
JAVA
배열

1 배열

1. 1차원 배열

2. 객체 배열

3. 2차원 배열

자바에서 JVM의 역할은 무엇일까?

구조로 알아보는 JVM의 동작 원리

클래스 파일 - **개발자가 만드는 혹은 이미 만들어진 파일.**
컴파일 과정을 거쳐 생성된 클래스 파일은 JVM에서 실행 가능

클래스 로더 서브 시스템 - **JVM은 실행할 클래스 파일을 읽고, JVM 메모리에 올려놓는 과정이 필요.** 이 과정을 클래스 로딩(Class loading)이라 하며 JVM의 클래스 로더 서브 시스템(클래스 로더)이 담당

메서드 영역 | 스택 영역 | 힙 영역 | 레지스터 영역 | 네이티브 메서드 영역

실행 데이터 영역

Execution Engine ↔ Native method interface

Native method librarys

JVM의 내부 구조를 큰 형태로 분리

JVM의 동작 원리

- **실행 데이터 영역**

클래스 로더로부터 분석된 데이터를 저장하고 실행 도중 필요한 데이터를 저장하는 영역

메소드 영역

클래스 로더에 의해서 로딩된 클래스가 저장되는 곳
JVM에서 클래스를 실행하면 메소드 영역에서 클래스 정보를 복사

스택 영역

호출된(실행된) 메소드 정보가 저장되는 곳으로실행이 끝나면 저장된 정보는 삭제
메소드가 실행될 때마다 저장되는 메소드 정보에는 **매개변수, 지역 변수** 등

힙 영역

JVM의 실행 데이터 영역 중에서 가장 중요한 역할을 담당
객체는 **클래스가 실행될 때 생성되어서 힙 영역에 저장**
힙 영역은 JVM에서 가장 중요한 데이터를 저장함과 동시에 세밀한 관리가 이뤄지는 곳

JVM의 동작 원리

- **실행 데이터 영역**

클래스 로더로부터 분석된 데이터를 저장하고 실행 도중 필요한 데이터를 저장하는 영역

레지스터 영역

현재 JVM이 수행할 명령어의 주소를 저장하는 메모리 공간

네이티브 메소드 스택 영역

네이티브 메소드 : OS의 시스템 정보, 리소스를 사용하거나 접근하기 위한 코드
자바 프로그램과 OS 사이에 JVM이 존재, 자바 프로그램은 시스템에 직접 접근하기 어려움
JNI(Java Native Interface) API를 사용하면 자바 프로그램에서 OS 시스템에 대한 접근이 가능

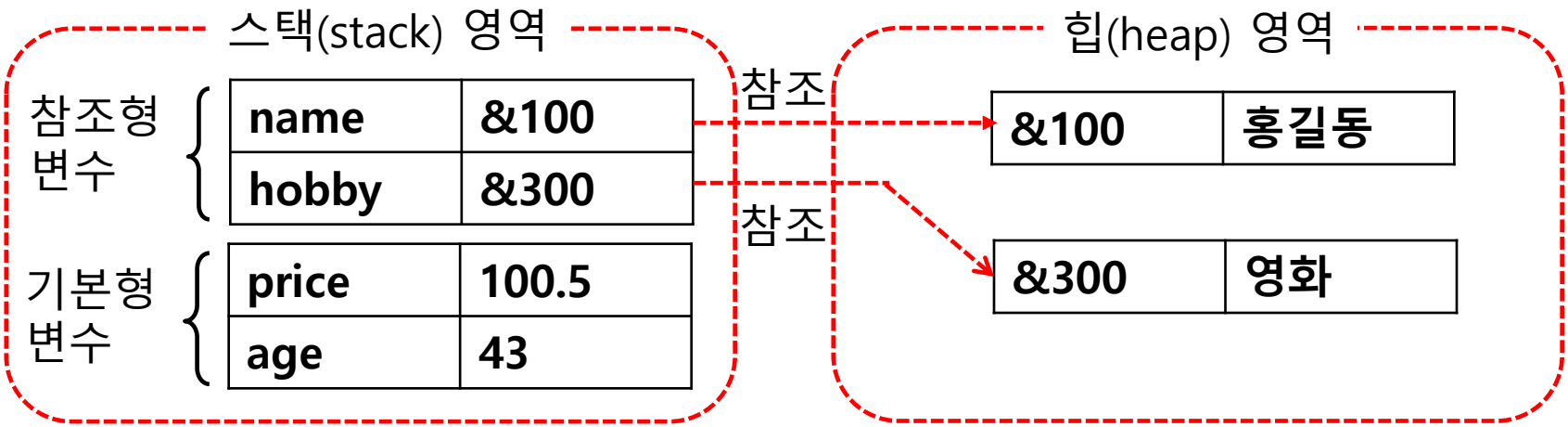
데이터 타입 분류

❖변수의 메모리 사용

- 기본 타입 변수 – 실제 값을 변수 안에 저장
- 참조 타입 변수 – 주소를 통해 객체 참조

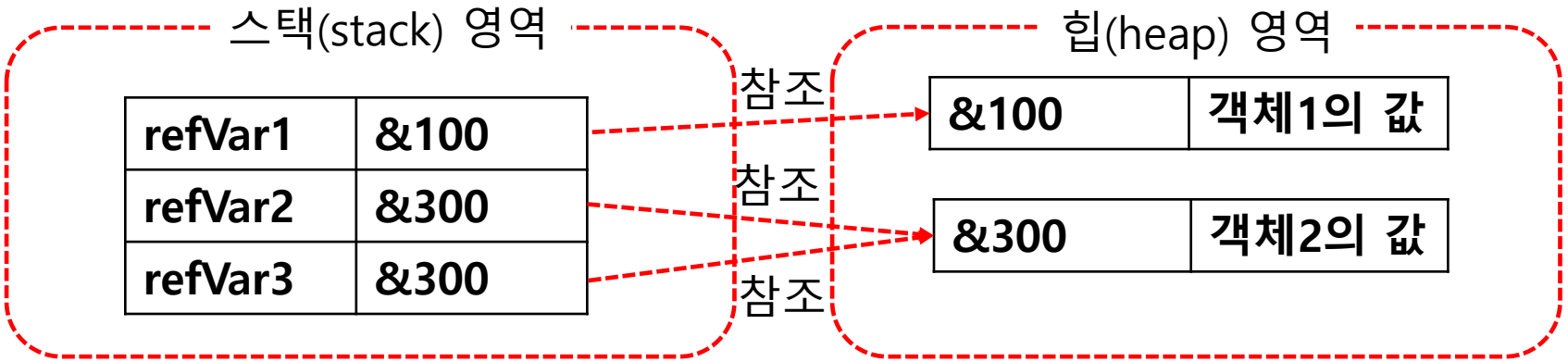
```
[기본 타입 변수]
int age = 43;
double price = 100.5;

[참조 타입 변수]
String name  = new String("홍길동");
String hobby = new String("영화");
```



참조 변수의 ==, != 연산

- 변수의 값이 같은지 다른지 비교
  - 기본 타입: byte, char, short, int, long, float, double, boolean
    - 의미 : 변수의 값이 같은지 다른지 조사
  - 참조 타입: 배열, 열거, 클래스, 인터페이스
    - 의미 : 동일한 객체를 참조하는지 다른 객체를 참조하는지 조사



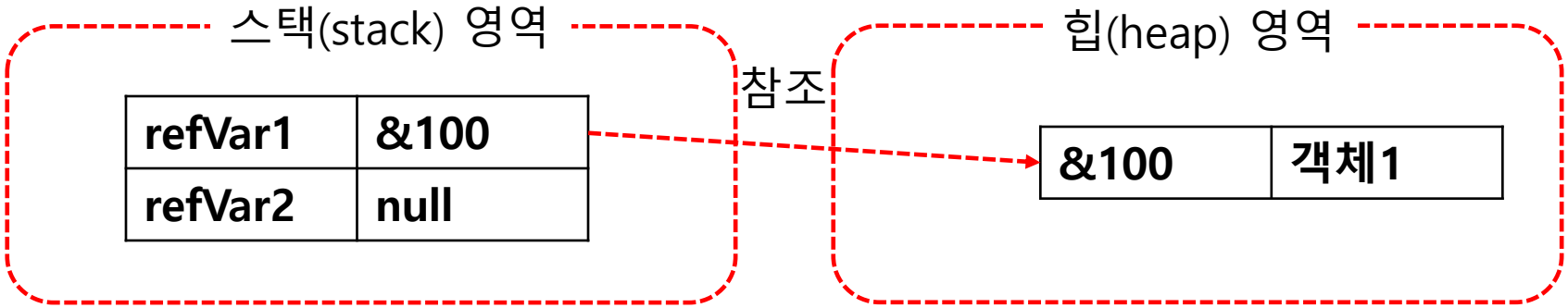
refVar1 == refVar2	결과: false
refVar1 != refVar2	결과: true

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

null과 NullPointerException

❖null(널)

- 변수가 참조하는 객체가 없을 경우 초기값으로 사용 가능
- 참조 타입의 변수에만 저장가능
- null로 초기화된 참조 변수는 스택 영역 생성



- ==, != 연산 가능

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

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

null과 NullPointerException

NullPointerException의 의미

- 예외(Exception)
  - 사용자의 잘못된 조작 이나 잘못된 코딩으로 인해 발생하는 프로그램 오류

- NullPointerException
  - 참조 변수가 null 값을 가지고 있을 때
    - 객체의 필드나 메소드를 사용하려고 했을 때 발생

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

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

배열 개념

여러 저장 장소가 필요한 경우에 배열을 사용한다면 변수 이름 관리뿐만 아니라 프로그램도 매우 편리하게 작성 가능

❖배열의 개념

- 배열(array)이란 컴퓨터가 시스템에서 사용되는 데이터 구조의 한 형식으로 동일한 형의 자료들을 모아서 하나의 변수에 연속적으로 저장하여 사용하는 자료의 집합체
- 배열로 선언되면 메모리 내의 연속적인 기억 공간에 할당되며, 해당 배열의 위치를 첨자(index)로 표현 가능
- 첫 번째 원소의 첨자는 0이고 다음부터 1씩 증가
- 첨자의 범위는 0부터 원소개수 -1까지

❖배열의 종류

선언된 첨자의 형태에 따라서 1차원 배열, 다차원배열(2차원, 3차원, n차원 등) 등으로 나뉨

배열의 선언과 사용

- 배열이란?

  같은 타입의 데이터를 연속된 공간에 저장하는 자료구조

  (여러 변수를 하나로 묶어 하나의 이름으로 다루는 것이다. 물리적으로 연속적인 메모리에 데이터를 저장하는 구조이다. 이때 배열의 이름에는 연속적인 메모리의 시작 주소만 저장한다.)

- 각 데이터 저장 위치는 인덱스 부여해 접근

- (하나의 이름으로 관리되는 배열은 각각의 메모리를 식별하기 위해 배열의 이름에 저장된 주소로부터 상대적으로 얼마만큼 떨어져 있는지를 나타내는 오프셋(offset)을 사용한다. 오프셋은 인덱스라고도 한다)

```
int num1 = 89;
int num2 = 90;
int num3 = 77;
int num4 = 58;
int num5 = 99;
```

항목 접근: 배열이름[인덱스] ex) num[0]

배열 장점

배열의 장점

- 중복된 변수 선언 줄이기 위해 사용
- 반복문 이용해 요소들을 쉽게 처리

```
int sum = 0;
sum += num1;
sum += num2;
sum += num3;
sum += num4;
sum += num5;

int avg = sum/5.0;
```

```
int sum = 0;
for(int i=0; i <5; i++){
        sum += num[i];
}
int avg = sum/5.0;
```

배열 선언

❖ 배열 선언

- 배열을 사용하기 위해 우선 배열 변수 선언

타입[] 변수;

```
int[]   intArray;
double[] doubleArray;
String[] strArray;
```

타입 변수[];

```
int intArray[];
double doubleArray[];
String strArray[];
```

배열을 선언하면 이 때 사용된 배열명은 배열의 시작 주소가 된다

배열의 이름 = 주솟값

- **배열 변수는 참조 변수 - 배열 생성되기 전 null로 초기화 가능**
  - **배열 변수가 null 값을 가진 상태에서 항목에 접근 불가**
    - **변수[인덱스] 못함**
      - » NullPointerException 발생

```
타입[] 변수 = null;
```

배열의 선언과 사용

① 먼저 배열 참조변수부터 선언하여 보자. 배열 참조 변수는 배열 객체를 가리키는 변수이다.

```
int[ ] numbers;          // 배열 참조 변수 선언
```

② 배열 참조 변수를 선언했다고 해서 배열이 생성된 것이 아니다. 자바에서는 배열도 객체이므로 반드시 new연산자를 사용하여서 생성하여야 한다.

```
numbers = new int[5];  // 배열 객체 생성
```

대괄호 안의 숫자가 배열의 크기이다.

배열 선언

```
타입[] 배열명 = null;
배열명 = new 타입[첨자];

타입[] 배열명 = new 타입[첨자];
```

배열의 특징

배열은 다음과 같은 특징을 가진다.

① 배열은 선언과 동시에 공간을 할당 받아 요소로 관리된다.
② 상수를 기억하는 배열의 메모리는 연속적으로 할당된다.
③ 배열의 요소는 각괄호 '[]'안의 정수 첨자, 즉 인덱스로 구분되며 0 부터 시작된다.
④ 배열로 선언되는 변수를 참조 변수라고 한다.
⑤ 배열의 요소의 개수는 '배열명.length'로 얻을 수 있다.

배열의 선언과 사용

어떤 자료형의 배열도 생성이 가능하다.

```
int[] number = new int[5];                  // 정수 배열
float[] distances = new float[20];          // 실수 배열
double[] average = new double[10];          // 실수 배열
char[] letters = new char[50];              // 문자 배열
String[] word = new String[30];             // 문자열 배열
```

```
final int SIZE = 30;
int[ ] studentNumber = new int[SIZE];
```

배열의 선언과 생성(정리)

배열을 참조하는 참조변수가 스택 메모리에 할당된다.

- 배열의 선언

```
자료형[] 배열명;
자료형 배열명[];
```

초기값으로 "가리키는 값이 없다"라는 의미의 null로 초기화 된다.

- 배열의 생성

① 선언 후 new 연산자를 이용하여 배열을 생성하는 방법.

```
배열명 = new 자료형[배열의 크기] ;
```

② 선언과 동시에 배열을 생성하는 방법.

```
자료형[] 배열명 = new 자료형[배열의 크기] ;
```

배열 선언시 자료형의 디폴트값

배열은 생성시 크기를 지정해 주면 new 연산자에 의해서 생성시 각 자료형에 초기화가 주어진다. 그때 초기화의 기본값(default)이다.

데이터 타입	초기화	데이터 타입	초기화
byte	0	short	0
int	0	long	0l
float	0.0f	double	0.0
char	'₩u0000'(null)	boolean	false
Object	null	참조형	null

배열 초기화

2) 배열 초기화

배열은 자동으로 초기화가 이루어진다. 그런데 기본 초기값 말고 특정한 값으로 초기화 하는 방법은 **배열을 생성할 때 초기값을 지정해서 배열 생성과 초기화를 동시에 진행하는 방법**이다. 배열을 선언과 동시에 초기화하려면 배열을 선언한 다음에 중괄호를 사용하여 배열 원소의 초기값을 적어 넣는다.

```
자료형[] 배열명 = new 자료형[] { 값1, 값2, 값3, ...};

자료형[] 배열명 = {값1, 값2, 값3, ...};
```

배열 초기화

2) 배열 초기화

```
int[] score = new int[]{90, 85, 78, 100, 98};

char[] alphabet = {'a', 'b', 'c', 'd', 'e'};

String[] name = {"홍길동", "김하늘", "오정임"};
```

(오류 주의)
자바에서는 배열 변수를 선언할 때 배열 크기를 지정할 수 없다. 다음은 잘못된 배열 선언의 예이다

- **int arrayValue[5];**
- **int arrayValue[5] = {1, 2, 3, 4, 5};**
- **double[] doubleArray;**
  doubleArray = {5.0, 7.0, 8.0, 9.0};

배열 인덱스 범위 검사

자바에서는 배열 인덱스에 대하여 범위 검사를 한다. 즉 인덱스가 허용되는 범위를 넘으면 오류가 발생한다.

```
public class ArrayTest {
        public static void main(String[] args) {
                int[] arrayData= new int[5];
                arrayData[5] = 100;
        }
}
```

배열 인덱스 범위 검사

```
arrayData[5] = 100;
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at ArrayTest.main(ArrayTest.java:5)
```

배열의 크기

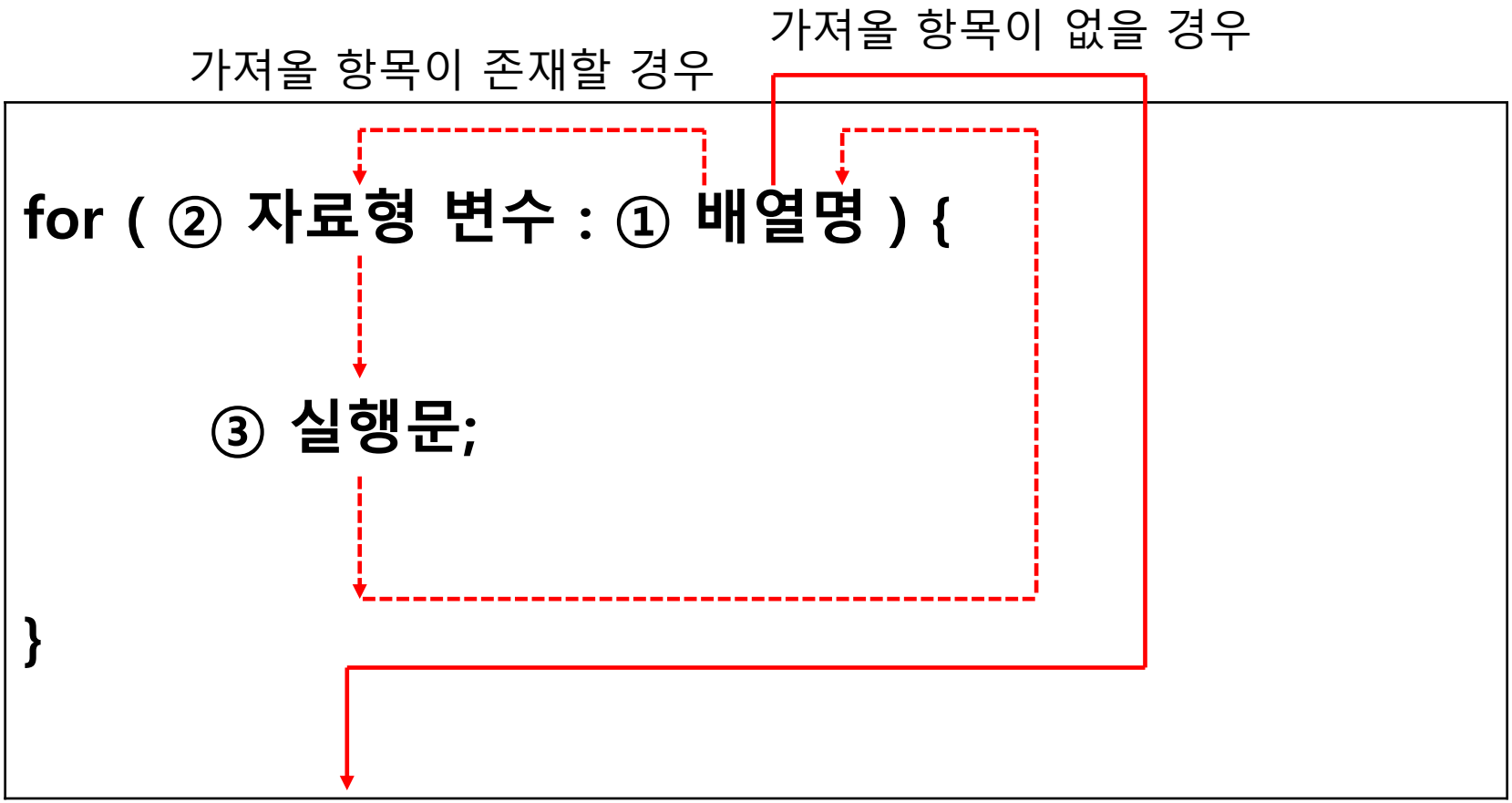
각 배열은 length라는 필드를 가지고 있다. length 필드는 배열의 크기를 나타낸다. 따라서 이것을 이용하면 배열의 크기만큼 반복할 수 있다.

```
배열명.length
```

```
for(int i=0; i < numbers.length; i++)
        numbers[i] = (int)(Math.random()*10);
```

향상된 for문

- 배열의 항목 요소를 순차적으로 처리
- 인덱스 이용하지 않고 바로 항목 요소 반복



for문을 이용하여 배열값 출력

```
public class Numbers {
        public static void main(String[] args) {
                int[] numbers = new int[5];

                for (int i = 0; i < numbers.length; i++)
                        numbers[i] = (int) (Math.random()*10);

                for (int i = 0; i < numbers.length; i++)
                        System.out.println(numbers[i]);
        }
}
```

향상된 for문

```
public class Numbers {
        public static void main(String[] args) {
                int[] numbers = new int[5];

                for (int i = 0; i < numbers.length; i++)
                        numbers[i] = (int) (Math.random()*10);

                for (int value : numbers)
                        System.out.println(value);
        }
}
```

향상된 for문

```
public class Strings {
        public static void main(String[] args) {
                String[] language =  { "Java", "C", "C#" };

                for (String lang : language) {
                        System.out.println(lang);
                }
        }
}
```

```
for(int i=0; i < str.length ; i++){
        System.out.println(str[i]);
}
```

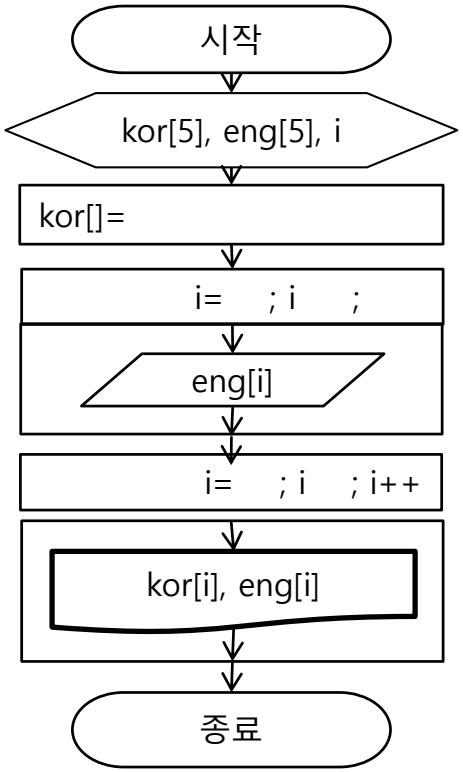
1차원 배열 기초

5명의 국어점수는 배열에 초기화 하고 영어점수는 입력을 받아 배열에 저장하여 출력하는 순서도와 슈도코드를 작성하시오.

참고 배열에 값을 저장하는 2가지 방법에 주목한다.

80	95	90	85	100
kor[0]	kor[1]	kor[2]	kor[3]	kor[4]

flowchart



pseudocode

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



}
```

실행결과 예

```
영어 점수를 입력해 주세요. 80
영어 점수를 입력해 주세요. 52
영어 점수를 입력해 주세요. 78
영어 점수를 입력해 주세요. 69
영어 점수를 입력해 주세요. 99
국어 : 80영어 : 80
국어 : 95영어 : 52
국어 : 90영어 : 78
국어 : 85영어 : 69
국어 :100영어 : 99
```

1차원 배열 기초

공장에 있는 온도 센서에서 1시간 마다 측정된 온도 값에 따라 에어컨을 가동한다. 에어컨이 가동된 시간을 구하여 출력하는 슈도코드이다. 빈칸을 채우시오.

참고

1. 공장은 하루 8시간 가동한다.
2. 온도는 실시간으로 입력되지만, 여기서는 배열에 값을 저장해 놓는 것으로 대신한다.
3. 온도가 30도가 넘으면 에어컨을 가동 시킨다.

pseudocode

```
1    public static void main(String[] args) {
2            int degree[]={28, 30, 29, 32, 31, 28, 29, 30};
3            int airConditioner = 0;









4            System.out.printf("에어컨은 총 %d시간 동안 가동되었다.₩n", airConditioner);
5    }
```

1차원 배열 기초

배열 A의 내용을 다음과 같이 배열 B에 저장하는 슈도코드를 작성하라.

참고

배열 A:	1	2	3	4	5	6	7	8	9	10

배열 B:	10	9	8	7	6	5	4	3	2	1

pseudocode

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




}
```

1차원 배열 기초

배열 A의 내용을 다음과 같이 배열 B에 저장하는 슈도코드를 작성하라.

참고

배열 A:	1	2	3	4	5	6	7	8	9	10

배열 B:	6	7	8	9	10	1	2	3	4	5

pseudocode

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




}
```

명령행 매개변수

명령행 매개변수란 프로그램을 실행할 때 함께 전달하는 정보를 의미한다.

```
java 클래스 문자열0 문자열1 문자열2 ... 문자열n-1
```

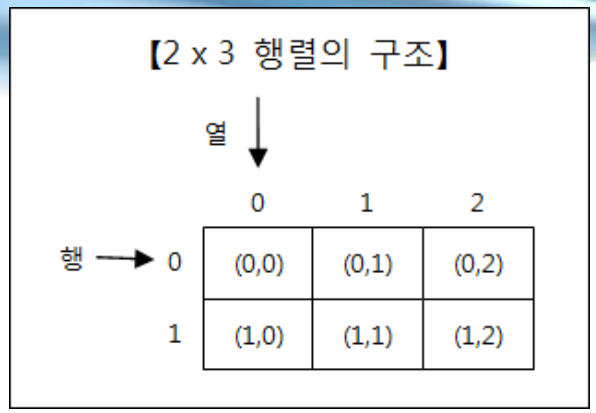
```
String[] args = {문자열0, 문자열1, 문자열2, ...문자열n-1}
```

main() 메서드 호출 시 전달

```
public static void main(String[] args){
    ...
}
```

args는 String 타입의 배열 주소를 가지는 변수인데 이것을 명령행 매개변수라고 한다.

배열 타입

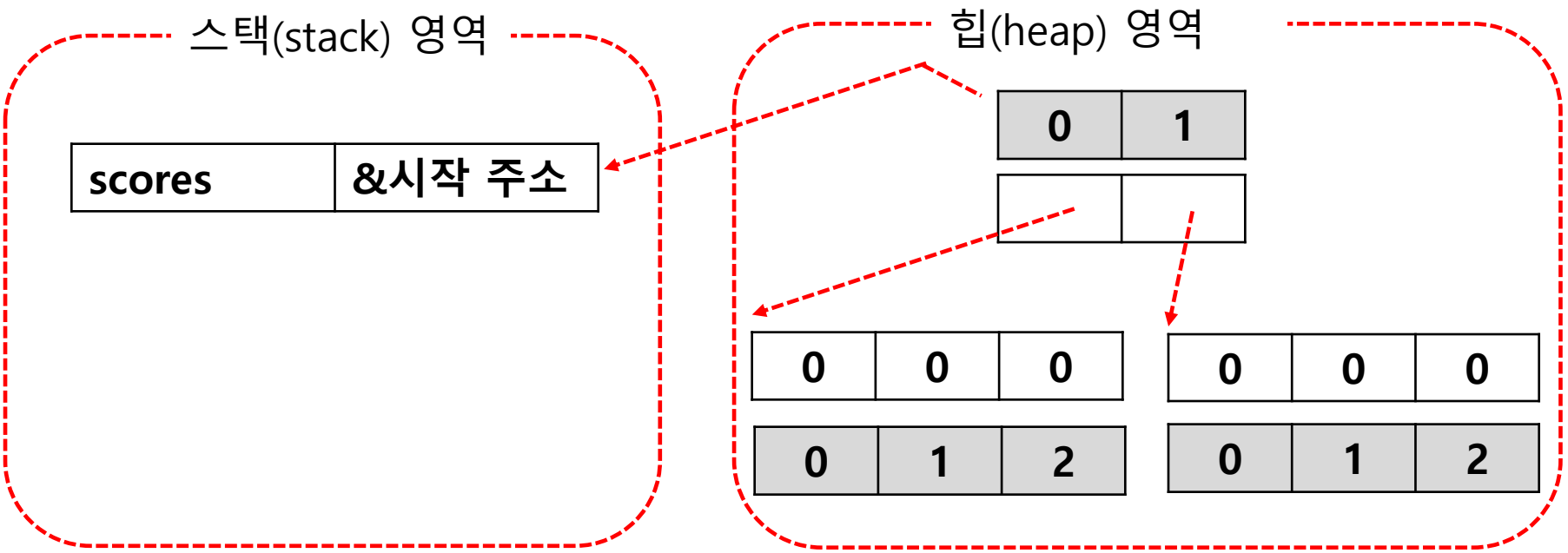


- ❖ 다차원 배열
  - 2차원 배열 이상의 배열
    - 수학의 행렬과 같은 자료 구조
  - 자바는 1차원 배열을 이용해 2원 배열 구현

```
int[][] scores = new int[2][3];
```

행의 수

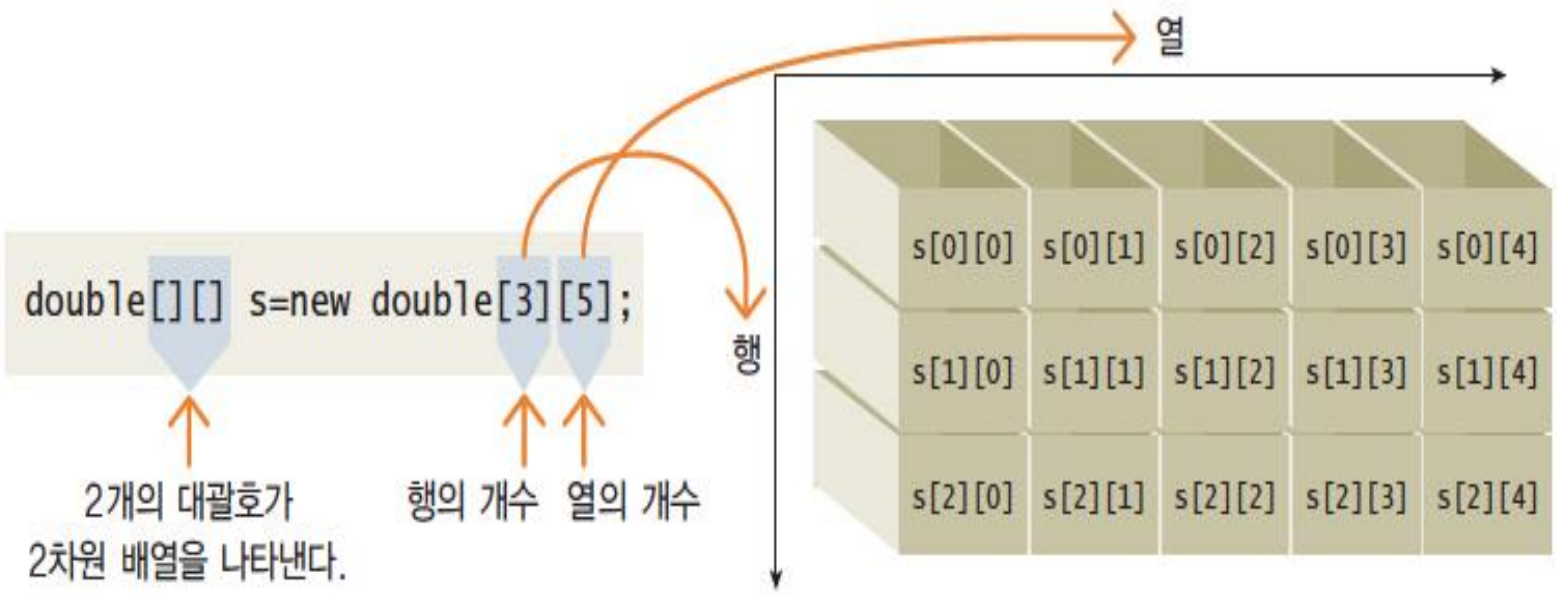
열의 수



배열 타입

1차원 배열은 행이 하나로 고정된 구조이고 2차원 배열은 여러 개의 행과 열을 가지는 구조이다.

```
double [ ][ ] s = new double[3][5];
```



2차원 배열

2차원 배열을 처리하는 방법은 중첩된 반복문을 사용한다.

```
 for(int i=0; i<3; i++){
      for(int j=0; j<5; j++)
           System.out.printf("s[%d][%d]=%.1f ₩t", i, j, s[i][j]);
      System.out.println();
}
```

[결과]

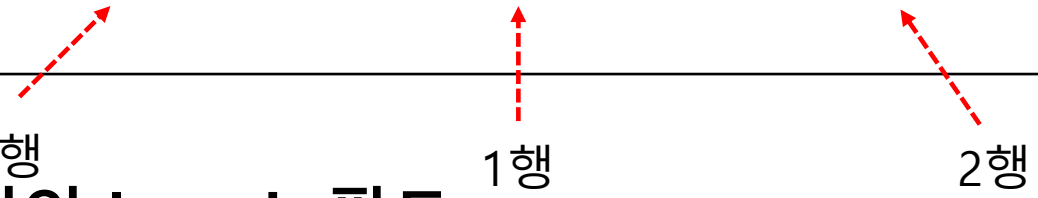
```
s[0][0]=0.0     s[0][1]=0.0     s[0][2]=0.0     s[0][3]=0.0     s[0][4]=0.0
s[1][0]=0.0     s[1][1]=0.0     s[1][2]=0.0     s[1][3]=0.0     s[1][4]=0.0
s[2][0]=0.0     s[2][1]=0.0     s[2][2]=0.0     s[2][3]=0.0     s[2][4]=0.0
```

2차원 배열

1) 2차원 배열의 초기화

2차원 배열의 초기화도 중괄호를 이용한다. 2차원 배열에서는 같은 행의 원소를 중괄호로 묶으면 된다.

```
int[ ] [ ] testArray = {
                    {10, 20, 30}, {40, 50, 60}, {70, 80, 90}
};
```



2) 2차원 배열에서의 length 필드

전체적으로 하나의 length필드가 있고 행의 개수를 나타낸다. 각 행마다 별도의 length필드가 있고 이것은 각 행이 가지고 있는 열의 개수를 나타낸다.

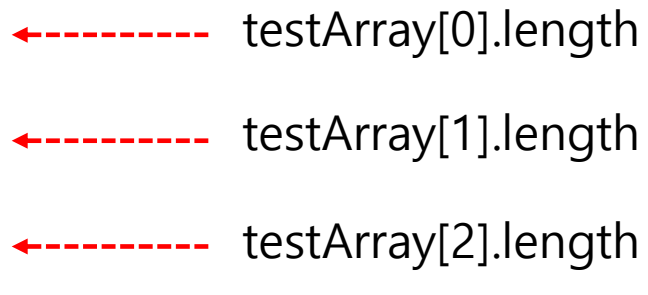
2차원 배열

```
int[ ][ ] testArray = {
                    {10, 20, 30}, {40, 50, 60}, {70, 80, 90}
};
```

testArray.length	2차원 배열에서 행의 길이
testArray[행 인덱스].length	2차원 배열에서 열의 길이

testArray.length

10	20	30
40	50	60
70	80	90



testArray.length	2차원 배열에서 행의 길이
testArray[0].length	2차원 배열에서 0번째 열의 길이
testArray[1].length	2차원 배열에서 1번째 열의 길이
testArray[2].length	2차원 배열에서 2번째 열의 길이

2차원 배열

3) 2차원 배열을 메소드에 넘기기

2차원 배열도 물론 메소드에 전달할 수 있다. 그러면 메소드의 정의는 어떻게 하여야 할까? 또 무엇이 전달되는 것일까? 2차원 배열의 경우에도 배열 참조변수가 전달된다.

4) 다차원 배열

자바에서도 얼마든지 다차원 배열을 생성할 수 있다.

```
double[ ] [ ] [ ] sales = new double[3][2][12];
```

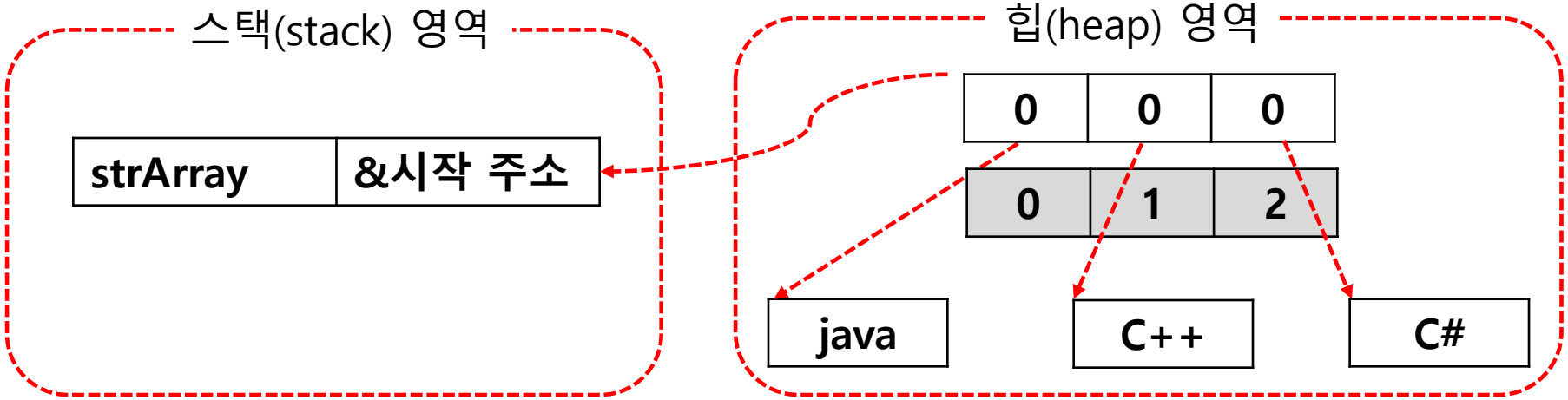
객체를 참조하는 배열

❖객체를 참조하는 배열

- 기본 타입(byte, char, short, int, long, float, double, boolean) 배열
  - 각 항목에 직접 값을 가지고 있음

- 참조 타입(클래스, 인터페이스) 배열 - 각 항목에 객체의 번지 가짐

```
String[] strArray = new String[3];
strArray[0] = "java";
strArray[1] = "C++"
strArray[2] = "C#";
```

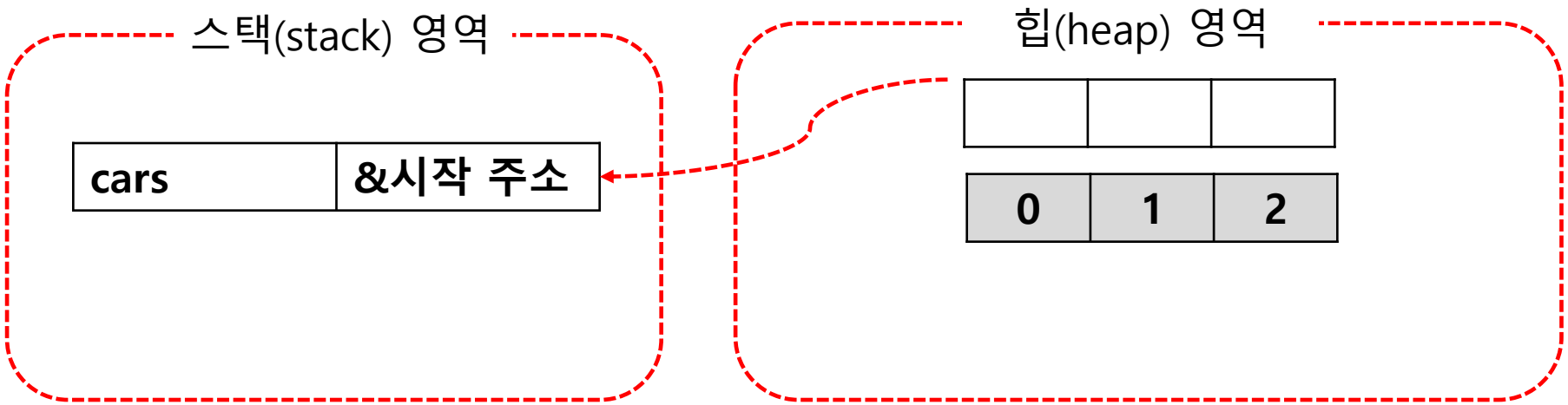


객체들의 배열

배열도 기초형 변수들의 모임이 될 수도 있고 참조형 변수들의 모임이 될 수도 있다.

```
class Car{
      private int speed;
      private int gear;
}
```

```
Car[ ] cars = new Car[3];
```



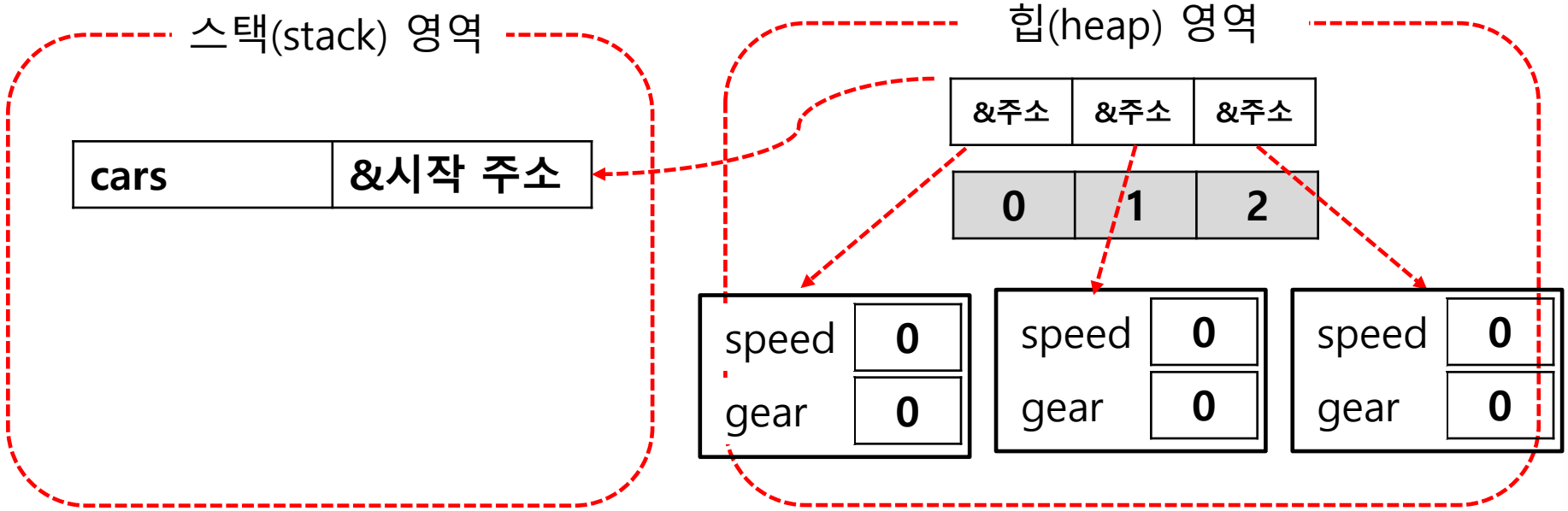
객체들의 배열

실제 객체는 다음과 같이 생성하여야 한다.

```
cars[0] = new Car();          cars[2] = new Car();
cars[1] = new Car();
```

또는 for 루프를 이용하여서 다음과 같이 모든 객체들을 생성할 수 있다.

```
for ( int i=0; i < cars.length; i++)
      cars[i] = new Car();
```



배열 복사

1) 배열 복사

- 배열은 한 번 생성하면 크기 변경 불가
- 더 많은 저장 공간이 필요하다면 보다 큰 배열을 새로 만들고 이전 배 열로부터 항목 값들을 복사

2) 배열 복사 방법

- for문 이용
- System.arraycopy() 메소드 이용
- Arrays 클래스 이용

배열 복사

```
public class ArrayCopyByForExample {
	public static void main(String[] args) {
		int[] oldIntArray = { 1, 2, 3 };
		int[] newIntArray = new int[5];

		for(int i=0; i<oldIntArray.length; i++) {
			newIntArray[i] = oldIntArray[i];
		}

		for(int i=0; i<newIntArray.length; i++) {
			System.out.print(newIntArray[i] + ", ");
		}
	}
}
```

배열 복사

```
public class ArrayCopyExample {
        public static void main(String[] args) {
                String[] oldStrArray = { "java", "array",
                                         "copy" };
                String[] newStrArray = new String[5];

                System.arraycopy( oldStrArray, 0,
                 newStrArray, 0, oldStrArray.length);

                for(int i=0; i<newStrArray.length; i++) {
                        System.out.print(newStrArray[i] + ", ");
                }
        }
}
```


Thank You