

Chapter

05

참조 타입



05-1. 참조 타입과 참조 변수

혼자 공부하는 자바(개정판) (신용권 저)

❖ 목차

- 시작하기 전에
- 기본 타입과 참조 타입
- 메모리 사용 영역
- 참조 변수의 ==, != 연산
- null과 NullPointerException
- String 타입
- 키워드로 끝내는 핵심 포인트



시작하기 전에

[핵심 키워드] : 기본 타입, 참조 타입, 메모리 사용 영역, 번지 비교, null, NullPointerException

[핵심 포인트]

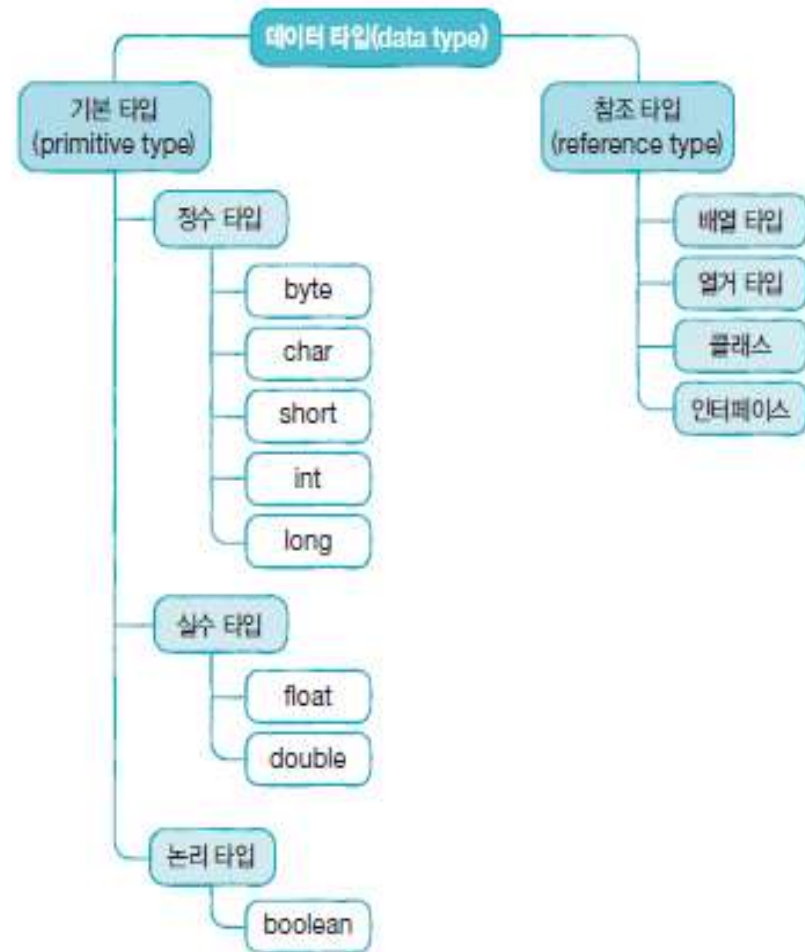
참조 타입의 종류와 참조 변수의 역할을 정확히 이해한다.

❖ 기본 타입 (primitive type)

- 정수, 실수, 문자, 논리 리터럴 저장

❖ 참조 타입 (reference type)

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



기본 타입과 참조 타입

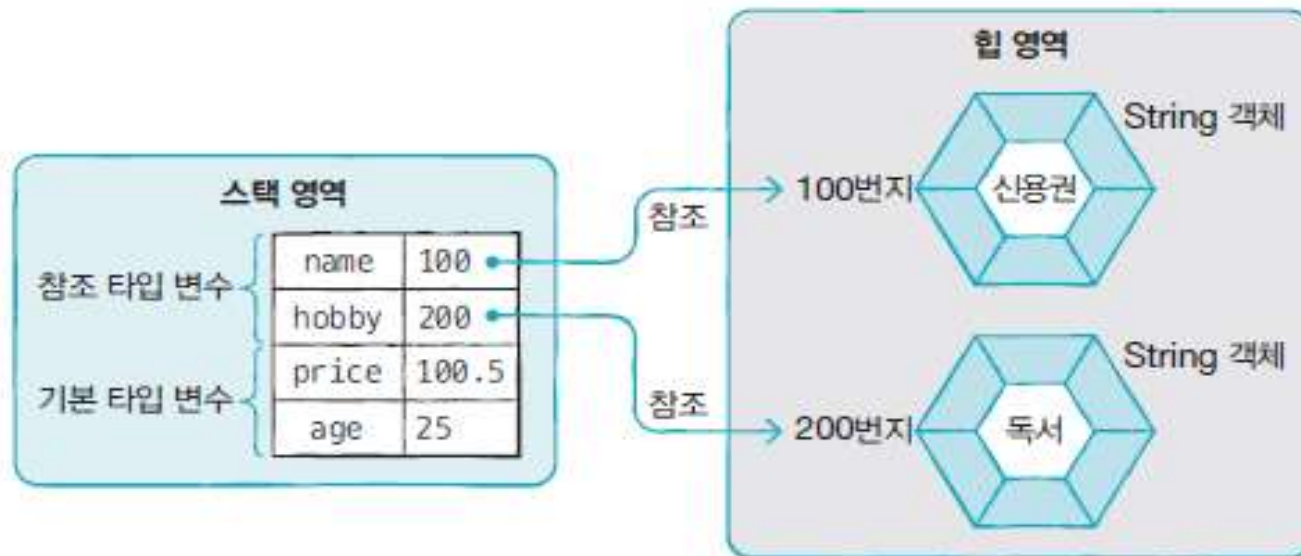
❖ 기본 타입 변수와 참조 타입 변수의 차이점

기본 타입 변수

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

참조 타입 변수

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



메모리 사용 영역

❖ 메모리 사용 영역 (Runtime Data Area)

■ 메소드 영역 (Method Area)

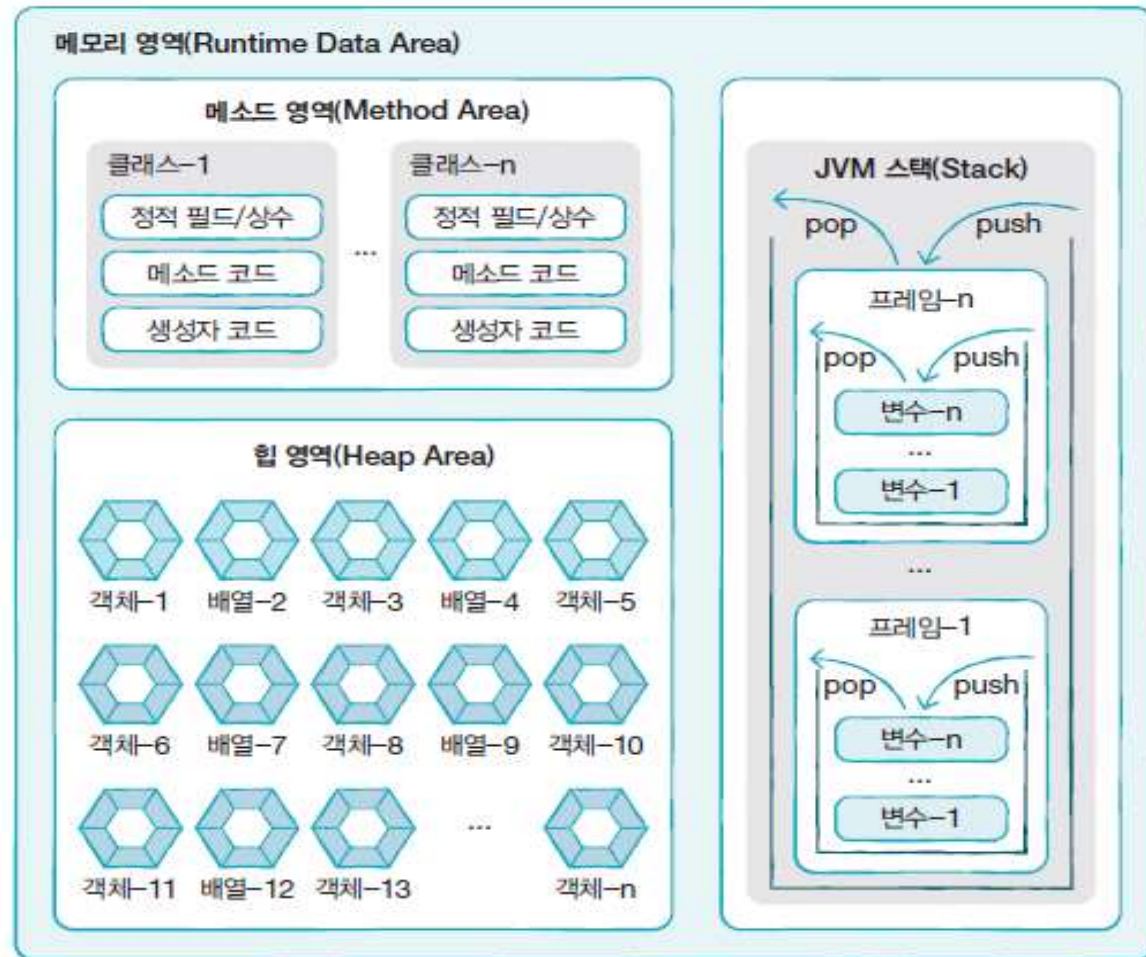
- -클래스별로
- 정적 필드(static field),
- 상수(constant),
- 생성자(creator)
- 메소드(method)
- 코드 등을 분류해 저장

■ 힙 영역 (Heap Area)

- -객체와 배열이 생성되는 영역

■ JVM 스택 영역

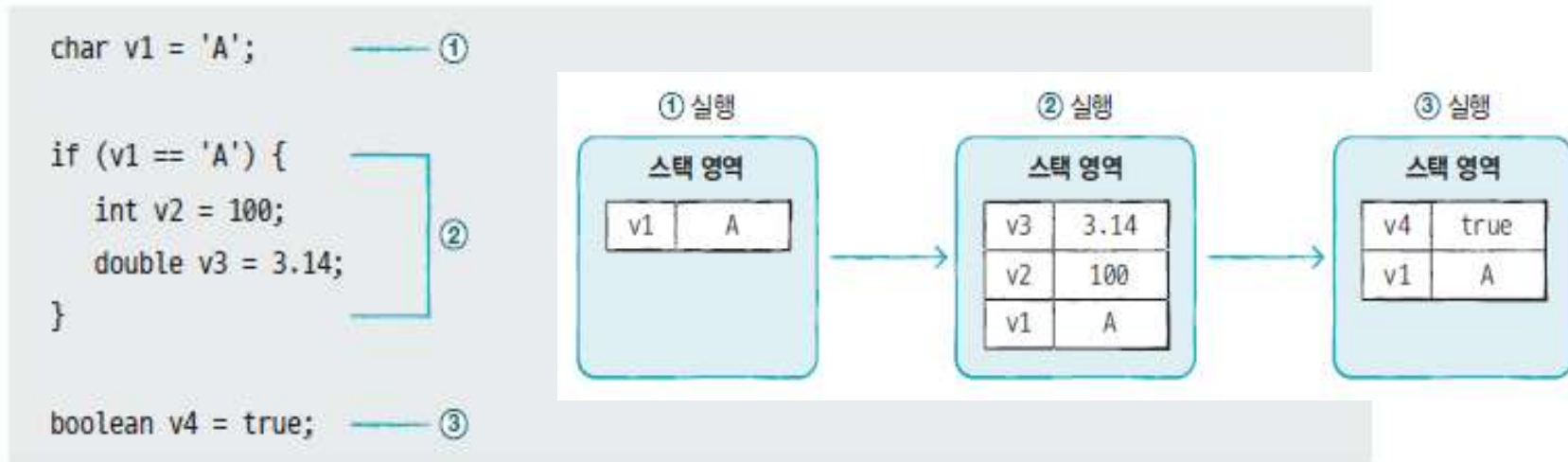
- -메소드가 호출되면
- 프레임이 추가되고,
- -메소드 종료되면
- 프레임이 제거됨



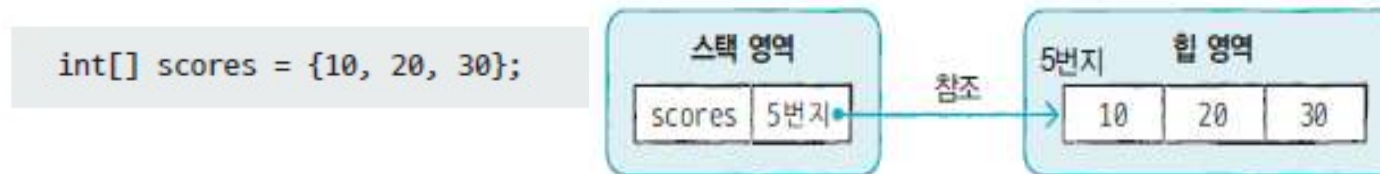
메모리 사용 영역

❖ JVM 스택 영역

- 메소드를 호출할 때마다 프레임이 추가되고, 메소드가 종료되면 해당 프레임이 제거
 - 프레임 내부의 변수 스택 이해



- 참조 타입 변수는 스택 영역에 힙 영역에 생성된 객체의 주소 가짐

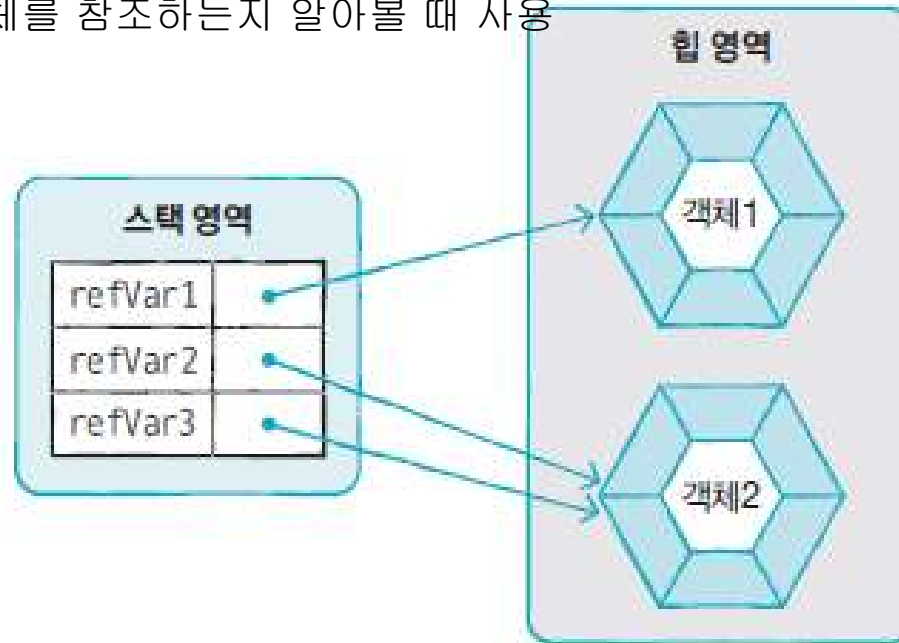


참조 변수의 ==, != 연산

❖ 참조 타입 변수 간의 ==, != 연산

- 동일 객체를 참조하는지, 다른 객체를 참조하는지 알아볼 때 사용
- 번지 값의 비교

- ==**
 - 같으면 true
 - 다르면 false
- !=**
 - 같으면 false
 - 다르면 true



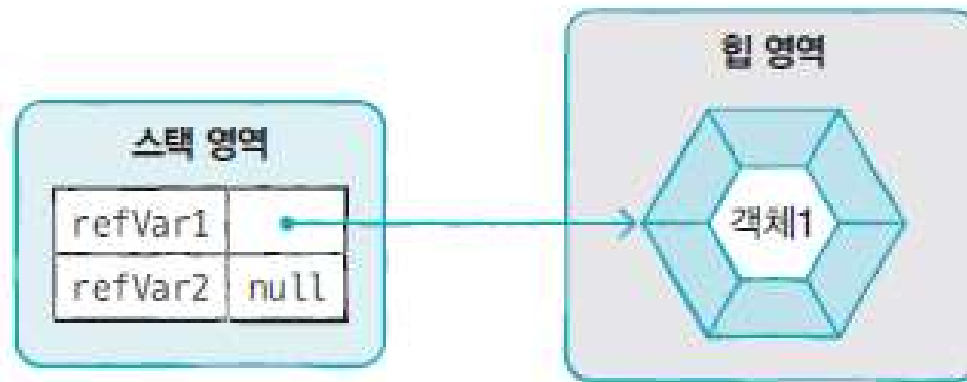
```
refVar1 == refVar2 //결과: false  
refVar1 != refVar2 //결과: true
```

```
refVar2 == refVar3 //결과: true  
refVar2 != refVar3 //결과: false
```



null과 NullPointerException

- ❖ 참조 타입 변수는 객체를 참조하지 않는다는 뜻으로 **null** 값 가질 수 있음
 - null로 초기화된 참조변수도 스택 영역에 생성



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

```
refVar2 == null //결과: true  
refVar2 != null //결과: false
```



null과 NullPointerException

❖ 예외 (Exception)

- 프로그램 실행 도중 발생하는 오류

❖ NullPointerException

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

```
int[] intArray = null;  
intArray[0] = 10; //NullPointerException
```

```
String str = null;  
System.out.println("총 문자수: " + str.length()); //NullPointerException
```



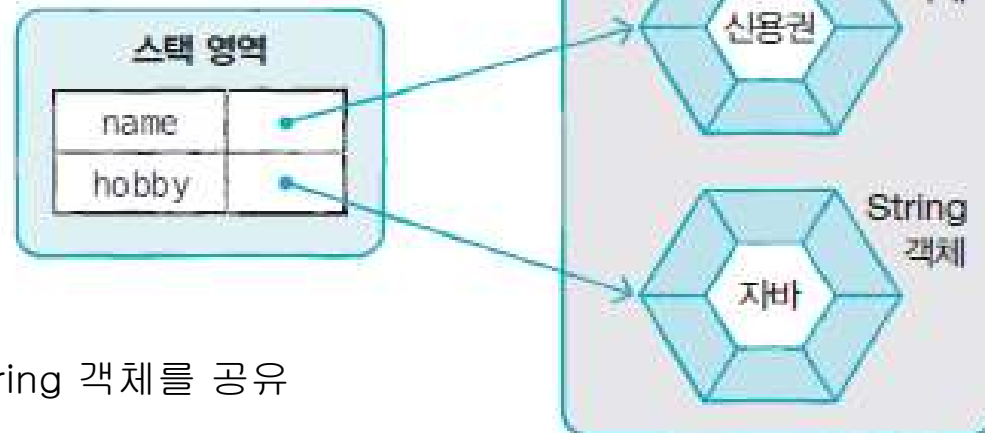
String 타입

❖ String 변수에 문자열 리터럴을 대입할 경우

- String 객체로 생성되고 변수가 String 객체를 참조

```
String name = "신용권";
```

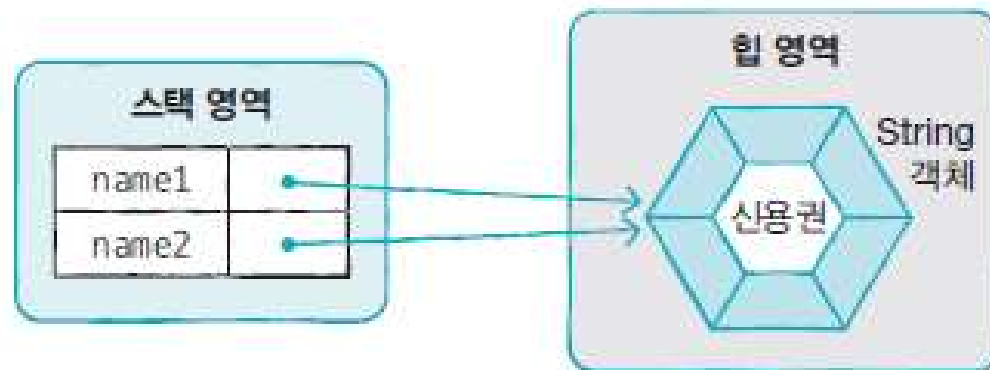
```
String hobby = "자바";
```



- 문자열 리터럴 동일한 경우 같은 String 객체를 공유

```
String name1 = "신용권";
```

```
String name2 = "신용권";
```

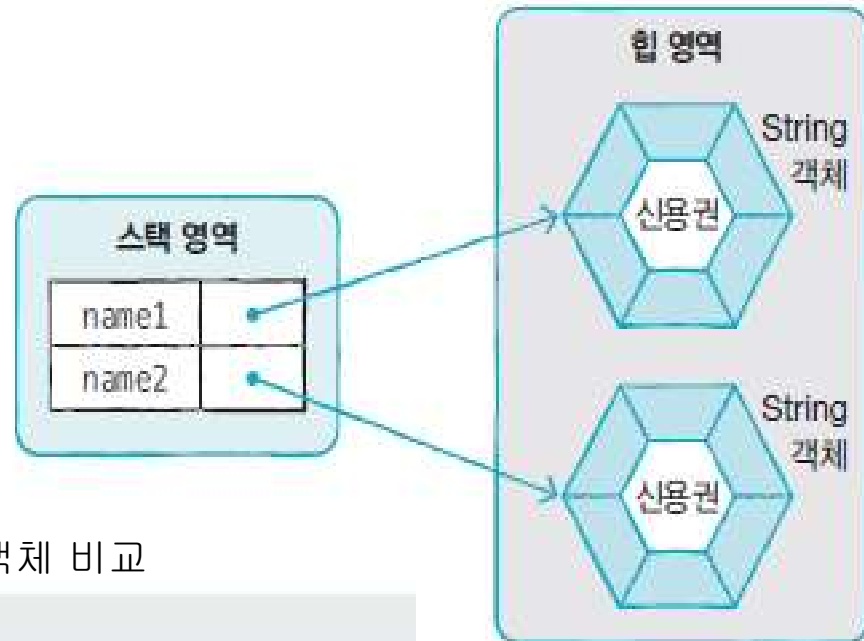


String 타입

❖ 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



String 타입

❖ 문자열 비교

- == : 번지 비교 (X)
- equals(): 문자열 비교 (O)

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

↑ ↑
원본 문자열 비교 문자열



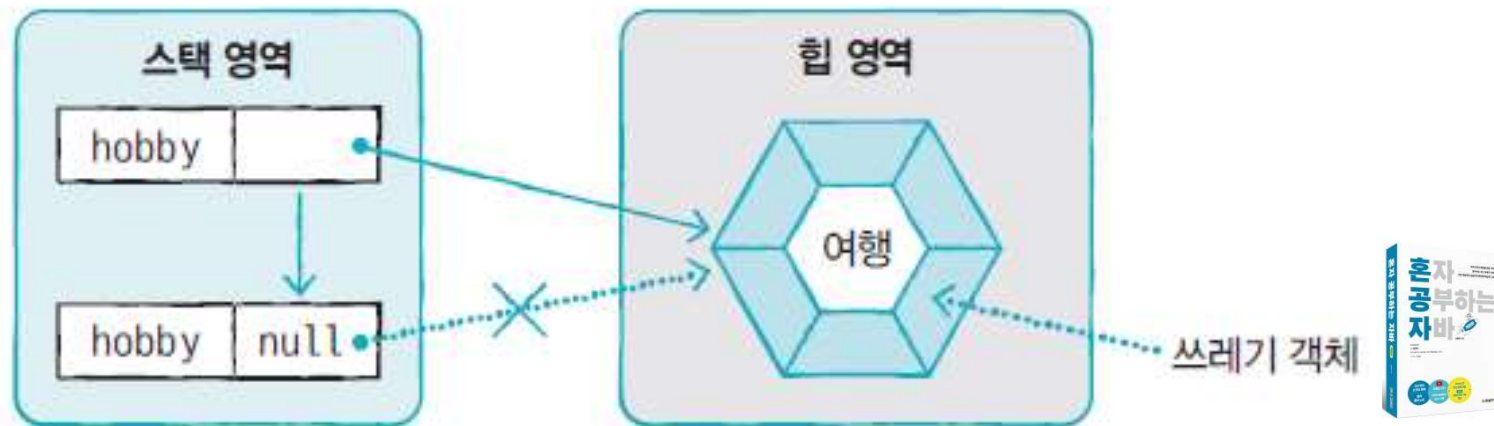
String 타입

- String 변수 초기값으로 null 대입
 - String 변수가 참조하는 객체가 없음을 의미

```
String hobby = null;
```

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

- 참조를 잃은 String 객체는 **쓰레기 수집기** (Garbage Collector) 통해 메모리에서 자동 제거

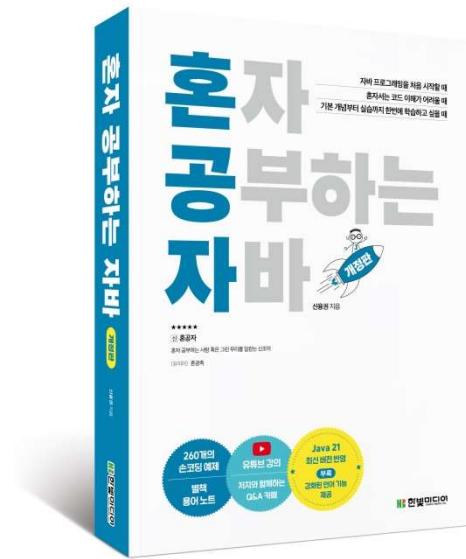


키워드로 끝내는 핵심 포인트

- **기본 타입**: byte, short, char, int, long, float, double, boolean 타입.
변수에 값 자체를 저장
- **참조 타입**: 기본 타입을 제외한 배열, 열거, 클래스, 인터페이스.
변수에 객체의 번지 저장
- **메모리 사용 영역**: JVM은 운영체제에서 할당받은 메모리 영역을
메소드 영역, 힙 영역, 스택 영역으로 구분해서 사용.
- **번지 비교**: 비교 연산자(==, !=)가 기본 타입에서 사용되면 값을 비교하지만
참조 타입에서 사용되면 번지를 비교
- **null**: 참조 변수는 객체를 참조하지 않는다는 뜻으로 null 값을 가질 수 있음
- **NullPointerException**: 참조 변수가 null 일 때
참조 변수를 통해 존재하지도 않는 객체를 사용하려고 할 경우 발



생



Thank You!