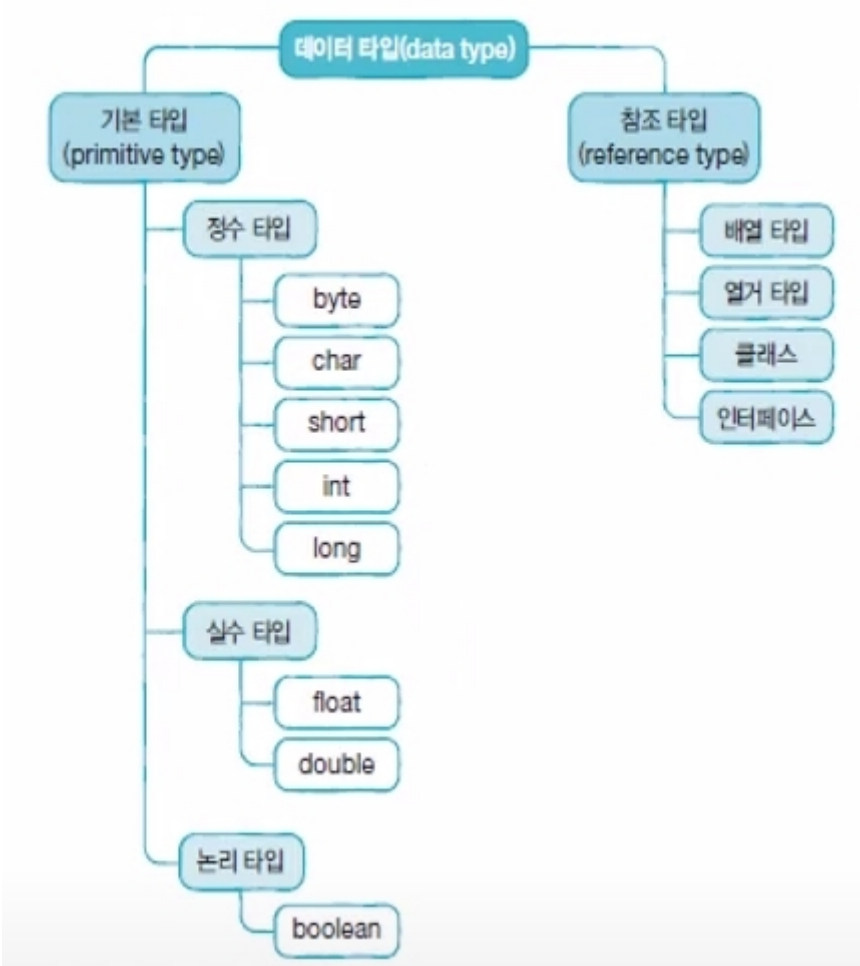
JAVA/혼자공부하는자바

[JAVA 기초] 참조 타입과 참조 변수(==/!=연산, NullPointerException)

JongHyun99 | 2020. 12. 11. 20:05



기본 타입(primitive type)

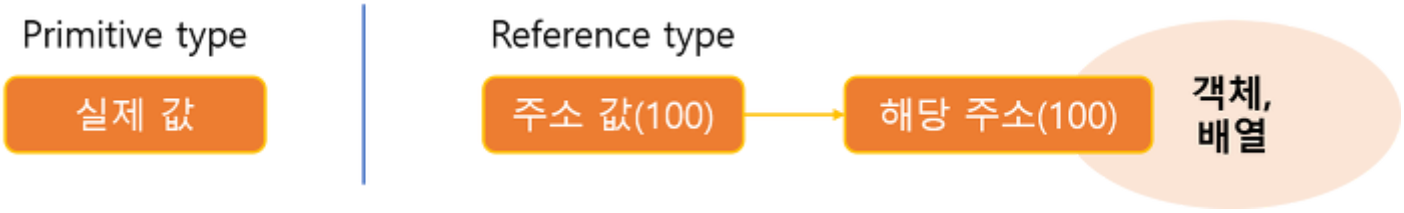
8개의 기본 타입으로 이루어져 있어 정수, 실수, 문자, 논리 리터럴을 저장한다

참조 타입(reference type)

객체(object)의 번지를 참조하는 타입
배열, 열거, 클래스, 인터페이스

기본 타입 변수와 참조 타입 변수의 차이점

기본 타입과 참조 타입의 차이는 저장되는 값이 무엇인가 이다. 기본 타입으로 선언된 변수는 실제 값을 변수에 저장하지만 참조 타입은 배열, 열거, 클래스, 인터페이스를 이용해서 선언된 변수는 메모리에 번지를 값으로 갖게 되며, 즉 번지를 통해 객체를 참조한다는 뜻으로 참조 타입이라고 부른다.



기본 타입 변수

```
int age = 25;
double price = 100.5;
```

참조 타입 변수

```
String name = "신용권";
String hobby = "독서";
```

메모리 사용 영역(Runtime Data Area)

메소드 영역(Method Area)

- 정적 필드(static field)
- 상수(constant)
- 생성자(constructor)
- 메소드(method)

코드 등을 분류해 저장한다

힙 영역(Healp Area)

- 객체와 배열이 생성되는 영역
- 인스턴스 필드들은 heap 영역에 올라간다 (이러한 이유로 static 한 메소드에서 인스턴스 멤버를 접근할 수가 없다. 어떤 인스턴스 인지도 알 수 없고, 존재하지도 않을 수도 있는 인스턴스를 사용하라는 것이기 때문)
- 메소드들은 static이 아니더라도 굳이 heap에 생기지 않는다. 어차피 같은 로직의 메소드이기 때문에, 여러 개 일 필요가 없다.
- stack영역에서 참조값을 이용하여 참조형 변수가 heap 영역에 있는 인스턴스를 가리키어 제어할 수 있음.
- 어떤 참조 변수도 힙 영역에 있는 인스턴스를 참조하지 않게 된다면, GC(가비지 컬렉터)에 의해 메모리에서 사라 지진다.
- 상속을 이용한 인스턴스를 만들었다면 상위 클래스들의 인스턴스들도 같이 생성된다. (최상위인 Object까지)

JVM 스택 영역

- 여는 중괄호 '{'를 만날 때마다 스택 프레임이 하나씩 생기고, 닫는 중괄호 '}'를 만나게 되면 스택 프레임이 사라짐, 그러므로 메소드가 실행될 뿐 만 아니라, if문, 반복문, 예외처리를 위한 try문 등도 모두 스택 프레임이 생긴다.
- stack 내부에서 선언된 지역변수는 stack 영역에 올라간다.
- 기본형 타입 변수의 값들은 stack영역에 저장되고, 참조형 타입 변수는 참조값만 저장된다. (이 참조값은 heap 영역에 존재하는 인스턴스(객체)를 가리키는 역할을 한다. 엄격한 표현은 아니지만 인스턴스(객체) 주소 값 정도로 이해하자.)
- 외부 스택 프레임에서는 내부 스택 프레임의 변수에 접근하는 것은 불가능하나 그 역은 가능하다. 쉽게 생각하면 메소드안에 for문 스택 프레임을 만든 경우, for문에서는 자신을 호출한 메소드의 변수는 사용 가능하나, 메소드에서는 for문에서 선언한 변수를 사용할 수 없다.
- + 메소드를 호출하는 것은 별개의 스택 프레임 이기 때문에 스택 프레임을 넘어서 접근할 수 없다.
- 스레드도 stack 영역에 생기게 된다. 하나의 쓰레드는 내부적으로 별개의 T메모리 구조 static, stack, heap영역을 갖게 된다. 이런 이유로 하나의 쓰레드는 다른 쓰레드로 접근 할 수 없지만, static 영역과 heap 영역은 공유해서 사용할 수있는 특징을 가지게 된다. (이런 특징이 멀티 프로레스 구조보다 멀티 쓰레드 구조가 메모리를 적게 사용 할 수 있는 이유)

> 참고도서 - 스프링 입문을 위한 자바 객체지향의 원리와 이해 (김종민 저자)
> / siyoon210.tistory.com/124

참조 변수의 ==, != 연산

- 동일 객체를 참조하는지, 다른 객체를 참조하는지 알아볼 때 사용
- 참조 타입에 저장된 번지 값을 비교하여 연산한다. (기본 타입은 값을 비교)

	==	!=
같으면	true	false
다르면	false	true

이는 단순히 변수끼리의 값이 같으냐를 확인하는것이 아니라 참조하는 객체가 같은지를 확인한다.

Null

참조 타입 변수는 객체를 참조하지 않는다는 뜻으로 null 값 가질 수 있음
null로 초기화된 참조 변수도 스택 영역에 생성이 된다.

```
refVar1 == null //결과 : false
refVar1 != null //결과 : true
```

참조 타입 변수가 null 값을 가지는지 확인하려면 다음 코드와 같이 ==, != 연산을 수행

NullPointerException

- 참조 타입 변수가 null 상태에서 존재하지 않는 객체의 데이터나 메소드 사용할 경우 발생
- 해당 참조 변수가 객체를 참조하도록 수정하여 해결

```
int[] intArray = null;       //특정 배열 값을 참조해주어 해결 ex)new int[3];
intArray[0] = 10;  //NullPointerException


String str = null;          //특정 문자열을 대입하여 해결 ex)"ABC"
System.out.println("총 문자수: " + str.length()); //NullpointerException
```

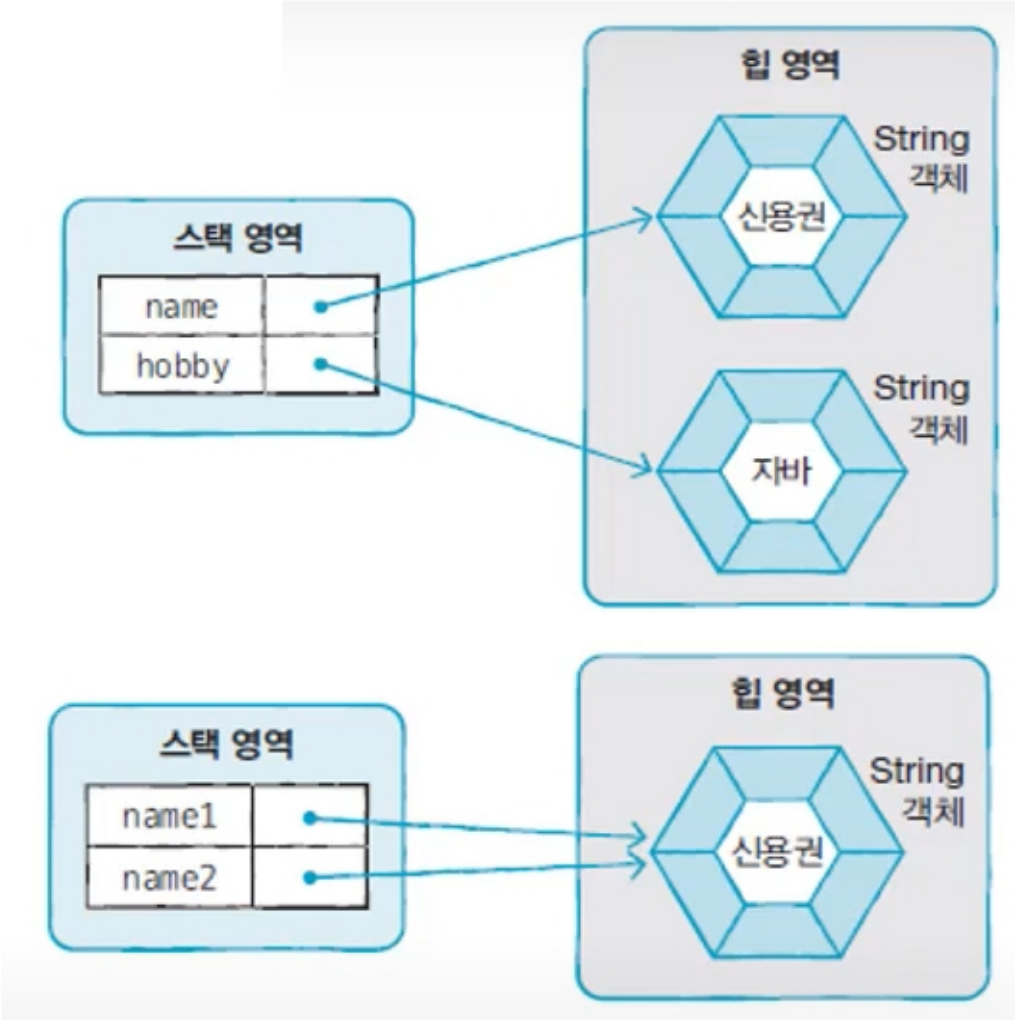
예외(Exception) : 프로그램 실행 도중 발생하는 오류
java에서는 error를 하드웨어, os에서 발생하는 것으로 취급하고
exception을 프로그램이 실행하는 도중 발생하는 것으로 취급한다.

JAVA/혼자공부하는자바

[JAVA 기초] 참조 타입과 참조 변수 2 (New 연산자, equals 연산자)

JongHyun99 | 2020. 12. 14. 01:32

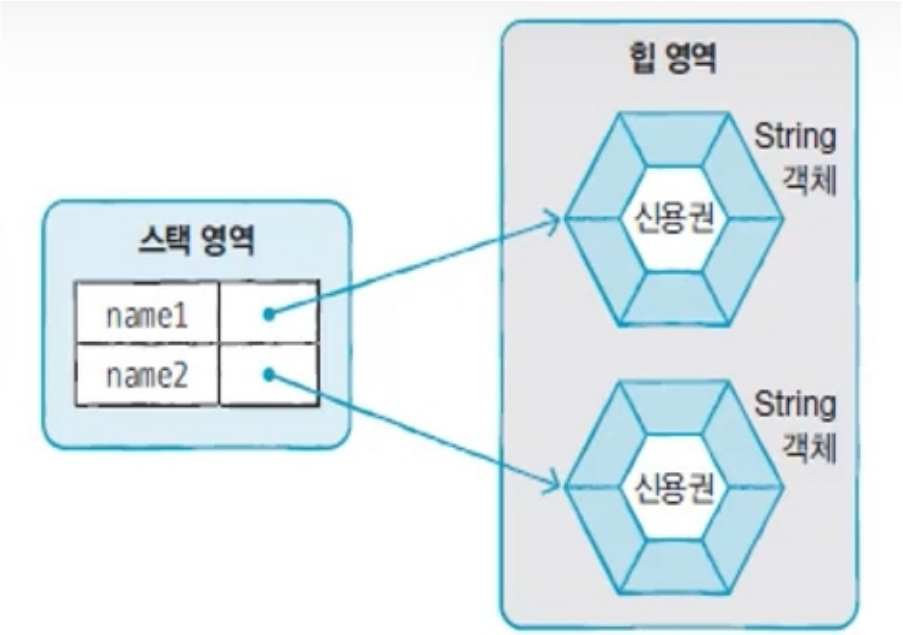
String 타입



String 변수에 문자열 리터럴을 대입할새 경우 스택 영역에 변수가 선언이 되고 힙 영역에 리터럴(객체)
가 생성되어 스택 영역에 저장이 된다. 만약 같은 리터럴로 변수 값을 선언하면 힙 영역에 생성된 하나의 객체로 두 변수가 공유하여 가지게 된다.

New 연산자

- 객체 생성 연산자
- 힙 영역에 새로운 String 객체를 생성




```
String name1 = new String("신용권");
String name2 = new String("신용권");
```

문자열 리터럴과 new 연산자로 생성된 객체 비교

```
String name1 = "신용권";
String name2 = "신용권";
String name3 = new String("신용권");
```

name1 == name2 : true
name1 == name3 : false

new를 이용하여 string객체를 만들게 되면 새로운 힙 영역에 저장이 되어 서로 같은 문자열 이어도
다른 힙 영역에 저장되기 때문에 다른 스트랭 객체로 인식된다.

문자열 비교

== : 번지 비교 (X)
equals(): 문자열 비교 (O)

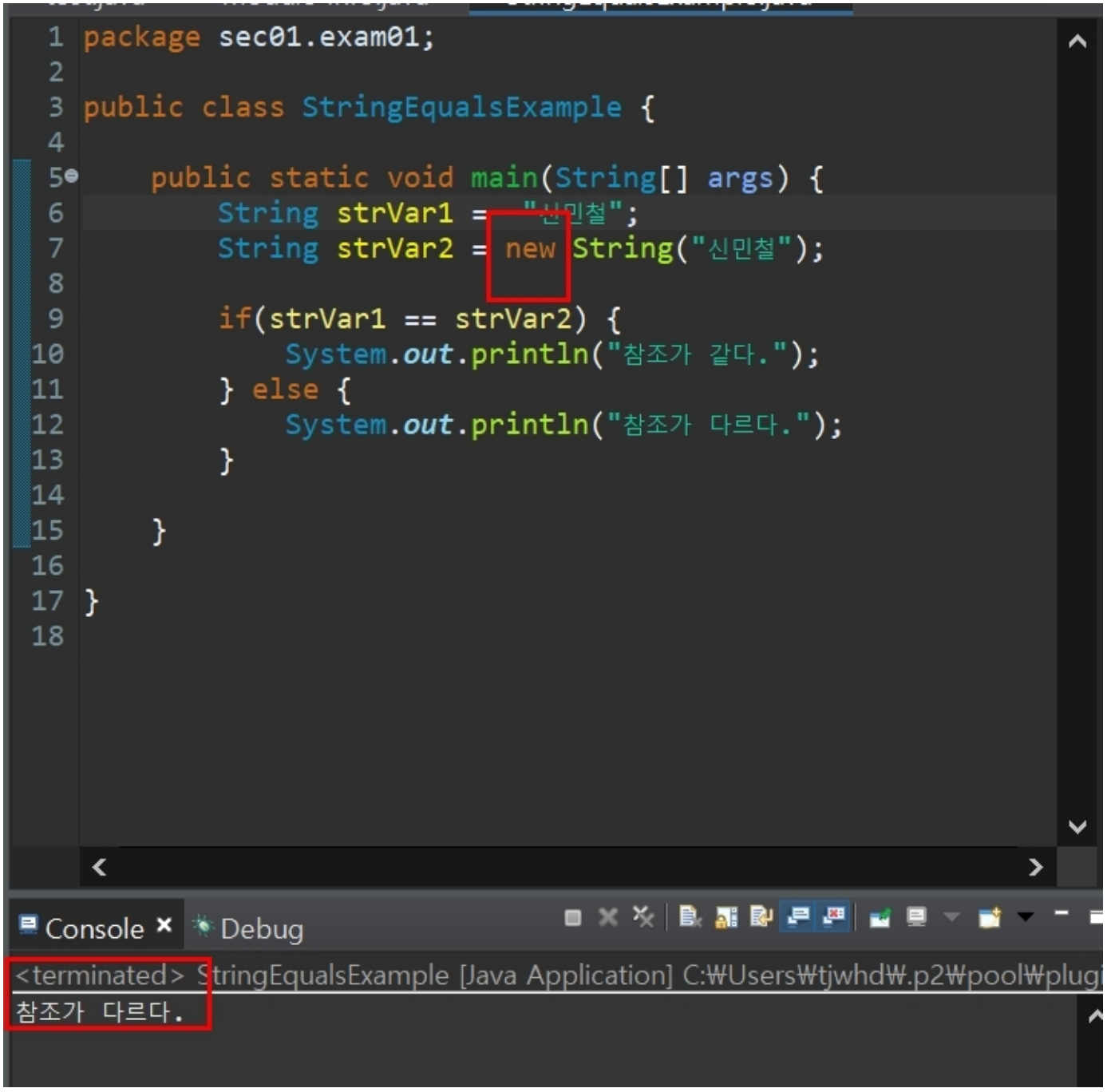
== 연산자는 문자열의 번지를 비교하고, equals연산자는 문자열 자체를 비교를 한다.
그래서 문자열을 비교할 때에는 주로 equals연산자를 사용한다.

```
boolean result = str1.equals(str2);
                 (원본)    (비교대상)
```

```java
package sec01.exam01;

public class StringEqualsExample {

    public static void main(String[] args) {
        String strVar1 = "신민철";
        String strVar2 = "신민철";

        if(strVar1 == strVar2) {
            System.out.println("참조가 같다.");
        } else {
            System.out.println("참조가 같다.");
        }

    }

}
```

```
Console × Debug
<terminated> StringEqualsExample [Java Application] C:₩Users₩tjwhd₩.p2₩pool₩plu
참조가 같다.
```

str1과 str2가 서로 동일한 객체를 참조하고 있다.



new String(); 연산자를 이용해 선언을 하니 서로 다른 참조를 하고 있다.

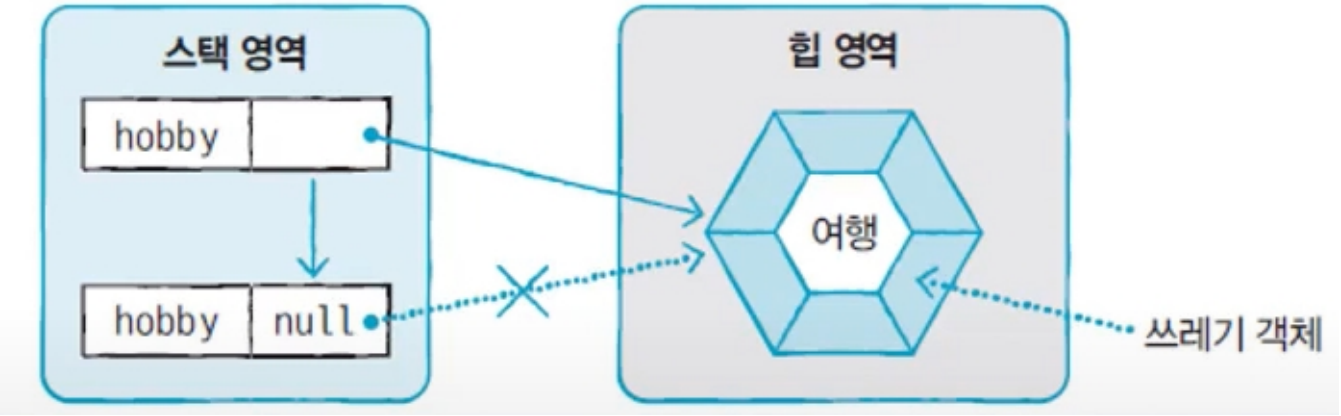
String 변수 초기값으로 null 대입이 가능하다.

```
String hobby = null;
```

이는 String 변수가 참조하는 객체가 없음을 의미한다.

```
String hobby = "여행";
hobby = null;
```

다른 값 대입 후에 다시 null값 대입 가능



참조를 잃은 String 객체는 쓰레기 수집기 (Garbage Collector) 통해 메모리에서 자동 제거된다.

```java
package sec01.exam02;

public class NullExample {

    public static void main(String[] args) {
        String hobby = null;

        System.out.println(hobby.length());
    }

}
```

null 값의 length를 구하려 시도하자 NullPointerException 발생

객체를 메모리에서 지우기 위해서는 위와 같이 null값을 대입해 변수에 저장된 참조를 끊어준다.
null값의 이전에 참조되어 있던 객체는 쓰레기 객체가 되고 쓰레기 수집기에 의해 메모리가 부족하거나 cpu가 한가할 때 메모리에서 제거가 된다.

C++ 에는 객체 제거 코드가 있지만 자바는 쓰레기 수집기가 있어 그런 코드가 따로 없다.

'JAVA > 혼자공부하는자바' 카테고리의 다른 글

[JAVA 기초] 자바 배열 2 (명령라인에서의 실행) (0) 2020.12.16
[JAVA 기초] 자바 배열 1 (선언, 생성, length) (0) 2020.12.16