

# Primitives and References: Know Your Variables

Samkeun Kim <skim@hknu.ac.kr>

<http://cyber.hankyong.ac.kr>

# 변수 선언

자바에서는 변수 타입을  
확실히 구분한다. Giraffe  
객체를 Rabbit 변수에  
저장할 수 없다!



자바는 타입을 철저히 따진다.

```
Rabbit hopper = new Giraffe();
```

**Giraffe** 레퍼런스를 **Rabbit** 변수에 대입하려고 하면 대부분 컴파일러가 걸러낸다.

변수는 크게 두 가지 유형으로 나뉜다:

- 원시변수 (primitive)  
Integers, Booleans, and Floating point numbers.
- 객체 레퍼런스 (object reference)

변수는 타입을 가져야 한다

변수는 이름을 가져야 한다

```
int count;
```

↑  
type

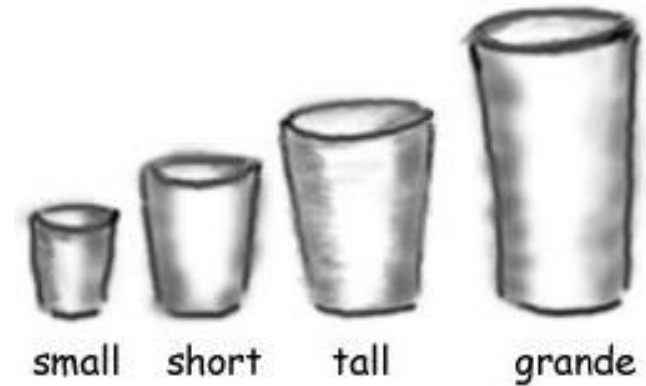
←  
name

# 자바 변수를 컵에 비유해 생각해보자

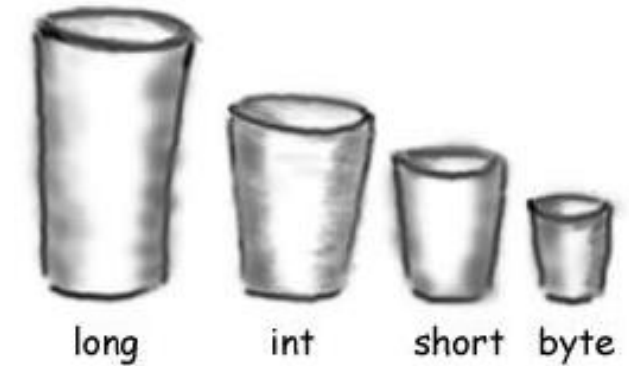
변수는 단순히 컵일 뿐이다. 뭔가를 담아 두기 위한 용도로 쓰인다.

컵은 **사이즈**와 **타입**을 갖는다.

**원시 변수**는 커피전문점에서 사용하는 컵에 비유할 수 있다.  
옆의 그림처럼 다양한 종류의 컵이 마련되어 있고 원하는  
사이즈로 주문하면 된다.



자바의 원시 변수 타입도 그 크기가 다양하며 각각 이름이 붙어있다.  
아래는 자바에서 사용하는 정수형 원시 변수 4 개에 해당한다:



한 가지 차이점: 자바에서는  
컵에 이름을 붙여줘야 한다.

# 자바에서 숫자 원시 변수 타입 6개

## Primitive Types

### boolean and char

Type	Bit Depth	Value Range
------	-----------	-------------

boolean	(JVM-specific)	<i>true</i> or <i>false</i>
char	16 bits	0 to 65535

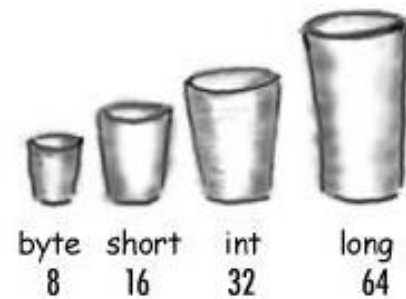
### numeric (all are signed)

#### integer

byte	8 bits	-128 to 127
short	16 bits	-32768 to 32767
int	32 bits	-2147483648 to 2147483647
long	64 bits	huge to huge

#### floating point

float	32 bits	varies
double	64 bits	varies



Primitive declarations  
with assignments:

```
int x;  
x = 234;  
byte b = 89;  
boolean isFun = true;  
double d = 3456.98;  
char c = 'f';  
int z = x;  
boolean isPunkRock;  
isPunkRock = false;  
boolean powerOn;  
powerOn = isFun;  
long big = 3456789;  
float f = 32.5f;
```

# 넘치면 안되죠!!

값이 변수에 들어갈 수 있는지 반드시 확인해야 한다.



*작은 컵에 너무 많이 집어넣을 수는 없다!*

```
int x = 24;  
(X) byte b = x;
```

변수에 값을 대입할 때 다음과 같은 방법을 쓸 수 있다:

- 등호 옆에 리터럴 값을 입력하는 방법 ( $x = 12$ ,  $\text{isGood} = \text{true}$  등)
- 한 변수의 값을 다른 변수에 대입하는 방법 ( $x = y$ )
- 위의 두 가지 방법을 결합한 방법 ( $x = y + 43$ )

## 실습과제 3-1 Sharpen Your Pencil

아래 코드가 하나의 메소드에 있다고 가정했을 때 적합한 라인들을 고르시오.

```
1. int x = 34.5;
2. boolean boo = x;
3. int g = 17;
4. int y = g;
5. y = y + 10;
6. short s;
7. s = y;
8. byte b = 3;
9. byte v = b;
10. short n = 12;
11. v = n;
12. byte k = 128;
```

# 키워드와 변수명

변수명으로 어떤 것을 쓸 수 있나?

규칙은 간단하다. 아래 규칙에 따라 클래스, 메소드, 변수에 이름을 붙일 수 있다:

- 반드시 알파벳 글자, 밑줄(\_), 또는 달러 기호(\$)로 시작해야 한다.
- 숫자로 시작하면 안 된다.
- 두 번째 문자부터는 숫자도 쓸 수 있다.
- 위의 두 가지 규칙을 지키고 자바 예약어만 사용하지 않으면 어떤 이름이든지 마음대로 사용 가능하다.

*예약어는 컴파일러가 인식할 수 있는 키워드(및 기타 사항)이다.*

*정말로 컴파일러를 혼란하게 만들고 싶다면 예약어를 변수명으로 사용해 보라!*



컴파일러에서 인식할 수 있는 키워드를 비롯한 단어들을 **예약어**(reserved words)라고 한다.

boolean	byte	char	double	float	int	long	short	public	private
protected	abstract	final	native	static	strictfp	synchronized	transient	volatile	if
else	do	while	switch	case	default	for	break	continue	assert
class	extends	implements	import	instanceof	interface	new	package	super	this
catch	finally	try	throw	throws	return	void	const	goto	enum

# Dog 객체 제어하는 방법

(이제 원시 변수를 선언하고 값을 대입하는 방법은 터득했다.)

하지만 원시 변수가 아닌 변수는 어떻게 해야 할까?

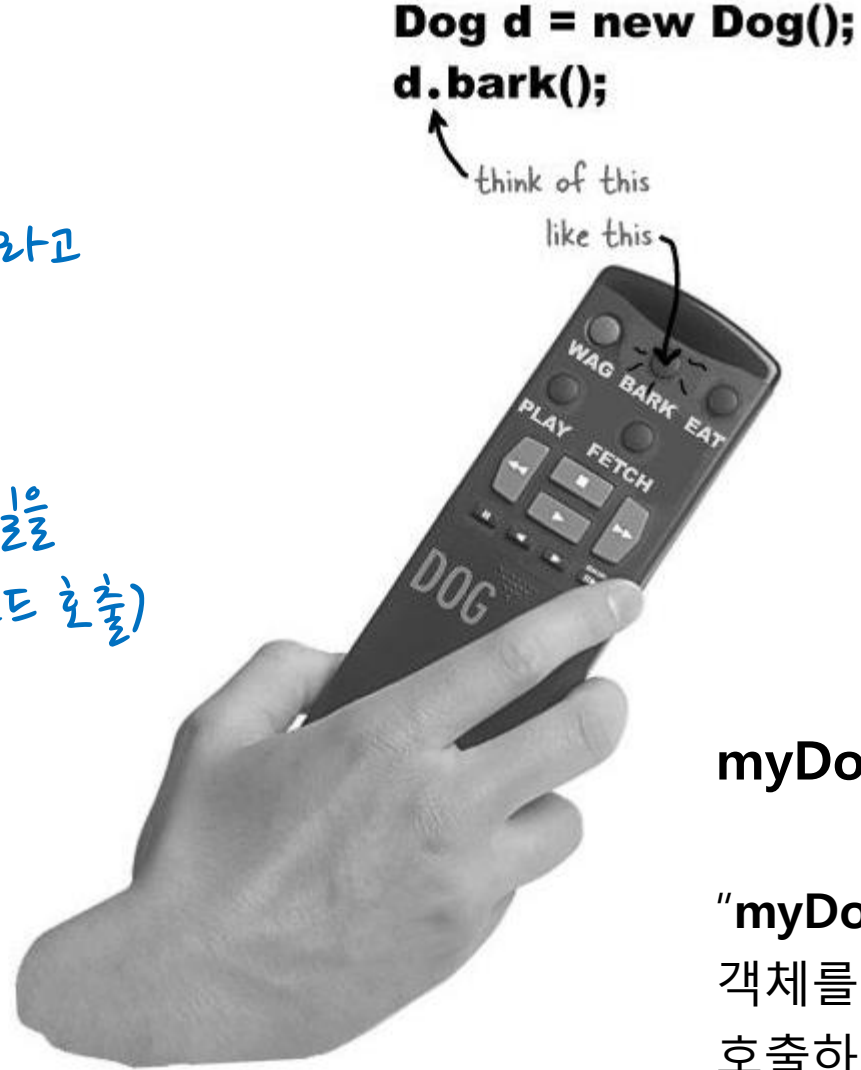
즉, 객체에 대해서는 어떻게 해야 하나?

- 객체 변수라는 것은 없다!!
- 객체 레퍼런스 변수라는 것은 있다.
- 객체 레퍼런스 변수에는 그 객체에 접근하는 방법을 알려주는 정보가 들어 있다.
- 객체 레퍼런스에 객체 자체가 들어있는 것이 아니다. 포인터 같은 것이 들어있을 뿐이다. 아니면 객체의 주소가 들어있다고 생각해도 무방하다!

객체 레퍼런스 변수 안에 무엇이 들어있는지 결코 알 수 없다.

다만 단 하나의 객체를 가리킨다는 사실만은 확실하다.

Dog 레퍼런스 변수는  
Dog에 대한 리모컨이라고  
생각해도 좋다.  
이 리모컨을 이용하여  
해당 객체에 어떤 일을  
지시할 수 있다 (메소드 호출)



**myDog.bark();**

"myDog라는 변수로 참조할 수 있는  
객체를 이용하여 **bark()** 메소드를  
호출하라"

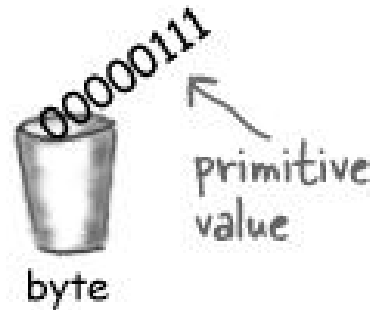
# 객체 레퍼런스는 단순히 또다른 변수에 불과하다

즉, 컵에 들어가는 것이라고 보면 된다.

다만 이번에는 그 컵 안에 리모컨이 들어간다고 볼 수 있다.

원시 변수:

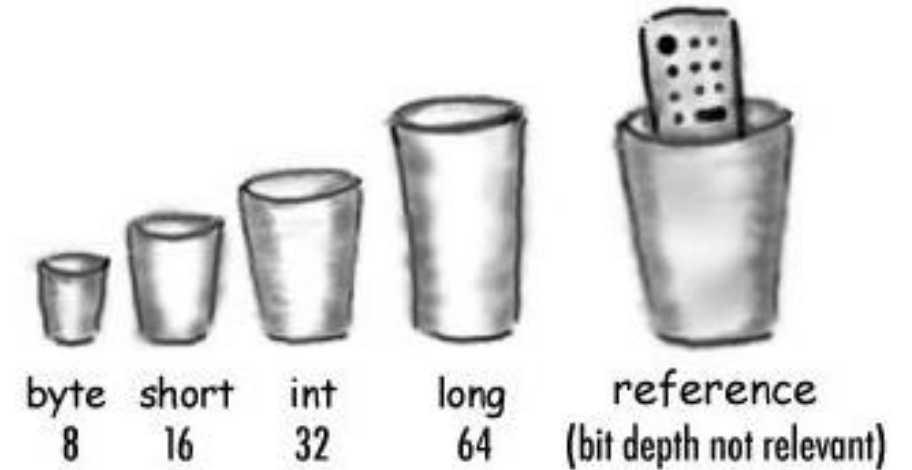
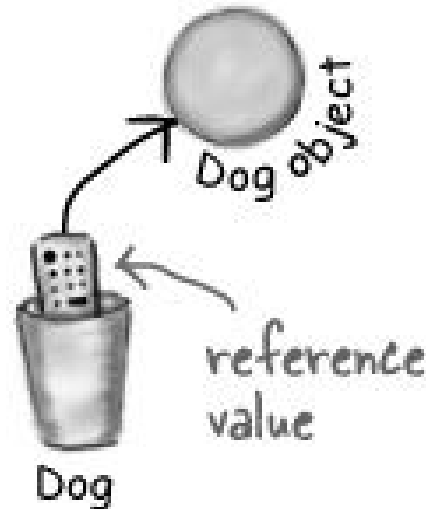
**byte x = 7;**



레퍼런스 변수:

**Dog myDog = new Dog();**

객체 자체는 변수에 저장되지 않는다.



# 객체 선언, 생성과 대입의 3단계

1. 레퍼런스 변수 선언

**Dog myDog = new Dog();**

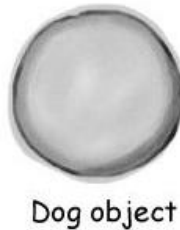


<sup>1</sup>  
Dog myDog <sup>3</sup> = <sup>2</sup>  
new Dog() ;

JVM에게 레퍼런스 변수를 저장할 공간을 할당해 달라고 요청

2. 객체 생성

**Dog myDog = new Dog();**

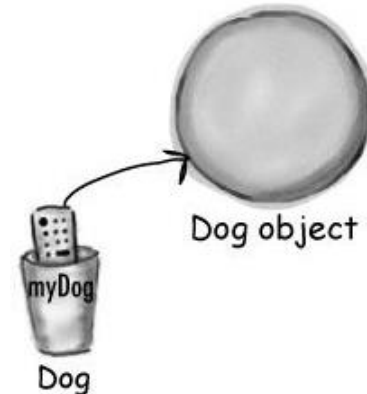


JVM에게 **힙**에 새로운 Dog 객체를 위한 공간을 할당해 달라고 요청

3. 객체와 레퍼런스 연결

**Dog myDog = new Dog();**

새로운 Dog 객체를 myDog라는 레퍼런스 변수에 대입한다.



# 가비지 컬렉션 기능이 있는 힙에서의 삶

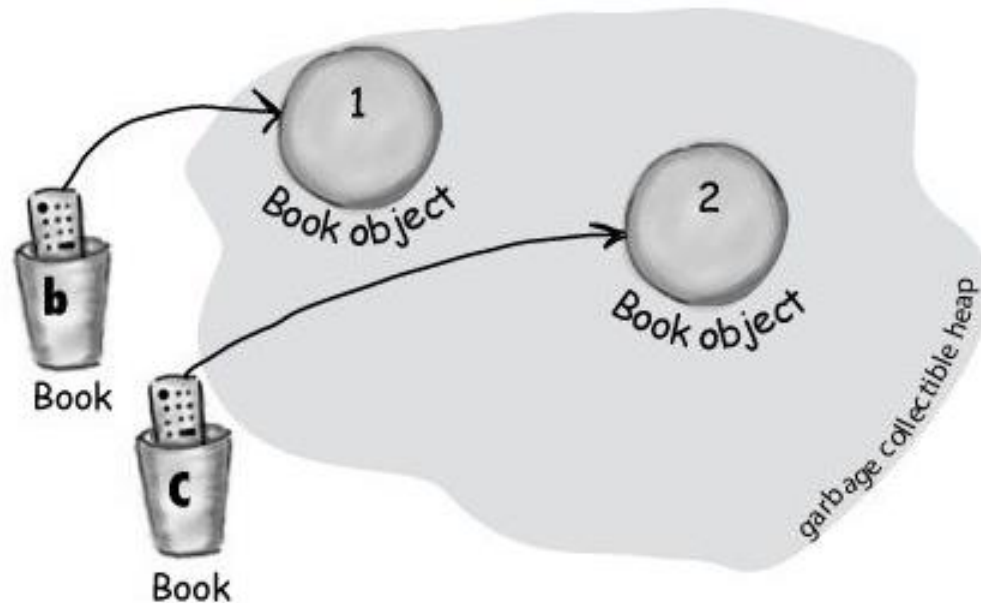
```
Book b = new Book();  
Book c = new Book();
```

**Book** 레퍼런스 두 개를 선언하자.  
아울러 두 개의 새로운 Book 객체도 생성한다.  
그리고 생성한 **Book** 객체를 레퍼런스 변수에 대입한다.

이제 두 **Book** 객체는 힙에서 살고 있다.

References: 2

Objects: 2



**Book d = c;**

새로운 **Book** 레퍼런스 변수를 선언한다.

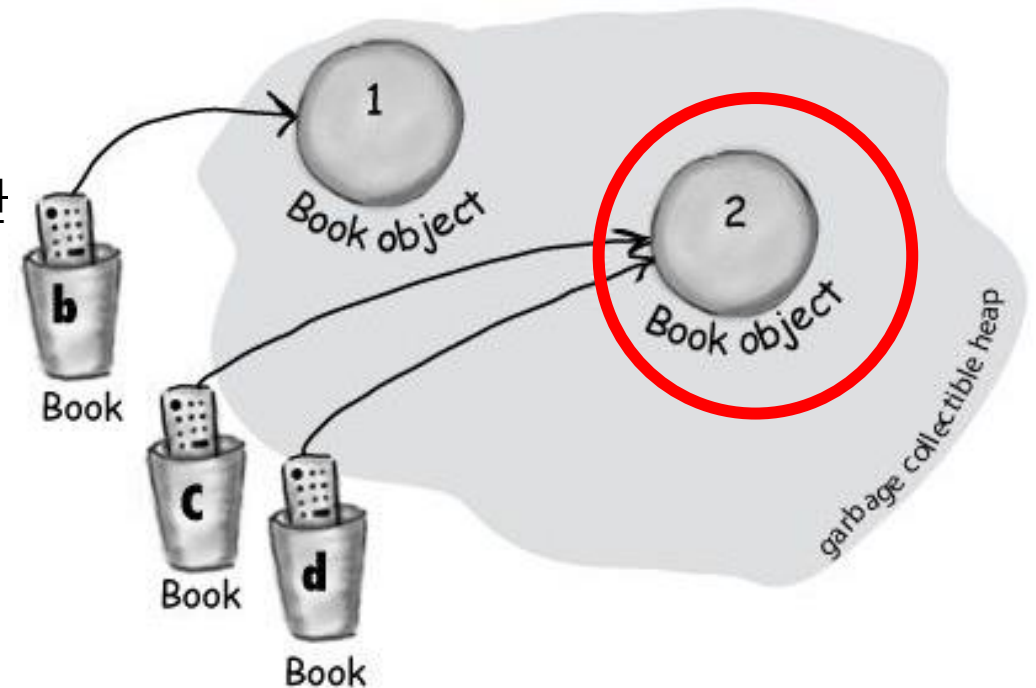
이번에는 새로운 **Book** 객체를 생성하는 것 대신 변수 **c** 의 값을 변수 **d** 에 대입한다.

⇒ 이제 **c** 와 **d** 는 동일한 객체를 참조한다.

변수 **c**, **d** 는 동일한 값의 두 개의 다른 사본을 저장한 하나의 TV에 두 개의 리모컨이 프로그래밍되었다.

References: 3

Objects: 2

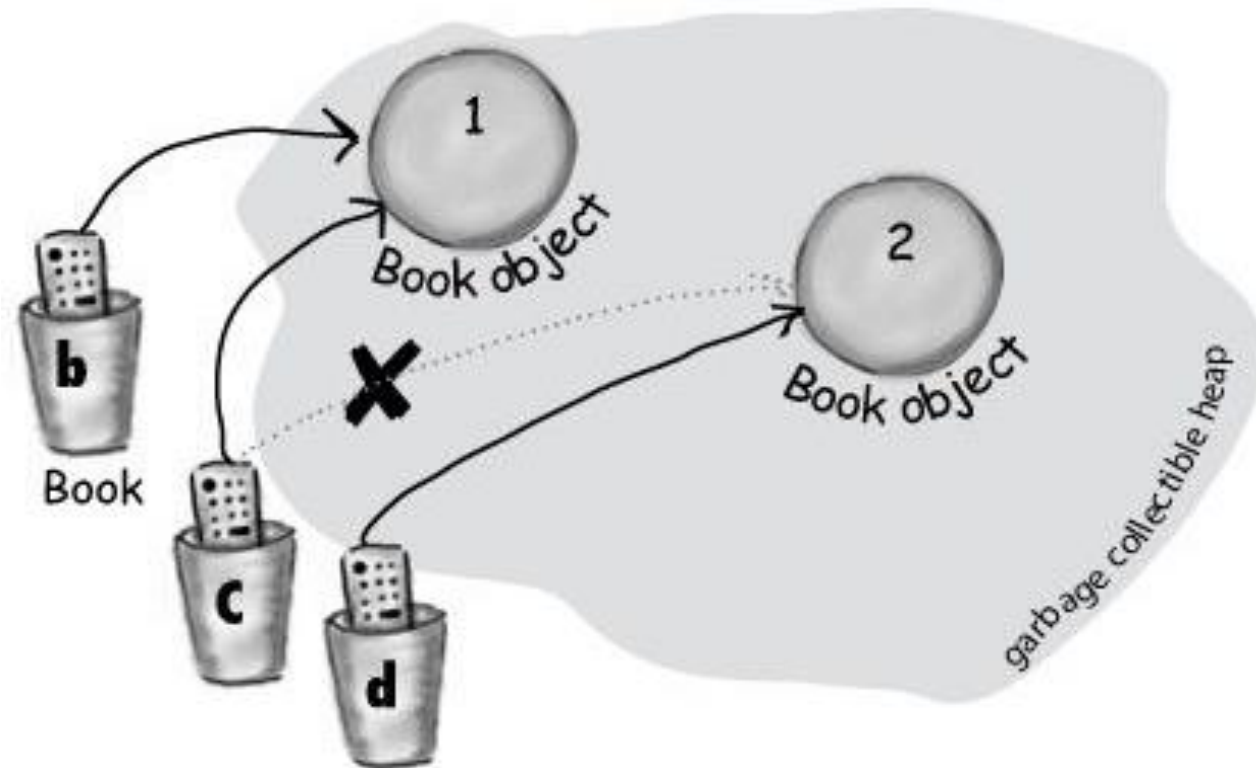


**c = b;**

변수 **b** 의 값을 변수 **c** 에 대입한다.

이제 **b** 와 **c** 는 동일한 객체를 참조한다.

References: 3  
Objects: 2





# 힙에서의 삶과 죽음

```
Book b = new Book();
```

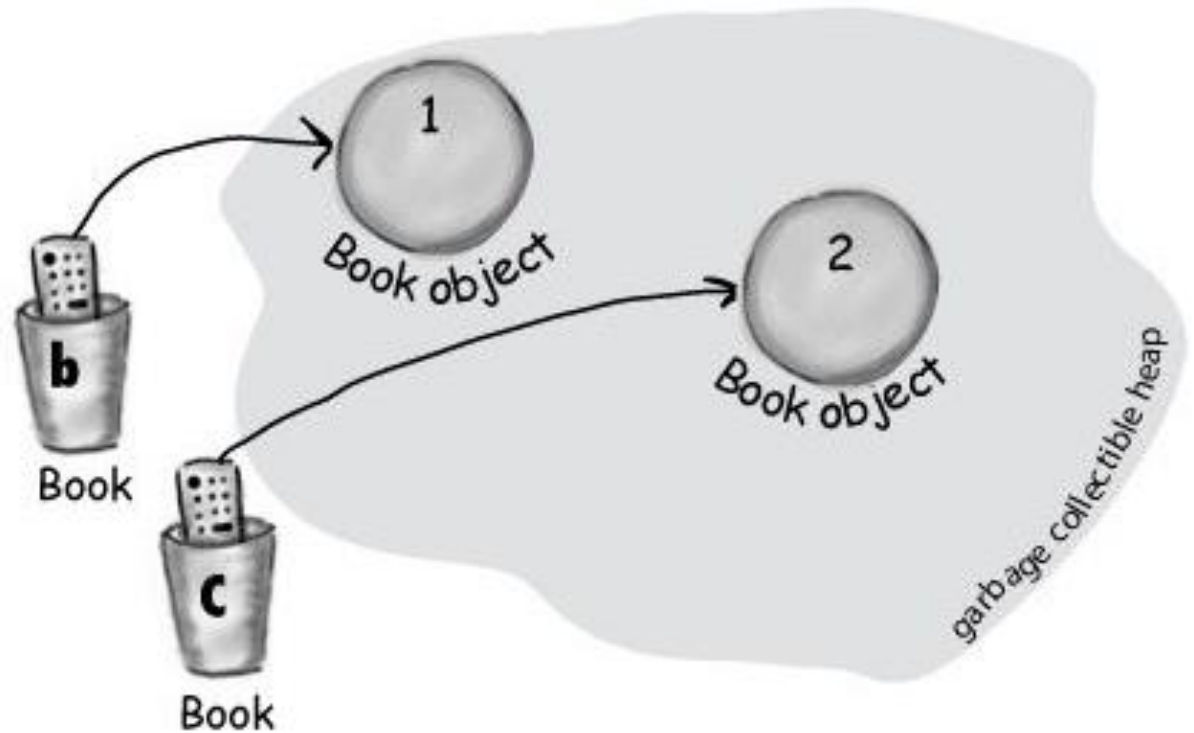
```
Book c = new Book();
```

두 개의 **Book** 레퍼런스 변수를 선언한다.  
두 개의 새로운 **Book** 객체를 생성한다.  
**Book** 객체를 레퍼런스 변수에 대입한다.

두 **Book** 객체는 이제 힙에서 살고 있다.

Active References: 2

Reachable Objects: 2



**b = c;**

변수 **c** 의 값을 변수 **b** 에 대입한다.

변수 **c** 의 비트가 복사되고 그 새로운 복사본이 변수 **b** 에 채워진다.

두 변수에는 동일한 값이 들어있다.

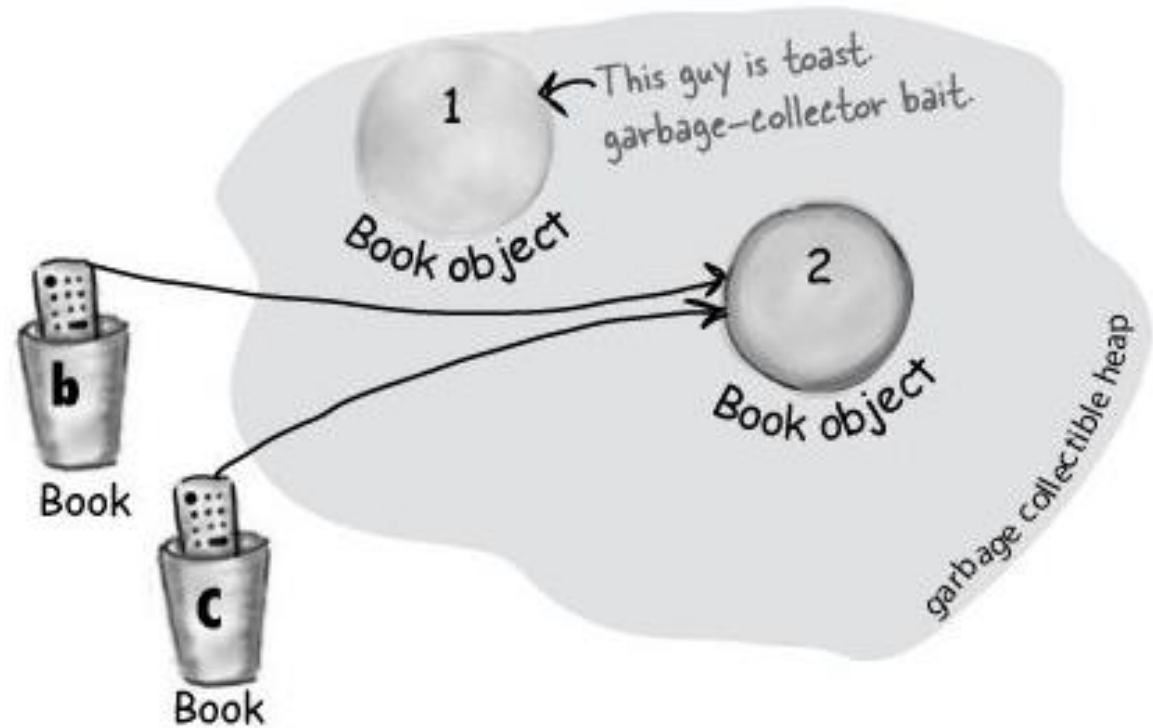
객체 1(Book object 1)은 버려지고 가비지 컬렉션 (GC) 대상이 된다.

Active References: 2

Reachable Objects: 1

Abandoned Objects: 1

처음 1번 객체는 더 이상 아무런 레퍼런스도 남아있지 않으므로 따라서 접근할 수가 없다.



**c = null;**

변수 **c** 에 **null** 값을 대입한다.

이것은 **c** 를 **null 레퍼런스**로 만들어준다 => 어떤 것도 참조하지 않는다는 것을 의미한다!

그래도 여전히 레퍼런스 변수 => 여전히 또 다른 **Book** 객체를 그것에 대입할 수 있다.

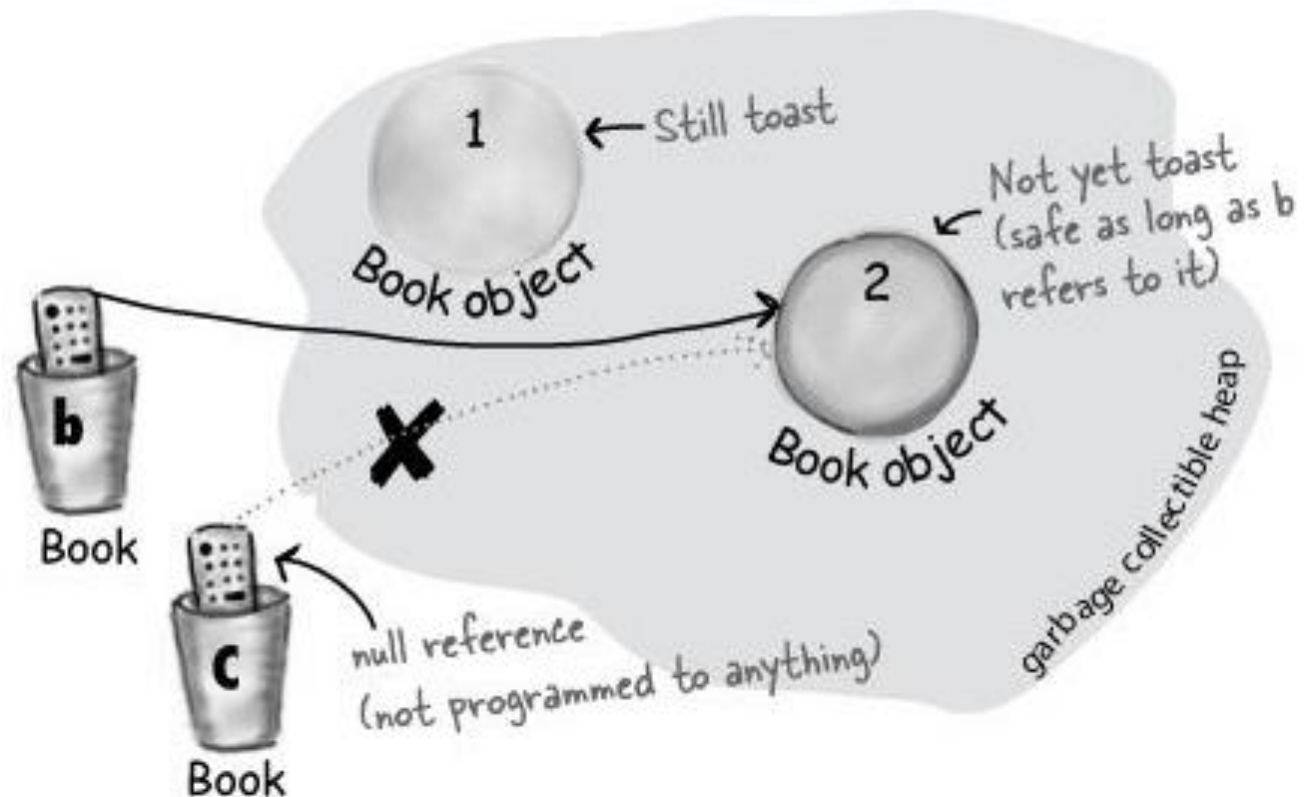
객체 2(**Book** object 2)는 여전히  
활성 레퍼런스 (**b**)가 있고 활성  
레퍼런스가 있는 한 **GC**의  
대상이 되지 않는다.

Active References: 1

*null* References: 1

Reachable Objects: 1

Abandoned Objects: 1



# 배열은 쟁반 위의 컵들과 같다

1. **int** 배열 변수를 선언한다. 배열 변수는 배열 객체의 리모컨이다.

```
int[] nums;
```

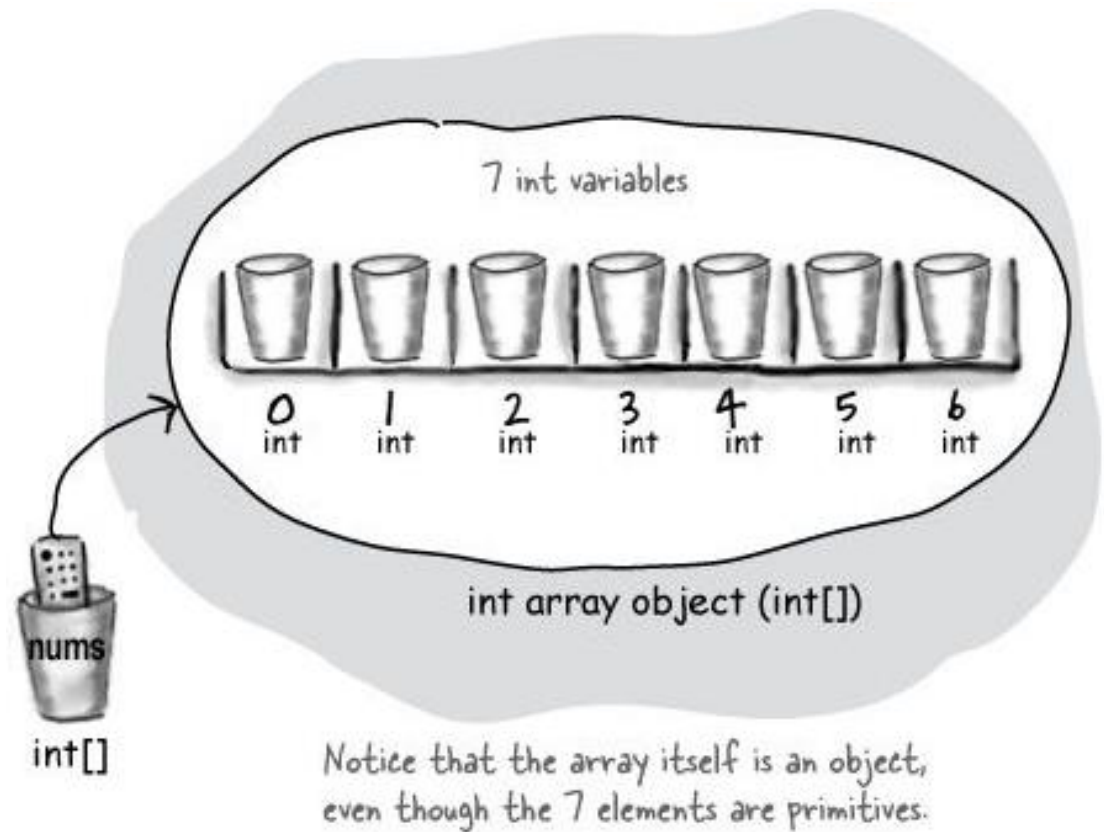
2. 길이가 7인 새로운 **int** 배열을 생성하고 앞에서 만든 **nums**라는 **int[]** 변수에 대입한다.

```
nums = new int[7];
```

3. 배열의 각 원소에 **int** 값을 대입한다. **int** 배열에 있는 원소는 **int** 변수만 있다.

7 int variables

```
nums[0] = 6;  
nums[1] = 19;  
nums[2] = 44;  
nums[3] = 42;  
nums[4] = 10;  
nums[5] = 20;  
nums[6] = 1;
```



# 배열도 객체이다

배열의 모든 원소는 그냥 변수이다.

즉 원시 변수이거나 레퍼런스 변수이다.

이런 유형의 변수에 집어 넣을 수 있는 것은 모두 그 유형의 배열 원소로 대입할 수 있다.

**int**형 배열 (**int**[])의 각 원소에는 **int**가 들어간다.

그렇다면 **Dog** 배열 (**Dog**[])의 각 원소에는 **Dog**가 들어가는가? NO.

레퍼런스 변수에는 객체 자체가 아닌 레퍼런스(리모컨)가 들어가게 된다.

*배열은 원시 변수의 배열이든 객체 레퍼런스에 대한 배열이든 상관없이 항상 객체이다!*

# Dog 배열을 만들어보자

1. Dog 배열 변수를 선언한다.

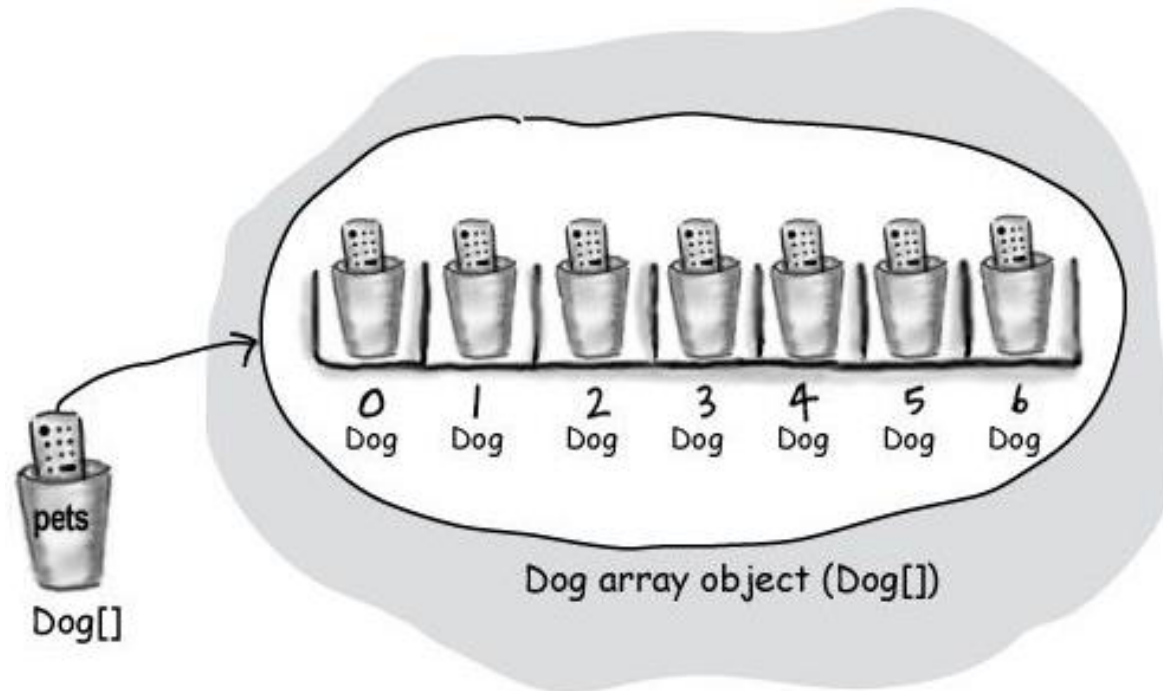
```
Dog[] pets;
```

2. 길이가 7 인 새로운 Dog 배열을 만들어서 앞서 선언한 Dog[] 변수 **pets**에 대입한다.

```
pets = new Dog[7];
```

*What's missing?*

*No actual Dog objects!*



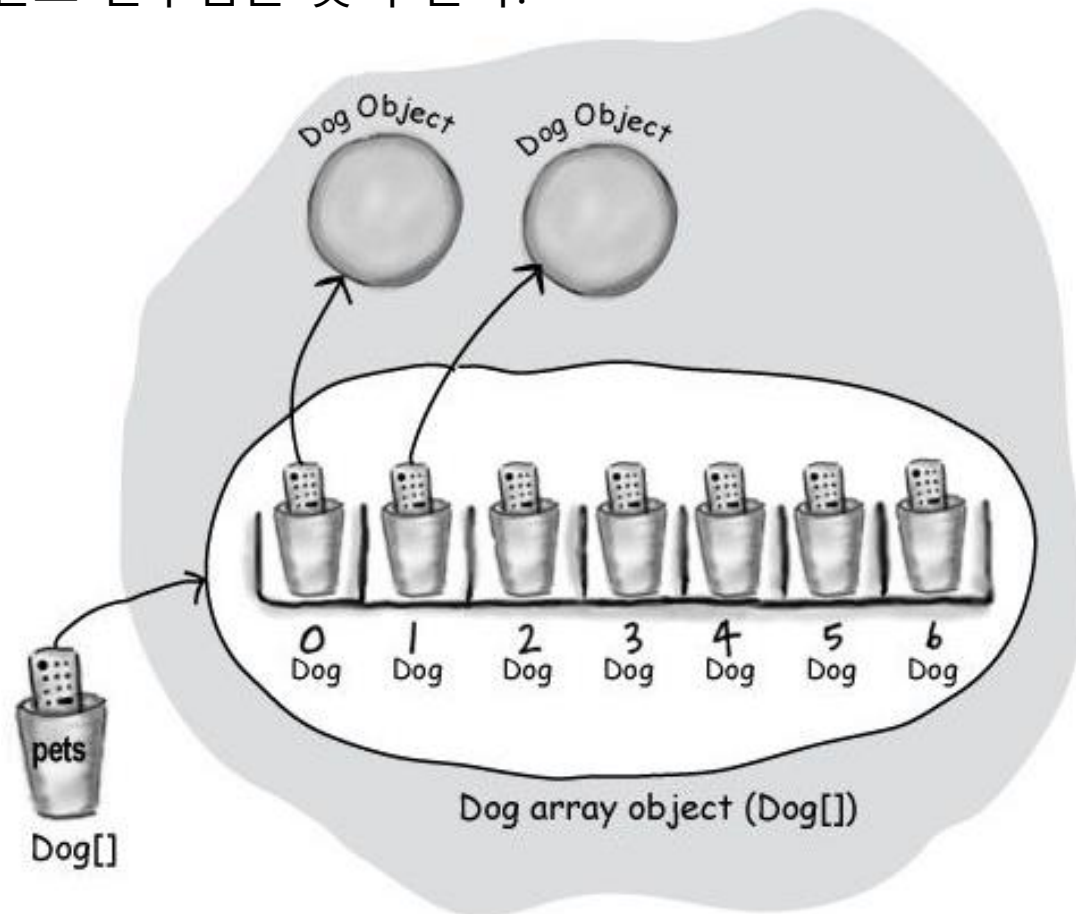
# Make an array of Dogs

3. 새로운 **Dog** 객체를 생성하고 그 객체를 배열 원소에 대입한다.  
**Dog** 배열에 들어있는 원소는 **Dog** 레퍼런스 변수임을 잊지 말자!  
여전히 실제 **Dog** 객체가 필요하다:

```
pets[0] = new Dog();  
pets[1] = new Dog();
```



Dog
name
bark() eat() chaseCat()



# Dog를 제어해보자 (레퍼런스 변수 사용)

```
Dog fido = new Dog();  
fido.name = "Fido";
```

**Dog** 객체를 만들고 **fido**라는 레퍼런스 변수에 대해 점 연산자를 사용하여 **name** 변수에 접근하였다.

**fido** 변수를 사용하여 **dog**로 하여금 짖거나 (**bark()**) 고양이를 쫓아가도록(**chaseCat()**) 할 수도 있다.

```
fido.bark();  
fido.chaseCat();
```





## *What happens if the Dog is in a Dog array?*

**Dog**가 배열에 들어있을 때는 **fido** 같은 실제 객체 변수명이 없다!

대신 배열 표기법을 이용하여 리모컨의 버튼(점 연산자)을 배열의 특정 **인덱스**에 있는 객체에 집어넣을 수 있다:

```
Dog[] myDogs = new Dog[3];  
myDogs[0] = new Dog();  
myDogs[0].name = "Fido";  
myDogs[0].bark();
```

## 실습과제 3-2 Dog 예제

Dog
name
bark() eat() chaseCat()

### Output

```
File Edit Window Help Howl
%java Dog
null says Ruff!
last dog's name is Bart
Fred says Ruff!
Marge says Ruff!
Bart says Ruff!
```

```
1 package com.ex3;
2
3 public class Dog {
4     String name;
5     public static void main(String[] args) {
6         Dog dog1 = new Dog();
7         dog1.bark();
8         dog1.name = "Bart";
9
10        Dog[] myDogs = new Dog[3];
11        myDogs[0] = new Dog();
12        myDogs[1] = new Dog();
13        myDogs[2] = dog1;
14
15        myDogs[0].name = "Fred";
16        myDogs[1].name = "Marge";
17
18        System.out.print("last dog's name is ");
19        System.out.println(myDogs[2].name);
20
21        int x = 0;
22        while (x < myDogs.length) {
23            myDogs[x].bark();
24            x = x+1;
25        }
26    }
27    public void bark() {
28        System.out.println(name + " says Ruff!");
29    }
30
31    public void eat() { }
32
33    public void chaseCat() { }
34 }
```

## 실습과제 3-3 Compile and run without exception

Each of the Java files on this page represents a complete source file. Your job is to play compiler and determine whether each of these files will compile and run without exception. If they won't, how would you fix them?

**A**

```
class Books {
    String title;
    String author;
}

class BooksTestDrive {
    public static void main(String [] args) {

        Books [] myBooks = new Books[3];
        int x = 0;
        myBooks[0].title = "The Grapes of Java";
        myBooks[1].title = "The Java Gatsby";
        myBooks[2].title = "The Java Cookbook";
        myBooks[0].author = "bob";
        myBooks[1].author = "sue";
        myBooks[2].author = "ian";

        while (x < 3) {
            System.out.print(myBooks[x].title);
            System.out.print(" by ");
            System.out.println(myBooks[x].author);
            x = x + 1;
        }
    }
}
```

**B**

```
class Hobbits {

    String name;

    public static void main(String [] args) {

        Hobbits [] h = new Hobbits[3];
        int z = 0;

        while (z < 4) {
            z = z + 1;
            h[z] = new Hobbits();
            h[z].name = "bilbo";
            if (z == 1) {
                h[z].name = "frodo";
            }
            if (z == 2) {
                h[z].name = "sam";
            }
            System.out.print(h[z].name + " is a ");
            System.out.println("good Hobbit name");
        }
    }
}
```

## 실습과제 3-4 Code Magnets

A working Java program is all scrambled up on the fridge. Can you reconstruct the code snippets to make a working Java program that produces the output listed below? Some of the curly braces fell on the floor and they were too small to pick up, so feel free to add as many of those as you need!

```
int y = 0;
```

```
ref = index[y];
```

```
islands[0] = "Bermuda";  
islands[1] = "Fiji";  
islands[2] = "Azores";  
islands[3] = "Cozumel";
```

```
int ref;  
while (y < 4) {
```

```
System.out.println(islands[ref]);
```

```
index[0] = 1;  
index[1] = 3;  
index[2] = 0;  
index[3] = 2;
```

```
String [] islands = new String[4];
```

```
System.out.print("island = ");
```

```
int [] index = new int[4];
```

```
y = y + 1;
```

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

File Edit Window Help Bkln

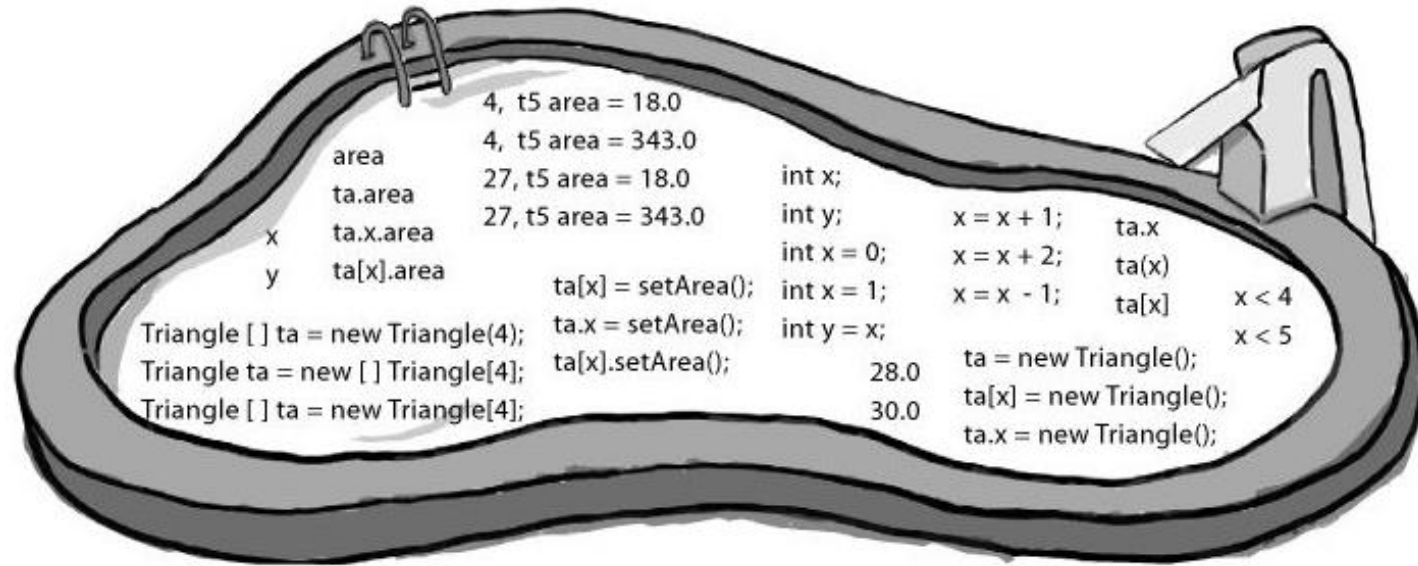
```
% java TestArrays  
island = Fiji  
island = Cozumel  
island = Bermuda  
island = Azores
```

# 실습과제 3-5 Pool Puzzle

```

class Triangle {
    double area;
    int height;
    int length;
    public static void main(String [] args) {
        _____
        while ( _____ ) {
            _____
            _____
            _____
            _____
            System.out.print("triangle "+x+", area");
            System.out.println(" = " + _____ .area);
            _____
        }
        _____
        x = 27;
        Triangle t5 = ta[2];
        ta[2].area = 343;
        System.out.print("y = " + y);
        System.out.println(", t5 area = "+ t5.area);
    }
    void setArea() {
        _____ = (height * length) / 2;
    }
}

```



## Output

```

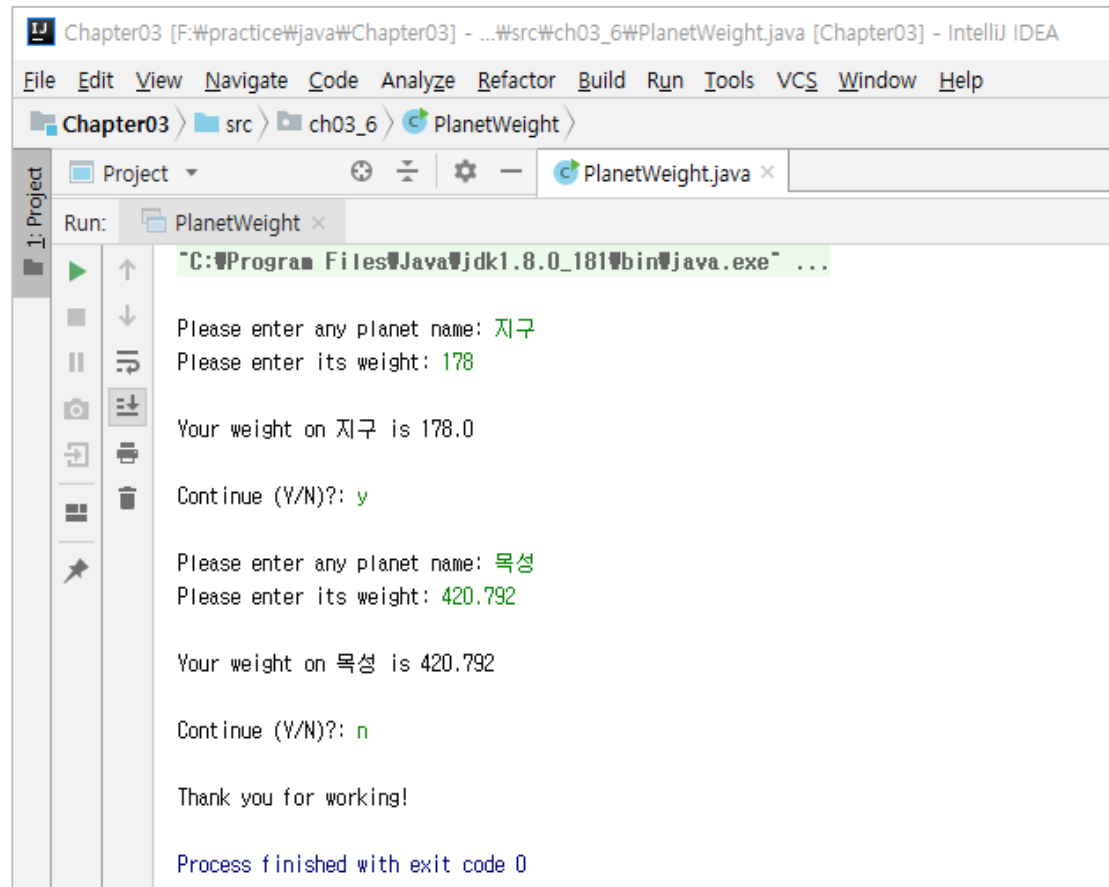
File Edit Window Help Bermuda
%java Triangle
triangle 0, area = 4.0
triangle 1, area = 10.0
triangle 2, area = 18.0
triangle 3, area = ____
y = ____

```

## 실습과제 3-6 The PlanetWeight Program

Planet name과 weight를 입력 받아 아래 Sample run처럼 출력해주는 프로그램을 작성하시오.  
단, 적어도 9개의 행성과 해당 행성의 정확한 무게를 입력하시오.

Sample run:



```
Chapter03 [F:\practice\java\Chapter03] - ...src\ch03_6\PlanetWeight.java [Chapter03] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Chapter03 > src > ch03_6 > PlanetWeight
Project PlanetWeight.java x
Run: PlanetWeight x
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Please enter any planet name: 지구
Please enter its weight: 178
Your weight on 지구 is 178.0
Continue (Y/N)? y
Please enter any planet name: 목성
Please enter its weight: 420.792
Your weight on 목성 is 420.792
Continue (Y/N)? n
Thank you for working!
Process finished with exit code 0
```

