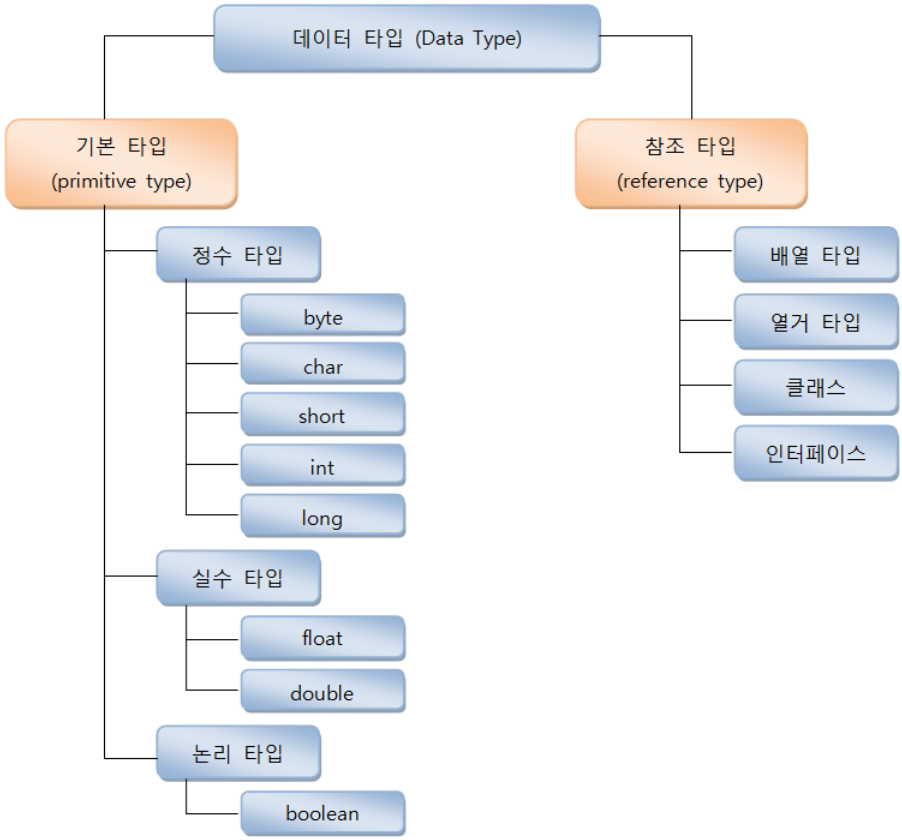
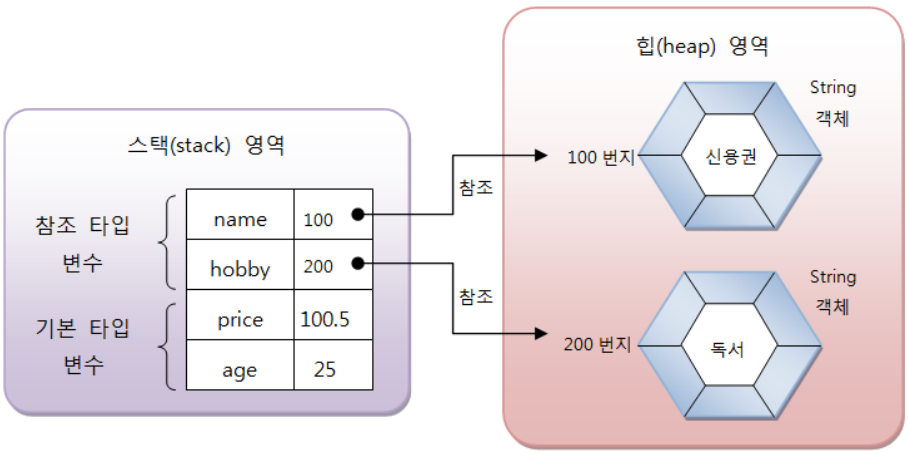
05장 참조 타입

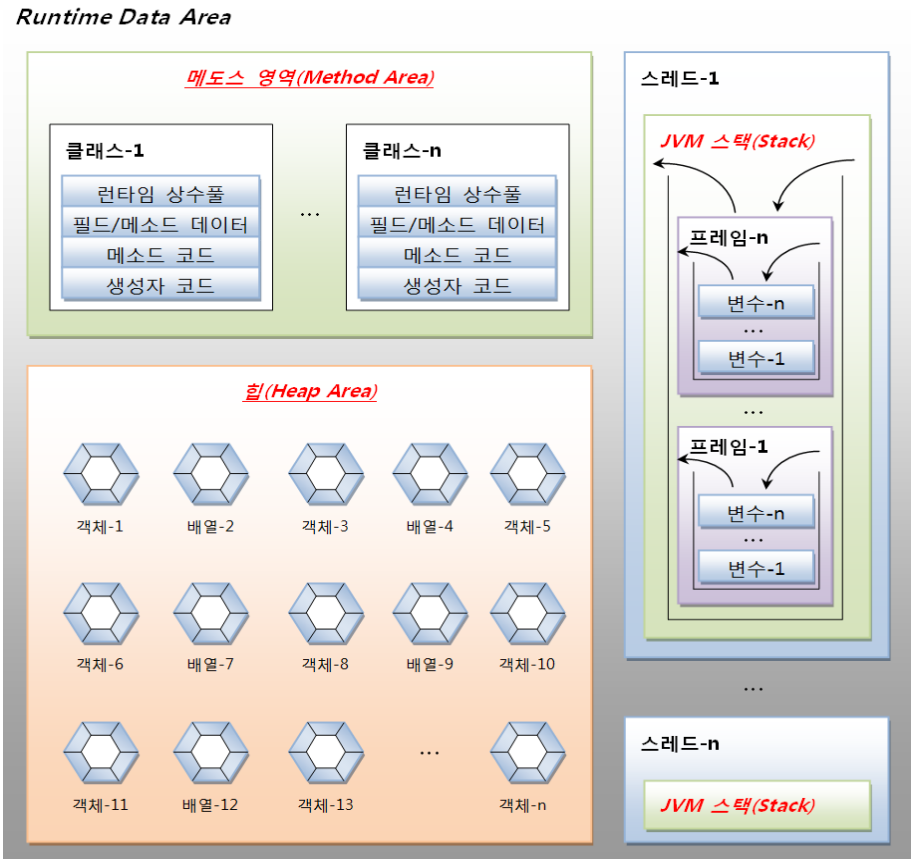
5.1 데이터 타입 분류




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

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



5.2 메모리 사용 영역



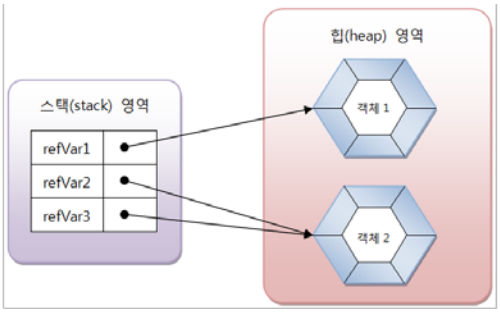
- 메소드(정적, Static) 영역
  - JVM 시작할 때 생성
  - 로딩된 클래스 바이트 코드 내용을 분석 후 저장
  - 모든 스레드가 공유
- 힙 영역
  - JVM 시작할 때 생성
  - 객체/배열 저장
  - 사용되지 않는 객체는 Garbage Collector 가 자동 제거
- 스택 영역
  - 스레드 별 생성
  - 메소드 호출할 때마다 Frame을 스택에 추가(push)
  - 메소드 종료하면 Frame 제거(pop)

```
char v1 = 'A';  // 변수와 값이 스택 영역에 생성된다.

if (v1=='A') {  // if 블록 내부가 실행되고 있을 때만 스택 영역에 존재하고 if블록을 빠져나가면 소멸된다.
   int v2 = 100;
   double v3 = 3.14;
}

boolean v4 = true;

int[] scores = {10, 20, 30};  // scores는 스택 영역에 생성되지만 실제 배열은 힙 영역에 생성된다.
```

5.3 참조 변수의 ==, != 연산

■ 참조 타입 변수의 값은 힙 영역의 객체 주소이므로 결국 주소 값을 비교하는 것이 된다. 동일한 주소 값을 갖고 있다는 것은 동일한 객체를 참조한다는 의미이다.

```
String refVar1 = new String("홍길동");
String refVar2 = "홍길동";
String refVar3 = "홍길동";
```



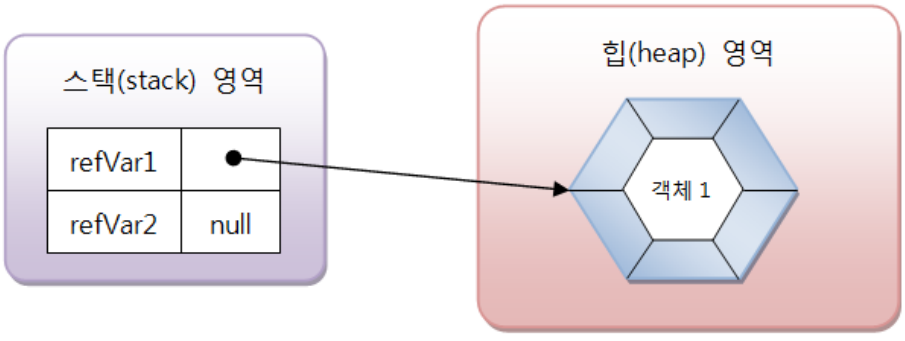
refVar1 == refVar2 결과: false
refVar1 != refVar2 결과: true

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

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

5.4 null과 NullPointerException

■ null(널)
- 변수가 참조하는 객체가 없을 경우 초기값으로 사용 가능
- null로 초기화된 참조 변수는 스택 영역 생성
- 참조 타입 변수가 null 값을 가지는 확인하려면 ==, != 연산 가능
  예) refVar1 == null //false
  refVar2 == null //true



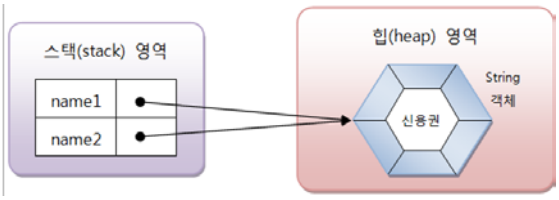
■ NullPointerException의 의미
- 예외(Exception): 사용자의 잘못된 조작 이나 잘못된 코딩으로 인해 발생하는 프로그램 오류
- NullPointerException: 참조 타입 변수가 null 값을 가지고 있을 때, 객체의 필드나 메소드를 사용하려고 했을 때 발생한다.

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

String str = null;
System.out.println(str); //NullPointerException 발생
```

5.5 String 타입

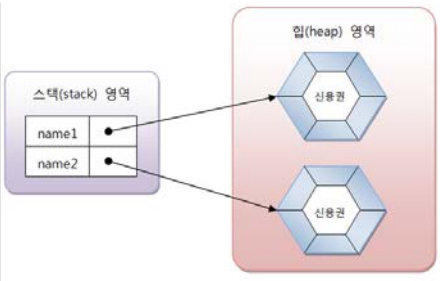
- 문자열 리터럴이 동일하다면 String 객체를 공유하도록 되어 있다.

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



- new 연산자를 사용하여 직접 String 객체를 생성할 경우, 서로 다른 String 객체를 참조한다.

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



- 주소값의 비교는 ==,!= 으로 할 수 있으며 객체의 문자열만을 비교할 때에는 String 객체의 equals() 메소드를 사용해야 한다.

```
String name1 = "신용권";
String name2 = "신용권";
String name3 = new String("신용권");
String name4 = new String("신용권");

if(name1 == name2)  //true
if(name1 == name3)  //false
if(name3 == name4)  //false
boolean result = name3.equals(name4);  //true
```

- String 변수는 참조 타입이므로 null을 대입할 수 있으며 이때 참조를 잃은 String 객체는 JVM의 쓰레기 수집기(Gabage Collector)에 의해 메모리에서 자동 제거된다.

```
name1 = null; //참조를 잃은 String 객체("신용권")는 메모리에서 자동 제거
```

[StringEqualsExample.java]

```
package sec05.exam01_string_equals;

public class StringEqualsExample {
	public static void main(String[] args) {
		String strVar1 = "신민철";
		String strVar2 = "신민철";
```

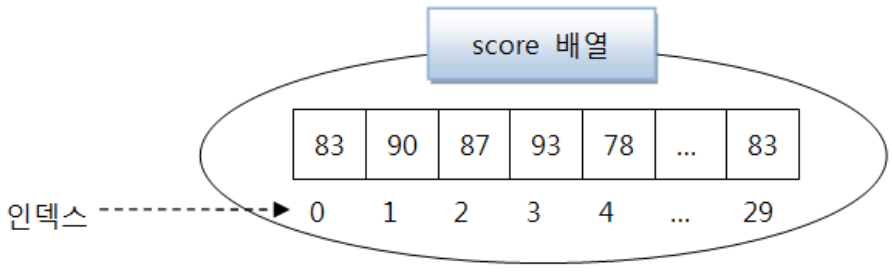
```
                if (strVar1 == strVar2) {
                        System.out.println("strVar1과 strVar2는 참조가 같음");
                } else {
                        System.out.println("strVar1과 strVar2는 참조가 다름");
                }

                if (strVar1.equals(strVar2)) {
                        System.out.println("strVar1과 strVar2는 문자열이  같음");
                }

                String strVar3 = new String("신민철");
                String strVar4 = new String("신민철");

                if (strVar3 == strVar4) {
                        System.out.println("strVar3과 strVar4는 참조가 같음");
                } else {
                        System.out.println("strVar3과 strVar4는 참조가 다름");
                }

                if (strVar3.equals(strVar4)) {
                        System.out.println("strVar3과 strVar4는 문자열이  같음");
                }
        }
}
```

5.6 배열 타입

5.6.1 배열이란?

■ 같은 타입의 데이터를 연속된 공간에 나열시키고, 각 데이터에 인덱스를 부여해 놓은 자료구조이다. 한 번 생성된 배열은 길이를 늘리거나 줄일 수 없다.

```
int score1 = 83;
int score2 = 90;
....
int score30 = 83;

int[] score = {83,90,...,83}; //배열: score[0]=83, score[1]=90,...,score[29]=83
```



■ 배열의 장점: 중복된 변수 선언 줄이기 위해 사용, 반복문 이용해 요소들을 쉽게 처리

```
int[] score = {83,90,83};
int sum = 0;
//for(int i=0; i<3; i++) {
for(int i=0; i<score.length; i++) {
      sum += score[i];
```

```
}
```

5.6.2 배열 선언

■ 기본형: 타입[] 변수;
타입 변수[];
타입 변수[] = null;

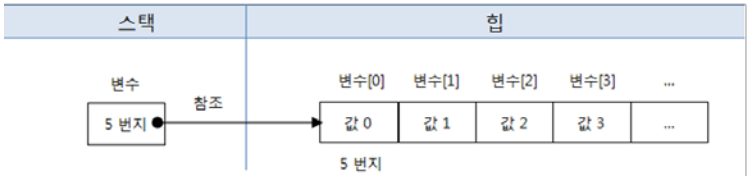
■ 배열 변수는 배열 생성되기 전 null로 초기화 가능, 배열 변수가 null 값을 가진 상태에서 항목에 접근 불가 --> NullPointerException 발생

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

5.6.3 값 목록으로 배열 생성

■ 변수 선언과 동시에 값 목록을 대입할 수 있다.

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



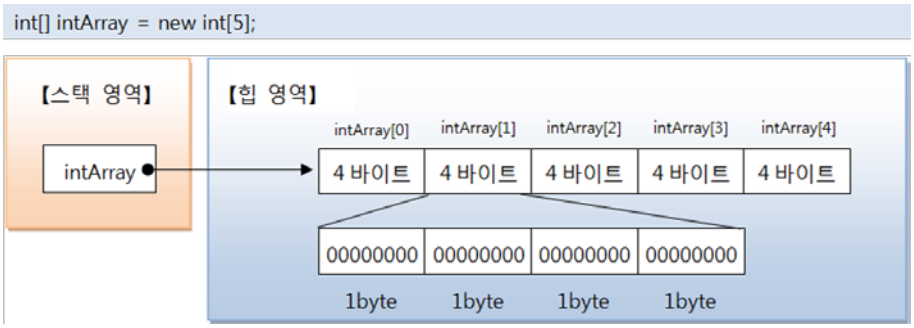

```
String[] name1 = {"신용권","홍길동","김자바"};
String[] name2 = new String[] {"신용권","홍길동","김자바"};
String[] name3 = null;
name3 = new String[] {"신용권","홍길동","김자바"};

int add(int[] scores) {...}  //int[] 배열을 매개변수로 갖는 메소드를 선언한 경우,
int result = add( {95,85,90} );  //컴파일 에러
int result = add( new int[] {95,85,90} );  //new 연산자를 사용해야 한다.
```

5.6.4 new 연산자로 배열 생성

■ 기본형: 타입[] 변수 = new 타입[길이];
타입[] 변수 = null;
변수 = new 타입[길이];

■ 배열 생성시 값 목록을 가지고 있지 않음

■ 향후 값들을 저장할 배열을 미리 생성하고 싶을 경우



■ 타입 별 항목의 기본값

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

5.6.5 배열 길이

■ 기본형: 배열변수.length

■ length 필드는 읽기 전용 필드이기 때문에 값을 바꿀 수가 없다.

■ for문을 사용해서 배열 전체를 루핑할 때 매우 유용하게 사용할 수 있다.

```
int[] intArray = {10,20,30};
int num = intArray.length;
intArray.length = 10; //잘못된 코드, length는 읽기 전용 필드이다.
for(int i=0; i<intArray.length; i++) {...}.
```

[과제] 배열을 이용한 최대값과 최소값

■ 키보드를 이용해서 정수 5개를 입력 받은후 int형 배열에 저장한다. 이때 배열에 저장된 값 중에서 최대값과 최소값을 구하는 프로그램을 작성하라.

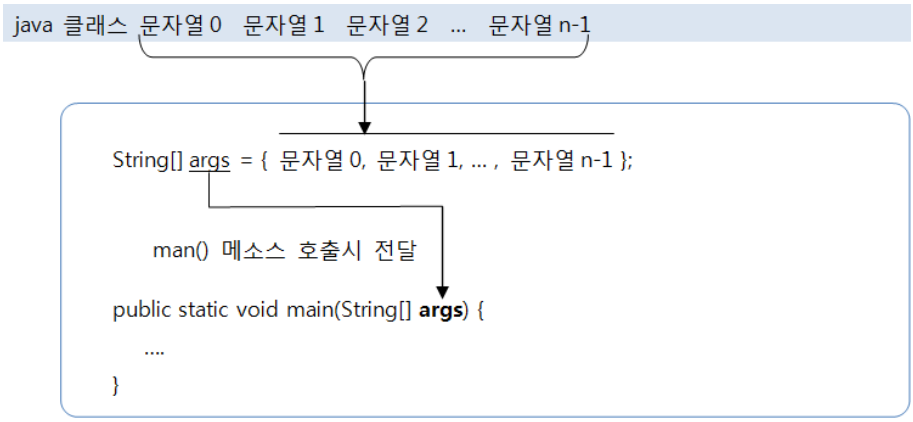
[ArrayTest.java]

```
01   import java.util.Scanner;
02
03   public class ArrayTest {
04
05            public static void main(String[] args) {
06                     // TODO Auto-generated method stub
07
```

```
08                      int max, min;
09                      int[] s = new int[5];
10
11                      System.out.print("정수 5개를 입력 하세요?");
12                      Scanner sc = new Scanner(System.in);
13
14                      ...
15              }
16
17      }
```

5.6.6 커맨드 라인 입력

■ "c:\>java 클래스"로 프로그램을 실행하면 JVM은 길이가 0인 String 배열을 먼저 생성하고 main() 메소드를 호출할 때 매개값으로 전달한다.

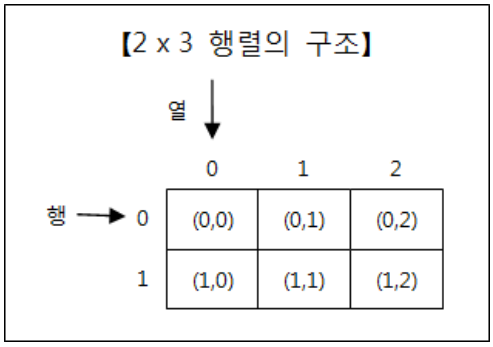


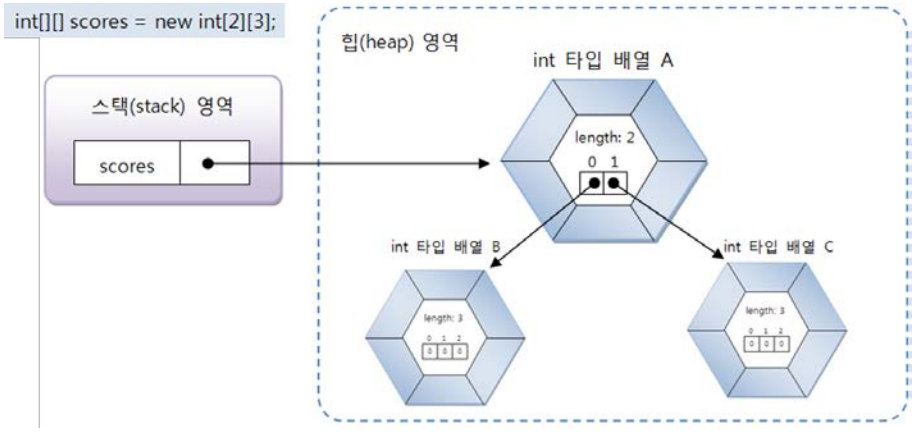
[MainStringArrayArgument.java]

```
package sec06.exam04_main_argument;

public class MainStringArrayArgument {
          public static void main(String[] args) {
                    if (args.length != 2) {
                              System.out.println("프로그램의 사용법");
                              System.out.println("java MainStringArrayArgument num1 num2");
                              System.exit(0);
                    }

                    String strNum1 = args[0];
                    String strNum2 = args[1];

                    int num1 = Integer.parseInt(strNum1);
                    int num2 = Integer.parseInt(strNum2);

                    int result = num1 + num2;
                    System.out.println(num1 + " + " + num2 + " = " + result);
          }
}
```

5.6.7 다차원 배열

■ 자바는 다차원 배열을 중첩 배열 방식으로 구현한다.





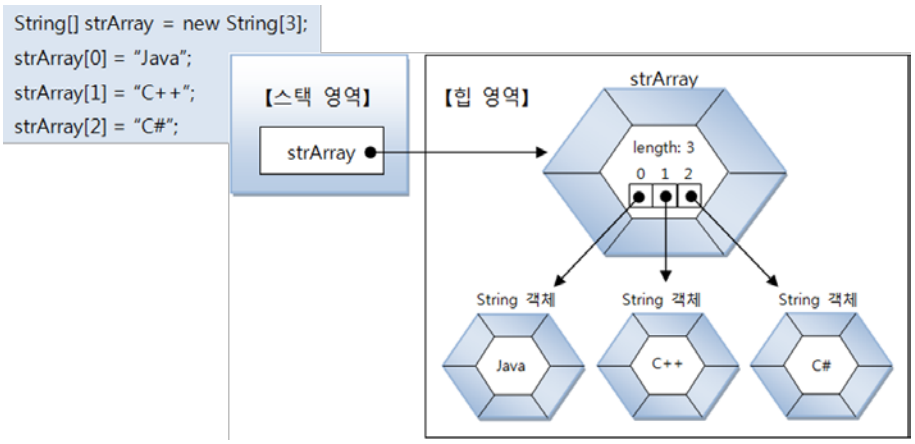
[ArrayInArrayExample.java]

```
package sec06.exam05_array_in_array;

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

		int[][] mathScores = new int[2][3];
		for (int i = 0; i < mathScores.length; i++) {
			for (int k = 0; k < mathScores[i].length; k++) {
				System.out.println("mathScores[" + i + "][" + k + "]=" +
mathScores[i][k]);
			}
		}
		System.out.println();

		int[][] englishScores = new int[2][];
		englishScores[0] = new int[2];
		englishScores[1] = new int[3];
		for (int i = 0; i < englishScores.length; i++) {
			for (int k = 0; k < englishScores[i].length; k++) {
				System.out.println("englishScores[" + i + "][" + k + "]=" +
englishScores[i][k]);
			}
		}
		System.out.println();

		int[][] javaScores = { { 95, 80 }, { 92, 96, 80 } };
		for (int i = 0; i < javaScores.length; i++) {
			for (int k = 0; k < javaScores[i].length; k++) {
				System.out.println("javaScores[" + i + "][" + k + "]=" +
javaScores[i][k]);
```

```
                        }
                }
        }
}
```

5.6.8 객체를 참조하는 배열

■ 기본 타입 배열은 각 항목에 직접 값을 갖고 있지만, 참조 타입 배열은 각 항목에 객체의 번지를 가지고 있다.



[ArrayReferenceObjectExample.java]

```
package sec06.exam06_array_reference_object;

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

                System.out.println(strArray[0] == strArray[1]);  //true (같은 객체를 참조)
                System.out.println(strArray[0] == strArray[2]);  //false (다른 객체를 참조)
                System.out.println(strArray[0].equals(strArray[2]));   //true (문자열이 동일)
        }
}
```

5.6.9 배열 복사

■ 배열은 한 번 생성하면 크기를 변경할 수 없기 때문에 더 많은 저장 공간이 필요하다면 보다 큰 배열을 새로 만들고 이전 배열로부터 항목 값들을 복사해야 한다.

■ 배열 간의 항목 값들을 복사하려면 for문을 사용하거나 System.arraycopy() 메소드를 사용하면 된다.

■ 기본형: System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length);

```
//for문으로 배열 복사
int[] oldIntArray = {1,2,3};
int[] newIntArray = new int[5];
for(int i=0; i<oldIntArray.length; i++) {
      newIntArray[i] = oldIntArray[i];
}
//System.arraycopy()로 배열 복사
System.arraycopy(oldIntArray, 0, newIntArray, 0, oldIntArray.length);
```

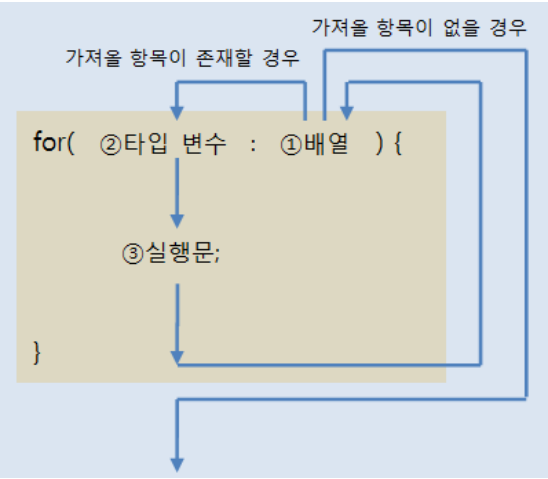
[ArrayCopyExample.java]

```
package sec06.exam07_array_copy;

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] + ", ");
                    }
          }
}
```

5.6.10 향상된 for문

■ 기본형: for(타입 변수 : 배열) {
　　실행문;
　}

■ 인덱스 이용하지 않고 바로 항목 요소 반복




```
int[] scores = {95,71,84,93,87};
int sum = 0;
for( int score : scores ) {
      sum = sum + score;
}
```

[과제] 구구단 및 성적표 구하기

1. 구구단(2~9단)의 연산 결과를 향상된 for문으로 사용하여 2차원 배열에 저장하고, 배열에 저장된 결과를 이용해서 구구단을 출력하는 프로그램을 작성하세요.

[ArrayEx.java]

```
package verify;

public class ArrayEx {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		int[][] s = new int[8][9]; // 8행 9열

		// 구구단 세로로 한줄로 출력
		for (int dan = 0; dan <= 7; dan++) { // 행
			System.out.println("[" + (dan + 2) + "단]");
			for (int i = 0; i <= 8; i++) { // 열
				s[dan][i] = (dan + 2) * (i + 1);
				System.out.println((dan + 2) + "*" + (i + 1) + "=" + s[dan][i]);
			}
			System.out.println("");
		}

		// 확장 for문을 이용
		System.out.println("*** 향상된 for문을 이용 ***");
		int dan = 2;
		int num;
		for (int[] i: s) {
			System.out.println("[" + dan + "단]");
			num = 1;
			// 작성 위치
			for( ) {

			}
			System.out.println("");
			dan++;
		}
	}
}
```

```
// 실행 결과

*** 향상된 for문을 이용 ***
[2단]
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18

[3단]
```

```
3*1=3
3*2=6

...(중략)...

[9단]
9*1=9
9*2=18
9*3=27
9*4=36
9*5=45
9*6=54
9*7=63
9*8=72
9*9=81
```

2. 2차원 배열을 이용해서 5명 학생들의 국어,영어,수학 점수를 저장 했을때, 아래 그림과 같이 과목별 총점과 학생별 총점을 출력하는 프로그램을 작성하세요?

```
각 과목별 총점구하기
국어:430
영어:370
수학:425

학생별 총점구하기
1번학생:215
2번학생:265
3번학생:255
4번학생:245
5번학생:245
```

[Arr04.java]

```java
public class Arr04 {

	public static void main(String[] args) {

		int[][] score = { { 85, 60, 70 }, // 0 행
				  { 90, 95, 80 }, // 1 행
				  { 75, 80, 100 }, // 2 행
				  { 80, 70, 95 }, // 3 행
				  { 100, 65, 80 } // 4 행
		};
		int[] subject = new int[3]; // 과목총점 저장
		int[] student = new int[5]; // 학생의 총점 저장
		String[] subName = { "국어:", "영어:", "수학:" };
		String[] stuName = { "1번학생:", "2번학생:", "3번학생:", "4번학생:", "5번학생:" };
		// subject[0]=0, student[0]=0;
		int r, c;

		System.out.println("각 과목별 총점구하기 ");

		// 이곳에 코드를 추가합니다.

		System.out.println();
		System.out.println("학생별 총점구하기");

		// 이곳에 코드를 추가합니다.

	}// main() end
```

```
}// class end
```

5.7 열거 타입

■ 한정된 값만을 갖는 데이터 타입이 열거 타입(enumeration type)이다.
- 요일: 월,화,수,목,금,토,일 이라는 일곱 개의 값만을 갖는다.
- 계절: 봄,여름,가을,겨울 이라는 네 개의 값만을 가진다.

5.7.1 열거 타입 선언

■ 파일 이름과 동일한 이름으로 선언 (첫 글자 대문자)

■ 한정된 값인 열거 상수 정의

■ 열거 상수 이름은 관례적으로 모두 대문자로 작성

■ 다른 단어가 결합된 이름일 경우 관례적으로 밑줄(_)로 연결

■ 기본형: public enum 열거타입이름 {...}

[Week.java]

```
package sec07.exam01_enum;

public enum Week {
        MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}
```

5.7.2 열거 타입 변수

■ 열거 타입 변수 선언 - 기본형: 열거타입 변수;
■ 열거 상수 값 저장 - 기본형: 열거타입 변수 = 열거타입.열거상수;
■ 열거 타입 변수는 참조 타입 - 열거 타입 변수는 참조 타입이므로 null 값 저장 가능
■ 열거 타입 Week의 경우, MONDAY부터 SUNDAY까지의 열거 상수는 총 7개의 Week 객체로 생성된다. 그리고 메소드 영역에 생성된 열거 상수가 해당 Week 객체를 각각 참조하게 된다.

```
Week today;  //열거 타입 변수 선언, today는 스택영역에 생성
Week today = Week.SUNDAY;  //열거 상수를 저장, Week.SUNDAY는 메소드 영역에 생성
Week today = null; //참조 타입이므로 null값을 저장할 수 있다.

today == Week.SUNDAY  //true, today와 Week.SUNDAY는 힙영역에 같은 Week객체의 주소값을 참조한다.
Week week1 = Week.SATURDAY;
Week week2 = Week.SATURDAY;
System.out.println( week1 == week2 );  //true
```

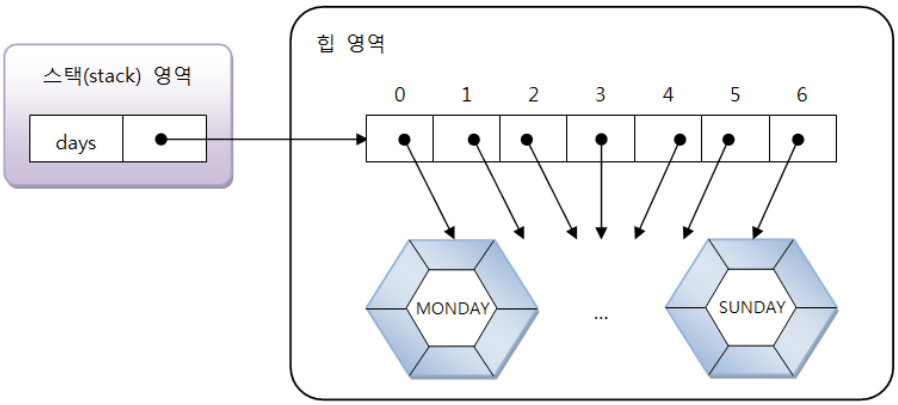
5.7.3 열거 객체의 메소드

■ 메소드는 java.lang.Enum 클래스에 선언된 메소드인데, 열거 객체에서 사용할 수 있는 이유는 모든 열거 타입은 컴파일 시에 Enum 클래스를 상속하게 되어 있기 때문이다.

■ 열거 객체는 java.lang.Enum 클래스의 메소드 사용 가능

리턴타입	메소드(매개변수)	설명
String	name()	열거 객체의 문자열을 리턴
int	ordinal()	열거 객체의 순번(0 부터 시작)를 리턴
int	compareTo()	열거 객체를 비교해서 순번 차이를 리턴
열거타입	valueOf(String name)	주어진 문자열의 열거 객체를 리턴
열거배열	values()	모든 열거 객체들을 배열로 리턴

```
Week today = Week.SUNDAY;
String name = today.name(); //name="SUNDAY"

int ordinal = today.ordinal(); //전체 열거 객체에서 몇 번째 순번인지 알아낸다.

Week day1 = Week.MONDAY;
Week day2 = Week.WEDNESDAY;
int result1 = day1.compareTo(day2); //-2
int result2 = day2.compareTo(day1); //2

Week weekDay = Week.valueOf("SATURDAY"); //weekDay는 Week.SATURDAY 열거 객체를 참조.

Week[] days = Week.values(); // 모든 열거 객체들을 배열로 리턴한다.
for( Week day : days) {
      System.out.println(day);
}
```



[과제] 확인문제

1. 참조 타입에 대한 설명으로 틀린 것은 무엇입니까?

(1) 참조 타입에는 배열, 열거, 클래스, 인터페이스가 있다.
(2) 참조 타입 변수의 메모리 생성 위치는 스택이다.
(3) 참조 타입에서 ==, != 연산자는 객체 번지를 비교한다.
(4) 참조 타입은 null 값으로 초기화할 수 없다.

2. 자바에서 메모리 사용에 대한 설명으로 틀린 것은 무엇입니까?

(1) 로컬 변수는 스택 영역에 생성되며 실행 블록이 끝나면 소멸된다.
(2) 메소드 코드나, 상수, 열거 상수는 정적(메소드) 영역에 생성된다.
(3) 참조되지 않는 객체는 프로그램에서 직접 소멸 코드를 작성하는 것이 좋다.
(4) 배열 및 객체는 힙 영역에 생성된다.

3. String 타입에 대한 설명으로 틀린 것은 무엇입니까?

(1) String은 클래스이므로 참조 타입이다.
(2) String 타입의 문자열 비교는 ==를 사용해야 한다.
(3) 동일한 문자열 리터럴을 저장하는 변수는 동일한 String 객체를 참조한다.
(4) new String("문자열")은 문자열이 동일하더라도 다른 String 객체를 생성한다.

4. 배열을 생성하는 방법으로 틀린 것은 무엇입니까?

(1) int[] array = {1, 2, 3};
(2) int[] array; array = {1, 2, 3};
(3) int[] array = new int[3];
(4) int[][] array = new int[3][2];

5. 배열의 기본 초기값에 대한 설명으로 틀린 것은 무엇입니까?

(1) 정수 타입 배열 항목의 기본 초기값은 0이다.
(2) 실수 타입 배열 항목의 기본 초기값은 0.0f 또는 0.0이다.
(3) boolean 타입 배열 항목의 기본 초기값은 true이다.
(4) 참조 타입 배열 항목의 기본 초기값은 null이다.

6. 배열의 길이에 대한 문제입니다. array.length의 값과 array[2].length의 값은 얼마입니까?

```
int[][] array = {
     {95, 86},
     {83, 92, 96},
     {78, 83, 93, 87, 88}
}
// array.length -> (     #1      ), array]2].length -> (      #2      )
```

7. 주어진 배열의 항목에서 최대값을 구해보세요. (for문을 이용하세요).

[Exercise07.java]

```
package verify;

public class Exercise07 {
          public static void main(String[] args) {
                    int max = 0;
```

```
                    int[] array = { 1, 5, 3, 8, 2 };

                    // 작성 위치

                    System.out.println("max: " + max);  //8
          }
}

// 실행 결과
// max: 8
```

8. 주어진 배열의 전체 항목의 합과 평균값을 구해보세요. (중첩 for문을 이용하세요).

[Exercise08.java]

```
package verify;

public class Exercise08 {
          public static void main(String[] args) {
                    int[][] array = { { 95, 86 }, { 83, 92, 96 }, { 78, 83, 93, 87, 88 } };

                    int sum = 0;
                    double avg = 0.0;

                    // 작성 위치

                    System.out.println("sum: " + sum);  //881
                    System.out.println("avg: " + avg);  //88.1
          }
}

// 실행 결과
// sum: 881
// avg: 88.1
```

9. 다음은 키보드로부터 학생 수와 각 학생들의 점수를 입력받아서, 최고 점수 및 평균 정수를 구하는 프로그램입니다. 실행 결과를 보고, 알맞게 작성해보세요. (참고로 Scanner의 nextInt() 메소드는 콘솔에 입력된 숫자를 읽고 리턴합니다).

[Exercise09.java]

```
package verify;

import java.util.Scanner;

public class Exercise09 {
          public static void main(String[] args) {
                    boolean run = true;

                    int studentNum = 0;
                    int[] scores = null;

                    Scanner scanner = new Scanner(System.in);

                    while (run) {
                              System.out.println("-----------------------------------------------------------
--");
                              System.out.println("1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료");
```

```
                                System.out.println("-----------------------------------------------------------
--");
                                System.out.print("선택> ");

                                int selectNo = scanner.nextInt();

                                // 작성 위치

                        }

                        System.out.println("프로그램 종료");
                }
}

// 실행 결과
// ------------------------------------------------------------
// 1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
// ------------------------------------------------------------
// 선택> 1
// 학생수> 3
// ------------------------------------------------------------
// 1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
// ------------------------------------------------------------
// 선택> 2
// scores[0]> 85
// scores[1]> 95
// scores[2]> 93
// ------------------------------------------------------------
// 1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
// ------------------------------------------------------------
// 선택> 3
// scores[0]: 85
// scores[1]: 95
// scores[2]: 93
// ------------------------------------------------------------
// 1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
// ------------------------------------------------------------
// 선택> 4
// 최고 점수: 95
// 평균 점수: 91.0
// ------------------------------------------------------------
// 1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
// ------------------------------------------------------------
// 선택> 5
// 프로그램 종료
```