System.out.println(a);
        System.out.println(b);
    }


    // 초기화 블록
    // 주로 인스턴스 변수를 초기화시키는 코드를 둔다.
    // 생성자가 호출 되기 전에 실행된다.
    // 인자값을 받지 않는다.
    {
        System.out.println("initialization block");
        System.out.println(a);
        System.out.println(b);
    }


    // 생성자 블록
    // new 연산자를 사용하여 인스턴스가 만들어 질때 실행되는 블록
    Bara() {
        System.out.println("const");
        System.out.println(a);
        System.out.println(b);
    }
}
```

인스턴스 **초기화 블록** 強制的

⇒ 매개 변수는 받아서 처리 **불가능**하다 !!

```
class Bar {
  // 공통으로 실행되는 코드를 초기화 블럭에 지정 (가장 먼저 실행되는 것이 확정일 때)
  // 반드시 객체가 생성될 때 초기화시킴

  {
    System.out.println("k");
  }


  Bar() {
    // System.out.println("k"); // 중복
  }


  Bar(int argA) {
    // System.out.println("k"); // 중복
    a = argA;
  }
```

```
  Bar(int argA, int argB) {
    // System.out.println("k"); // 중복
    a = argA;
    b = argB;
  }
}


public class MyProject {
  public static main(String args[]) {
    Bar b1 = new Bar();
  }
}
```

★ `this( );` ★ 非強制的 ↔ `super( )`

⇒현재 생성된 자기 자신의 객체의 주소를 나타낸다 ⇒ **매개 변수도 받아서 처리 가능하다 !!**

```
class Bar {

  Bar() { // 이렇게 써도 된다. 물론
    System.out.println("k"); // 중복
  }

  Bar(int argA) {
    this();
    a = argA;
  }
  Bar(int argA, int argB) {
    this();
    a = argA;
    b = argB;
  }
}



public class MyProject {
  public static main(String args[]) {
      Bar b1 = new Bar();
  }
}
```

클래스 초기화 블록

클래스 메소드를 작업 할 때 그전에 초기화 작업이 필요할 때 사용한다.

```
class Bar {

static int x =3 ;

  static {
    System.out.println("스태틱 초기화 블록 호출");
  }

  {
    System.out.println("인스턴스 초기화 블록 호출");
  }

  Bar() {
    System.out.println("생성자 호출");
  }
}

public class MyProject {
  public static void main(string args[]) {
    System.out.println("betty!");
  }
}
```

```
class Bar {

static int x  ;
static int y  ;

  static {
          x = 3;
          y = 3;
  }

  {
    System.out.println("인스턴스 초기화 블록 호출");
  }

  Bar() {
    System.out.println("생성자 호출");
  }
}

public class MyProject {
```

```
  public static void main(string args[]) {
    System.out.println(Bar.x);
    System.out.println("hello");
  }
}
```

출력 순서

```
package test;

class Bar {
  static {
    System.out.println("스태틱 초기화 블록 호출");
  }

  {
    System.out.println("인스턴스 초기화 블록 호출");
  }

  Bar() {
    System.out.println("생성자 호출");
  }
}

public class test1 {

  public static void main(String[] args) {

    Bar betty1 = new Bar();

    System.out.println(betty1);

  }

}
```