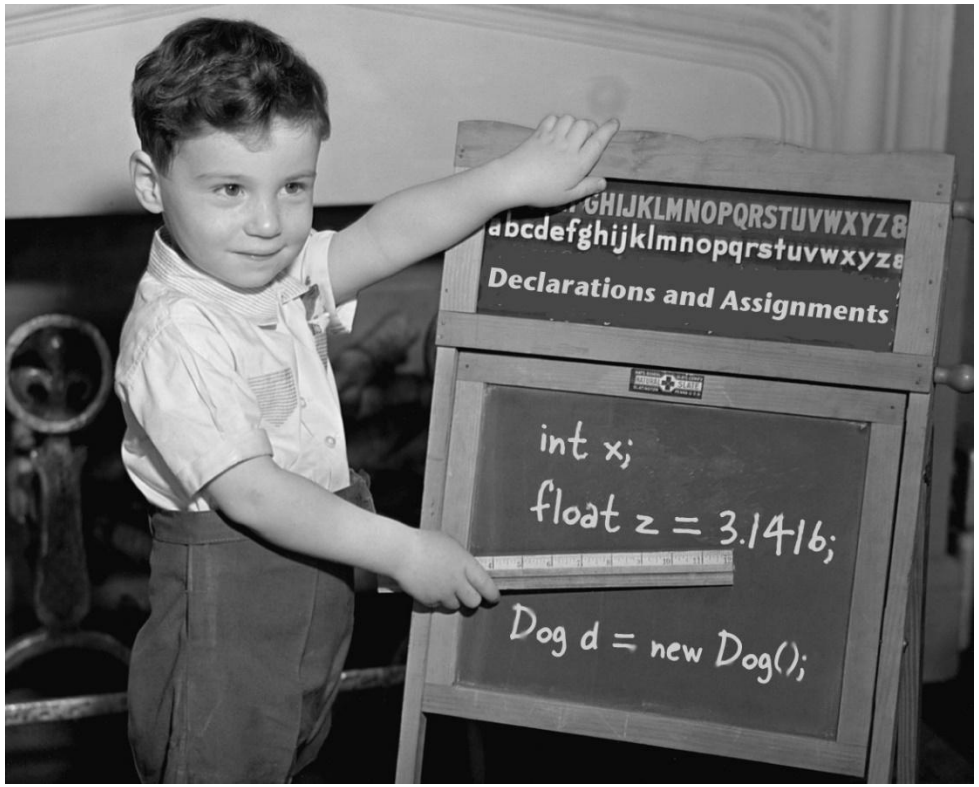
3장. 네 변수를 알라

1. 원시 변수와 레퍼런스 변수에 대해 알아봅니다.
2. 변수가 저장되는 힙에 대해 알아봅니다.
3. 배열에 대해 알아봅니다.

네 변수를 알라

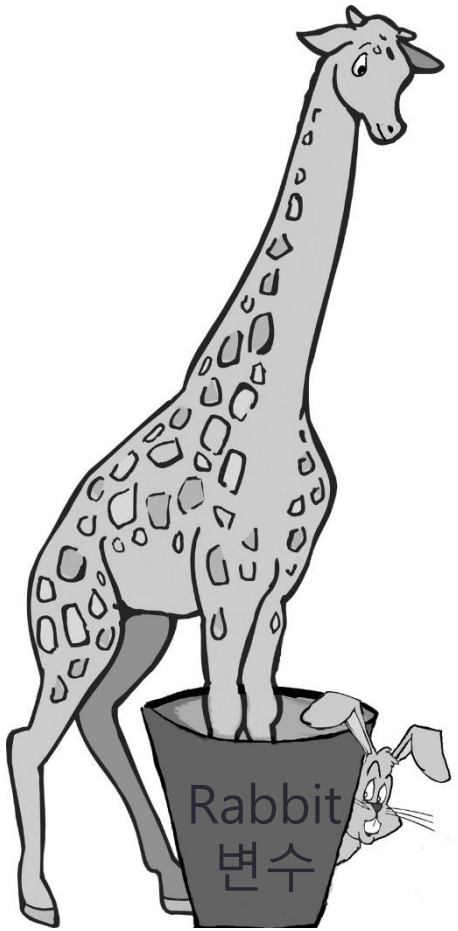
- 변수의 용도
  - 객체의 상태
  - 지역변수
  - 매개 변수(parameters) - 인자(arguments)
  - 리턴 유형



변수 선언

- 유형 안전성 (type-safety)
  - 자바에서는 유형을 철저하게 따집니다.
    → NetBeans IDE 사용시 변수 유형에 오류 있을 때 빨간 밑줄이 보임.

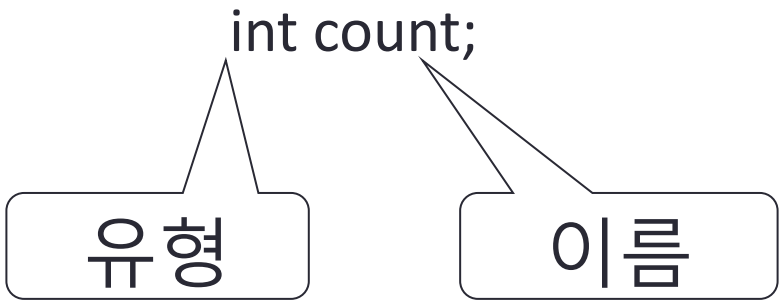
~~Rabbit hopper = new Giraffe();~~
~~hopper.hop();~~



p.92

변수 선언

- 변수를 선언하는 데 있어서 필요한 것
  - 변수 ~~유형~~ 자료형 (int, double, String 등)
  - 변수 이름 (count, size, name, myDog 등)



원시 유형(Primitive Type)

- 변수를 컵에 비유해봅시다.

모 커피 가게에서 파는 컵 크기

small short tall grande

자바 원시 정수 변수의 크기

long int short byte

원시 유형

- 정수와 부동소수점 유형

정수 유형

byte short int long
8 16 32 64

부동소수점 유형

float double
32 64

원시 유형의 크기

유형	비트 수	범위
부울과 문자		
boolean	JVM에 따라 다름	true, false
char	16 비트	0 ~ 65535
숫자 (모두 부호가 있음)		
정수 *cf.* java.math.BigDecimal		
byte	8 비트	-128 ~ 127
short	16 비트	-32768 ~ 32767
int	32 비트	-2147483468 ~ 2147483467
long	64 비트	-아주 큰 값 ~ 아주 큰 값
부동소수점 소수		
float	32 비트	다양함
double	64 비트	다양함

참고. BigDecimalTest.java

```
6   package cse.oop2.ch03.bigdecimal;
7
8   import java.math.BigDecimal;
9
10  /**...4 lines */
14  public class BigDecimalTest {
15
16      /**
17       * @param args the command line arguments
18       */
19      public static void main(String[] args) {
20          BigDecimal n1 = new BigDecimal("12345.12345");
21          BigDecimal n2 = new BigDecimal("123456789012345678901");
22          System.out.format("n1 = %s, n2 = %s%n", n1, n2);
```

```
24          double d1 = 12345.12345;
25          double d2 = 12345678901234567890.0;
26          System.out.format("d1 = %f, d2 = %f%n", d1, d2);
27
28          // https://stackoverflow.com/questions/322749/retain-precision-with-double-in-java
29          n1 = new BigDecimal("5.6");
30          n2 =new BigDecimal("5.8");
31
32          BigDecimal result1 = n1.add(n2);
33          System.out.format("n1 = %s, n2 = %s, result1 = %s%n", n1, n2, result1);
34
35          d1 = 5.6;
36          d2 = 5.8;
37          double result2 = d1 + d2;
38          System.out.format("d1 = %s, d2 = %s, result2 = %s%n", n1, n2, result2);
39      }
40
41  }
```

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ java_maven ---
n1 = 12345.12345, n2 = 12345678901234567890
d1 = 12345.123450, d2 = 12345678901234567000.000000
n1 = 5.6, n2 = 5.8, result1 = 11.4
d1 = 5.6, d2 = 5.8, result2 = 11.399999999999999
```

원시 변수 선언 및 대입 예

- 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 = 3456789L;
- float f = 32.5f;

원시값 대입 시 주의 사항: 오버플로우

- 넘치게 하지 마세요. → overflow

변수에 값을 대입할 때는 유형을 잘 맞춰야 합니다. 해당 유형의 변수에 들어갈 수 없는 너무 큰 값을 대입하면 (컵의 크기에 비해 너무 많은 양을 집어넣으려고 하면) 넘치고 말 테니까요.

~~int x = 24;~~
~~byte b = x;~~

byte b = (byte)x; // type casting

이와 반대로 큰 컵에 적은 양의 내용물을 집어넣는 것은 가능합니다.

```
byte b = 24;
int x = b;
```

p.94

원시값 대입 방법

- 변수에 값을 대입하는 방법
  - 등호 옆에 리터럴 값 입력
    - x = 12, isGood = true 등
  - 한 변수의 값을 다른 변수에 대입
    - x = y
  - 위의 두 가지 방법을 결합한 방법
    - x = y + 43

```
int size = 32;
char initial = 'j';
double d = 456.709;
boolean isCrazy;
isCrazy = true;
int y = x + 456;
```

키워드와 변수명

- 명명 규칙 (클래스, 메서드, 변수)
  - 알파벳, 밑줄(_), 달러 기호($)로 시작해야 합니다. ($는 주로 컴파일러가 식별자 만들 때 사용)
  - 두 번째 문자부터는 숫자를 쓸 수도 있습니다.
    - 하지만 숫자로 시작할 수는 없습니다.
  - 예약어는 이름으로 쓸 수 없습니다.
  - 대소문자 구분합니다.
  - Camel Case: 클래스 이름 - 대문자로 시작, 인스턴스 변수 & 메서드 이름 - 소문자로 시작
  - 한글도 식별자로 사용 가능하며, 이 경우에는 대소문자 구분 없음.

boolean char byte short int long float double

B C B S I L F D

Be Careful! Bears Shouldn’t Ingest Large Furry Dogs.

예약어

예약어에는 다음과 같은 것이 있습니다.
이런 예약어를 클래스, 메서드, 변수 이름으로 사용하면
반드시 문제가 생깁니다. 절대 쓰지 마세요.

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

(참고) 키워드 (Keywords) in JLS 11

- Keywords a.k.a. reserved words (예약어)
- 51 character sequences, formed from ASCII letters, are reserved for use as keywords and cannot be used as identifiers (§3.8).

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

(참고) 식별자 (Identifiers)

- An identifier is an unlimited-length sequence of Java letters and Java digits, the first of which must be a Java letter.

Identifier:
IdentifierChars but not a *Keyword* or *BooleanLiteral* or *NullLiteral*

IdentifierChars:
JavaLetter {JavaLetterOrDigit}

JavaLetter:
any Unicode character that is a "Java letter"

JavaLetterOrDigit:
any Unicode character that is a "Java letter-or-digit"

(참고) 리터럴 (Literals)

- A literal is the source code representation of a value of a primitive type (§4.2), the String type (§4.3.3), or the null type (§4.1).

- Literal:
  - IntegerLiteral : 3, -7, 1_000, 1__000, 50l (문자 ell; 숫자 1과 비슷), 50L (preferred),
    0x5A, 0X5A, 0xabcd_1234 (-1412623820),
    0177 (127), 0_123_456 (42798),
    0b1100, 0B1100, 0b011_1100_0111 (967), ...
  - FloatingPointLiteral : 1.2f, 3.0, ...
  - BooleanLiteral : true, false
  - CharacterLiteral: 'a', '안', '₩t', '₩u03a9', '₩177', ...
  - StringLiteral: "", "₩"", "This is a string", ...
  - NullLiteral : null

```
jshell> long a = 1231231231212123L;
a ==> 1231231231212123

jshell> long a = 1231231231212123;
|  Error:
|  integer number too large
|  long a = 1231231231212123;
|           ^

jshell> long a = 1231231231212123L;
a ==> 1231231231212123
```

객체 레퍼런스(Object Reference)

- 객체란 무엇일까?
  - 객체 변수라는 것은 없습니다.
  - 객체 레퍼런스라는 것만 있습니다.
  - 객체 레퍼런스에는 객체에 접근하는 방법을 알려주는 비트가 들어있습니다.
  - 객체 레퍼런스에는 실제 객체가 들어있는 것은 아니고 그 객체를 가리키는 무언가가 있을 뿐입니다.

There are two kinds of types in the Java programming language: **primitive types** (§4.2) and **reference types** (§4.3).

p.96

객체 레퍼런스

```
Dog d = new Dog();
d.bark();
```

Dog 레퍼런스 변수는 Dog 객체에 대한 리모컨이라고 생각하면 됩니다. 이 리모컨을 가지고 해당 객체에 어떤 일을 지시할 수 있습니다.

객체 레퍼런스

byte short int long
8 16 32 64

레퍼런스
(비트 수는 중요하지 않음)

객체 레퍼런스

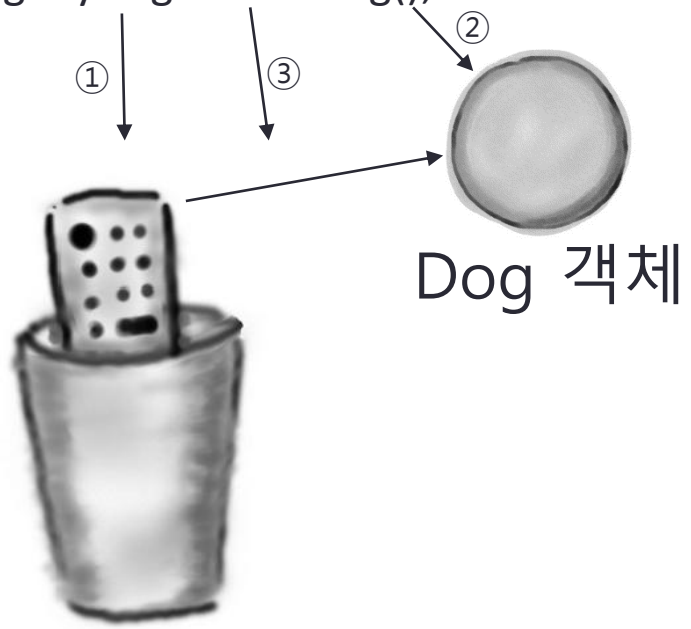
원시 변수

byte x = 7;

byte

레퍼런스 변수

Dog myDog = new Dog();



Dog

객체 선언, 생성 및 대입

Dog myDog = new Dog();

1. 레퍼런스 변수 선언

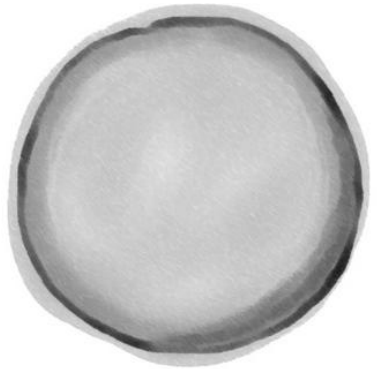
Dog myDog = new Dog();

Dog

Dog myDog = new Dog();

2. 객체 생성

Dog myDog = new Dog();

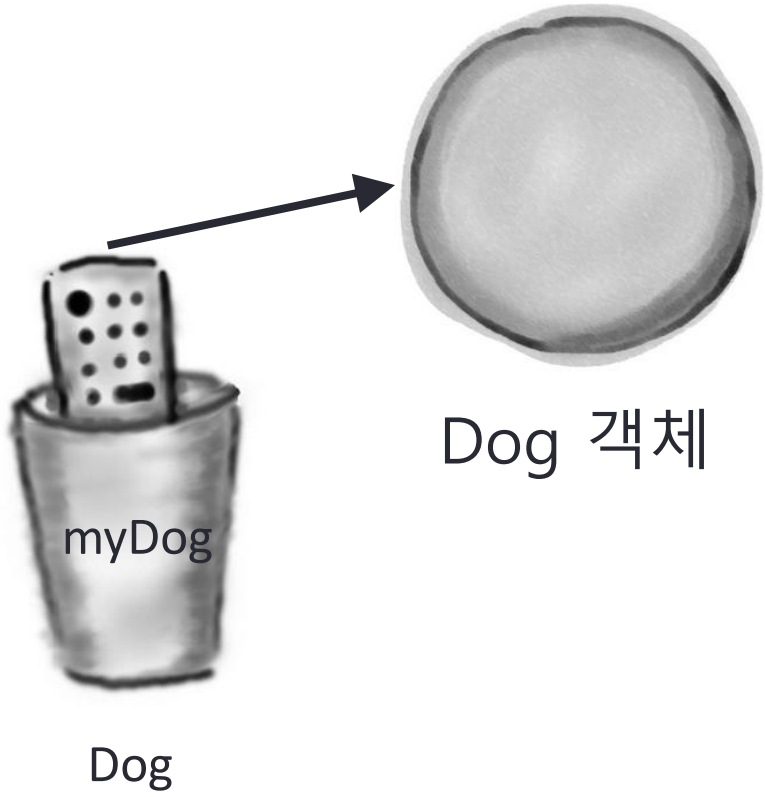


Dog 객체 생성

Dog myDog = new Dog();

3. 객체와 레퍼런스 연결

Dog myDog = new Dog();



바보 같은 질문은 없습니다

- 레퍼런스 변수의 크기는 얼마인가요?
  - 알 수 없습니다. 레퍼런스의 내부적인 표현 방식은 공개되어있지 않고, 프로그래머 입장에서 굳이 알 필요도 없습니다. 메모리 할당과 관련된 문제를 생각할 때도 중요한 것은 객체 레퍼런스의 개수가 아니라 객체의 개수, 그리고 객체의 크기입니다.

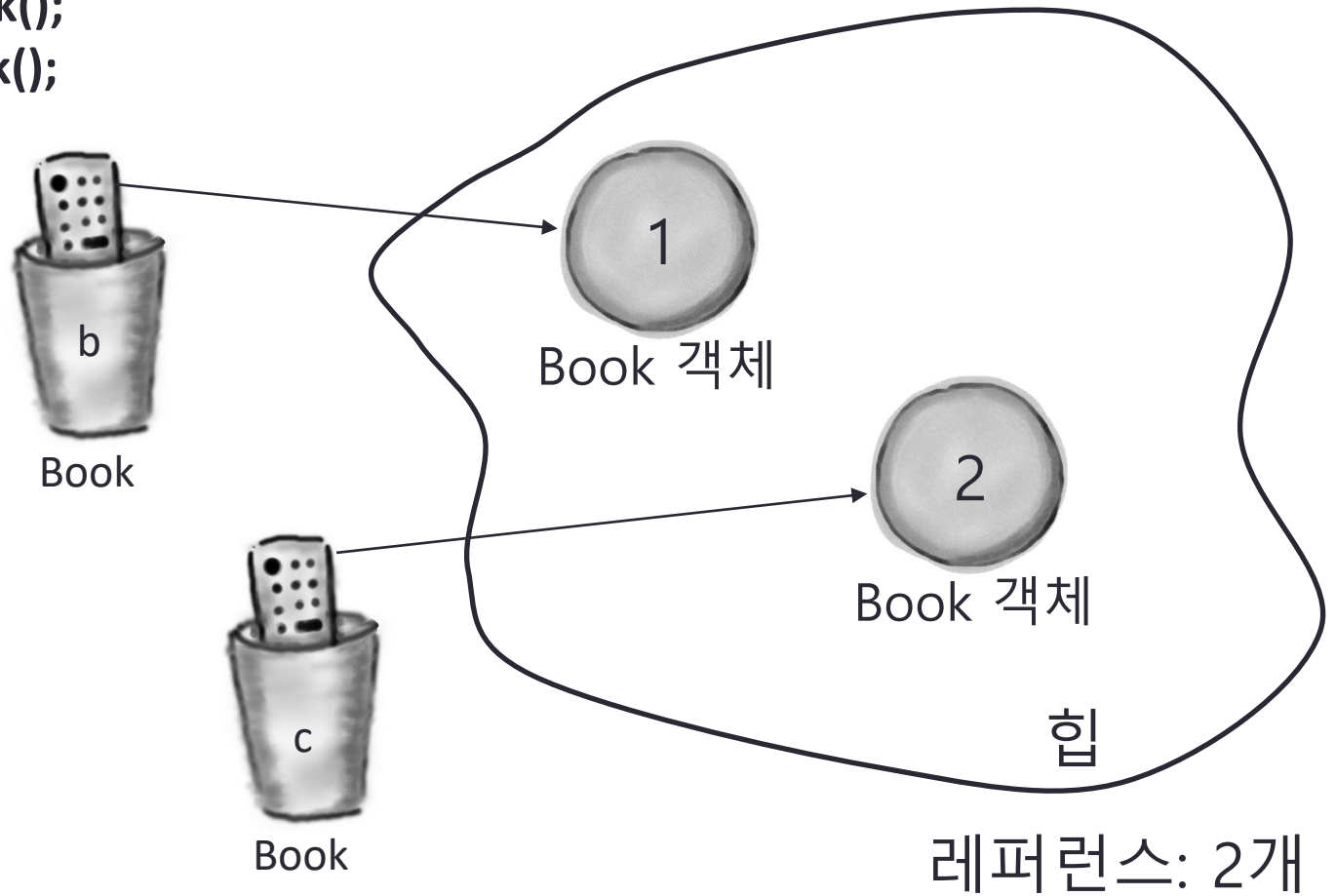
바보 같은 질문은 없습니다

- 그러면 모든 객체 레퍼런스의 크기가 객체의 실제 크기와는 상관없이 똑같은가요?
  - 예. 같은 JVM에서는 객체의 크기와는 상관없이 레퍼런스의 크기는 모두 같습니다. 다만, JVM에 따라서 레퍼런스의 크기가 다를 수는 있습니다.

- 레퍼런스 변수에 대해 C에서처럼 증가 연산 같은 것을 적용시킬 수 있나요?
  - 아닙니다. 자바는 C가 아니니까요.

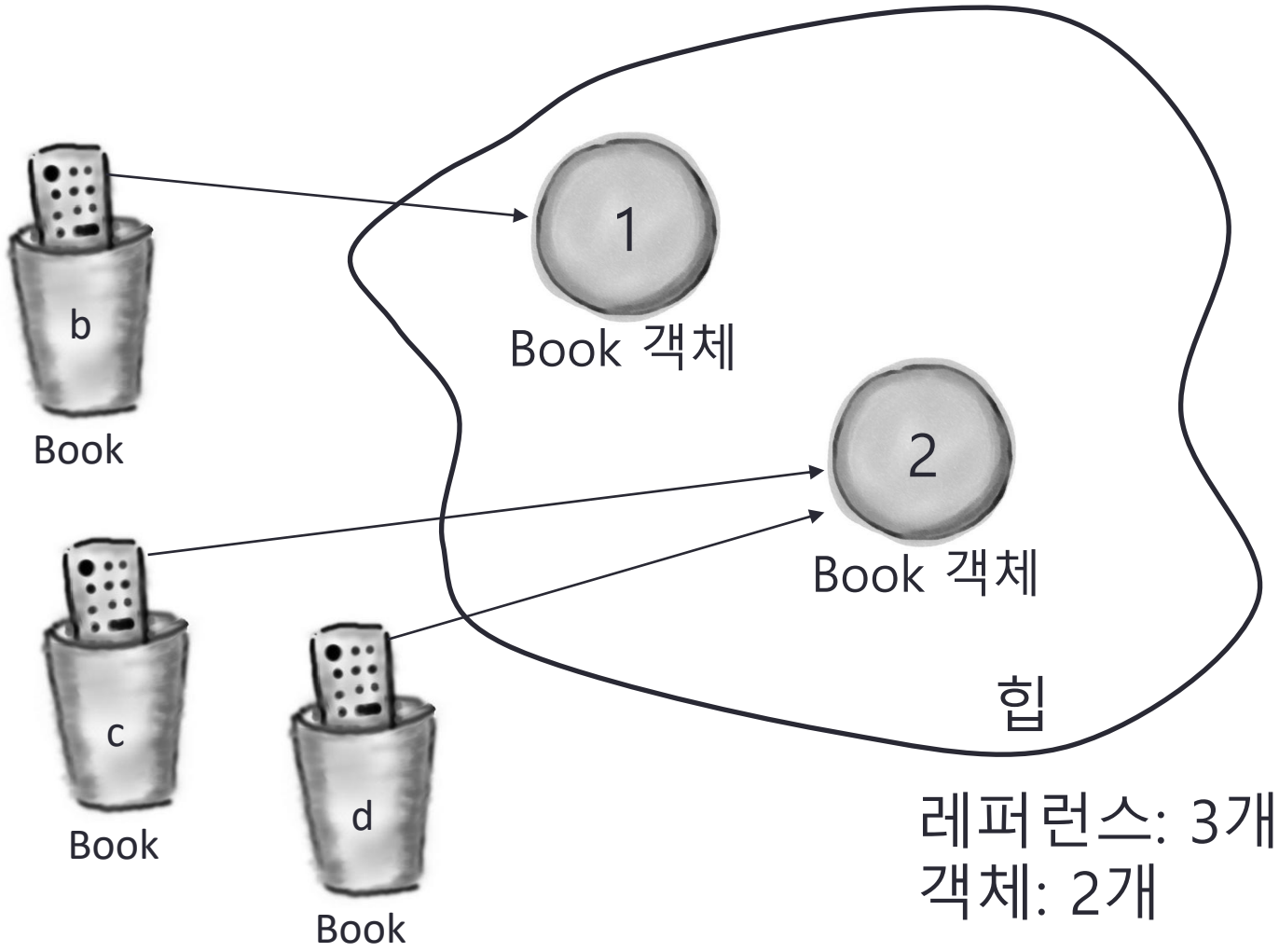
힙(heap) 메모리: 객체 생성 장소

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

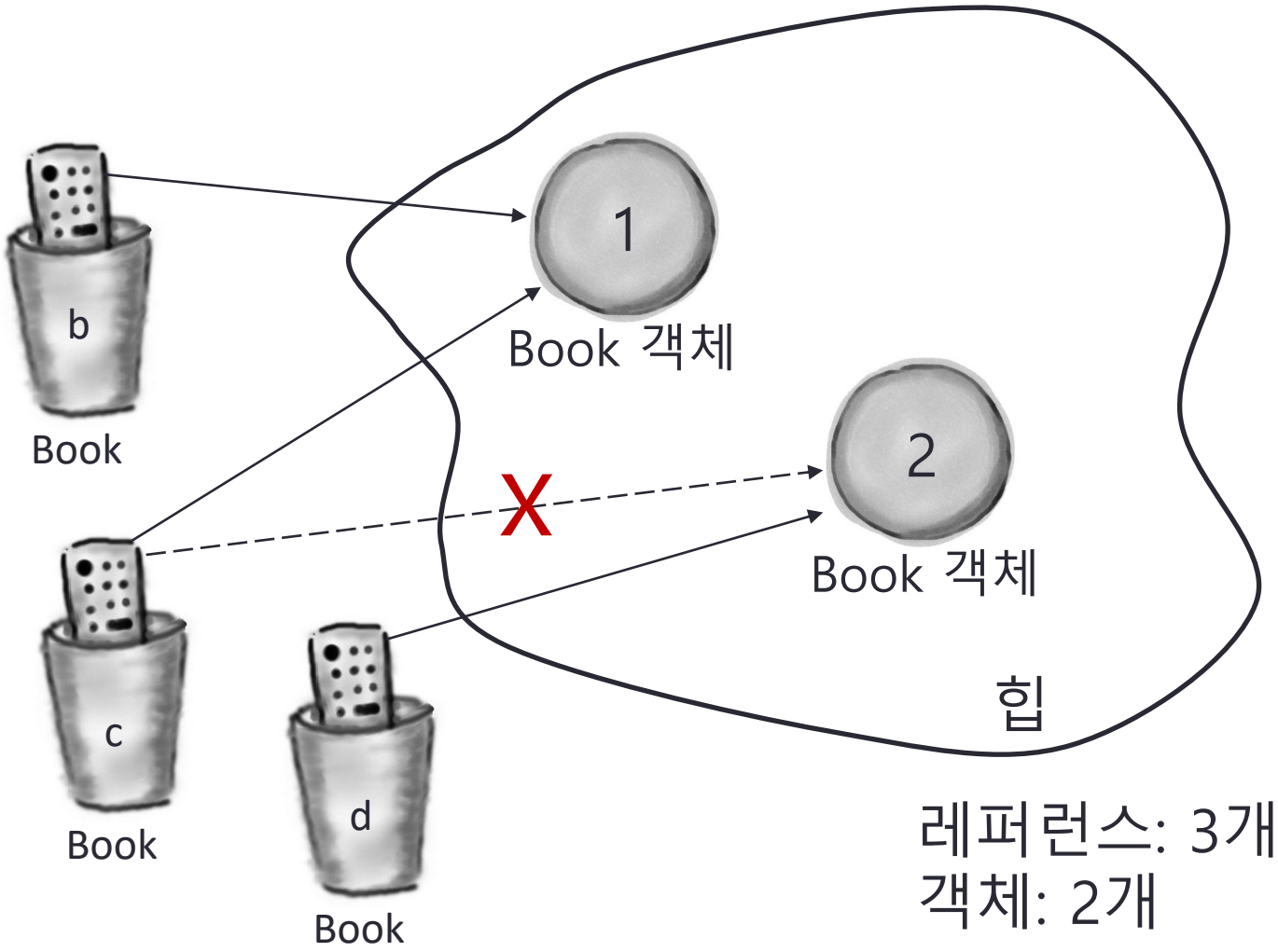


레퍼런스: 2개
객체: 2개

Book d = c;

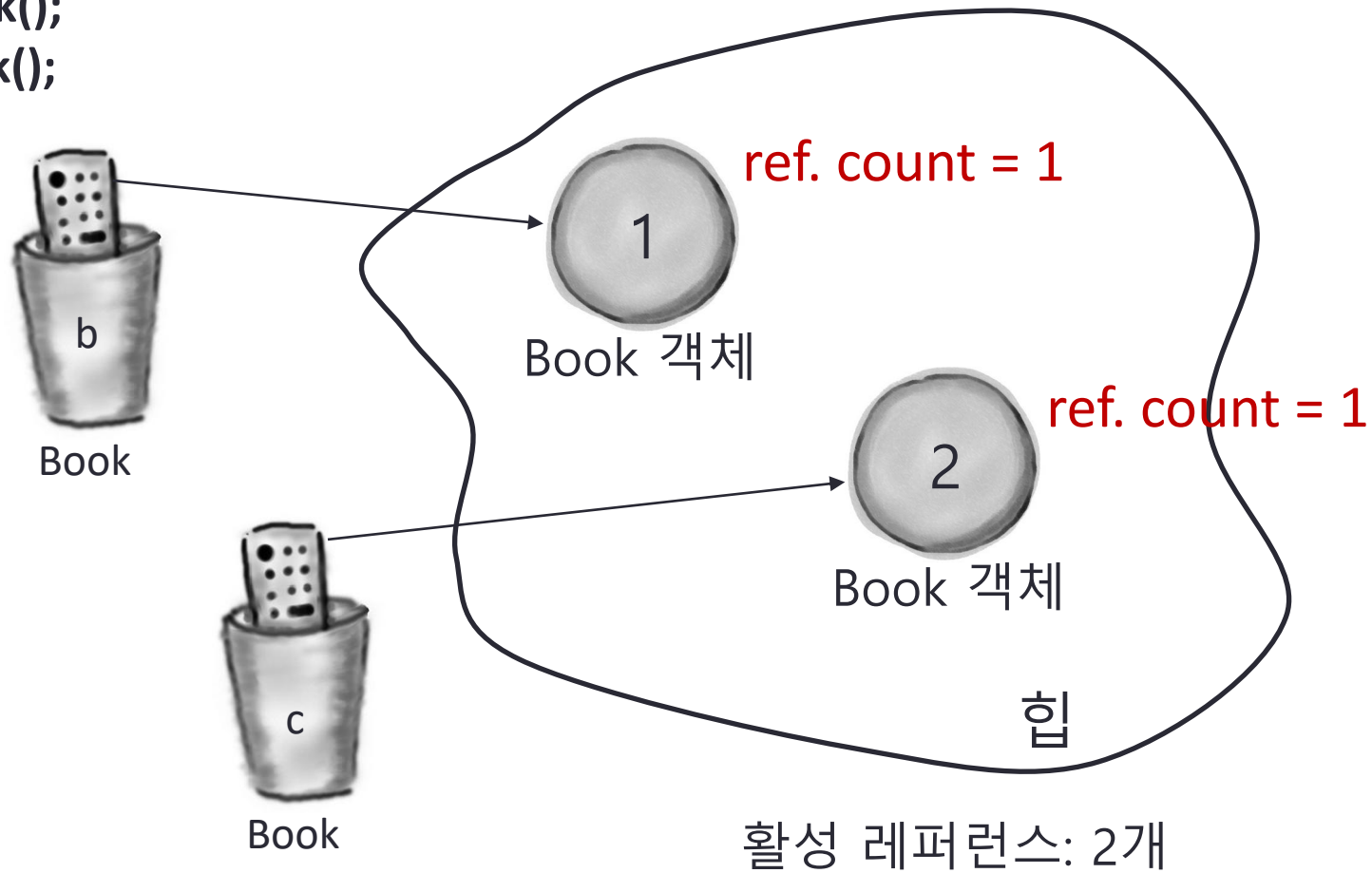


c = b;



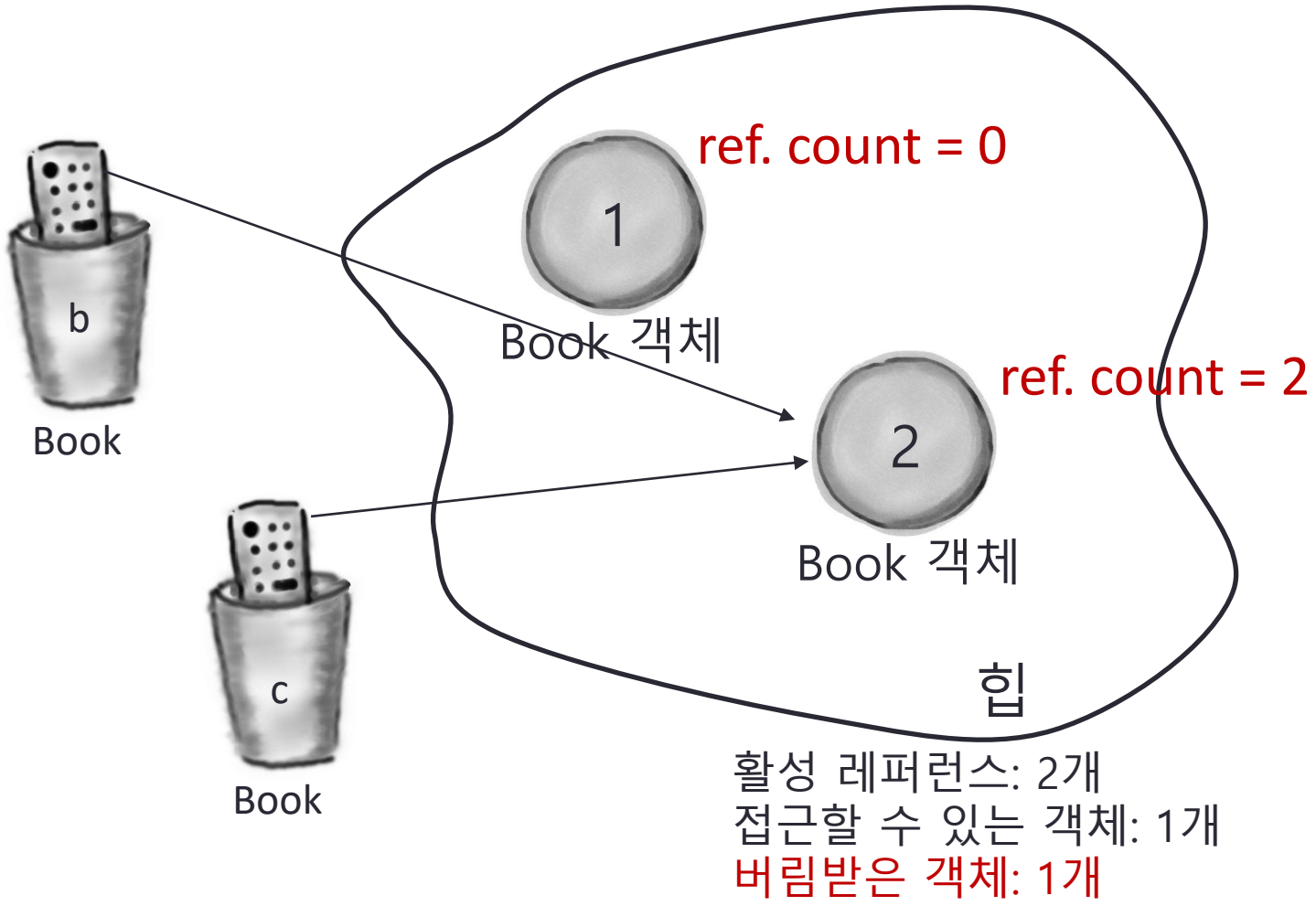
참조 계수(ref. count)와 가비지 컬렉션

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

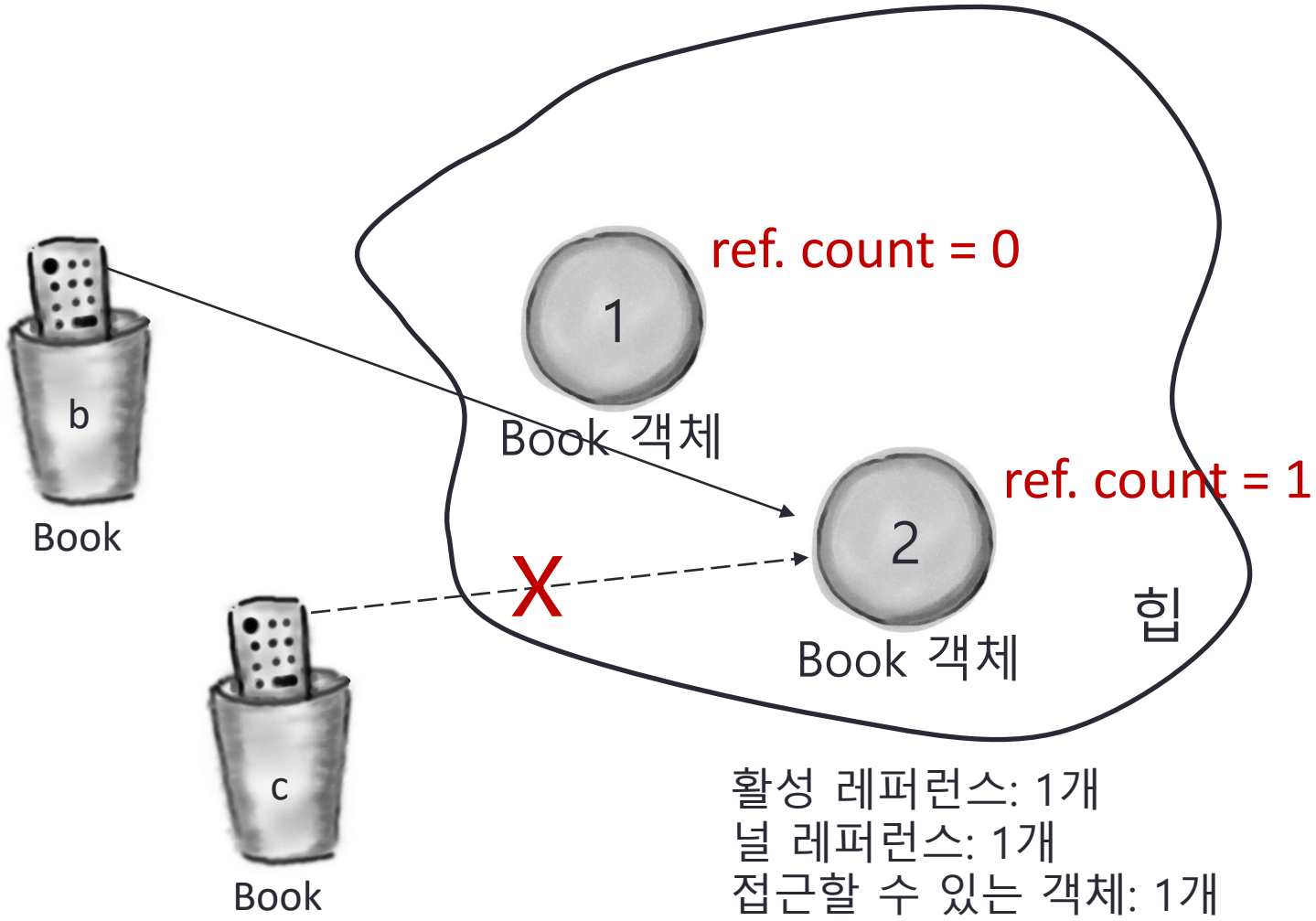


활성 레퍼런스: 2개
접근할 수 있는 객체: 2개

b = c;


ref. count = 0
1
Book 객체
b
Book
ref. count = 2
2
Book 객체
c
Book
힙
활성 레퍼런스: 2개
접근할 수 있는 객체: 1개
버림받은 객체: 1개

c = null;


b
Book
c
Book
1
ref. count = 0
Book 객체
2
ref. count = 1
Book 객체
X
힙
활성 레퍼런스: 1개
널 레퍼런스: 1개
접근할 수 있는 객체: 1개
버림받은 객체: 1개

배열

1. int 배열 변수 선언

```
int[] nums;
```

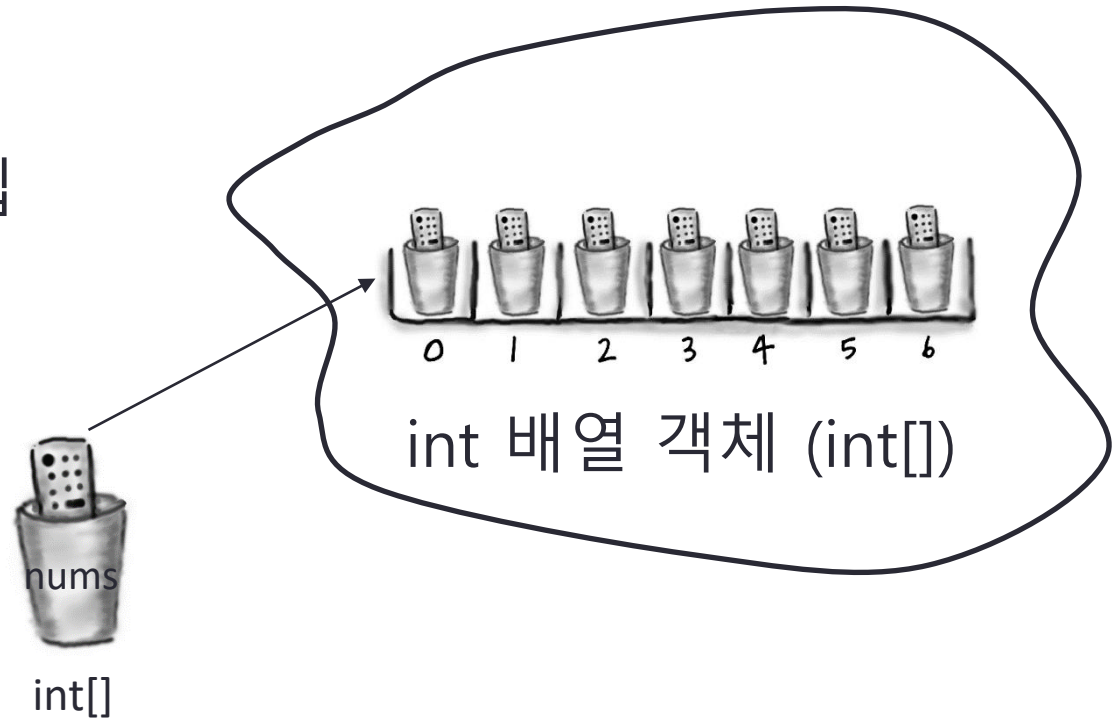
배열은 객체이다.
cf. nums***.length***

2. 길이가 7인 int 배열을 만들어서 변수에 대입

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

3. 각 원소에 int 값 대입

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



p.101

배열

1. Dog 배열 변수 선언

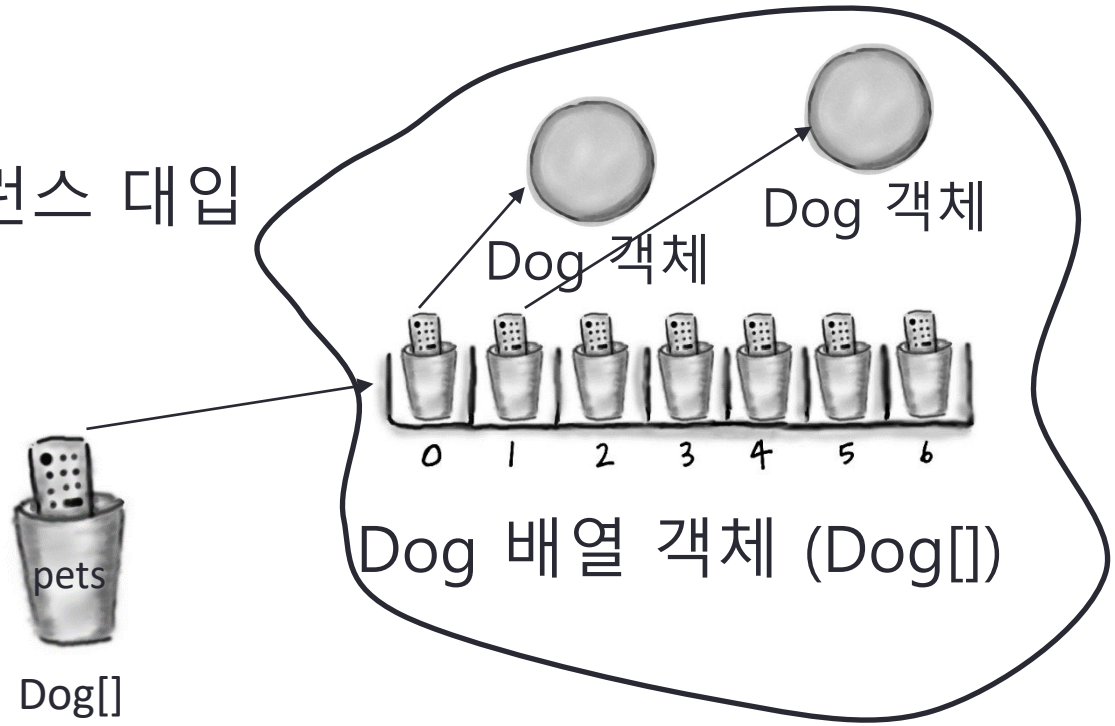
```
Dog[] pets;
```

2. 길이가 7인 Dog 배열을 만들어서 변수에 대입

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

3. 각 원소에 Dog 레퍼런스 대입

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



레퍼런스 변수를 이용한 객체 조작

Dog
name
bark() eat() chaseCat()

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

배열과 객체 레퍼런스

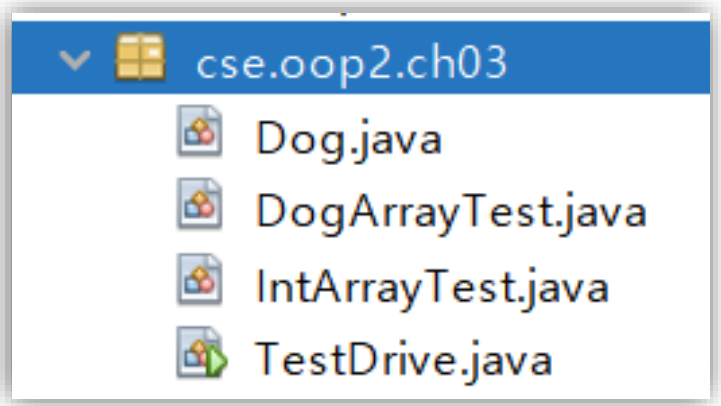
배열에 들어있는 객체 레퍼런스를 사용할 때도 똑같은 식으로 사용하면 됩니다.

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

101, 104 페이지에 있는 예제를 직접 돌려봅시다.

p.101, p.104 실습

- cse.oop2.ch03 패키지 내 자바 파일
  - IntArrayTest.java
  - Dog.java
  - DogArrayTest.java
  - TestDrive.java



IntArrayTest.java

```
6   package cse.oop2.ch03;
7
8   import java.util.OptionalDouble;
9   import java.util.OptionalInt;
10
11  /** 교재 p ...5 lines */
16  public class IntArrayTest {
17      public static final int MAX_NUMS = 1000;
18
19      public void test() {
20          int[] nums;
21          long sum = 0;
22
23          nums = new int[MAX_NUMS];
24          for (int i = 0; i < MAX_NUMS; i++) {
25              nums[i] = (int) (Math.random() * 100); // 형 변환
26              sum = sum + nums[i];
27          }
28          System.out.println("평균 1: " + (float) sum / MAX_NUMS);
29          this.calculateSumUsingWrapper(nums);
30      }
```

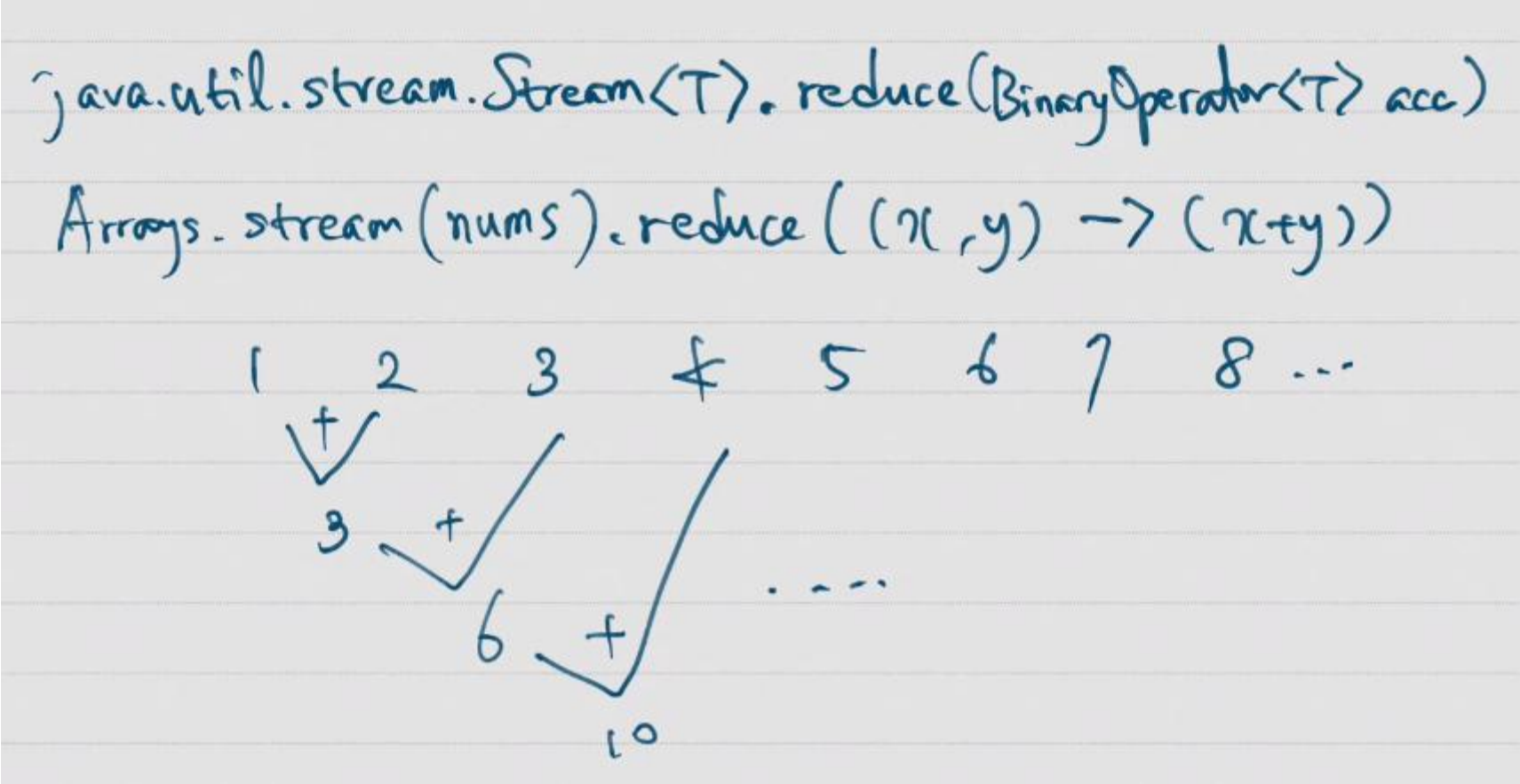
p.101

```
32      private void calculateSumUsingWrapper(int[] nums) {
33          OptionalInt sum = java.util.Arrays.stream(nums).reduce((x, y) -> (x + y));
34          OptionalDouble avg = java.util.Arrays.stream(nums).average();
35          if (sum.isPresent() && avg.isPresent()) {
36              System.out.printf("합계 = %d, 평균 2: %f%n",
37                      sum.getAsInt(), avg.getAsDouble());
38          }
39      }
40  }
```

cf. {OptionalInt, OptionalDouble}.isPresent()

cf. 람다 표현식, map, reduce, filter
스트림 API: Stream<T>, IntStream, DoubleStream, LongStream

reduce 개념


java.util.stream.Stream<T>.reduce(BinaryOperator<T> acc)
Arrays.stream(nums).reduce((x,y) -> (x+y))
1 2 3 4 5 6 7 8 ...
+
3
+
6
+
10
....

Dog.java

```
6   package cse.oop2.ch03;
7
8   /**...4 lines */
12  public class Dog {
13
        private String name;
15
16      public Dog(String name) {
17          this.name = name;
18      }
19
20      public String getName() {
21          return name;
22      }
23
24      public void bark() {
25          System.out.println(name + "이(가) 왈!하고 짖습니다.");
26      }
27  }
```

DogArrayTest.java

```
6   package cse.oop2.ch03;
7
8   /**...4 lines */
12  public class DogArrayTest {
13
14      public void test() {
15          Dog[] myDogs = new Dog[3];
16
17          myDogs[0] = new Dog("Fred");
18          myDogs[1] = new Dog("Marge");
19          myDogs[2] = new Dog("Bart");
20
21          System.out.print("마지막 개의 이름: ");
22          System.out.println(myDogs[myDogs.length - 1].getName());
23
24          int x = 0;
25          while (x < myDogs.length) {
26              myDogs[x].bark();
27              x = x + 1;
28          }
29      }
30  }
```

```
java.util.Arrays.stream(myDogs)
    .forEach(e -> e.bark());
```

TestDrive.java

```
6   package cse.oop2.ch03;
7
8   /**...4 lines */
12  public class TestDrive {
13
14      public static void main(String[] args) {
15          IntArrayTest test1 = new IntArrayTest();
16          test1.test();
17
18          DogArrayTest test2 = new DogArrayTest();
19          test2.test();
20      }
21
22  }
```

실행 결과

```
--- exec-maven-plugin:1.5.0:exec (de
평균 1: 50.054
합계 = 50054, 평균 2: 50.054000
마지막 개의 이름: Bart
Fred이(가) 왈!하고 짖습니다.
Marge이(가) 왈!하고 짖습니다.
Bart이(가) 왈!하고 짖습니다.
```

```
--- exec-maven-plugin:1.5.0:exec (
평균 1: 48.261
합계 = 48261, 평균 2: 48.261000
마지막 개의 이름: Bart
Fred이(가) 왈!하고 짖습니다.
Marge이(가) 왈!하고 짖습니다.
Bart이(가) 왈!하고 짖습니다.
```

```
--- exec-maven-plugin:1.5.0:exec (d
평균 1: 49.542
합계 = 49542, 평균 2: 49.542000
마지막 개의 이름: Bart
Fred이(가) 왈!하고 짖습니다.
Marge이(가) 왈!하고 짖습니다.
Bart이(가) 왈!하고 짖습니다.
```

java.util.stream 패키지

- Interface BaseStream<T,S extends BaseStream<T,S>>
- public interface Stream<T> extends BaseStream<T,Stream<T>>
- public interface IntStream extends BaseStream<Integer,IntStream>
- public interface LongStream extends BaseStream<Long,LongStream>
- public interface DoubleStream extends BaseStream<Double,DoubleStream>

Stream 인터페이스 실습

- cse.oop2.ch03.stream 패키지
  - Gender.java: 열거형
  - Person.java
  - StreamTestDrive.java

Gender.java

```
1   package cse.oop2.ch03.stream;
2
3   /** 열거형 ...5 lines */
8   public enum Gender {
9       FEMALE("female"),
10      MALE("male");
11
        private String info;
13
14      private Gender(String info) {
15          this.info = info;
16      }
17
        public String toString() {
19          return info;
20      }
21  }
```

Person.java

```
1   package cse.oop2.ch03.stream;
2
3   /** Stream 인터페이스 실습에 사용할 Person 클래스 ...5 line
8   public class Person implements Comparable<Person> {
9
10      private String name;
11      private int age;
12      private Gender gender;
13
14      public Person(String name, int age, Gender gender) {
15          super();
16          this.name = name;
17          this.age = age;
18          this.gender = gender;
19      }
20
21      public String getName() {
22          return name;
23      }
```

```
25      public int getAge() {
26          return age;
27      }
28
29      public Gender getGender() {
30          return gender;
31      }
32
33      @Override
        public int compareTo(Person o) {
35          return name.compareTo(o.name);
36      }
37
38  }
```

StreamTestDrive.java

```
1   package cse.oop2.ch03.stream;
2
3   import java.util.LinkedList;
4   import java.util.List;
5   import java.util.OptionalDouble;
6   import java.util.stream.Stream;
7
8   /** Stream 인터페이스 실습 ...5 lines */
13  public class StreamTestDrive {
14
15      public static List<Person> persons = new LinkedList<>();
16
17      public static void main(String[] args) {
18          initialize();
```

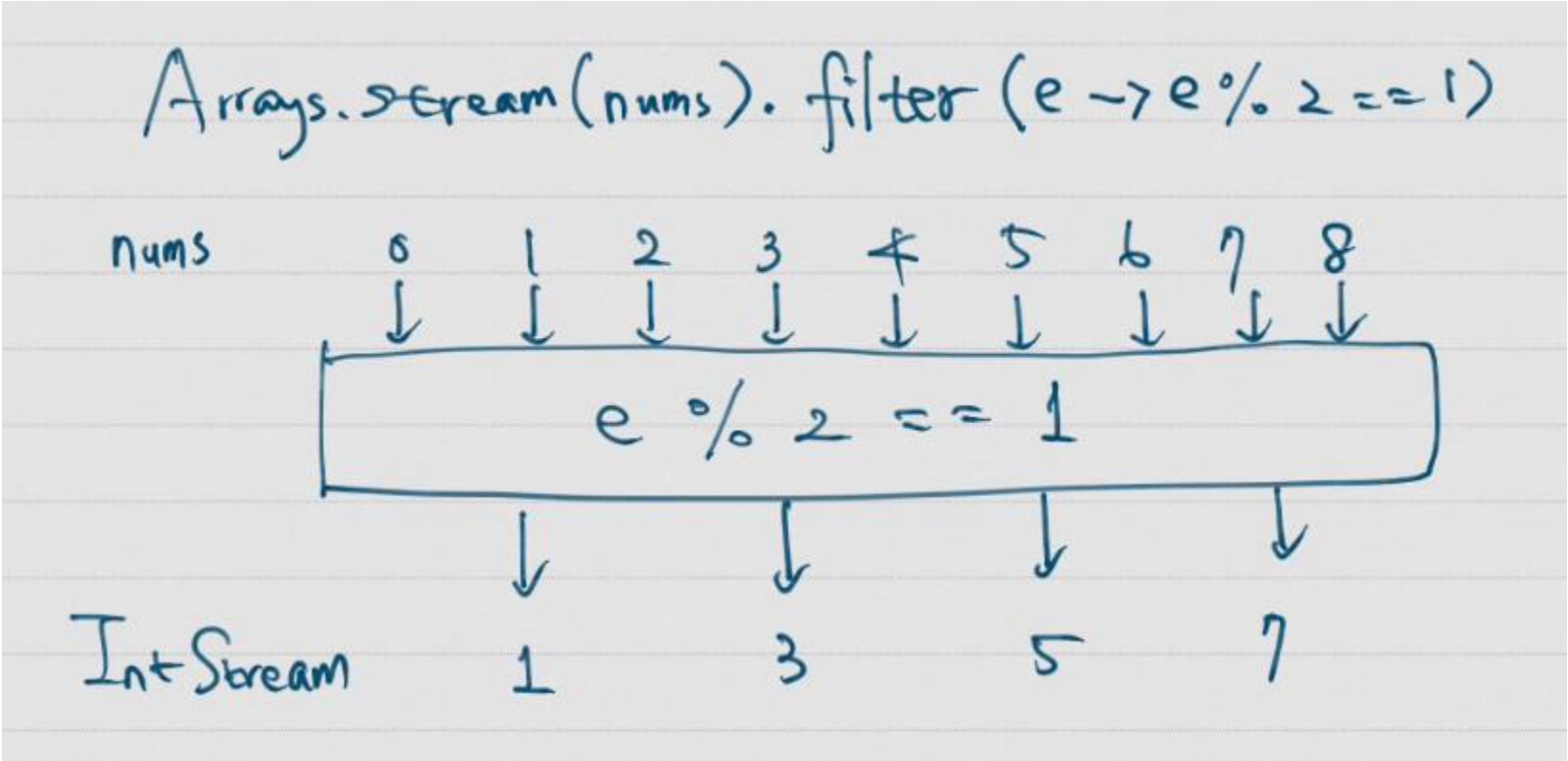
```
20          // 남자는 몇 명?
21          long maleCount = persons.stream()
22                  .filter(e -> e.getGender() == Gender.MALE).count();
23          System.out.printf(">>> 남자는 %d명입니다.%n", maleCount);
24
25          // 여자의 평균 나이는?
26          OptionalDouble femaleAverageAge = persons.stream()
27                  .filter(e -> e.getGender() == Gender.FEMALE)
28                  .mapToInt(Person::getAge)
29                  .average();
30          if (femaleAverageAge.isPresent()) {
31              System.out.printf(">>> 여자의 평균 나이는 %.2f입니다.%n",
32                      femaleAverageAge.getAsDouble());
33          }
34
35          System.out.println(">>> 20~25살인 여자의 이름을 정렬해서 출력");
36          Stream<Person> sp = persons.stream()
37                  .filter(e -> e.getGender() == Gender.FEMALE)
38                  .filter(e -> e.getAge() >= 20 && e.getAge() <= 25)
39                  .sorted();
40          sp.forEach(e -> System.out.printf("%s%n", e.getName()));
41      }
```

```
43      public static void initialize() {
44          Person[] data = {
45              new Person("Linda", 21, Gender.FEMALE),
46              new Person("Oliver", 25, Gender.MALE),
47              new Person("Alice", 27, Gender.FEMALE),
48              new Person("Noah", 19, Gender.MALE),
49              new Person("Abby", 23, Gender.FEMALE),
50              new Person("Daisy", 25, Gender.FEMALE),
51              new Person("Samuel", 31, Gender.MALE),
52              new Person("Crystal", 31, Gender.FEMALE),
53              new Person("Tadeo", 33, Gender.MALE)
54          };
55
56          // persons.addAll(Arrays.asList(data));와 동일
            for (Person p : data) {
58              persons.add(p);
59          }
60      }
61
62  }
```

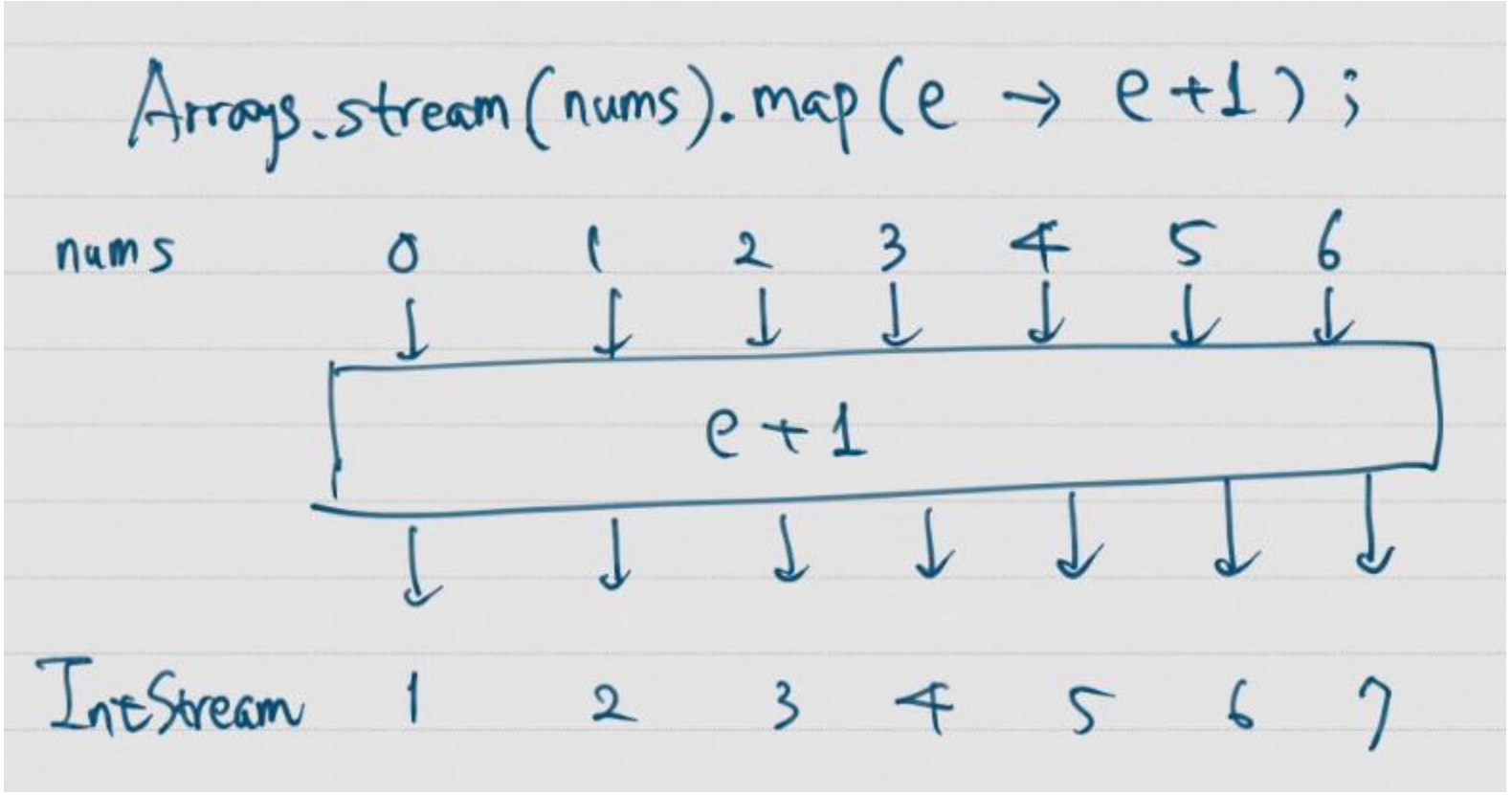
실행 결과

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ java_maven ---
>>> 남자는 4명입니다.
>>> 여자의 평균 나이는 25.40입니다.
>>> 20~25살인 여자의 이름을 정렬해서 출력
Abby
Daisy
Linda
```

filter 개념


Arrays.stream(nums).filter(e -> e % 2 == 1)
nums
0 1 2 3 4 5 6 7 8
e % 2 == 1
IntStream
1 3 5 7

map 개념


Arrays.stream(nums).map(e → e+1);
nums
0 1 2 3 4 5 6
e+1
IntStream
1 2 3 4 5 6 7

핵심 정리

- 변수에는 원시 변수와 레퍼런스 변수가 있습니다.
- 변수를 선언할 때는 이름과 유형이 필요합니다.
- 원시 변수의 값은 그 값을 표시하는 비트로 구성됩니다.
- 레퍼런스 변수의 값은 힙에 들어있는 객체를 건드릴 수 있는 방법을 나타내는 비트입니다.
- 레퍼런스 변수는 리모컨과 같습니다. 레퍼런스 변수에 대해 점 연산자(.)를 사용하는 것은 리모컨의 버튼을 눌러서 메서드나 인스턴스 변수를 액세스하는 것과 비슷합니다.
- 모든 배열은 객체입니다.