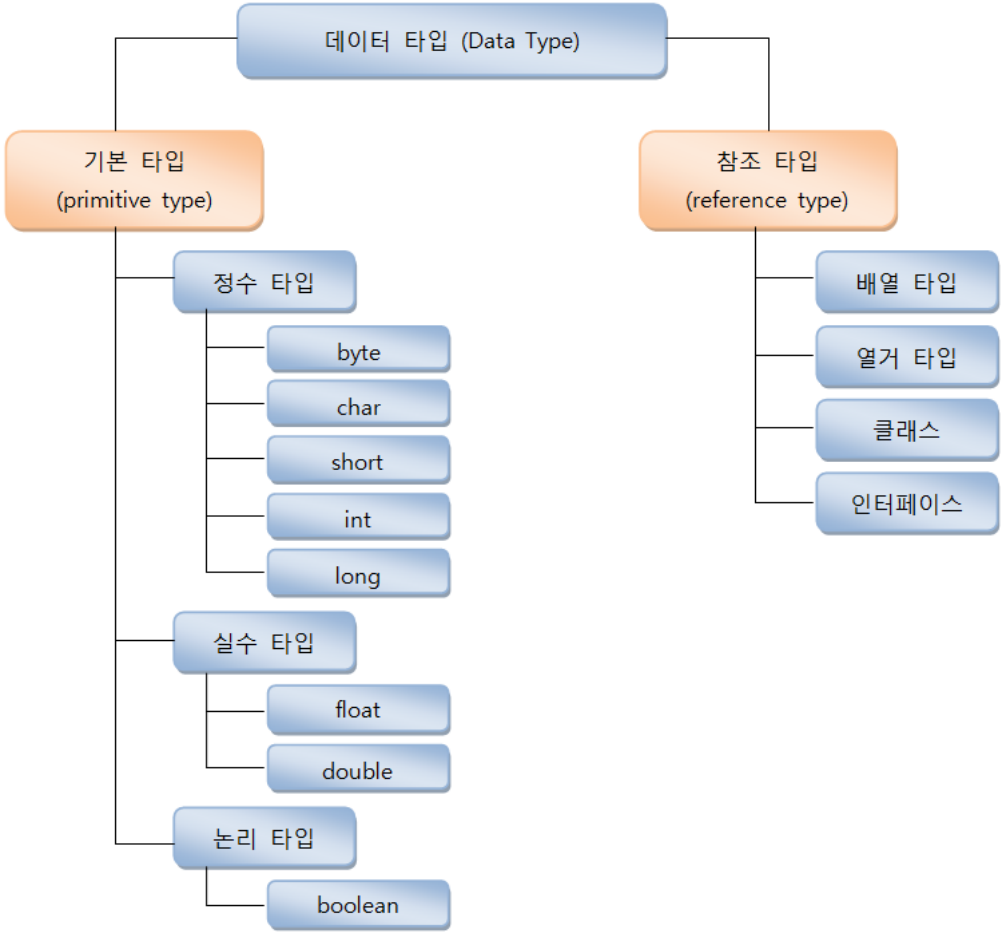
5장. 참조 타입

Contents

1절. 데이터 타입 분류

❖ 데이터 타입 분류

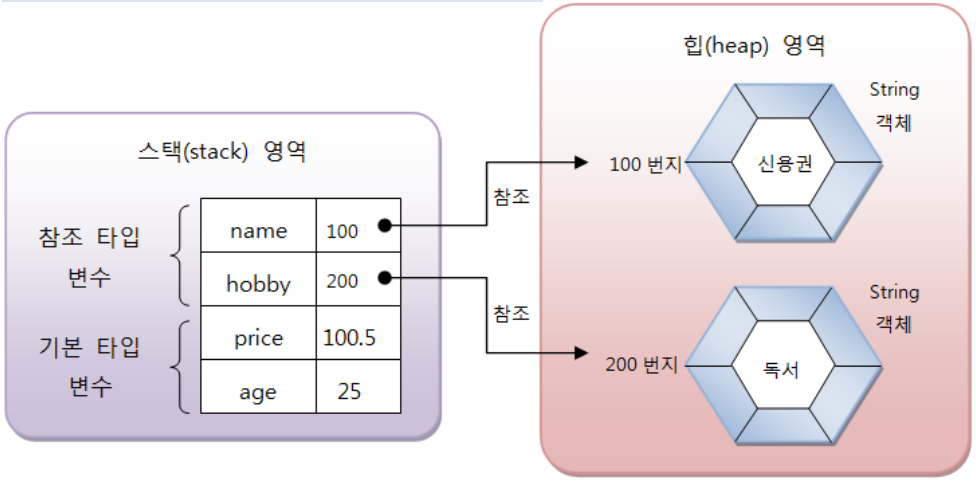


1절. 데이터 타입 분류

❖ 변수의 메모리 사용

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

```
[기본 타입 변수]
int age = 25;
double price = 100.5;

[참조 타입 변수]
String name = "신용권"
String hobby = "독서";
```

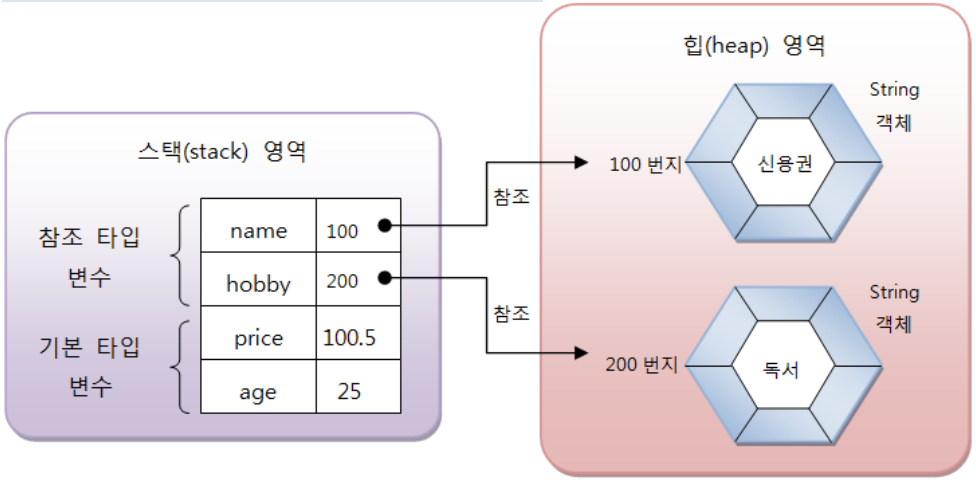


1절. 데이터 타입 분류

❖ 변수의 메모리 사용

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

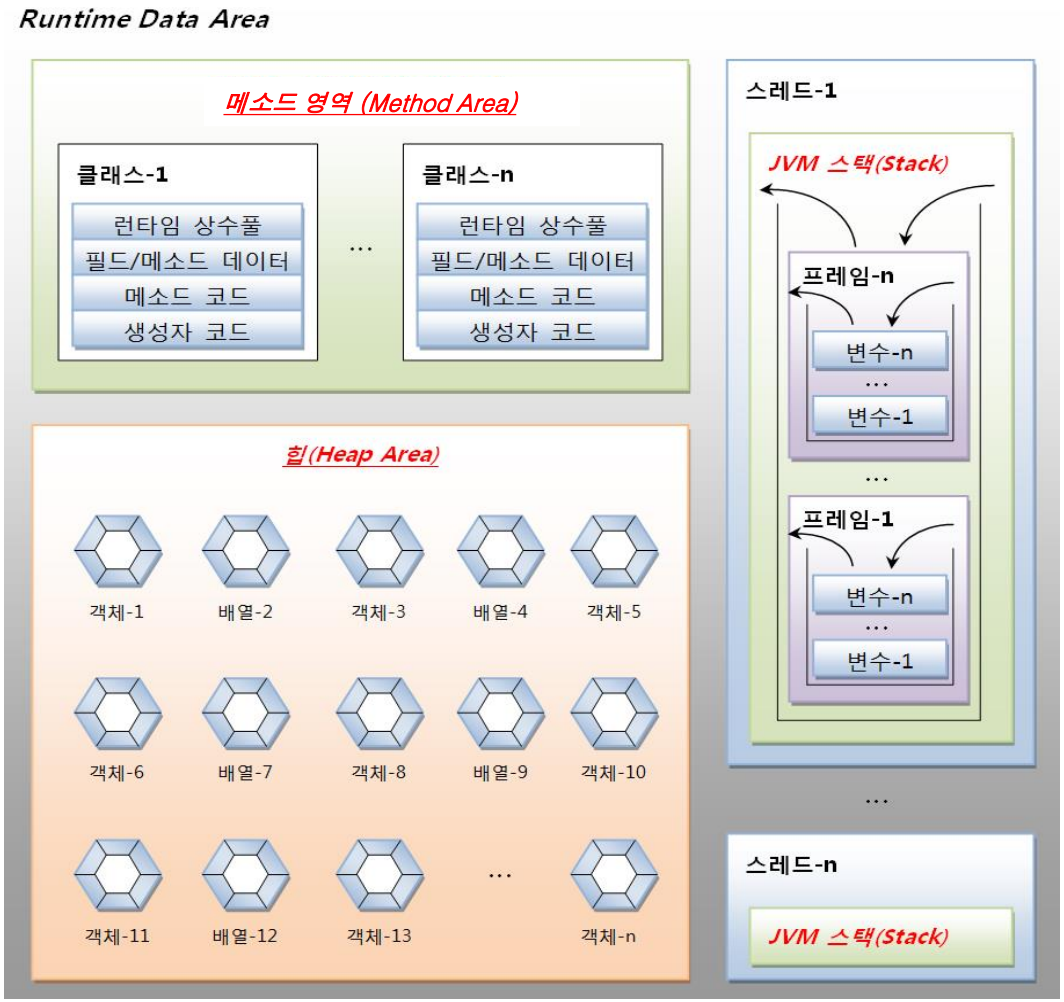
```
[기본 타입 변수]
int age = 25;
double price = 100.5;

[참조 타입 변수]
String name = "신용권"
String hobby = "독서";
```



2절. 메모리 사용 영역

❖ JVM이 사용하는 메모리 영역

- OS에서 할당 받은 메모리 영역(Runtime Data Area)을 세 영역으로 구분

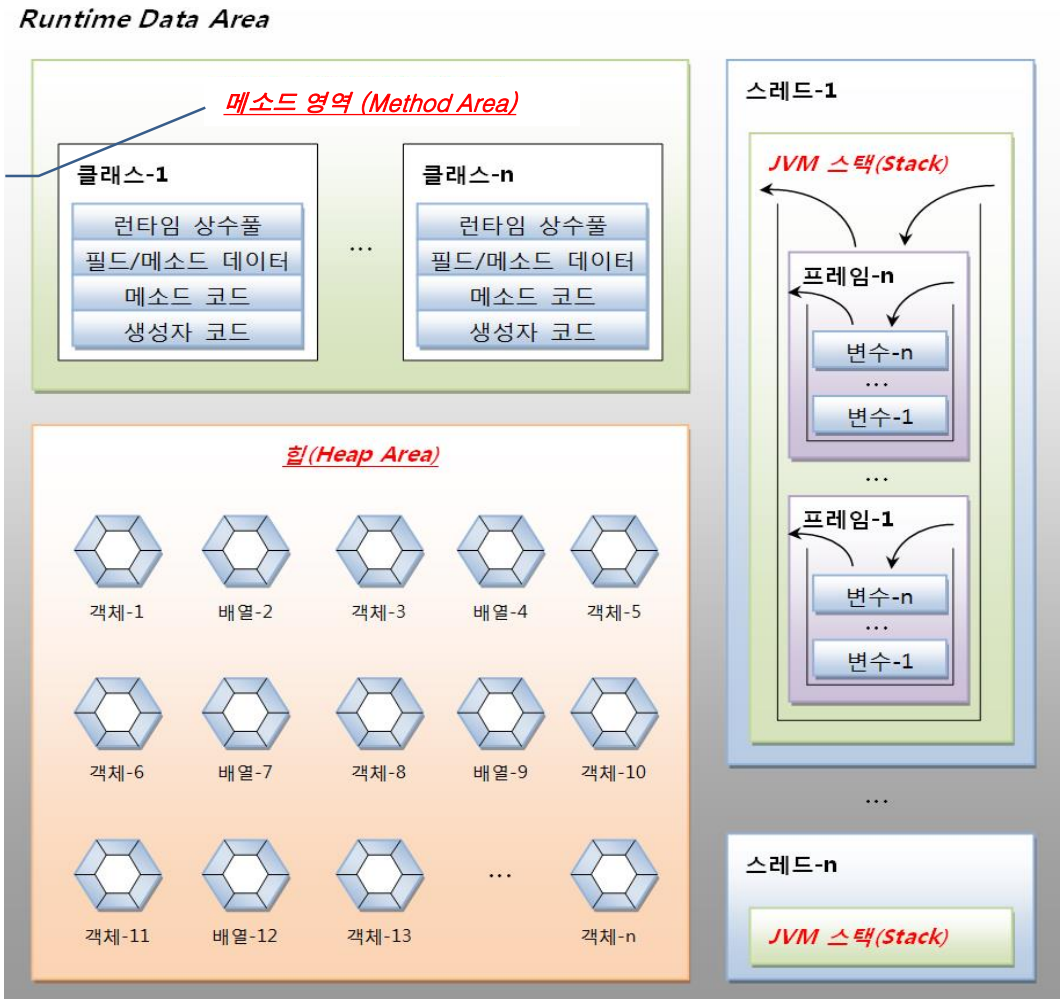


2절. 메모리 사용 영역

❖ JVM이 사용하는 메모리 영역

- OS에서 할당 받은 메모리 영역(Runtime Data Area)을 세 영역으로 구분

- JVM 시작할 때 생성
- 로딩된 클래스바이트 코드 내용을 분석 후 저장
- 모든 스레드가 공유



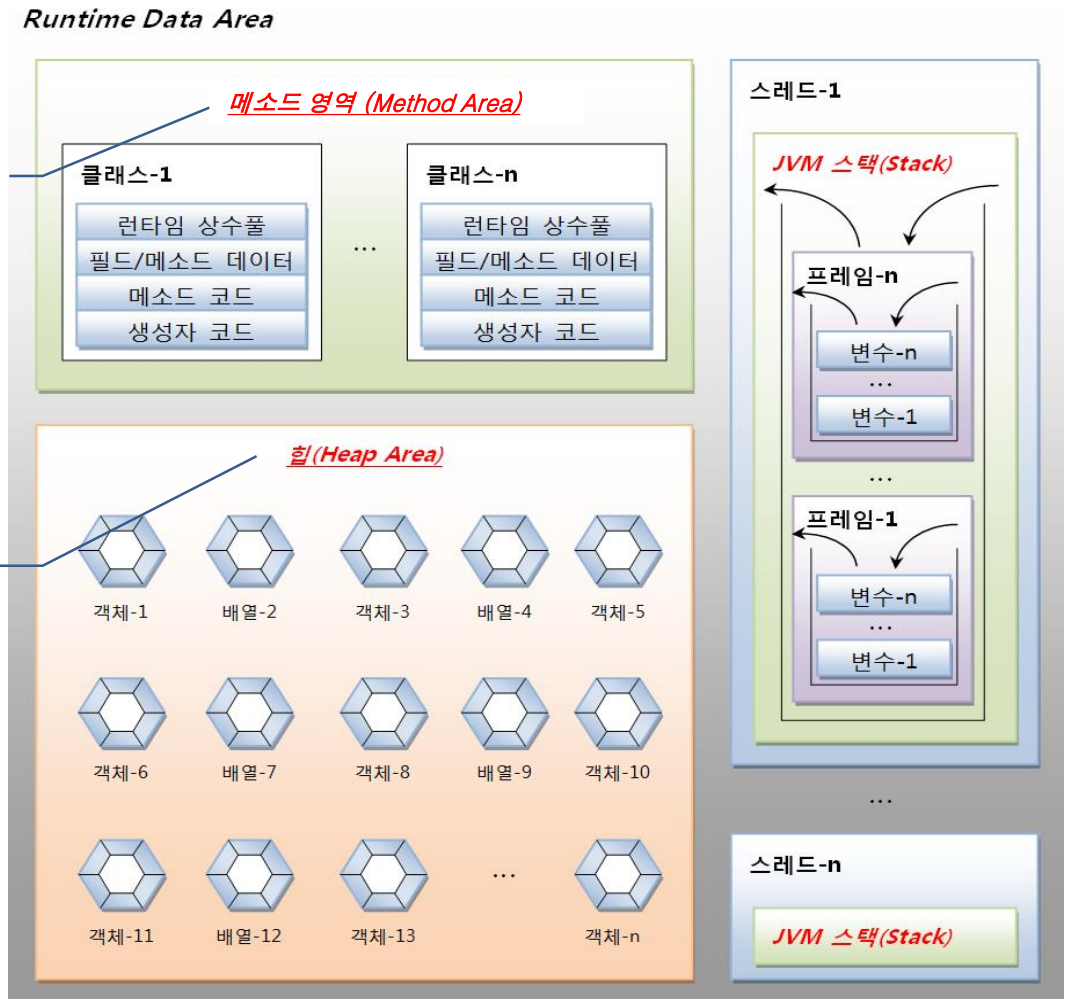
2절. 메모리 사용 영역

❖ JVM이 사용하는 메모리 영역

- OS에서 할당 받은 메모리 영역(Runtime Data Area)을 세 영역으로 구분

- JVM 시작할 때 생성
- 로딩된 클래스바이트 코드 내용을 분석 후 저장
- 모든 스레드가 공유

- JVM 시작할 때 생성
- 객체/배열 저장
- 사용되지 않는 객체는 자동 제거

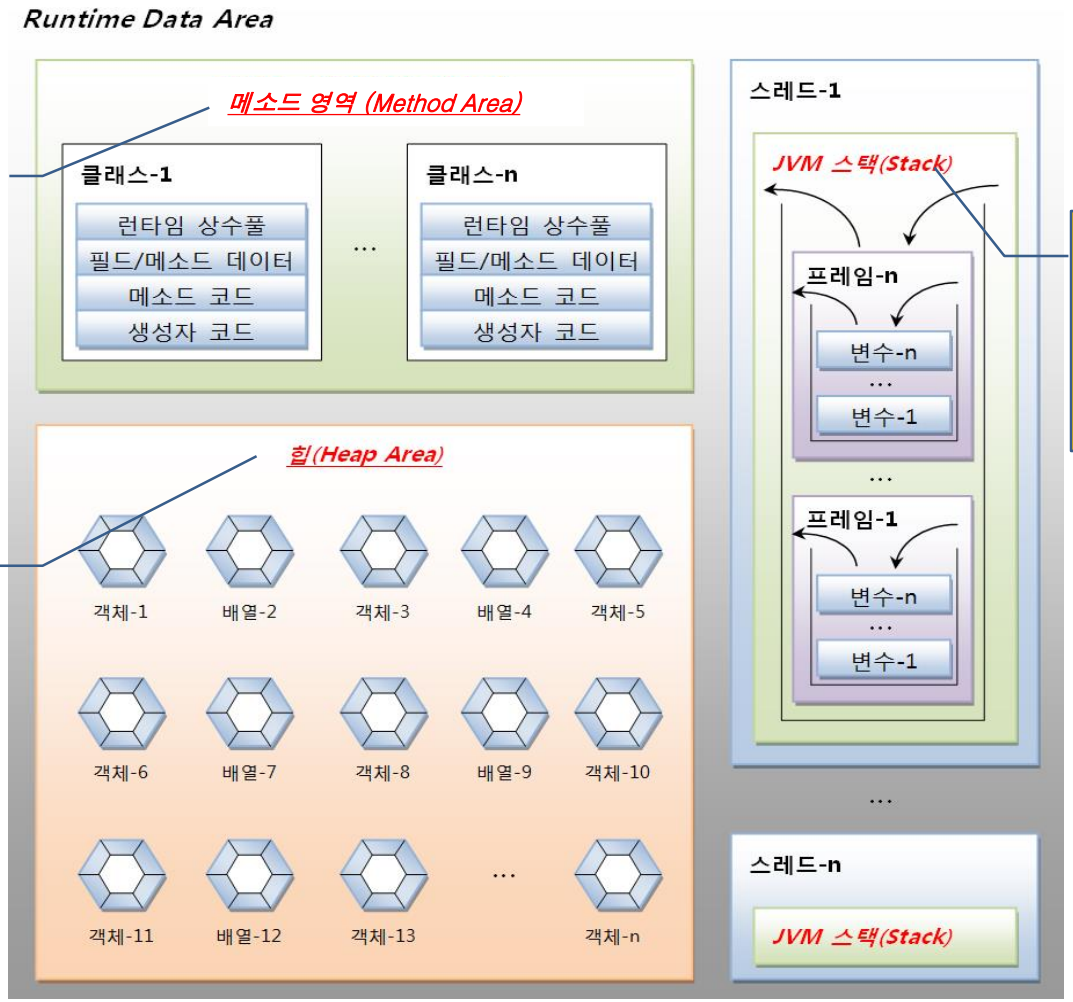


2절. 메모리 사용 영역

❖ JVM이 사용하는 메모리 영역

- OS에서 할당 받은 메모리 영역(Runtime Data Area)을 세 영역으로 구분

- JVM 시작할 때 생성
- 로딩된 클래스바이트 코드 내용을 분석 후 저장
- 모든 스레드가 공유



- JVM 시작할 때 생성
- 객체/배열 저장
- 사용되지 않는 객체는 자동 제거

- 스레드 별 생성
- 메소드 호출할 때마다 Frame을 스택에 추가(push)
- 메소드 종료하면 Frame 제거(pop)

2절. 메모리 사용 영역

❖ JVM이 사용하는 메모리 영역

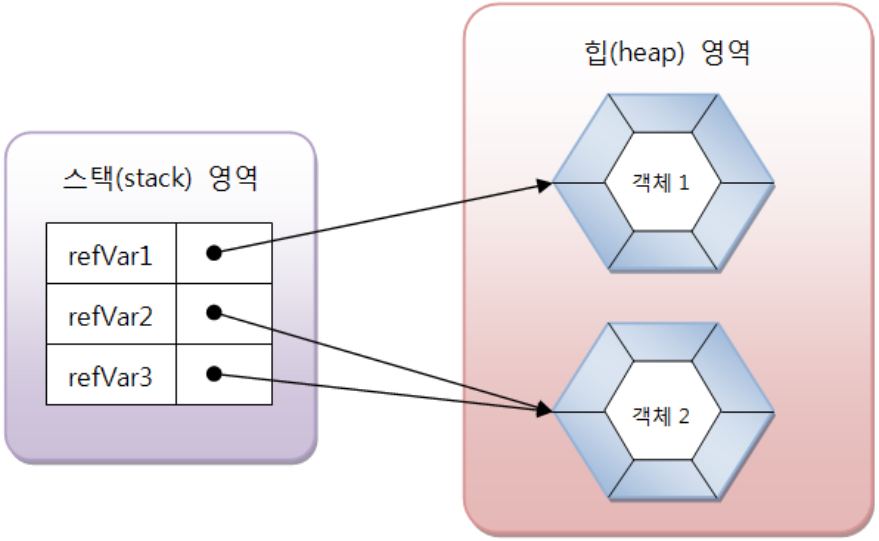
- 메소드 영역
  - JVM 시작할 때 생성
  - 로딩된 클래스 바이트 코드 내용을 분석 후 저장
  - 모든 스레드가 공유
- 힙 영역
  - JVM 시작할 때 생성
  - 객체/배열 저장
  - 사용되지 않는 객체는 Garbage Collector 가 자동 제거
- JVM 스택
  - 스레드 별 생성
  - 메소드 호출할 때마다 Frame을 스택에 추가(push)
  - 메소드 종료하면 Frame 제거(pop)

3절. 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
```

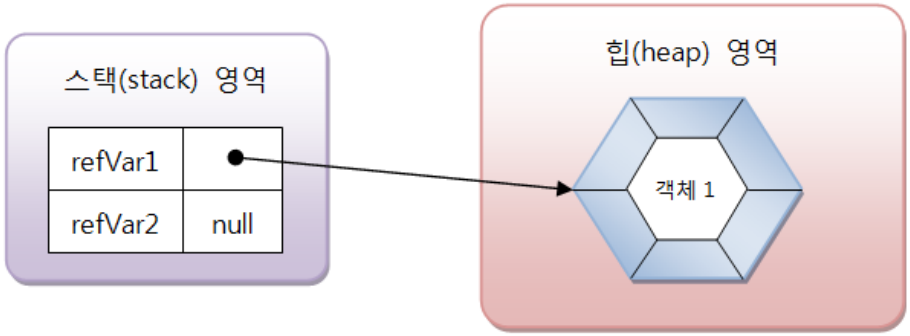
```
refVar2 == refVar3      결과:   true
refVar2 != refVar3      결과:   false
```

```
if( refVar2 == refVar3 ) { ... }
```

4절. 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

그림에서 refVar1 은 힙 영역의 객체를 참조하므로 연산의 결과는 다음과 같다.

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

refVar2 는 null 값을 가지므로 연산의 결과는 다음과 같다.

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

4절. null과 NullPointerException

❖ NullPointerException의 의미

- 예외(Exception)
  - 사용자의 잘못된 조작이나 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
```

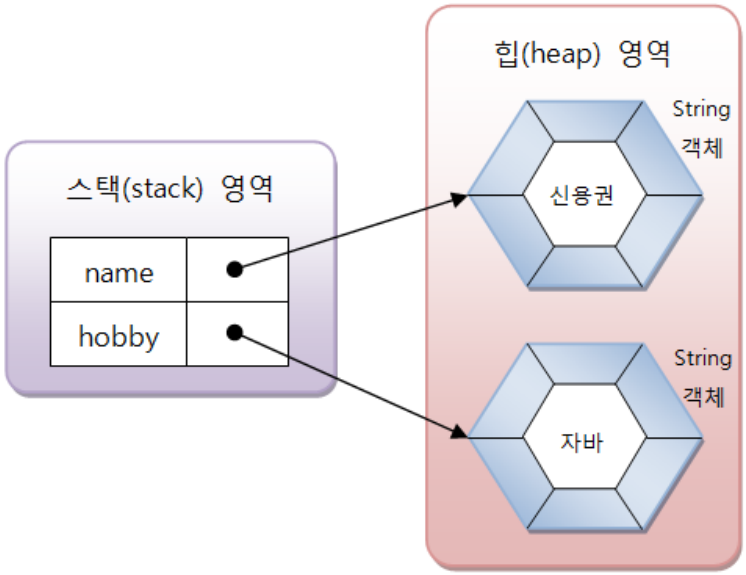
```
String str = null;
System.out.println("총 문자수: " + str.length());    //NullPointerException
```

5절. String 타입

❖ String 타입

- 문자열을 저장하는 클래스 타입

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

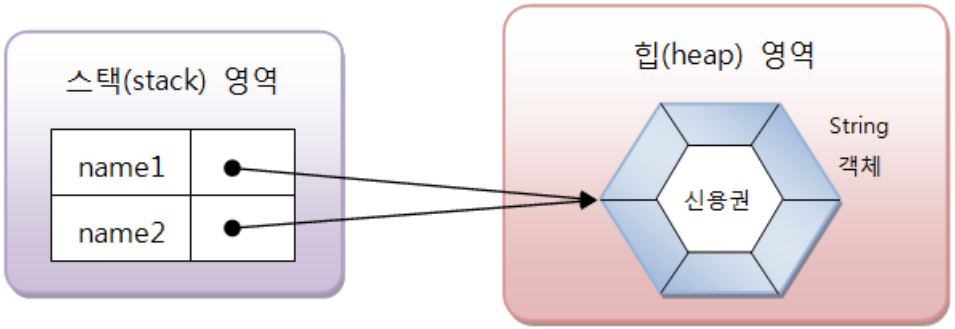


5절. String 타입

❖ String 타입

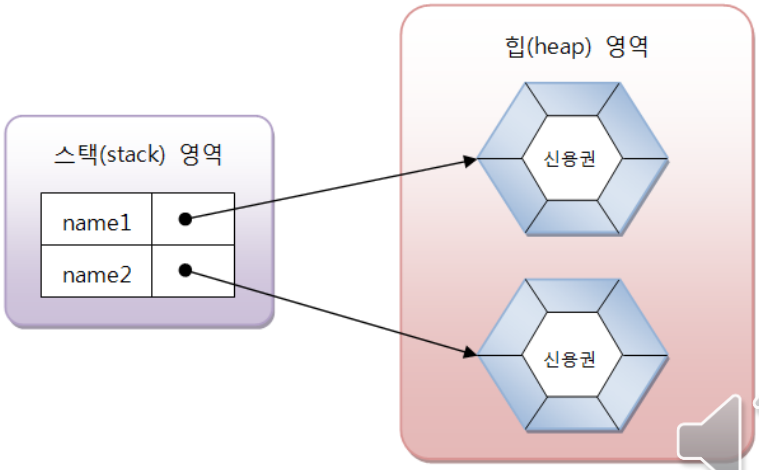
- 문자열 리터럴 동일하다면 String 객체 공유

```
String name1 = "신용권";
String name2 = "신용권";
```



- **new** 연산자를 이용한 String 객체 생성
  - 힙 영역에 새로운 String 객체 생성
  - String 객체를 생성한 후 번지 리턴

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



참조 타입 실습 (Reference1.java)

Reference1.java

```java
package week6;

public class Reference1 {
    public static void main(String[] args) {
        //기본 데이터 타입의 변수는 스택영역에 생성되면서 값도 같이 저장된다
        int iVal1 = 100;
        int iVal2 = 100;
        System.out.println("iVal1 == iVal2 : " + (iVal1 == iVal2));

        //참조 타입의 변수는 스택영역에 생성되면서 해당 객체의 주소를 저장한다
        String sVal1 = "Java";
        String sVal2 = "Java";

        //자바는 문자열 리터럴이 동일하다면 String 객체를 공유한다
        System.out.println("sVal1==sVal2:"+(sVal1==sVal2?"참조가 같음":"참조가 다름"));

        //참조타입의 변수는 초기값으로 null을 사용할 수 있다(스택영역에만 생성됨)
        String sVal5 = null;

        //참조타입의 변수가 null 값을 가지고 있을 때는 해당 변수를 사용할 수 없다
        // =>NullPointerException 에러 발생
        //System.out.println("sVal5.length = " + sVal5.length());

        //값을 대입하는 순간 힙영역에 객체가 생성되고 해당 주소가 스택영역의 변수값에 저장됨
        sVal5 = "Java";
        System.out.println("sVal5 = " + sVal5);
        System.out.println("sVal1==sVal5:"+(sVal1==sVal5?"참조가 같음":"참조가 다름"));
    }
}
```

참조 타입 실습 (Reference1.java)

Reference1.java

```
1  package week6;
2
3  public class Reference1 {
4      public static void main(String[] args) {
5          //기본 데이터 타입의 변수는 스택영역에 생성되면서 값도 같이 저장된다
6          int iVal1 = 100;
7          int iVal2 = 100;
8          System.out.println("iVal1 == iVal2 : " + (iVal1
9
10         //참조 타입의 변수는 스택영역에 생성되면서 해당 객체의 주소를 저장한다
11         String sVal1 = "Java";
12         String sVal2 = "Java";
13
14         //자바는 문자열 리터럴이 동일하다면 String 객체를 공유한다
15         System.out.println("sVal1==sVal2:"+(sVal1==sVal2?"참조가 같음":"참조가 다름"));
16
17         //참조타입의 변수는 초기값으로 null을 사용할 수 있다(스택영역에만 생성됨)
18         String sVal5 = null;
19
20         //참조타입의 변수가 null 값을 가지고 있을 때는 해당 변수를 사용할 수 없다
21         // =>NullPointerException 에러 발생
22         //System.out.println("sVal5.length = " + sVal5.length());
23
24         //값을 대입하는 순간 힙영역에 객체가 생성되고 해당 주소가 스택영역의 변수값에 저장됨
25         sVal5 = "Java";
26         System.out.println("sVal5 = " + sVal5);
27         System.out.println("sVal1==sVal5:"+(sVal1==sVal5?"참조가 같음":"참조가 다름"));
28     }
29 }
```

Console

```
<terminated> Reference1 [Java Application] C:₩Program Files₩
iVal1 == iVal2 : true
sVal1==sVal2:참조가 같음
sVal5 = Java
sVal1==sVal5:참조가 같음
```

참조 타입 실습 (Reference2.java)

Reference2.java

```java
package week6;

public class Reference2 {
    public static void main(String[] args) {
        //자바는 문자열 리터럴이 동일하다면 String 객체를 공유한다
        String sVal1 = "Java";
        String sVal2 = "Java";
        System.out.println("sVal1==sVal2:"+(sVal1==sVal2?"참조가 같음":"참조가 다름"));

        //new 연산자를 이용하는 경우에는 힙 영역에 새로운 String 객체가 생성된다
        String sVal3 = new String("Java");
        String sVal4 = new String("Java");
        System.out.println("sVal3==sVal4:"+(sVal3==sVal4?"참조가 같음":"참조가 다름"));

        //String 객체가 동일한 문자열을 가지고 있는지 비교할 경우에는
        //반드시 equals() 메소드를 사용한다
        if (sVal3.equals(sVal4))
            System.out.println("sVal3와 sVal4의 문자열은 동일하다.");
        else
            System.out.println("sVal3와 sVal4의 문자열은 다르다.");
    }
}
```

참조 타입 실습 (Reference2.java)

Reference2.java

```
1  package week6;
2
3  public class Reference2 {
4      public static void main(String[] args) {
5          //자바는 문자열 리터럴이 동일하다면 String 객체를 공유한
6          String sVal1 = "Java";
7          String sVal2 = "Java";
8          System.out.println("sVal1==sVal2:"+(sVal1==sVal2?"참조가 같음":"참조가 다름"));
9
10         //new 연산자를 이용하는 경우에는 힙 영역에 새로운 String 객체가 생성된다
11         String sVal3 = new String("Java");
12         String sVal4 = new String("Java");
13         System.out.println("sVal3==sVal4:"+(sVal3==sVal4?"참조가 같음":"참조가 다름"));
14
15         //String 객체가 동일한 문자열을 가지고 있는지 비교할 경우에는
16         //반드시 equals() 메소드를 사용한다
17         if (sVal3.equals(sVal4))
18             System.out.println("sVal3와 sVal4의 문자열은 동일하다.");
19         else
20             System.out.println("sVal3와 sVal4의 문자열은 다르다.");
21     }
22 }
```

Console

```
<terminated> Reference2 [Java Application] C:₩Program Files₩
sVal1==sVal2:참조가 같음
sVal3==sVal4:참조가 다름
sVal3와 sVal4의 문자열은 동일하다.
```

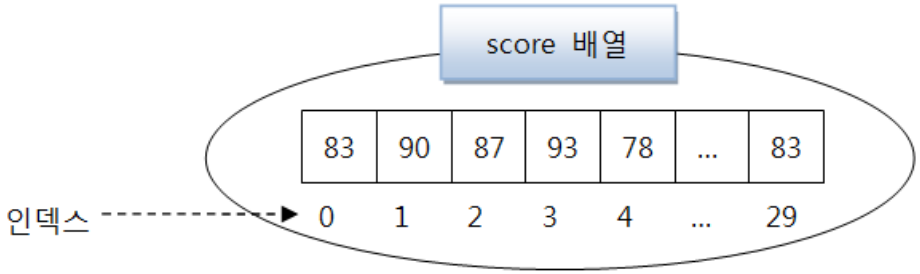
6절. 배열 타입

❖ 배열이란?

- 같은 타입의 데이터를 연속된 공간에 저장하는 자료구조

- 각 데이터 저장 위치는 인덱스 부여해 접근

```
int score1= 83;
int score2 = 90;
int score3 = 87;
 :
int score30= 75;
```



항목 접근: 배열이름[인덱스] ex) score[0], score[3]

6절. 배열 타입

❖ 배열의 장점

- 중복된 변수 선언 줄이기 위해 사용

- 반복문 이용해 요소들을 쉽게 처리

```
int sum = score1;
sum += score2;
sum += score3;
  :
sum += score30;
int avg = sum / 30;
```

```
int sum = 0;
for(int i=0; i<30; i++) {
   sum += score[i];
}
int avg = sum / 30;
```

6절. 배열 타입

❖ 배열 선언

- 배열을 사용하기 위해 우선 배열 변수 선언

```
타입[ ] 변수;
```

```
타입 변수[ ];
```

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

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

- 배열 변수는 참조 변수 – 배열 생성되기 전 null로 초기화 가능

```
타입[] 변수 = null;
```

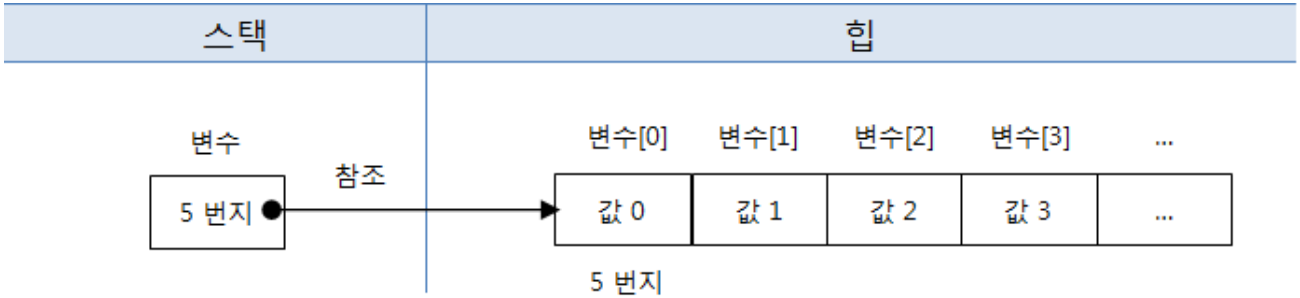
  - 배열 변수가 null 값을 가진 상태에서 항목에 접근 불가
    - "변수[인덱스]" 로 접근 못함
      - » NullPointerException 발생

6절. 배열 타입

❖ 값 목록으로 배열 생성하는 방법

▪ 변수 선언과 동시에 값 목록 대입

```
데이터타입[] 변수 = { 값 0, 값 1, 값 2, 값 3, ... };
```



▪ 변수 선언 후 값 목록 대입

```
데이터타입[] 변수;
변수 = new 타입[] { 값 0, 값 1, 값 2, 값 3, ... };
```

단, 다음과 같이 생성하면 에러발생

```
데이터타입[] 변수;
변수 = {값0, 값1, 값2, 값3,...};
```

6절. 배열 타입

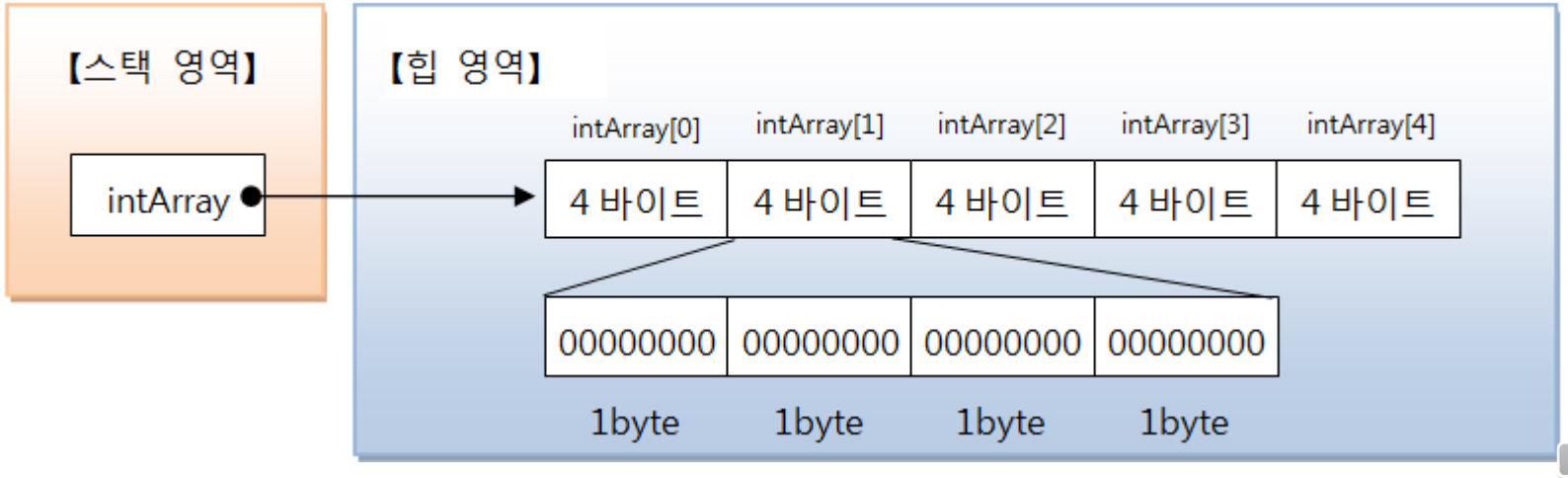
❖ new 연산자로 배열 생성

- 배열 생성시 값 목록을 가지고 있지 않음
- 향후 값들을 저장할 배열을 미리 생성하고 싶을 경우

```
타입[] 변수 = new 타입[길이];
```

```
타입[] 변수 = null;
변수 = new 타입[길이];
```

```
int[] intArray = new int[5];
```



6절. 배열 타입

❖ 타입 별 항목의 기본값

- new 연산자로 배열을 처음 생성할 경우, 배열은 자동적으로 기본값으로 초기화된다

분류	데이터 타입	초기값
기본 타입 (정수)	byte[]	0
	char[]	'₩u0000'
	short[]	0
	int[]	0
	long[]	0L
기본 타입 (실수)	float[]	0.0F
	double[]	0.0
기본 타입 (논리)	boolean[]	false
참조 타입	클래스[]	null
	인터페이스[]	null

6절. 배열 타입

❖ 배열의 길이

- 배열에 저장할 수 있는 전체 항목 수
- 코드에서 배열의 길이 얻는 방법

```
배열변수.length;
```

```
int[] intArray = { 10, 20, 30 };
int num = intArray.length;
```

- 배열의 길이는 읽기 전용

```
intArray.length = 10;   //잘못된 코드
```

배열의 인덱스 범위는 0 ~ (길이-1) 이다.
(조건식에 < 를 사용)

- 배열의 길이는 for문의 조건식에서 주로 사용

```
int[] scores = { 83, 90, 87 };

int sum = 0;
for(int i=0; i<scores.length; i++) {
   sum += scores[i];
}
System.out.println("총합 : " + sum);
```

3

인덱스 범위를 초과해서 사용하게 되면
ArrayIndexOutOfBoundsException 에러

배열 실습 (Array1.java)

Array1.java

```java
package week6;

public class Array1 {
    public static void main(String[] args) {
        //값 목록으로 배열을 생성하는 방법
        int[]    score  = {70, 90, 65, 83, 50};
        String[] pNames = {"java", "c", "python", "c#"};
        double[] dArray1;
        dArray1 = new double[] {0.1, 0.2, 0.3, 0.4};

        //new 연산자로 배열을 생성하면 자동적으로 기본값으로 초기화된다
        int[] intArray = new int[5];
        System.out.println("intArray[0] = " + intArray[0]);

        double[] dArray2 = new double[5];
        System.out.println("dArray2[0] = " + dArray2[0]);

        String[] sArray  = new String[4];
        System.out.println("sArray[0] = " + sArray[0]);

        //배열에 새로운 값을 저장하려면 대입 연산자를 이용한다
        for (int i=0; i<4; i++) {
            sArray[i] = pNames[i];
            System.out.printf("sArray[%d] = %s\n", i, sArray[i]);
        }
    }
}
```

배열 실습 (Array1.java)

Array1.java

```java
package week6;

public class Array1 {
    public static void main(String[] args) {
        //값 목록으로 배열을 생성하는 방법
        int[]    score  = {70, 90, 65, 83, 50};
        String[] pNames = {"java", "c", "python", "c#"};
        double[] dArray1;
        dArray1 = new double[] {0.1, 0.2, 0.3, 0.4};

        //new 연산자로 배열을 생성하면 자동적으로 기본값으로 초기화된다
        int[] intArray = new int[5];
        System.out.println("intArray[0] = " + intArray[0]);

        double[] dArray2 = new double[5];
        System.out.println("dArray2[0] = " + dArray2[0]);

        String[] sArray  = new String[4];
        System.out.println("sArray[0] = " + sArray[0]);

        //배열에 새로운 값을 저장하려면 대입 연산자를 이용한다
        for (int i=0; i<4; i++) {
            sArray[i] = pNames[i];
            System.out.printf("sArray[%d] = %s\n", i, sArray[i]);
        }
    }
}
```

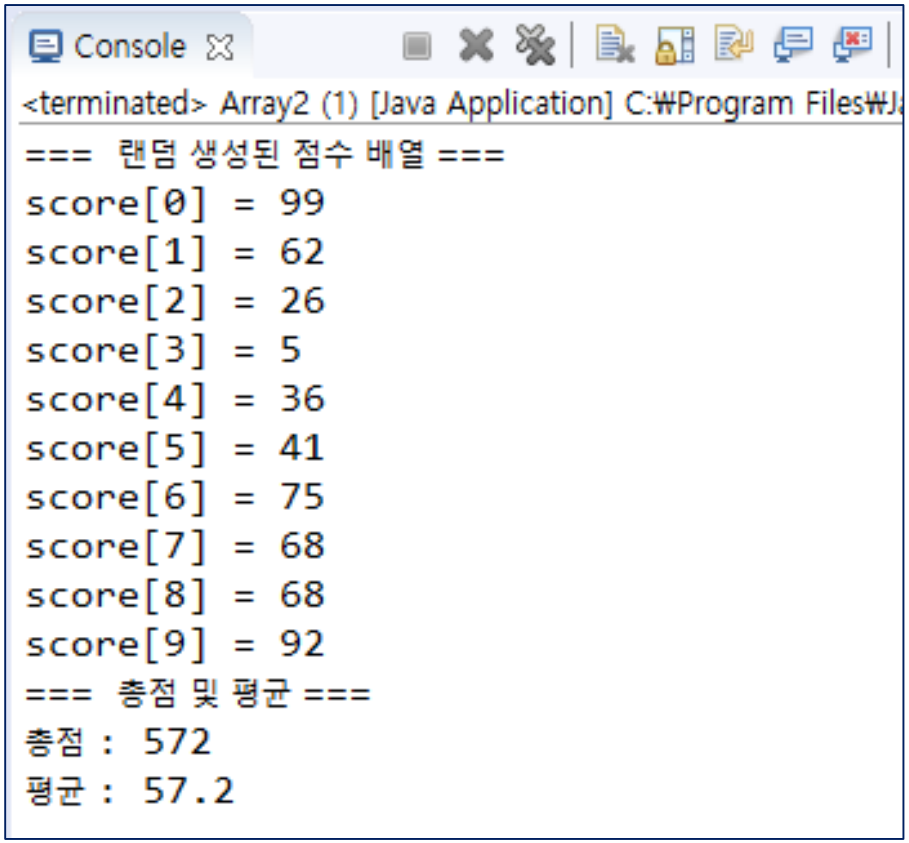
Console

<terminated> Array2 [Java Application] C:₩Program Files₩Java

```
intArray[0] = 0
dArray2[0] = 0.0
sArray[0] = null
sArray[0] = java
sArray[1] = c
sArray[2] = python
sArray[3] = c#
```

배열 실습 (Array2.java)

- 길이가 10인 정수 타입의 배열을 선언한다
- 각 배열 값에 랜덤하게 생성된 점수(0~99)를 대입한다
- 최종 생성된 점수값과 총점 및 평균을 출력하시오



배열 실습 (Array2.java)

- 길이가 10인 정수 타입의 배열을 선언한다
- 각 배열 값에 랜덤하게 생성된 점수(0~99)를 대입한다
- 최종 생성된 점수값과 총점 및 평균을 출력하시오

Array2.java

```
1  package week6;
2
3  public class Array2 {
4      public static void main(String[] args) {
5          int[] score = new int[10];
6
7          System.out.println("=== 랜덤 생성된 점수 배열 ===");
8          for (int i=0; i<score.length; i++) {
9              score[i] = (int)(Math.random() * 100);
10             System.out.printf("score[%d] = %d\n", i, score[i]);
11         }
12
13         System.out.println("=== 총점 및 평균 ===");
14         int    sum = 0;
15         double ave = 0.0;
16
17         for (int i=0; i<score.length; i++)
18             sum += score[i];
19
20         ave = (double)sum / score.length;
21         System.out.println("총점 : " + sum);
22         System.out.println("평균 : " + ave);
23     }
24 }
```

Console

```
<terminated> Array2 (1) [Java Application] C:₩Program Files₩J
=== 랜덤 생성된 점수 배열 ===
score[0] = 99
score[1] = 62
score[2] = 26
score[3] = 5
score[4] = 36
score[5] = 41
score[6] = 75
score[7] = 68
score[8] = 68
score[9] = 92
=== 총점 및 평균 ===
총점 : 572
평균 : 57.2
```

6절. 배열 타입

❖ 다차원 배열

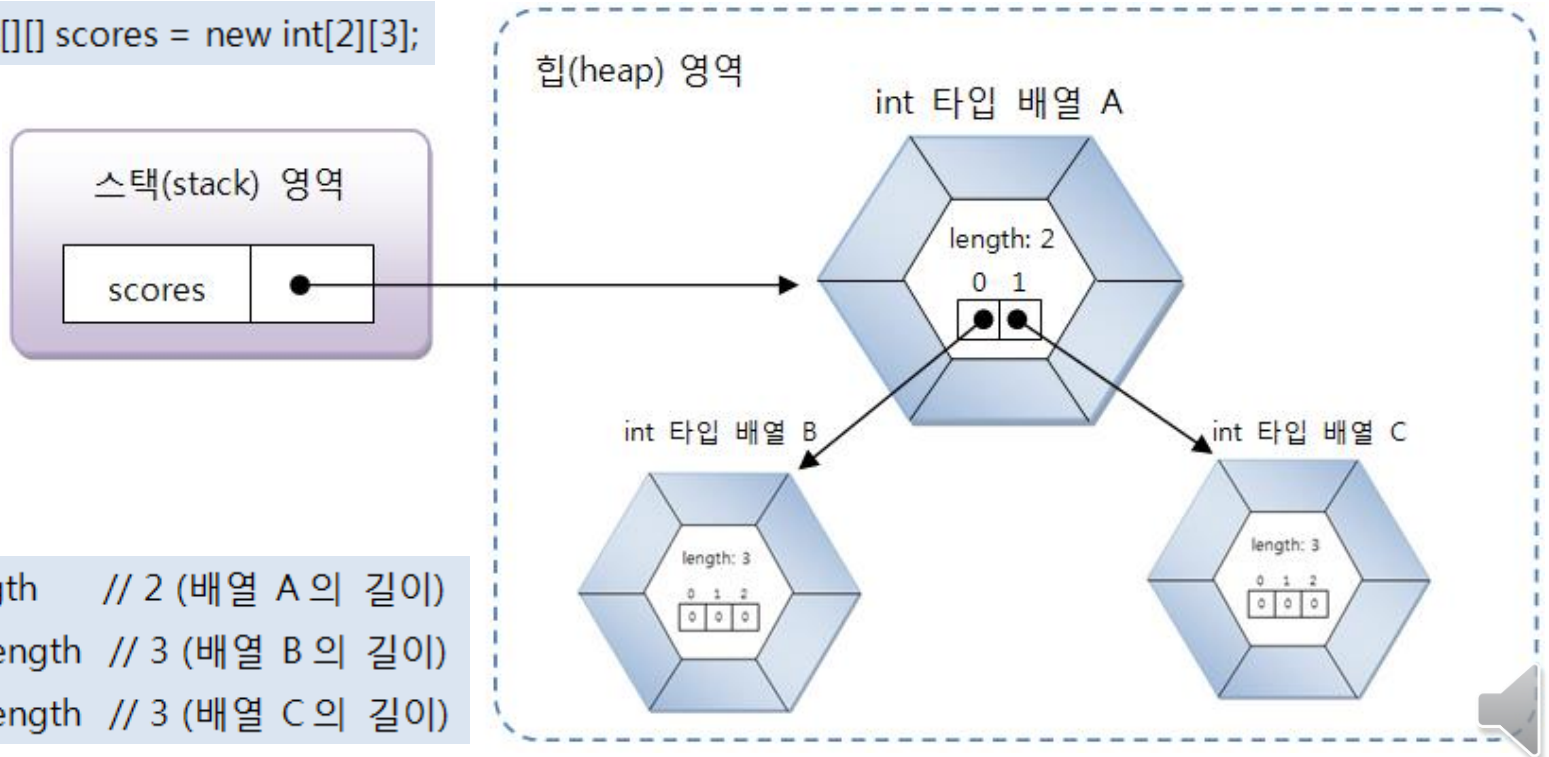
- 2차원 배열 이상의 배열
  - 수학의 행렬과 같은 자료 구조

【2 x 3 행렬의 구조】

행 \ 열	0	1	2
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)

- 자바는 1차원 배열을 이용해 2차원 배열 구현

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




```
scores.length     // 2 (배열 A의 길이)
scores[0].length  // 3 (배열 B의 길이)
scores[1].length  // 3 (배열 C의 길이)
```

다차원 배열 실습 (TwoDimension.java)

TwoDimension.java

```java
package week6;

public class TwoDimension {
    public static void main(String[] args) {
        int[][] math = new int[2][3];
        for (int i=0; i<math.length; i++) {
            for (int j=0; j<math[i].length; j++) {
                System.out.printf("math[%d][%d] = %d\n", i, j, math[i][j]);
            }
        }
        System.out.println();

        int[][] eng = new int[2][];
        eng[0] = new int[2];
        eng[1] = new int[3];
        for (int i=0; i<eng.length; i++) {
            for (int j=0; j<eng[i].length; j++) {
                System.out.printf("eng[%d][%d] = %d\n", i, j, eng[i][j]);
            }
        }
        System.out.println();

        int[][] kor = {{95, 80}, {92, 96, 80}};
        for (int i=0; i<kor.length; i++) {
            for (int j=0; j<kor[i].length; j++) {
                System.out.printf("kor[%d][%d] = %d\n", i, j, kor[i][j]);
            }
        }
    }
}
```

다차원 배열 실습 (TwoDimension.java)

TwoDimension.java

```java
package week6;

public class TwoDimension {
    public static void main(String[] args) {
        int[][] math = new int[2][3];
        for (int i=0; i<math.length; i++) {
            for (int j=0; j<math[i].length; j++) {
                System.out.printf("math[%d][%d] = %d\n", i, j, math
            }
        }
        System.out.println();

        int[][] eng = new int[2][];
        eng[0] = new int[2];
        eng[1] = new int[3];
        for (int i=0; i<eng.length; i++) {
            for (int j=0; j<eng[i].length; j++) {
                System.out.printf("eng[%d][%d] = %d\n", i, j, eng[i
            }
        }
        System.out.println();

        int[][] kor = {{95, 80}, {92, 96, 80}};
        for (int i=0; i<kor.length; i++) {
            for (int j=0; j<kor[i].length; j++) {
                System.out.printf("kor[%d][%d] = %d\n", i, j, kor[i][j]);
            }
        }
    }
}
```

Console

<terminated> TwoDimension [Java Appli

```
math[0][0] = 0
math[0][1] = 0
math[0][2] = 0
math[1][0] = 0
math[1][1] = 0
math[1][2] = 0

eng[0][0] = 0
eng[0][1] = 0
eng[1][0] = 0
eng[1][1] = 0
eng[1][2] = 0

kor[0][0] = 95
kor[0][1] = 80
kor[1][0] = 92
kor[1][1] = 96
kor[1][2] = 80
```

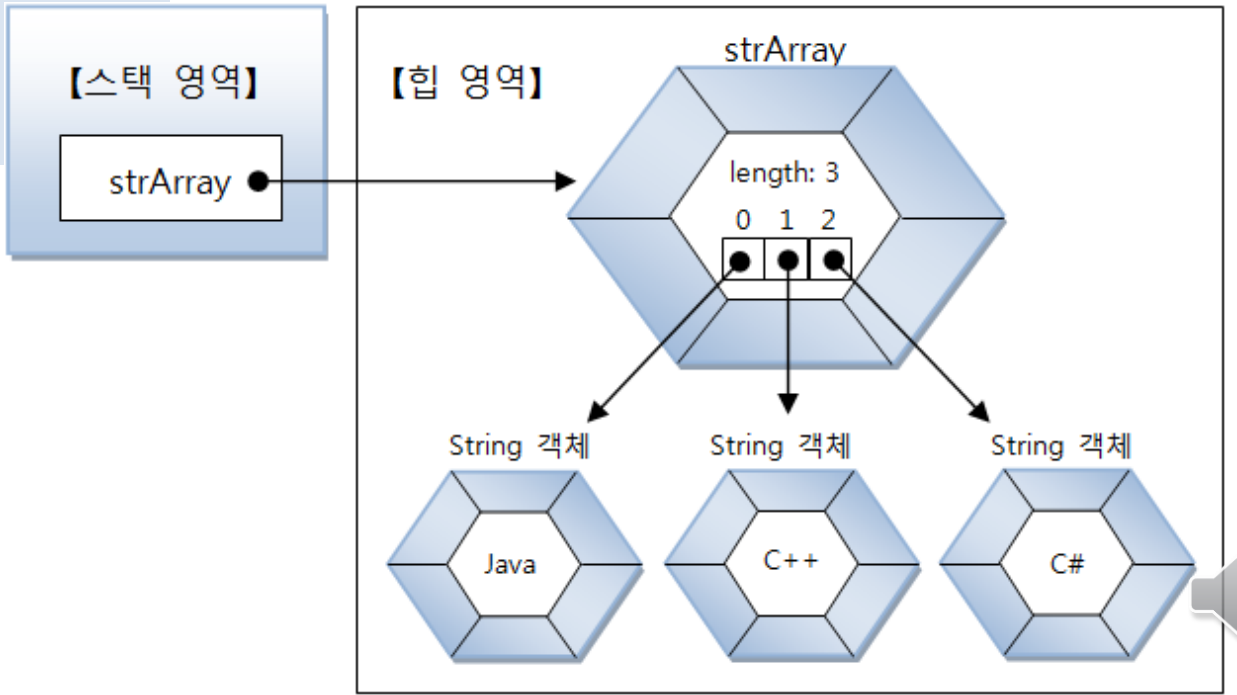
6절. 배열 타입

❖ 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#";
```



객체 참조 배열 실습 (ReferenceArray.java)

ReferenceArray.java

```java
package week6;

public class ReferenceArray {
    public static void main(String[] args) {
        String[] sArray = new String[3];
        sArray[0] = "Java";
        sArray[1] = "Java";
        sArray[2] = new String("Java");

        //자바는 문자열 리터럴이 동일하다면 String 객체를 공유한다
        boolean bVal = (sArray[0]==sArray[1]);
        System.out.println("sArray[0],[1]:"+(bVal?"참조가 같다":"참조가 다르다"));

        //new 연산자를 이용하여 생성된 객체는 힙 영역에 새로운 String 객체를 생성한다
        bVal = (sArray[0]==sArray[2]);
        System.out.println("sArray[0],[2]:"+(bVal?"참조가 같다":"참조가 다르다"));

        //String 객체가 동일한 문자열을 가지고 있는지 비교할 경우에는 equals() 메소드를 사용한다
        if (sArray[0].equals(sArray[2]))
            System.out.println("sArray[0]와sArray[2]의 문자열은 동일하다.");
        else
            System.out.println("sArray[0]와sArray[2]의 문자열은 동일하다.");
    }
}
```

객체 참조 배열 실습 (ReferenceArray.java)

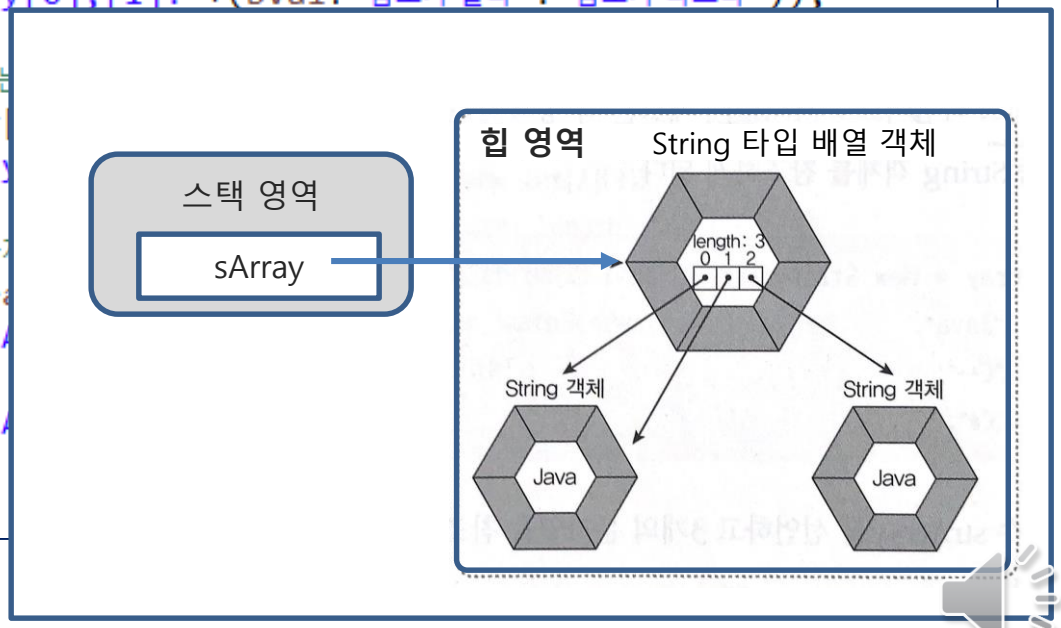
ReferenceArray.java

```
1  package week6;
2
3  public class ReferenceArray {
4      public static void main(String[] args) {
5          String[] sArray = new String[3];
6          sArray[0] = "Java";
7          sArray[1] = "Java";
8          sArray[2] = new String("Java");
9
10         //자바는 문자열 리터럴이 동일하다면 String 객체를 공유한다
11         boolean bVal = (sArray[0]==sArray[1]);
12         System.out.println("sArray[0],[1]:"+(bVal?"참조가 같다":"참조가 다르다"));
13
14         //new 연산자를 이용하여 생성된 객체는
15         bVal = (sArray[0]==sArray[
16         System.out.println("sArray
17
18         //String 객체가 동일한 문자열을 가
19         if (sArray[0].equals(sArra
20             System.out.println("sA
21         else
22             System.out.println("sA
23     }
24 }
```

Console

<terminated> ReferenceArray [Java Application] C:₩Program Fil

```
sArray[0],[1]:참조가 같다
sArray[0],[2]:참조가 다르다
sArray[0]와sArray[2]의 문자열은 동일하다.
```



6절. 배열 타입

❖ 배열 복사

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

❖ 배열 복사 방법

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

배열 복사 실습 (ArrayCopy.java)

ArrayCopy.java

```java
package week6;

import java.util.Arrays;

public class ArrayCopy {
    public static void main(String[] args) {
        int[] arr1 = {1,2,3,4,5};
        int[] arr2 = new int[7];

        //for문을 이용한 복사
        for (int i=0; i<arr1.length; i++)
            arr2[i] = arr1[i];
        for (int i=0; i<arr2.length; i++)
            System.out.print(arr2[i] + " ");

        System.out.println();
        //System.arraycopy() 메소드를 이용한 복사
        System.arraycopy(arr1, 0, arr2, 0, arr1.length);
        for (int i=0; i<arr2.length; i++)
            System.out.print(arr2[i] + " ");

        System.out.println();
        //Arrays 클래스 이용한 복사
        arr2 = Arrays.copyOf(arr1, 3);
        for (int i=0; i<arr2.length; i++)
            System.out.print(arr2[i] + " ");
    }
}
```

배열 복사 실습 (ArrayCopy.java)

ArrayCopy.java

```
1  package week6;
2
3  import java.util.Arrays;
4
5  public class ArrayCopy {
6      public static void main(String[] args) {
7          int[] arr1 = {1,2,3,4,5};
8          int[] arr2 = new int[7];
9
10         //for문을 이용한 복사
11         for (int i=0; i<arr1.length; i++)
12             arr2[i] = arr1[i];
13         for (int i=0; i<arr2.length; i++)
14             System.out.print(arr2[i] + " ");
15
16         System.out.println();
17         //System.arraycopy() 메소드를 이용한 복사
18         System.arraycopy(arr1, 0, arr2, 0, arr1.length);
19         for (int i=0; i<arr2.length; i++)
20             System.out.print(arr2[i] + " ");
21
22         System.out.println();
23         //Arrays 클래스 이용한 복사
24         arr2 = Arrays.copyOf(arr1, 3);
25         for (int i=0; i<arr2.length; i++)
26             System.out.print(arr2[i] + " ");
27     }
28 }
```

Console

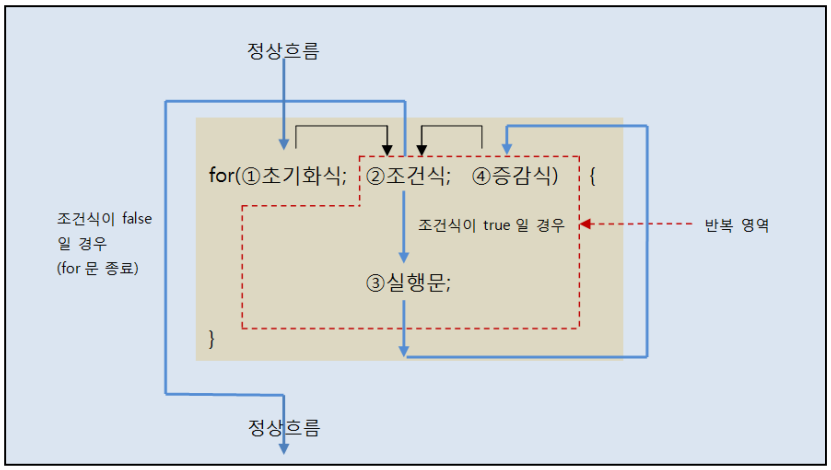
```
<terminated> ArrayCopy [Java Application] C:₩Pr
1 2 3 4 5 0 0
1 2 3 4 5 0 0
1 2 3
```

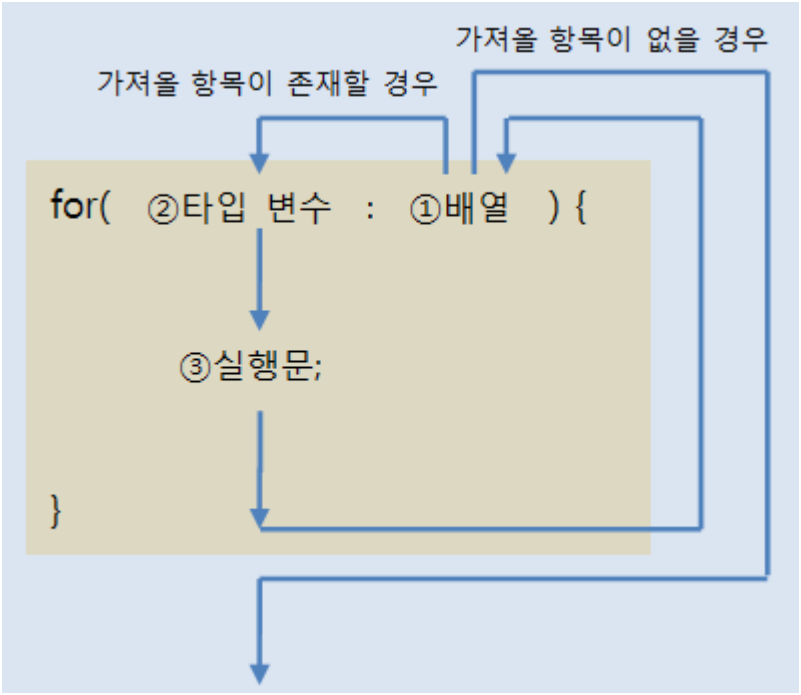
6절. 배열 타입

❖ 향상된 for 문

- 배열 및 컬렉션의 항목 요소를 순차적으로 처리
- 인덱스 이용하지 않고 바로 항목 요소 반복

< 기본 for문 >






```
int[] scores = { 95, 71, 84, 93, 87 };

int sum = 0;
for (int score : scores) {
   sum = sum + score;
}
```

과제물

소스와 결과창이 포함된 이클립스 화면 캡쳐 후 제출
과제 제출 파일명 : 학번+이름

과제 내용

- 배열을 이용하여 각 학생들의 총점과 평균 점수를 구하는 프로그램을 만들어 보자.

학생	A	B	C	D	E
국어 점수	49	80	20	100	80
수학 점수	43	60	85	30	90
영어 점수	49	82	48	50	100

실행 결과

```
A 학생 총점 : ***점, 평균 : **.*점
B 학생 총점 : ***점, 평균 : **.*점
C 학생 총점 : ***점, 평균 : **.*점
D 학생 총점 : ***점, 평균 : **.*점
E  학생 총점 : ***점, 평균 : **.*점
```