자바 스터디 2주차

- 프리미티브 타입 종류와 값의 범위 그리고 기본 값
- 프리미티브 타입과 레퍼런스 타입
- 리터럴
- 변수 선언 및 초기화하는 방법
- 변수의 스코프와 라이프타임
- 타입 변환, 캐스팅 그리고 타입 프로모션
- 1차 및 2차 배열 선언하기
- 타입 추론, var

1. 프리미티브 타입 종류와 값의 범위 그리고 기본 값

- C의 자료형과 동일하지 않음. (long long X)
자료형	설명	바이트(크기)	범위
byte	부호있는 정수	1바이트	-128 ~ 127
short		2바이트	-32768 ~ 32767
int		4바이트	약 -21억 ~ 21억
long		8바이트	
float	실수형	4바이트	
double		8바이트	
char	문자형	2바이트	
boolean	논리형	1바이트	true, false

2. 프리미티브 타입과 레퍼런스 타입

- 기초형은 실제 값이 저장(위의 자료형)
- 참조형은 실제 값 저장X, 실제 객체를 가르키는 주소 저장(클래스, 인터페이스, 배열)
  - 레퍼런스 변수를 객체의 이름으로 생각해도 좋음
- ☆기초형은 스택에 선언 / 참조형은 힙에 선언된 클래스의 주소를 참조☆

3. 리터럴

- 리터럴: X = 100; 에서 `100`과 같이 소스코드에 쓰여 있는 값을 의미

1. 정수형 리터럴
   - 여러 진법으로 표시 가능
     - 10진수
     - 8진수: 정수 앞에 `0` 붙이면 8진수 처리 ex. 016
     - 16진수: 정수 앞에 `0x` 붙이면 16진수 처리 ex. 0xe10

        - 2진수: 정수 앞에 `0b` 붙이면 2진수 처리 ex. 0b101
    - JDK 7부터 정수형 리터럴 앞에 밑줄 기호 포함 가능
    - ```
      int num = 123_456;  // 컴파일러가 밑줄 기호 무시
      ```

2. 실수형 리터럴
    - 부동소수점형 리터럴은 double형이 기본
    - ```
      float num = 10.04 // double형이 기본이므로 오류
      float num = 10.04f. // 숫자 끝이 f나 F 붙이면 float형 리터럴이 됨
      ```
    - 부동소수점 오류는 필연적으로 발생함
        - 0.1은 이진법에서 정확하게 표현x

          0.100000001490...과 같이 표현 됨

          흑흑 이해가 안 됨..부동소수점 오류 이해
        - 큰 그액을 정확하게 계산하려면 `java.math.BigDecimal` 클래스 사용 필요

3. 문자형 리터럴
    - 문자형인 `char`는 하나의 문자를 저장할 수 있음
    - 자바에서는 유니코드를 지원하기 위해 문자 하나가 16비트(2바이트)로 표현 됨
    - c처럼 특수 문자들은 문자 앞에 `\`사용

4. 논리형 리터럴
    - 논리형은 true/false만 가질 수 있음
    - ```
      boolean x = 1 > 2; // x에 false가 저장
      ```

5. 기호 상수
    - `상수(constant)`란 프로그램이 실행하는 동안 값이 변하지 않는 수를 의미
    - 리터럴도 상수의 일종
    - `final`: 키워드를 이용해 리터럴을 기호상수로 나타낼 수 있음

      ```
      final double PI = 3.141592; // PI를 기호상수로 정의
      ```

4. 변수 선언 및 초기화하는 방법

1. 프리미티브 자료형 선언

```
int a = 10;
```

2. 레퍼런스 자료형 선언
```

```java
// 클래스 정의
// ExampleReference.java로 독립하는게 원칙
class ExampleReference {
    int a = 10;
    String str;

    String print(str) {
        return system.out.println(str);
    }
}
// 하나의 소스 파일에는 하나의 public 클래스만 있어야 하고 public 클래스의 이름은 소스파일 이름과 동일
public class Example {
    // main 메서드는 어느 클래스에서든 만들 수 있음
    public static void main(String[] args) {
        ExampleReference obj;           // 참조 변수 선언
        obj = new ExampleReference();   // 객체 생성
        obj.a = 11;                     // 객체의 필드 접근
        obj.str = "java study";
        String method = obj.print();    // 객체 메소드 접근
    }
}
```

레퍼런스 자료형 선언 순서

1. 클래스 정의
   - ☆클래스는 객체(인스턴스)가 아니다☆
   - 클래스 안에 필드와 메소드(클래스 멤버) 정의
2. 메인 메소드에서 레퍼런스 변수 선언

   `ExampleReference obj;`
   - 위 문장만으로 인스턴스 생성x

     인스턴스를 가르킬 수 있는 참조 변수만 생성
     - ☆자바의 모든 객체는 `new` 연산자를 이용해야만 생성 가능☆
3. `new` 연산자를 사용해 객체 생성, 객체의 참조값을 레퍼런스 변수에 저장

   `obj = new ExampleReference();`
   - new 연산자는 객체를 `히프메모리`에 생성된 후에 객체의 참조값 반환

     obj에 ExampleReference 클래스 주소가 할당
   - 객체는 아직 초기화 되지 않음
4. 객체의 필드와 메소드 사용
   - 객체의 필드, 메소드 접근시 점 연산자(.) 이용

```java
// 객체의 필드 접근
obj.a = 11;
obj.str = "java study";
// 객체의 메소드 접근
String method = obj.print();
```

5. 변수의 스코프와 라이프타임

변수 스코프: 변수의 활동 영역

변수 라이프타임: 변수가 메모리에서 존재하는 기간

1. 인스턴스 변수: 클래스 내부에서 선언된 변수

- *scope*: static 메소드를 제외한 모든 클래스에서 사용 가능
- *life time*: 클래스 객체가 선언된 이후 부터 메모리에 남아있을 때 까지

2. 클래스 변수: 클래스 안에서 static으로 선언된 변수

- *scope*: 클래스 전체
- *life time*: 클래스 초기화 후 프로그램 종료까지

3. 지역 변수: 블록 내에서 선언된 변수

- *scope*: 선언된 블록 내
- *life time*: 변수 선언 후 블록에서 벗어날 때 까지

참조

- 변수 스코프와 라이프타임 참조1
- 변수 스코프와 라이프타임 참조2
- 변수 스코프와 라이프타임 참조3

6. 타입 변환, 캐스팅 그리고 타입 프로모션

1. 타입 변환

하나의 자료형을 다른 자료형으로 변환하는 것

1. 자동적인 형변환
    - 컴퓨터는 산술적 연산 하기 전 피연산자의 타입 통일해야 함
    - 컴퓨터는 정수 계산 하드웨어와 실수 계산 하드웨어가 다름
    - 수식 계산 시 범위가 넓은 피연산자 타입으로 변환

```
double sum = 1.3 + 12; // 12가 12.0으로 형 변환되어 계산 됨
```

2. 강제적인 형변환(캐스팅)
    - 형변환 연산자 사용

```
int x = 12;
double y = (double)x;
```

- **축소변환**

  더 작은 크기의 자료형에 값을 저장시 값의 누스가 발생할 수 있음

```
int x = (int)12.5;  // x = 12, 소수점 아래 저장x
```

3. 타입 프로모션
   - 산술적 연산시 피연산자의 범위가 초과되면 범위가 더 큰 자료형으로 변환
   - `byte`, `short`, `char`를 계산시 `int`로 프로모션 됨

   타입 프로모션 참고

7. 1차 및 2차 배열 선언하기

- 배열은 같은 자료형의 여러 개의 변수를 하나로 묶어 넣은 것
- 배열의 초기값
  1. 숫자 배열: 0으로 초기화
  2. boolean 배열: false
  3. 문자열 배열: null

- 배열 선언과 생성

  1차원 배열 선언

```
int[] arr = new int[10];
// 배열 선언 int[] arr; -> 배열 생성x 변수만 생성
// arr = new int[10];  -> new 연산자를 통해 생성

int[] arr = {1,2,3,4,5}; // 초기값 입력하면 new 연산자 없이 배열 생성 가능
```

2차원 배열 선언

```
int[][] arr2 = new int[2][3];

int[][] arr2 = {
    {10,20,30},
    {40,50,60}
};
```

8. 타입 추론, var

타입추론: 타입을 명시하지 않았지만 컴파일러가 타입을 추론하는 것

var: 타입 추론을 지원하는 자료형

- 선언된 후 타입 변환 불가
- 지역변수만 사용 가능
- 클래스 가르키는 레퍼런스 변수로 사용 가능