# 📖 Java Study Skill Up Team Review

## 실행문과 세미콜론

[ 실행문: 실행을 하는 코드 ]

- main
  - main() 메서드 블록 내부에는 다양한 실행문이 작성된다.
  - 즉 main 메서드는 프로그램 실행 진입점이다.
- System.out.println();
  - System.out.println("출력값"); 은 () 안의 내용을 출력하는 실행문이다.
  - 즉 '출력값'을 출력하기 위해 출력문을 사용하여 출력한 실행문이라고 할 수 있겠다.
- 아래와 같은 것들도 실행문이다.
  - **변수 선언**
  - 변수에 숫자 값 1을 **저장**
  - 변수의 숫자 값 1을 2로 **증가 후 저장**
  - 콘솔에 변수의 값을 출력하는 println() **메서드 호출**

[ 세미콜론; 콜론: ]

- 실행문과 실행문을 구분하기 위해 세미콜론으로 구분한다.
- 즉 실행문 뒤에는 무조건 세미콜론을 입력해 주어야 한다.
- ex) int x = 1; int y = 1;

## 변수

- 변수란?
  - 프로그램에 의해 수시로 값이 변동될 수 있기 때문에 변수라는 이름을 갖게 되었다.
  - 변수는 복수 개의 값을 저장할 수 없으며 단 하나의 값만 저장할 수 있다.
  - 또한 다양한 타입의 값을 저장할 수 없으며 한 가지 타입의 맞는 값만 저장할 수 있다.
- 변수는 왜 사용할까?
  - 프로그램은 작업을 처리하는 과정에서 필요에 따라 데이터를 메모리에 저장해야 한다. 이때 사용할 수 있는 게 변수다.
  - 즉 변수는 값을 저장할 수 있는 메모리의 공간을 의미한다.
- 변수 선언
  - 어떤 타입의 데이터를 저장할 것인지, 이름을 무엇으로 할지 결정한다.
  - type은 변수에 저장되는 값의 종류와 범위를 결정짓는 요소이기 때문에 고민이 필요하다.
    - int 예시
      - int가 사용하는 4바이트가 대부분의 CPU에서 처리하기 적합한 크기이다.
      - 사람이 보기에는 메모리를 꽤 차지할 거 같지만 컴퓨터 기준에서는 큰 숫자가 아니다.
      - 즉 데이터를 아껴 쓰려고 short와 byte 사용의 고민이 필요할 정도는 아니다.
  - 같은 타입의 변수는 콤마(,)를 통해 한 번에 선언할 수도 있다.

        - int age, year, pageNumber;
    - 결정된 변수 이름은 메모리 주소에 붙여진 이름이다.
        - 프로그램은 이 변수 이름을 통해 메모리 주소에 접근하여 저장된 값을 읽어오거나 저장하거나 수정한다.

- 변수 이름
    - 변수 이름은 명명 규칙을 따라야 하며 자바 예약어를 변수 이름으로 사용할 수 없다.
        - ex) public, int, class, void 등
    - 문자 수의 길이 제한은 없다.

- 변수 사용
    - 처음 값을 저장할 경우 이러한 값을 초기값이라고 한다.
        - 이런 초기값을 주는 행위는 변수의 초기화라고도 한다.
    - 소스 코드 내에서 직접 입력된 값을 리터럴(literal)이라고 한다.
        - 즉 "안녕하시렵니까?" 와 같은 값
    - 리터럴 == 상수 / 같은 의미지만 프로그램에서는 다른 용어로 사용한다.
        - 상수: 값을 한 번 저장하면 변경할 수 없는 변수
        - 리터럴: 리터럴은 변수에 저장되지 않고 그 자체로 사용되는 값

## 이스케이프

[ 문자와 용도 ]

- 문자열 사이에서 사용
    - '\t' : 수평 탭
    - '\n' : 줄 바꿈
    - '\r' : 리턴
    - '\'"' : 큰 따옴표(")
    - '\'' : 작은 따옴표(')
    - '\' : 역슬래쉬(\)

## 정수 자료형

- byte: 1
- char: 2
- int: 4
- long: 8

양수에 1씩 적은 이유는 0이 양수는 아니지만 한정된 자릿수에 0을 포함해야 하기 때문에 양 수에서 자리를 내준 것이다.

## 명시적 형변환

```
byte byteNum;
int smallIntNum = 11111;

byteNum = (byte) smallIntNum;
```

- 명시적(강제) 형변환
- 개발자: "그냥 강제로 변환해요."
- 이처럼 강제로 범주 외의 값을 넣을 경우 값은 손실됨
- 실제 대입 값: 11111 % 128 의 나머지 값이 대입된다. = 103

## 이항 연산자

```
int a = 2 + 2; // a: 4
int b = a - 2; // b: 2
```

- a가 4인 상태에서 b에 a-2를 계산하기 위해 사용되더라도 a의 값은 변하지 않는다.
- 즉 부수 효과를 일으키지 않는다고 할 수 있다.

```
int a = 1;
int b = 2;

byte c;
c = a + b; // 불가능
c = (short) (a + b); // 명시적으로 가능

int int1 = 10 / 3; // 3
```

- 작은 자료형에 더 큰 자료형에 대입할 수 없다.
- 정수 자료형 int 계산은 소수점 아래를 '버림'

## 복합 대입 연산자

- 부수 효과를 일으키는 연산자

```
int a = 1;
a = a + 2;
a += 3;

int b = a += 4;
```

- a와 b 모두 10이 된다.

## boolean연산자

- true/false 둘 중 한 값을 가짐
- 1바이트 (8비트)를 가짐: 1비트를 차지하지 않고 8비트를 차지하는 이유는 컴퓨터가 메모리상에서 잡을 수 있는 가장 작은 단위가 1 바이트 이기 때문이다.

## 논리 연산자

&&: and, ||: or

- boolean 변수는 isXXX로 많이 사용하자.
- && 연산자에서 좌측이 false이면 볼 필요가 없이 반환된다.
- || 연산자에서는 좌측이 true이면 뒤의 것을 평가할 필요가 없다.

```
// 단축평가(short circuit)
boolean bool1 = a < b && c ++ < (d +=3);
```

- 사소하지만 연산 부하가 적은 코드를 앞에 작성함으로써 리소르를 절약할 수 있다.

## 삼항 연산자

```
a ? b : c
```

- a: boolean 값
- b: a가 true일 때 반환될 값
- c: a가 false일 때 반환될 값

## 문자열 literal과 객체 인스턴스

```
String hello1 = "Hello";
String hello2 = "Hello";
String world = "World";

// 리터럴 끼리는 비교 연산자 == '는는'을 사용해서 비교가 가능하다.
hello1 == hello2; // true
String hello3 = new String("Hello");
String hello4 = new String("Hello");
String hello5 = hello4; // hello5를 hello4 그대로 초기화 (같은 주소를 참조)

// 인스턴스와 비교할 때에는 equals 메서드를 사용해야 한다.
hello3 == hello4;       // false
hello3.equals(hello4);  // true
```

- 자바에서 문자열 리터럴은 String Constant Pool 이라는 특별한 영역에 저장이 된다.
- String pool은 Heap 메모리 내에 위치하지만, 일반적인 Heap 영역과는 다른 메모리 영역이다.
- 문자열 리터럴은 프로그램이 실행될 때 메모리에 한 번만 생성되며, 같은 문자열 리터럴을 참조하는 모든 부분은 동일한 객체를 공유한다.
- new String() 키워드를 사용하여 문자열을 생성하면 새로운 string 객체가 Heap 메모리에 생성된다.
- 이 경우 항상 새로운 객체가 생성되며, 문자열 내용이 같더라도 독립적인 객체가 된다.
- 정리하자면, 문자열 리터럴은 String Pool에서 관리되며 동일한 객체를 봉유한 반면 new String을 사용하면 새로운 객체가 생성되며, 동일한 문자열 내용이라도 독립적인 객체가 된다.

```
String str_a1 = "안녕";
boolean bool_a1 = str_a1.equals("안녕");

// 리터럴로 선언했어도 객체 인스턴스로 만들어짐
// 때문에 객체의 기능인 메서드 사용이 가능하다.
boolean bool_a2 = "안녕".equals("안녕");
boolean bool_a3 = "안녕".equals(str_a1);
```

- 문자열은 리터럴에 바로 .을 찍고 메서드를 사용하는 것이 가능하다.
- 코드에서는 리터럴이지만 객체의 인스턴스 이기 때문이다.
- 더하기를 제외하고 문자열에서는 복합 대입 연산이 불가하다.