



참조변수



- 참조변수(**reference variables**)에 대하여 개념을 안다.
- 자바에서 활용되는 메모리 구조를 알고, 활용할 수 있다.
- 참조 변수에서 활용되는 비교연산자(**==,!=**)의 의미를 파악한다.
- **NullPointer** 개념을 이해하고, 에러가 발생했을 때, 활용할 수 있다.
- 문자열(**String**), 배열타입, 열거(**enum**) 타입에 대한 메모리 할당과 활용에 대하여 안다



생각해봅시다:

정수
실수
문자형

- **java**의 메모리 구조의 기본은 어떻게 시작될까?
- 많은 데이터를 활용하는 객체 단위는 어떤 메모리 구조를 가지는 것이 효율적인가?

스택

도서목록표(카드) -
index

힙

도서관 실재 책이 있는 곳.

- 주소값을 비교한다는 의미와 실제 데이터값을 비교한다는 의미의 차이는 무엇?



데이터 타입의 종류 :

기본 데이터 type

정수형
실수형
논리형

실제값 = 변수호출
`int age = 25;`

`<<age>>`

25

정수형

참조 데이터 type
reference
address value

객체-클래스
배열타입
인터페이스

stack

100번지(heap영역
주소)
Person **p1**
p1:heap 주소값

heap

100번(실제
객체,배열..)
p1 = new
Heap();



참조형 변수의 확인 :

- 메모리에서 `==`, `!=` 연산
 - 기본데이터형에서 데이터와 메모리가 동일하기에 같은 데이터는 같은 메모리로 동일한 값(boolean) 값이 나타난다.
 - `int num01 = 25;`
 - `int num02 = 25;`
 - `System.out.println(num01 == num02)` **true**
 - 참조형에서는 데이터와 메모리가 동일하지 않기 때문에 같은 객체를 호출하더라도 주소값이 다르기에 다른 주소값으로 인식하여 나타난다.
 - `Person p1 = new Person();`
 - `Person p2 = new Person();`
 - `System.out.println(p1 == p2)` **false**

기본예제 :



```
/*
```

##reference data type

참조형 데이터타입(객체, 배열, 인터페이스) 할당.

변수 저장영역에는 실제 데이터 메모리의 주소값을 할당하고,
실제 데이터는 **heap** 영역에 할당되는 것을 말한다.

javaexp.a05_reference.A01_basic@15db9742

객체명@**heap** 영역의 주소값을 **16**진수코드로 할당(**JVM**에서 자동할당)

stack 영역에는 **15db9742** 주소만 할당하고, 실제 **A01_basic()**는
heap 영역에 **15db9742** 주소에 객체가 할당되어 있다.

heap 영역

참조형 데이터가 들어가는 실제 메모리로 **stack**의 주소값에 의해
호출된다.

```
*/
```


```
int age=25;
```

```
System.out.println(age==25);
```

```
A01_basic p = new A01_basic();
```

```
System.out.println(p);
```

기본예제 :



```
/*
메모리의 ==, != 연산자 활용
stack 주소값에 대한 내용으로
기본 데이터형의 값이 동일하면 true로.
참조형데이터의 값은 객체가 동일하더라도,
heap 영역에 참조 객체가 다르므로 false가 나온다.
*/
int num01 = 25;
int num02 = 25;
System.out.println("기본데이터형:"+(num01==num02));
A01_basic a01 = new A01_basic();
A01_basic a02 = new A01_basic();
// 동일한 객체를 생성하더라도, heap 영역에 다른 위치를 참조하기에
// stack 영역의 주소값이 다르므로 false 값이 나온다.
System.out.println("참조데이터형:"+(a01==a02));
```



null, nullPointer●

- 참조형 데이터에서 참조변수만 선언하고, heap영역에 실제 객체를 생성하지 않을 때, null 할당한다.
 - `Person p = null;`
 - 객체가 heap영역에 할당되지 않는 상황에서 메서드를 호출하면 발생하는 예러가 **NullPointerException** 이다.
 - 배열이 없는 곳을 호출, 데이터 없는 상황에서 호출
 - ex) `String name=null;`
 - `System.out.println(name.length());`
 - `System.out.println(args[2]);`
-



기본예제 :

/*

null값의 할당!

실제 참조할 객체를 할당하지 않을 때, 처리 부분.

일반적으로 객체를 초기화할 때, 활용된다.

*/

A01_basic a03=**null**;

String name=**null**;

System.**out.println**("참조객체가 없는 참조변수");

System.**out.println**("a03:"+a03);

System.**out.println**("name:"+name);

//int num07=**null**; 기본데이터형은 참조형이 아니기에 **null**을 할당할 수 없다.

// 참조형에 데이터를 할당되었을 때와, 할당되지 않았을 때, 에러발생부분..


//**heap** 영역에 데이터값 즉, 실제 객체가 할당되지 않았기에..

// 하위에 메소드나 변수를 호출하면 **NullPointerException** 발생한다.

String name02 = "안녕하세요!!!";// **new String**("안녕하세요!!!")

System.**out.println**("객체가 할당된 경우:"+name02.length());

기본예제 :



```
// 문자열.length(): 문자열의 길이..  
// System.out.println("객체가 할당되지 않은  
경우:"+name.length());  
// NullPointerException : 참조되는 객체가 없는데, 해당 하위 메서드를  
호출  
//                      하시면!!?? 안되죠!!!  
// 배열 객체가 생성되지 않았을 때, 해당 내용을 호출하면,  
NullPointerException  
// 발생..  
String names[]=null;  
// System.out.println(names[0]);
```



String 객체 참조에 관해서.●

- String은 객체이다.
 - 첫글자가 대문자인 것은 자바에서 객체로 정의 한다.
 - 객체는 여러 메서드를 가지고 있다.
 - .length(), .substring(idx01,idx02), .concat()
 - "문자열"을 바로 할당 수 있다. new String("문자열")을 할당할 수 있다.
 - 차이점 "문자열"을 바로 할당하면, 같은 문자열은 같은 stack주소로 할당되어 있기에, 비교연산자를 쓰면 같은 문자열은 true값이 출력된다.
 - System.out.println("홍길동"=="홍길동"); true
 - new로 객체생성을 하면서 같은 문자열일지라도 다른 주소값을 호출되기에 false값이 출력된다.
 - String name01 = new String("홍길동");
 - String name02 = new String("홍길동");
 - System.out.println(name01==name02); false
 - System.out.println(name01.equals(name02)); 문자열 자체를 비교하는 메서드 활용해서 처리



*/**

String 값의 주소와 문자열의 비교..

1. String은 객체이다..

*하지만 대입연산자에 의해서 직접적으로 문자열을 할당하면,
문자열이 같은 것은 같은 주소에 할당이 된다... * */*

```
String name05 = "하이맨";
```

```
String name06 = "하이맨";
```

```
System.out.println(name05 == name06);
```



기본예제 :

```
/*
2. 객체를 원칙적으로는 String 참조변수= new String( 문자열 )로
   생성이 되면, 일반적인 프레임웍나, 데이터 로딩에 의해서 처리 되는 것은
   내부적으로 이와 같은 방식으로 처리된다..
   이렇게 될 때는 객체의 문자열이 같더라도 다른 참조변수를 활용하기에
   다른 주소로 할당되어, 주소값 비교는 false값이 된다.
*/
String name07=new String("홍길동");
String name08=new String("홍길동");
System.out.println(name07==name08);// false가 나타남..
/* 문자열 객체에 대한 비교는 원칙적으로 문자열 객체에서 지원하는
* 메서드(.equals)를 활용해서 처리하여야 한다.
* 문자열변수.equals("문자열")
*/
System.out.println(name07.equals(name08));
// 문자열이 같을 때, true값이 출력된다.
```



문자열 비교관련 확인예제 :

args 값으로 id password

args[0] ==> id값으로 할당 new String(args[0]);

args[1] ==> password값으로 할당. new String(args[1]);

id는 himan

password는 7777

입력받으면 인증성공 , 아니면 인증된 계정이 아닙니다..



문자열 비교관련 확인예제풀이 :

```
String id= new String(args[0]);  
String password = new String(args[1]);  
if(id.equals("himan")&&  
    password.equals("7777")){  
    System.out.println("인증성공");  
}else{  
    System.out.println("인증된 계정이 아닙니다.");  
}
```

The screenshot shows an IDE with two tabs: 'A01_basic.java' and 'A02_basicArgs.java'. The 'A02_basicArgs.java' tab is active, displaying the following code:

```
1 package javaexp.a05_reference;  
2  
3 public class A02_basicArgs {  
4  
5     public static void main(String[] args) {  
6         // TODO Auto-generated method stub
```

Below the code editor, there is a console window titled 'Name: A02_basicArgs'. It shows the execution output:

```
Main (Main Arguments) JRE Classpath Source Environment Common  
Program arguments:  
himan 7777
```



객체 배열 :

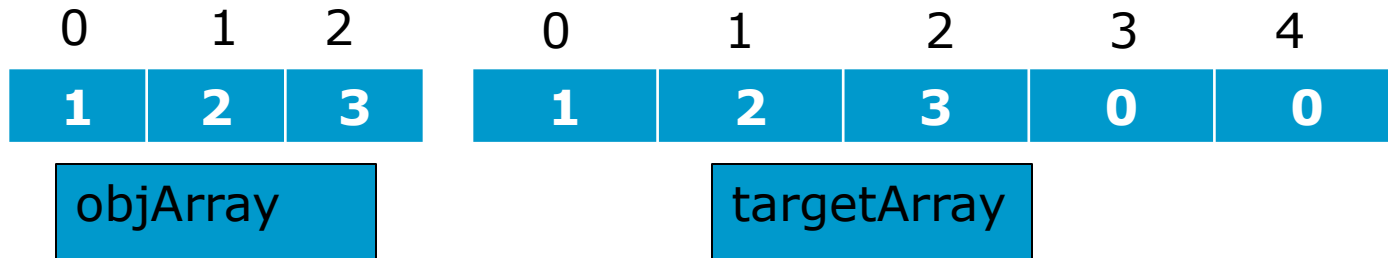
■ 배열 객체

- 선언 : 기본데이터형[] 이름 = new
기본데이터형[크기];
 - `int[] points= new int[5];`
- 할당 이름[index]=할당할 데이터..
 - `point[3] = 80;`
- 선언과 할당 : 기본 데이터형[]이름 =new
기본데이터형[] {데이터1, 데이터2, 데이터3};
 - `double[] weights = new double[]
{67.5,70.4,98.2}; // new double[]
생략가능..`



배열 복사 :

- 배열은 자바의 **api**에 의해 복사처리 메서드가 지원된다.



- System.arraycopy(objArray, int
원본시작위치, targetArray, int
복사본시작위치, int 길이)**



기본예제 :

```
// 배열 복사.  
//System.arraycopy( 원본배열객체, int 원본배열시작index,  
    복사본배열객체,  
// int 복사본배열시작위치, int 복사할길이)  
int[] orginArray={1,3,5};  
int[] targetArray= new int[5];  
System.arraycopy(orginArray, 1, targetArray, 3, 2);  
System.out.println("복사된 배열");  
for(int idx=0;idx<targetArray.length;idx++){  
    System.out.println("데이터:"+targetArray[idx]);  
}
```



확인예제 :

원본

사과

바나나

딸기

복사본

바나나

딸기



확인예제풀이 :

```
String[] orgFruits={"사과","바나나","딸기"};
String[] targetFruits= new String[5];
//          바나나(1) 4번째부터 2개복사
System.arraycopy(orgFruits, 1, targetFruits, 3, 2);
for(String fruit:targetFruits){
    System.out.println("복사된 과일"+fruit);
}
```



열거 타입(enum):

- 데이터 중에서 몇가지 한정된 값을 처리하는 경우 사용되는 형태.
 - 요일의 경우 월~일.
- 선언
 - 파일 지정. 지정할 enum 이름.java 파일 생성
 - Week.java
 - `public enum 열거타입이름{}`
 - 상수 선언
 - `public enum 열거타입이름{상수명1,상수명2,상수명3....}`
 - `public enum Week{MONDAY, TUESDAY, WEDNESDAY, THURSDAY,}`
- 활용 : `enum객체 변수 = enum객체명.상수`
 - `Week week01 = Week.SUNDAY;`
 - `Week week02 = 오늘 날짜 관련 된 내용 호출..`
 - `System.out.println(week01 == week02)`



enum을 활용한 예제 :

- 현재 날짜 관련 객체 Calendar
 - Calendar cal = Calendar.getInstance();
 - Calendar.YEAR(연도),
Calendar.MONTH(월 0~11),
Calendar.DAY_OF_MONTH(일),
Calendar.DAY_OF_WEEK(요일 1~7)
 - int week=cal.get(Calendar.DAY_OF_WEEK)
:요일정보
 - 1~7 Week.SUNDAY~SATURDAY
 - switch case문을 통해서 현재 날짜 정보 출력..
 - 오늘 요일 관련된 정보로 일요일이면, 등산을 가능
것으로 처리..
-



enum의 지원 메서드:

- `name()` : 열거 객체의 문자열
 - `ordinal()` : 열거 객체의 순번(0부터)
 - `values()` : 모든 열거 객체들을 배열로 리턴
 - `compareTo()`: 순번 차이를 가져옴
 - `valueOf(열거형문자)`: 문자에 해당 하는 열거 객체.
-




2차원.. n차원...

사과	오렌지	연필	볼펜	세탁기	TV
----	-----	----	----	-----	----

- 선언
 - `String[][] products=new String[1차원크기][2차원크기];`
 - 차원수만큼 `[][]`(2차), `[][][]`(3차)
 - 1차원크기 : 가장 외부에 있는 데이터크기
 - 2차원크기 : 1차원의 1개 데이터 안에 있는 데이터 크기
- 할당
 - `products[0][0] → "사과"`
- 선언 + 할당
 - `String [][]prods =
{{"사과","오렌지"},"연필","볼펜"},"세탁기","TV"}};`

기본예제 :



```
/* 다차원 배열 선언
 * 데이터type [][] 변수명=new 데이터type[1차원크기][2차원
크기][...][n차원크기] 차원수 만큼[...]..n개..* */
String [][] products=new String[3][2];
products[0][0]="사과";
products[0][1]="오렌지";
products[1][0]="연필";
products[1][1]="볼펜";
products[2][0]="세탁기";
products[2][1]="TV";
//products.length : 가장 외부배열 크기
for(int idx=0;idx<products.length; idx++){
    // products[idx].length: 해당 배열마다 포함된 배열의 크기..
    System.out.println("가장 외부idx:"+idx);
    for(int ix=0;ix<products[idx].length;ix++){
        System.out.println( products[idx][ix] );
    }
}
```



다차원 배열 확인예제 :

1~5반이 있는 학급에 포함된 성적 4명씩 출력하세요.
성적은 Math.random()활용(0~100)
1반 1번 @@@ 점 for문 활용 출력..



확인예제풀이 :

```
int [][]points=new int[5][4];
for(int classIdx=0;classIdx<points.length;classIdx++){
    for(int ptIdx=0;ptIdx<points[classIdx].length;ptIdx++){
        points[classIdx][ptIdx]=(int)(Math.random()*101);
        System.out.println((classIdx+1)+"번 "+(ptIdx+1)+"번
        "+points[classIdx][ptIdx]+"점");
    }
}
```



확인 및 정리 :

- 참조 타입에 대한 설명 중 틀린 것은?
 1. 참조 타입은 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.
- **String** 타입에 대한 설명으로 틀린 것은 무엇인가?
 1. **String**은 클래스이므로 참조 타입이다.
 2. **String**은 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.

- 주어진 배열을 **for**문을 이용해서 최대값을 산출하는 프로그램을 만들어 보세요.
 - `int max=0; //`
 - `int [] dataList = {1,7,3,10,6,9};`
 - `//for 문 처리.. if`
 - `System.out.println("최대값:"+max);`



감사합니다 !
