Chapter 5. 참조 타입

5.1 데이터 타입 분류

- **데이터 타입** : 기본타입(원시 타입 : primitive type)과 참조 타입 (reference type)
  - **기본 타입** : byte, char, short, int, long, float, double, boolean을 이용해서 선언된 변수이며 **실제 값을 변수 안에 저장**
- **참조 타입** : 객체(Object)의 번지를 참조하는 타입으로 배열, 열거, 클래스, 인터페이스 타입을 말한다. 위와 같은 타입을 이용해서 선언된 변수는 **메모리의 번지**를 값으로 갖는다.

- **기본 타입과 참조 타입 예시**

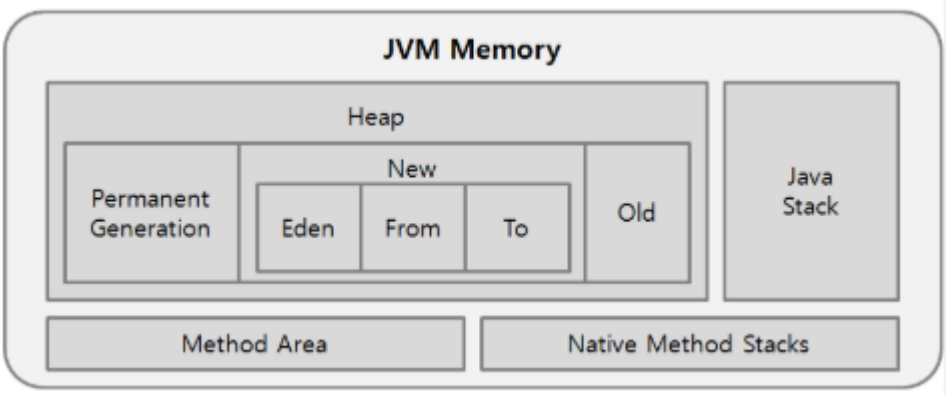
```
[기본 타입 변수]
int age = 25;
double price = 100.5;

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

> age와 price는 직접 값을 스택(stack) 영역에 **직접 저장**하고 name과 hobby는 힙(heap) 영역의 주소를 통해 **객체를 참조**한다.

5.2 메모리 사용 영역

- **JVM** : 운영체제에서 할당받은 메모리 영역을 다음과 같이 세부 영역으로 구분해서 사용한다.



: **메소드(Method)영역 , 힙(Heap) 영역 , 스택(Stack) 영역**

5.2.1) 메소드(Method) 영역

: **클래스(~.class) 들**을 클래스 로더로 읽어 클래스별로 런타임 상수풀, 필드 데이터, 메소드 데이터, 메소드 코드, 생성자 코드 등을 분류해서 저장한다. 메소드 영역은 JVM이 시작할 때 생성되고 **모든 스레드가 공유하는 영역**이다.

힙(Heap) 영역

: **객체와 배열이 생성되는 영역**이다. JVM 스택 영역의 변수나 다른 객체의 필드에서 참조한다. **참조하는 변수나 필드가 없다면** JVM은 쓰레기 수집기 (Garbage Collectoe)를 실행시켜 **자동으로 제거한다.**

JVM 스택(Stack) 영역

: 각 스레드마다 하나씩 존재하며 **스레드가 시작될 때 할당**된다. 추가적으로 스레드를 생성하지 않았다면 main 스레드만 존재하므로 JVM 스택도 하나이다. 메소드를 호출하면 **프레임을 추가(push)**하고 메소드가 종료되면 해당 **프레임을 제거(pop)**한다. 변수는 선언된 블록 안에서만 스택에 존재하고 **블록을 벗어나면 스택에서 제거**된다.

- **JVM 스택 영역 예시**

```
char v1 = 'A';          // 1번

if (v1=='A') {          // 2번
    int v2 = 100;
    double v3 = 3.14;
}

boolean v4 = true;      // 3번
```

> 1번 실행
>
> **스택 상태**: v1
>
> 2번 실행
>
> **스택 상태** : v1, v2, v3
>
> 3번 실행
>
> **스택 상태** : v1, v4

- **배열**

```
int[] scores = {10, 20, 30};
```

: 자바에서는 배열을 객체로 취급한다.

5.3 참조 변수의 ==, != 연산

: 기본 타입 변수의 비교 연산은 **변수의 값**이 같은지, 아닌지를 조사하지만 참조 타입 변수들 간의 비교 연산은 **동일한 객체**를 참조하는지, 아닌지를 비교한다.

- **예시**

```
// refVar1 --> 객체1
// refVar2 --> 객체2
// refVar3 --> 객체2
refVar1 == refVar2  // false
refVar1 != refVar2  // true, 서로 다른 객체를 참조

refVar2 == refVar3  // true
refvar2 != refVar3  // false, 서로 같은 객체를 참조
```

5.4 null과 NullPointerException

null : 참조 타입 변수는 힙 영역의 객체를 참조하지 않는다. 초기값으로 사용 가능.

- **예시**

```
// refVar1 --> 객체1
// refVar2 = null
refVar1 == null     // false
refVar2 != null     // true

refVar2 == null     // true
refVar2 != null     // false
```

- **예외(Exception)** : 프로그램 실행 도중에 발생하는 오류
  - **NullPointerException** : 참조 타입 변수를 잘못 사용하였을 때 발생

    예시

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

    > intArray 변수가 참조하는 배열 객체가 없기 때문에 예외 발생

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

> str 변수가 참조하는 String 객체가 없기 때문에 str.length( ) 에서 예외가 발생한다.
```

5.5 String 타입

- **선언 예시**

```
String name;            // String 변수;
name = "신용권";         // 변수 = "문자열";
String hobby = "자바";   // String 변수 = "문자열";
```

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

> name1과 name2 가 "신용권" 이라는 동일한 객체를 참조하게 된다.

- **new 연산자** : 힙 영역에 새로운 객체를 만들 때 사용하는 연산자

```
String name1 = new String("신용권");
String name2 = new String("신용권");
System.out.println(name1 == name2);     // false
```

> name1 과 name2는 서로 다른 String 객체를 참조하게 되어 비교 연산하였을 때 false가 산출되게 된다.

- **equals( ) 메소드** : 문자열 비교 메소드

```
boolean result = str1.equals(str2);
                 원본         비교
```

- **예제 코드**

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

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

        if(strVar1.equals(strVar2)) {
```

```
            System.out.println("문자열이 같음");
        }

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

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

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

실행 결과

```
참조가 같음
문자열이 같음
참조가 다름
문자열이 같음
```

- **참조 제거**

```
String hobby = "여행";
hobby = null;
```

> hobby 변수가 String 객체를 참조하였으나, null을 대입함으로써 더 이상 객체를 참조하지 않도록 할 수도 있다.

5.6 배열타입

5.6.1) 배열이란?

: 같은 타입의 데이터를 연속된 공간에 나열시키고, 각 데이터에 인덱스(index)를 부여해 놓은 자료구조이다.

- **데이터 타입** : 이미 정해져 있는 데이터 타입이 아닌 만약 다른 타입의 값을 저장하려고 하면 타입 불일치(Type mismatch) 컴파일 오류가 발생한다.
- **배열 길이** : 길이를 선언과 동시에 수정할 수 없다.

5.6.2) 배열 선언

: **타입[ ] 변수;** , **타입 변수[ ]; ex)** int[ ] intArray;, int intArray[ ]

- **null**

  : 배열 변수는 참조 변수에 속하므로 참조할 배열 객체가 없다면 배열 변수는 null 값으로 초기화될 수 있다.
  ex) 타입[ ] 변수 = null

  : 배열이 null인 상태로 변수[인덱스]로 값을 읽거나 저장하게 되면 NullPointerException이 발생한다.

5.6.3) 값 목록으로 배열 생성

- **예시**

```
String[] names = { "신용권" , "홍길동" , "감자바" };
```

- **값 목록으로 배열 생성 예제**

```
public class ArrayCreateByValueListExample {
    public static void main(String[] args) {
        int[] scores = {83, 90, 87};

        System.out.println("scores[0] : " + scores[0]);
        System.out.println("scores[1] : " + scores[1]);
        System.out.println("scores[2] : " + scores[2]);

        int sum = 0;
        for(int i=0; i<3; i++){
            sum += scores[i];
        }

        System.out.println("총합 : " + sum);
        double avg = (double)sum/3;
        System.out.println("평균 : " + avg);
    }
}
```

실행 결과

```
scores[0] : 83
scores[1] : 90
scores[2] : 87
총합 : 260
평균 : 86.66666666666667
```

- **주의 사항**

  : 배열 변수를 이미 선언한 후에 다른 실행문에서 중괄호를 사용한 배열 생성은 허용되지 않는다.

```
타입[] 변수;
변수 = { 값0, 값1, 값2, ... };   // 컴파일 에러
```

값 목록들이 나중에 결정되는 상황이라면 다음과 같이 **new 연사자**를 사용해야 한다.

```
String[] names = null;
names = new String[] { "신용권", "홍길동", "감자바"};
```

- **값의 리스트로 배열 생성 예제**

```
public class ArrayCreateByValueListExample2 {
    public static void main(String[] args) {
        int[] scores;
        scores = new int[] { 83, 90, 87};
        int sum1 = 0;
        for( int i=0; i<3; i++){
            sum1 += scores[i];
        }
        System.out.println("총합 : " + sum1);

        int sum2 = add(new int[]{83, 90, 87});
        System.out.println("총합 : " + sum2);
        System.out.println();
    }

    public static int add(int[] scores){
        int sum = 0;
        for(int i=0; i<3; i++){
            sum += scores[i];
        }
        return sum;
    }
}
```

실행 결과

```
총합 : 260
총합 : 260
```

5.6.4) new 연산자로 배열 생성

: 값의 목록을 가지고 있지 않지만, 향후 값들을 저장할 배열을 미리 만들 때 사용

```
타입[] 변수 = new 타입[길이];
int[] intArray = new int[5];
```

- **new 연산자로 배열 생성 예제 코드**

```
public class ArrayCreateByNewExample {
    public static void main(String[] args) {
        int[] arr1 = new int[3];
        for(int i=0; i<3; i++){
            System.out.println("arr1[" + i + "]" + arr1[i]);
        }
        arr1[0] = 10;
        arr1[1] = 20;
        arr1[2] = 30;
        for(int i=0; i<3; i++){
            System.out.println("arr1[" + i + "]" + arr1[i]);
        }

        double[] arr2 = new double[3];
        for(int i=0; i<3; i++){
            System.out.println("arr2[" + i + "]" + arr2[i]);
        }
        arr2[0] = 0.1;
        arr2[1] = 0.2;
        arr2[2] = 0.3;
        for(int i=0; i<3; i++){
            System.out.println("arr2[" + i + "]" + arr2[i]);
        }

        String[] arr3 = new String[3];
        for(int i=0; i<3; i++){
            System.out.println("arr3[" + i + "]" + arr3[i]);
        }
        arr3[0] = "1월";
        arr3[1] = "2월";
        arr3[2] = "3월";
        for(int i=0; i<3; i++){
            System.out.println("arr3[" + i + "]" + arr3[i]);
        }
    }
}
```

실행 결과

```
arr1[0]0
arr1[1]0
arr1[2]0
arr1[0]10
arr1[1]20
arr1[2]30
```

```
arr2[0]0.0
arr2[1]0.0
arr2[2]0.0
arr2[0]0.1
arr2[1]0.2
arr2[2]0.3
arr3[0]null
arr3[1]null
arr3[2]null
arr3[0]1월
arr3[1]2월
arr3[2]3월
```

5.6.5) 배열 길이

: 배열 변수에 도트(.) 연산자를 붙이고 length를 적어주면 된다.

```
배열변수.length;
```

- **배열의 legth 필드 예제**

```
public class ArrayLengthExample {
    public static void main(String[] args) {
        int[] scores = {83, 90, 87};

        int sum = 0;
        for(int i=0; i<3; i++){
            sum += scores[i];
        }
        System.out.println("총합 : " + sum);

        double avg = (double) sum / scores.length;
        System.out.println("평균 : " + avg);
    }
}
```

실행 결과

```
총합 : 260
평균 : 86.66666666666667
```

> 인덱스를 초과해서 사용하면 **ArrayIndexOutOfBoundsException** 예외가 발생한다.

5.6.6) 커맨드 라인 입력

- **main( ) 메소드의 매개값인 String[ ] args**

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

: 커맨드 라인에서 입력된 데이터의 수(배열의 길이) 와 입력된 데이터(배열의 항목 값)를 알 수 있게 된다.

- **main( ) 메소드의 매개 변수 예제**

```
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];

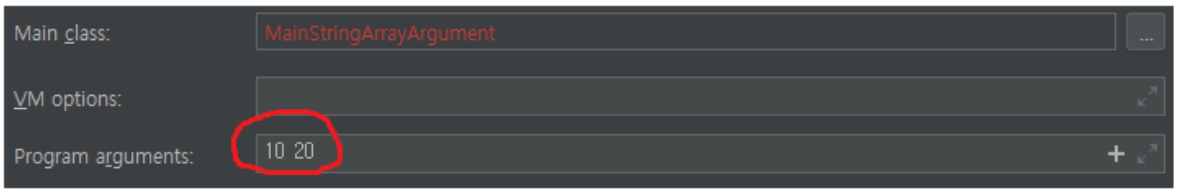
        int num1 = Integer.parseInt(strNum1);
        int num2 = Integer.parseInt(strNum2);

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

실행결과

```
프로그램의 사용법
java MainStringArrayArgument num1 num2
```

실행할 때 매개값을 주고 실행



실행결과

```
10+20 = 30
```

> 이와 같이 실행하면 **args는 { "10" , "20" } 배열을 참조**하게 되고 args[0]은 "10", args[1]은 "20"을 얻을 수 있다. 문자열은 산술 연살을 할 수 없기 때문에 **Interger.parseInt( ) 메소드**를 이용해서 정수로 변환시킨다.

5.6.7) 다차원 배열

- 예시

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

(0, 0)	(0, 1)	(0, 2)
(1, 0)	(1, 1)	(1, 2)

> 이 코드는 세 개의 배열 객체를 생성한다.
>
> **scores[2]** --> 배열 A 참조
>
> **배열 A 의 scores[0]** --> 길이 3인 배열 B 참조
>
> **배열 A 의 scores[1]** --> 길이 3인 배열 C 참조

```
int[][] scores = new int[2][];
scores[0] = new int[2];
scores[1] = new int[3];

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

- 값 목록 나열

```
int[][] scores = { {95, 80}, {92, 96} };
```

- 배열 속의 배열 예제

```
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]);
            }
        }
    }
}
```

실행 결과

```
mathScores[0][0]=0
mathScores[0][1]=0
mathScores[0][2]=0
mathScores[1][0]=0
mathScores[1][1]=0
mathScores[1][2]=0

englishScores[0][0]= 0
englishScores[0][1]= 0
englishScores[1][0]= 0
englishScores[1][1]= 0
englishScores[1][2]= 0

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

5.6.8) 객체를 참조하는 배열

: 참조 타입(클래스, 인터페이스) 배열은 각 항목에 객체의 번지를 가지고 있다.

- **예시**

  : String 은 클래스 타입이므로 객체의 주소를 가지고 있다. 즉 String 객체를 참조

```
String[] strArray = new String[3];
strArray[0] = "Java";
strArray[1] = "C++";
strArray[2] = "C#";
// 각 인덱스마다 다른 객체를 참조하고 있다.
```

객체, 문자열 비교

```
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 객체 참조
System.out.println( strArray[0] == strArray[2] ); // false, 다른 객체 참조
System.out.println( strArray[0].equals(strArray[1]) );    // true, 문자열 동일
```

- **객체를 참조하는 배열 예제**

```
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 객체 참조
        System.out.println( strArray[0] == strArray[2] );   // flase, 다른 객체 참조
        System.out.println( strArray[0].equals(strArray[1]) );  // true, 문자열 동일
    }
}
```

실행 결과

```

```

5.6.9) 배열 복사

: 더 많은 저장 공간이 필요하면 보다 큰 배열을 새로 만들고 이전 배열로부터 항목 값들을 복사해야 한다. 복사할 때는 for문을 사용하거나 System.arraycopy( ) 메소드를 사용.

- **for문으로 배열 복사 예제**

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

실행결과

```
1, 2, 3, 0, 0,
```

- **System.arraycopy( ) 메소드**

  System.arraycopy(**원본 배열** , **원본 배열 복사 시작 인덱스** , **새 배열** , **새 배열 붙여넣을 시작 인덱스** , **복사할 개수**)

예제 코드

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

실행결과

```
java, array, copy, null, null,
```

> **얕은 복사(shallow copy)** : 새 배열의 항목은 이전 배열의 항목이 참조하는 객체와 동일
>
> > **깊은 복사(deep copy)** : 참조하는 객체도 별도로 생성

5.6.10) 향상된 for문

- **예제 코드**

```
public class AdvancedForExample {
```

```
    public static void main(String[] args) {
        int[] scores = {95, 71, 84, 93, 87};

        int sum = 0;
        // scores 배열에서 값을 가져와 score에 값을 저장하여 반복하며
        // scores에 값이 존재하는지를 평가한다.
        for(int score : scores){
            sum = sum + score;
        }
        System.out.println("점수 총합 = " + sum);

        double avg = (double) sum / scores.length;
        System.out.println("점수 평균 = " + avg);
    }
}
```

실행결과

```
점수 총합 = 430
점수 평균 = 86.0
```

5.7) 열거 타입(enumeration type)

: 한정된 값만을 갖는 데이터 타입. 열거 타입은 몇 개의 열거 상수 중에서 하나의 상수를 저장하는 데이터 타입.

5.7.1) 열거 타입 선언

: 열거 타입 이름은 관례적으로 첫 문자를 대문자로 하고 나머지는 소문자로 구성한다.

- **열거 타입 소스 파일들의 이름**

```
Week.java
MemberGrade.java
ProductKind.java
```

- **열거 상수 선언**

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

// 여러 단어로 구성될 경우에는 단어 사이를 밑줄(_)로 연결
public enum LogicResult { LOGIN_SUCCESS, LOGIN_FAILED }
```

- **열거 타입 선언 예제**

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

5.7.2) 열거 타입 변수

- **열거 타입 변수 선언**

```
열거타입 변수;
Week today;
```

- **열거 상수 저장**

```
열거타입 변수 = 열거타입.열거상수;
Week today = Week.SUNDAY;
```

- **열거 타입 변수 값 저장**

```
Week birthday = null;   // 열거 타입 = 참조 타입
```

- **열거 변수와 상수 비교**

```
today == Week.SUNDAY    // true, 같은 객체를 참조하기 때문
```

- **Calendar를 이용해서 날짜와 시간 얻기**

```
Calendar now = Calendar.getInstance();  // Calendar 변수 선언 후 메소드 실행

int year = now.get(Calendar.YEAR);          // 년
int month = now.get(Calendar.MONTH) + 1;    // 월(1~12)
int day = now.get(Calendar.DAY_OF_MONTH);   // 일
int week = now.get(Calendar.DAY_OF_WEEK);   // 요일(1~7)
int hour = now.get(Calendar.HOUR);          // 시간
int minute = now.get(Calendar.MINUTE);      // 분
int second = noew.get(Calendar.SECOND);     // 초
```

- **열거 타입과 열거 상수 예제**

```
import java.util.Calendar;

public class EnumWeekExample {
    public static void main(String[] args) {
        Week today = null;

        // Calendar의 getInstance 메소드 실행
        Calendar cal = Calendar.getInstance();
        int week = cal.get(Calendar.DAY_OF_WEEK);

        switch (week){
            case 1:
                today = Week.SUNDAY; break;
            case 2:
                today = Week.MONDAY; break;
            case 3:
                today = Week.TUESDAY; break;
            case 4:
                today = Week.WEDNESDAY; break;
            case 5:
                today = Week.THURSDAY; break;
            case 6:
                today = Week.FRIDAY; break;
            case 7:
                today = Week.SUNDAY; break;
        }

        System.out.println("오늘 요일 " + today);

        if(today == Week.SUNDAY){
            System.out.println("일요일에는 축구를 합니다.");
        } else {
            System.out.println("열심히 자바 공부합니다.");
        }

        int year = cal.get(Calendar.YEAR);          // 년
        int month = cal.get(Calendar.MONTH) + 1;    // 월(1~12)
        int day = cal.get(Calendar.DAY_OF_MONTH);   // 일
        int hour = cal.get(Calendar.HOUR);          // 시간
        int minute = cal.get(Calendar.MINUTE);      // 분
        int second = cal.get(Calendar.SECOND);      // 초

        System.out.println(year);
        System.out.println(month);
        System.out.println(day);
        System.out.println(hour);
        System.out.println(minute);
        System.out.println(second);
    }
}
```

실행결과

```
오늘 요일 SUNDAY
일요일에는 축구를 합니다.
2018
12
2
4
3
37
```

5.7.3) 열거 객체의 메소드

열거 객체 : 열거 상수의 문자열을 내부 데이터로 가지고 있다. 모든 열거 타입은 컴파일 시에 Enum 클래스를 상속하게 되어 있다.

리턴 타입	메소드(매개 변수)	설명
String	name()	열거 객체의 문자열 리턴
int	ordinal( )	열거 객체의 순번(0부터 시작)을 리턴
int	compareTo( )	열거 객체를 비교해서 순번 차이를 리턴
열거 타입	valueOf(String name)	주어진 문자열의 열거 객체를 리턴
열거 배열	values( )	모든 열거 객체들을 배열로 리턴

name( ) 메소드

: 열거 객체의 문자열 리턴

- **예시**

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

ordinal( ) 메소드

: 전체 열거 객체 중 몇 번째 열거 객체인지 알려준다.

```
public enum Week {
    MONDAY,          // 0
    TUESDAY,         // 1
    WEDNESDAY,       // 2
    THURSDAY,        // 3
    FRIDAY,          // 4
    SATURDAY,        // 5
    SUNDAY           // 6
}

Week today = Week.SUNDAY;
int ordinal = today.ordinal();  // ordinal = 6
```

compareTo( ) 메소드

: 매개값으로 주어진 열거 객체를 기준으로 전후로 몇 번째 위치하는지를 비교한다.

```
Week day1 = Week.MONDAY;        // 0
Week day2 = Week.WEDNESDAY;     // 2
int result1 = day1.compareTo(day2);     // result1 = -2
int result2 = day2.compareTo(day1);     // result2 = 2
```

valueOf( ) 메소드

: 매개값으로 주어지는 문자열과 동일한 문자열을 가지는 열거 객체 리턴

```
Week weekDay = Week.valueOf("SATURDAY")     // weekday --> Week.SATURDAY 참조
```

values( ) 메소드

: 열거 타입의 모든 열거 객체들을 배열로 만들어 리턴

```
Week[] days = Week.values();
for(Week day : days){
    System.out.println(day);
}
```

출력 결과

```
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
```

- **열거 객체의 메소드 예제**

```
public class EnumMethodExample {
    public static void main(String[] args) {
        // name() 메소드
        Week today = Week.SUNDAY;
        String name = today.name();     // name = SUNDAY
        System.out.println(name);
        System.out.println();

        // ordinal() 메소드
        int ordinal = today.ordinal();
        System.out.println(ordinal);    // ordinal = 6
        System.out.println();

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

        // valueOf() 메소드
        if(args.length == 1){   // args에 SUNDAY 저장
            String strDay = args[0];
            Week weekDay = Week.valueOf(strDay);    //
            if(weekDay == Week.SUNDAY || weekDay == Week.SUNDAY){
                System.out.println("주말 이군요");
            } else {
                System.out.println("평일 이군요");
            }
        }
        System.out.println();

        // values() 메소드
        Week[] days = Week.values();
        for(Week day : days){
            System.out.println(day);
        }
        System.out.println();
    }
```

```
  }
```

실행결과

```
SUNDAY

6

-2
2

주말  이군요

MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
```

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. (X , 초기화할 수 있다.)**
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. (X , 참조되지 않는 객체는 자동 소멸된다.)**
    4. 배열 및 객체는 힙 형역에 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. (X , new 연산자를 사용하면 동일한 문자열이더라도 다른 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 }; (X , 배열을 선언하고 값을 지정할 수는 없다.)**

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 0.0 이다.
3. **boolean 타입 배열 항목의 기본 초기값은 true 이다. (X , false 이다.)**
4. 참조 타입 배열 항목의 기본 초기값은 null 이다.

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 = 3 , array[2].length = 5

7. 주어진 배열의 항목에서 최대값을 구해보세요. (for 문을 이용하세요.)

```
public class Exercise07 {
    public static void main(String[] args) {
        int max = 0;
        int[] array = {1, 5, 3, 8, 2};

        for(int i=0; i<array.length; i++){
            if(max < array[i]){
                max = array[i];
            }
        }

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

실행결과

```
8
```

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

```
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;
```

```
        for (int i=0; i<array.length; i++){
            for(int j=0; j<array[i].length; j++){
                sum += array[i][j];
            }
        }
        avg = (double) sum / ( array[0].length + array[1].length + array[2].length
);

        System.out.println("sum: " + sum);
        System.out.println("avg: " + avg);
    }
}
```

실행결과

```
sum: 881
avg: 88.1
```

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

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

            int selectNo = scanner.nextInt();

            if(selectNo == 1){
                System.out.print("학생수> ");
                studentNum = scanner.nextInt();
                scores = new int[studentNum];
            } else if(selectNo == 2){
                for(int i=0; i<studentNum; i++){
                    System.out.print("scores[" + i + "]> ");
                    scores[i] = scanner.nextInt();
                }
```

```
            } else if(selectNo == 3){
                for(int i=0; i<studentNum; i++){
                    System.out.println("scores[" + i + "]> " + scores[i]);
                }
            } else if(selectNo == 4){
                int sum = 0;
                int max = -1;
                double avg = 0.0;
                for(int i=0; i<studentNum; i++){
                    if(max < scores[i]){
                        max = scores[i];
                    }
                    sum += scores[i];
                }
                avg = (double) sum / studentNum ;
                System.out.println("최고 점수 : " + max);
                System.out.println("평균 점수 : " + avg);

            } else if(selectNo == 5){
                run = false;
            }
        }
        System.out.println("프로그램 종료");
    }
}
```

실행결과

```
-----------------------------------------------------
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> 4
최고 점수 : 95
평균 점수 : 91.0
-----------------------------------------------------
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> 5
프로그램 종료
```