2주차 과제

study 할래?

스터디 내용

1. 프리미티브 타입 종류와 값의 범위 그리고 기본값
2. 프리미티브 타입과 레퍼런스 타입
3. 리터럴
4. 변수 선언 및 초기화 하는 방법
5. 변수의 스코프와 라이프타임
6. 타입 변환, 캐스팅 그리고 타입 프로모션
7. 1차 및 2차 배열 선언하기
8. 타입 추론, var

1. 프리미티브 타입 종류와 값의 범위 그리고 기본값

- Primitive type (기본형 타입)은 총 8가지가 정의되어 제공된다.
- 기본값이 있기 때문에 Null은 존재하지 않는다.
- 실제 값을 Stack 메모리에 저장한다.
- 컴파일 시점에 담을 수 있는 크기를 벗어나면 컴파일 에러가 발생한다.

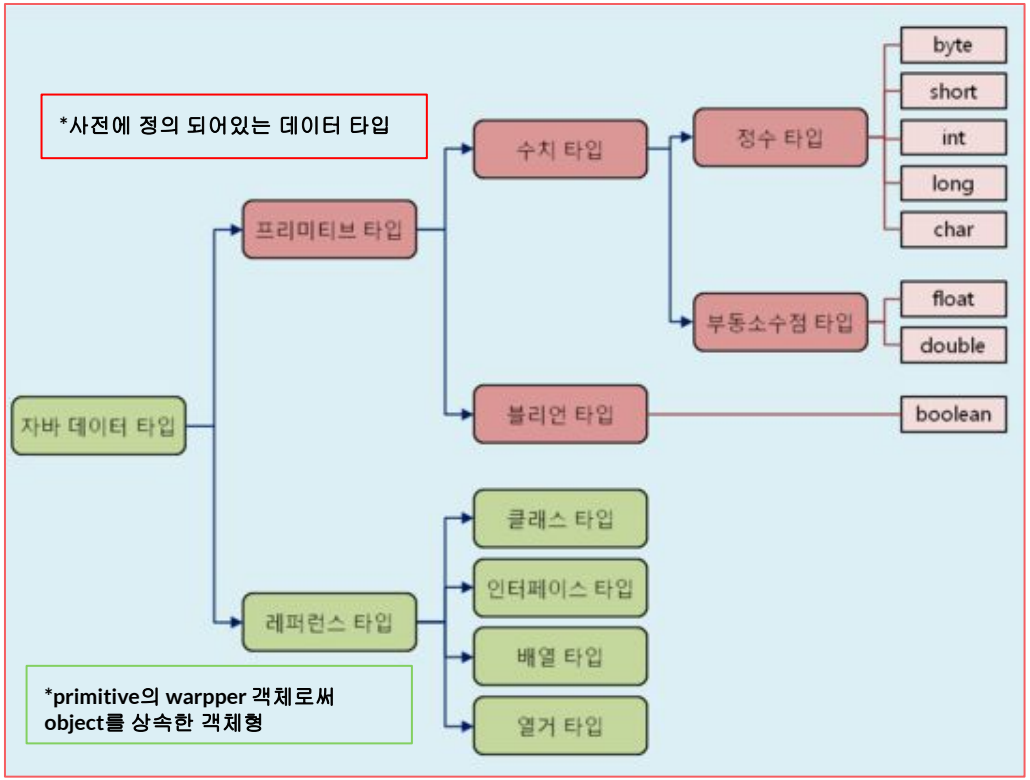
✦ : 기본형

	Type	Size (byte)	Default	Range
논리형	boolean	1	false	true or false
정수형	byte	1	0	128 ~ 127
	short	2	0	-32,768 ~ 32,767
	int ✦	4	0	-2,147,483,648 ~ 2,147,483,647
	long	8	0L	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수형	float	4	0.0F	(3.4 X 10-38) ~ (3.4 X 1038) 의 근사값
	double ✦	8	0.0	(1.7 X 10-308) ~ (1.7 X 10308) 의 근사값
문자형	char	2 (UniCode)	'₩u0000'	0 ~ 65,535

2. 프리미티브 타입과 레퍼런스 타입

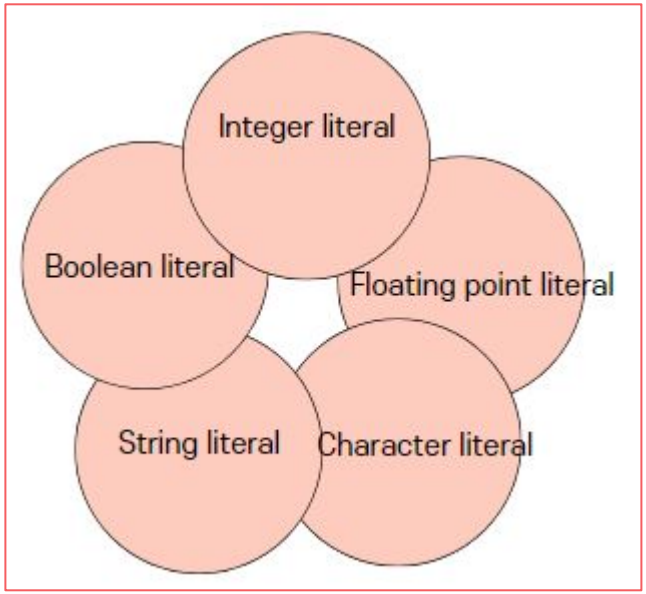
프로그래밍 시 둘의 차이점은 참조형 변수의 Integer나 Double 등은 Object로써 가공이 용이한 반면 단순대입 시에 불편하다. 하지만 기본형의 경우 단순연산에는 좋으나 복잡한 데이터 가공 시에 어려움이 있다.

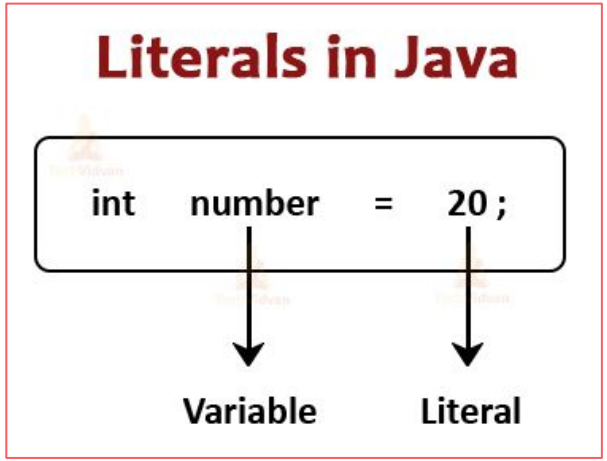
기본형 변수는 실제 값을 저장하지만 참조형 변수는 주소를 값으로 갖는다. 참조형 변수는 선언할 때 변수의 타입으로 클래스 이름을 사용하므로 클래스 이름이 변수의 타입이다. 즉 새로운 클래스를 작성하는 것은 새로운 참조형을 추가하는 것이 된다.



3. 리터럴(literal)

리터럴이란 Java 언어가 처리하는 실제 데이터를 리터럴이라고 한다.
Data type마다 리터럴을 다르게 표현하니 주의해야 한다.





3-1. 리터럴(String)

1. String literal (문자형 리터럴)
   문자형은 “A”, “hello”와 같이
   따옴표를 사용해서 표현합니다.
   영문자 이외의 유니코드 문자를
   표현할때는 ‘\u’를 이용해서
   표현합니다.

   ex) ‘\uC790’ → 자
   “\uBC14” → 바

```
5    public static void main(String[] args) {
6        char c1 ='A';
7        System.out.println("c1: "+c1);
8        //char c2 = 'AA';
9        //System.out.println(c2);
10      // 역슬래시u : 유니코드라는 기호를 상징하는 부호
11       char c3 ='\uC790';
12       System.out.println("c3: "+c3);
13       char c11 ='a';
14       char c12 ='\n';      // \ : escape 코드 : 출력제어
15       char c13 ='b';
16       System.out.print("c11: "+c11+" / c12: "+c12);
17   }
18 }
```

Problems @ Javadoc Declaration Console

<terminated> LiteralTest [Java Application] C:₩Users₩user₩Desktop₩java₩OOPSW₩jdk1

```
c1: A
c3: 자
c11: a / c12:
```

출처: https://codingisgame.tistory.com/3 [Coding or Gaming]

3-2. 리터럴(Integer)

2. 정수형 리터럴은 일반적인 숫자 즉 정수 데이터를 의미합니다. 표현할 수 있는 방법은 10진수, 8진수, 16진수, 2진수가 있습니다.

```
public class LiteralTest {

    public static void main(String[] args){
        // 정수형
         // byte, short, int,long
         // 10진수
         int i1 =10;
         System.out.println(i1);
          // 2진수 / 8진수 / 16진수
         int i2 =0b1010;
         System.out.println("2진수 0b1010 : "+i2);
        int i3 =030;
         int i4 =0xA4;
         System.out.println("8진수 030 :"+i3);
         System.out.println("16진수 0xA4 :"+i4);

         byte s1 =10;
         System.out.println(s1);
         byte s2 = (byte)300;        //   <-------------------byte 는127까지밖에 안들어간다 에러가 나는 코드
                                                //형변환으로 2개의 0~127 제외한 44가 출력됨
         System.out.println("byte에 127보다 큰수를 넣었을때: "+s2);

          long h1 =10;
         System.out.println("long h1:"+h1);//  <---------long 은int 보다 커서 i5 = h1이라는 공식이 성립이 안된다

    //  형변환
         // 형변환
         //묵시적 형변환 :작은 자료형 -> 큰 자료형
         // 명시적 형변환 : 작은 자료형 -> 작은 자료형 : 변환형의 이름 기입
         int i5 = (int)h1;  // <----------------- 이때 (int)로long 을 변환 시켜 준다면 이공식은 성립한다
         long h2 = i5;
         System.out.println("long h2:"+h2);

        // char <-> int
         char c1 ='A';
         System.out.println("char c1:"+c1);
        // ascii 코드 - 영문자, 숫자를 저장할 때 사용하는 내부 코드
         // 유니코드 -영문자 이외의 문자를 저장할때 사용하는 내부코드
         System.out.println("char (int)c1:"+(int)c1);

     // char 상태로 전환하면 그값에 따른 B로 바뀐다
         int c2 ='A' +1;
         System.out.println("(char)int A+1 :"+(char)c2);
```

```
10
2진수 0b1010 : 10
8진수 030 :24
16진수 0xA4 :164
10
byte에 127보다 큰수를 넣었을때: 44
long h1:10
long h2:10
char c1:A
char (int)c1:65
(char)int A+1 :B
```

출처: https://codingisgame.tistory.com/3 [Coding or Gaming]

3-3. 리터럴(Integer)

3. 실수형 리터럴은 소수점을 가진 실수형 데이터를 의미합니다.

3.14 //일반적인 실수 표현방식

6.02E23 // 지수를 이용한 표현 방식입니다 주로 큰실수 데이터를 표현할때 사용합니다(칸때문에)

2.718F // 간단한 float형을 표현 합니다

123.4E+306D //double형을 명시한 큰실수 데이터를 표현합니다

```
5      public static void main(String[] args){
6           // 실수형
7          // float /double
8
9          double d1 =3.14;
10         System.out.println("double 3.14 : "+d1);
11
12       // 모든 실수는 기본적으로 double형 자료형임
13         float f0 = 3.14f;
14         //  <------------ float 형을 실행하는 방법 2가지
15         float f1 = (float)3.14;
16         System.out.println("f0:"+f0);
17         System.out.println("f1:"+f1);
```

Problems Javadoc Declaration Console

```
<terminated> LiteralTest [Java Application] C:₩Users₩user₩Desktop₩java₩OOPS
double 3.14 : 3.14
f0:3.14
f1:3.14
```

출처: https://codingisgame.tistory.com/3 [Coding or Gaming]

3-4. 리터럴(boolean)

4. 논리형 리터럴은 참(true)과 거짓(False)을 표현할때 사용하는 논리데이터입니다.

```
5      public static void main(String[] args){
6            // 논리형 리터럴
7           // 정수나 대문자 (TRUE / FALSE) : 컴파일 에러
8           // 논리형 리터럴 을 쓸때는 boolean 을 쓴다
9              boolean bl = true;
10             System.out.println(bl);
11     }
12 }
13
```

Problems @ Javadoc Declaration Console

<terminated> LiteralTest [Java Application] C:₩Users₩user₩Desktop₩java₩

true

출처: https://codingisgame.tistory.com/3 [Coding or Gaming]

4. 변수 선언 및 초기화 하는 방법

변수는 “데이터타입 + 변수명 +;” 으로 선언이 가능하다.
참고 : https://doublesprogramming.tistory.com/73

```
데이터타입 변수명;
```

```
public class VariablesTest {
    int a;   //단일선언
    String b,c; //복수선언
    float[] d; //배열선언
    Object e; //객체선언

}
```

```
public class VariablesTest {

    public static void main(String[] args) {
        int a;   //단일선언
        a=10; //초기화
        String b,c; //복수선언
        b="hello"; //초기화
        c="world"; //초기화
    }
}
```

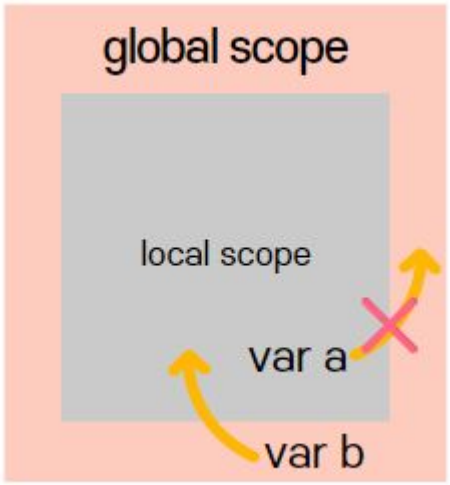
```
public class VariablesTest {

    public static void main(String[] args) {
     int a =10;
     String b="hello"; //선언과 동시에 초기화
    }
}
```

5. 변수의 스코프과 라이프타임

변수의 스코프(영역)이란? 변수에 접근하거나 접근할 수 있는 유효 범위/영역 {}
변수의 라이프타임이란? 변수가 메모리에서 살아있는 기간

변수는 크게 3가지의 스코프를 가지고 있다.

1. Instance var (정적 메서드를 제외한 클래스 전체)
2. Class var (프로그램이 끝날때까지 또는 클래스가 메모리에 로드되는 동안)
3. Local var (선언된 블록 내)



5. 변수의 스코프과 라이프타임

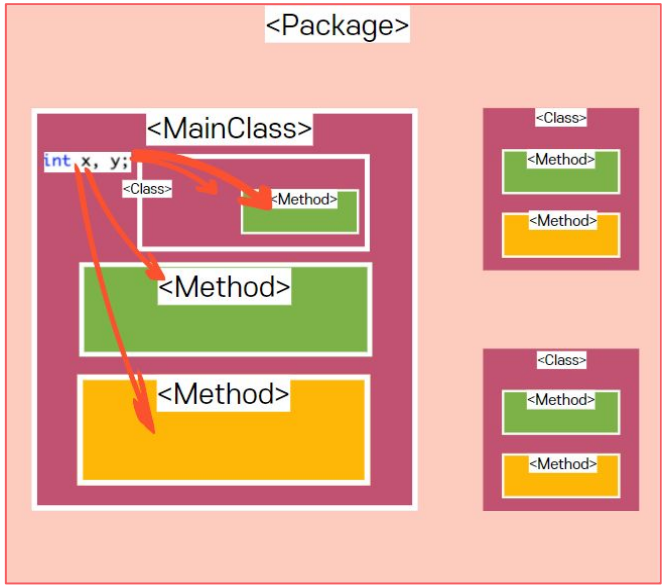
1. Instance 변수
   클래스 내부와 모든 메소드 및 블록 외부에서 선언된 변수
   라이프타임 : 객체가 메모리에 남아있을 때 까지

```
1  package com.oopsw.test;
2
3  public class VariablesTest {
4      int x, y;
5      static int result;
6
7      void add(int a, int b){
8          x=a;
9          y=b;
10         int sum = x+y;
11         System.out.println("Sum = "+sum);
12     }
13
14     public static void main(String[] args){
15         VariablesTest obj = new VariablesTest();
16         obj.add(10, 20);
17     }
18 }
19
```

Problems @ Javadoc Declaration Console

<terminated> VariablesTest [Java Application] C:₩Users₩user₩Desktop₩java₩OO

Sum = 30



5. 변수의 스코프과 라이프타임

2. Class 변수

   클래스 내부, 모든 블록 외부에서 선언되고 static으로 표시된 변수

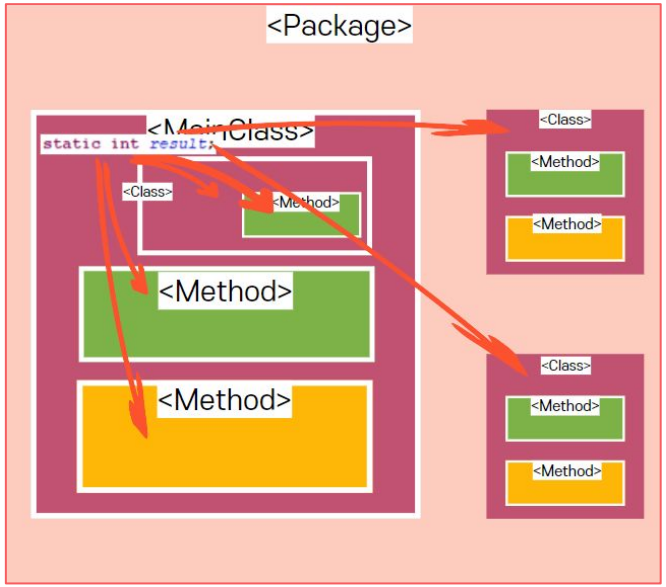
   라이프타임 : 프로그램이 끝날때까지또는 클래스가 메모리에 로드되는 동안

```
1  package com.oopsw.test;
2
3  public class VariablesTest {
4      int x, y;
5      static int result;
6
7      void add(int a, int b) {
8          x=a;
9          y=b;
10         int sum = x+y;
11         System.out.println("Sum = "+sum);
12     }
13
14     public static void main(String[] args) {
15         VariablesTest obj = new VariablesTest();
16         obj.add(10, 20);
17     }
18 }
19
```

Problems @ Javadoc Declaration Console

<terminated> VariablesTest [Java Application] C:₩Users₩user₩Desktop₩java₩OOP

Sum = 30



5. 변수의 스코프과 라이프타임

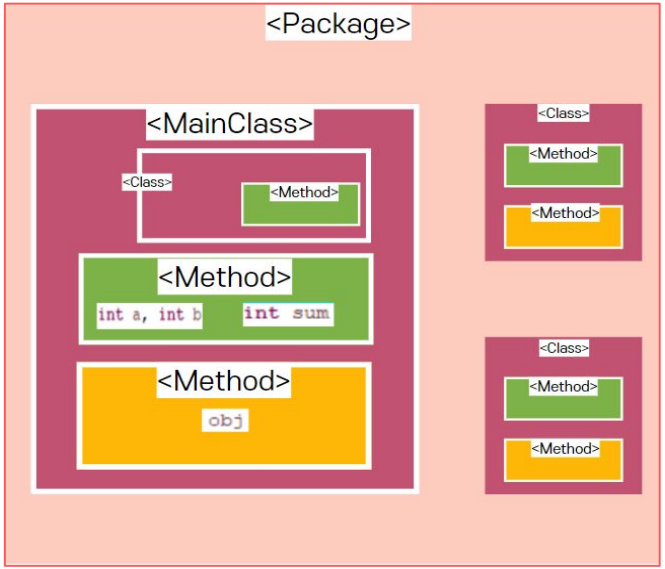
3. Local 변수
   선언된 블록내에 있음
   라이프타임 : 컨트롤이 선언된 블록을 떠날때 까지

```
1  package com.oopsw.test;
2
3  public class VariablesTest {
4      int x, y;
5      static int result;
6
7      void add(int a, int b) {
8          x=a;
9          y=b;
10         int sum = x+y;
11         System.out.println("Sum = "+sum);
12     }
13
14     public static void main(String[] args) {
15         VariablesTest obj = new VariablesTest();
16         obj.add(10, 20);
17     }
18 }
19
```

Problems @ Javadoc Declaration Console

<terminated> VariablesTest [Java Application] C:₩Users₩user₩Desktop₩java₩OOP

Sum = 30



6. 타입변환, 캐스팅 그리고 타입 프로모션

https://programmers.co.kr/learn/courses/5/lessons/135 배열