Java chapter 1

자바 시작하기
변수(Variable)
자료형(Data Type)
형 변환

1-1. 자바 시작하기

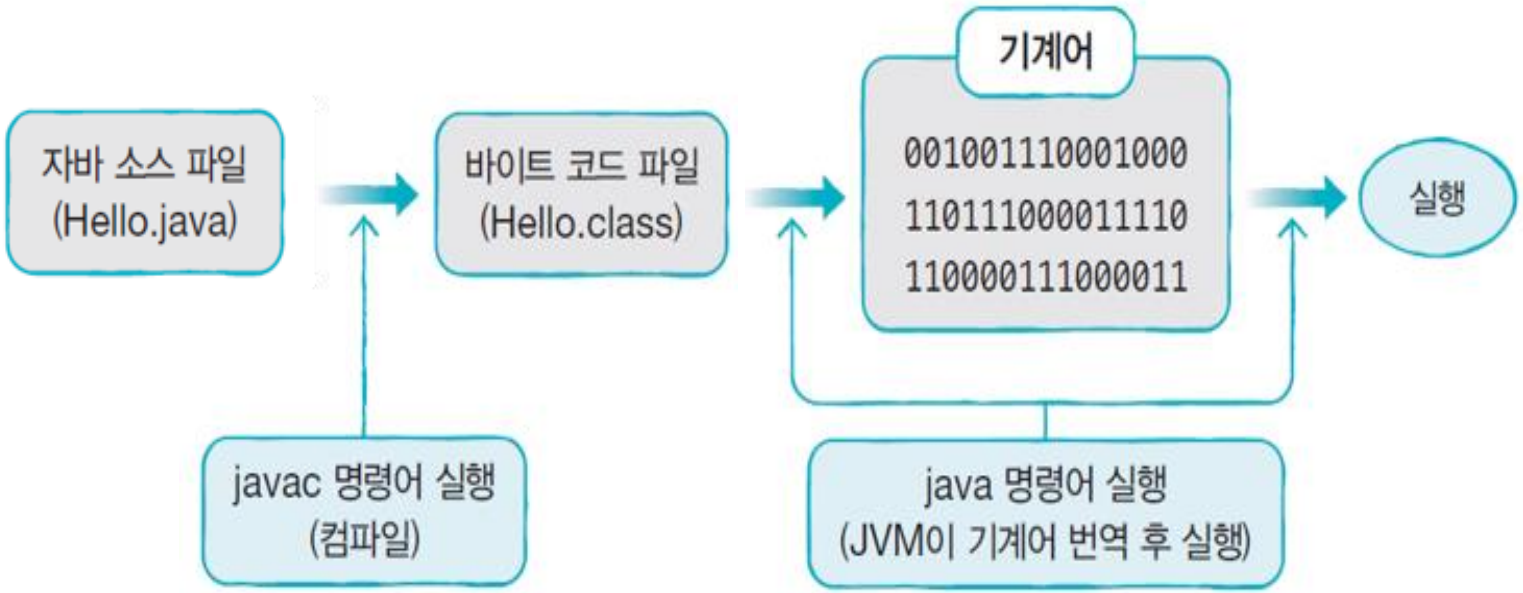
프로그래밍 언어 : 컴퓨터 프로그램을 개발할 때 사용하는 도구

대표적인 프로그래밍 언어

- 자바(Java) : 임베디드, 웹, 안드로이드 SW
- C / C++ : HW 장치 제어 및 임베디드 SW
- 파이썬(Python) : 빅데이터 분석 및 머신러닝 SW
- 코틀린(Kotlin) : 안드로이드 SW
- Object C / 스위프트(swift) : IOS SW

자바(Java) 언어의 특징

- 모든 운영체제에서 실행 가능(OS 독립성)
- 객체지향프로그래밍(OOP : Object-Oriented Programming)
- 메모리 자동 정리
- 풍부한 무료 라이브러리



- 소스코드를 작성하는 파일의 확장자는 .java 이다.
- .java 파일에 작성한 코드는 컴파일(compile) 과정을 거쳐 .class 확장자를 지닌 파일을 생성한다.
- IntelliJ에서는 작성한 코드를 저장하면 자동으로 컴파일을 진행한다.
- 컴파일을 통해 생성된 .class 파일은 JDK 설치 시 함께 설치되는 JVM(Java Virtual Machine)에 의해 컴퓨터가 실행가능한 파일을 생성 후 실행된다.
- IntelliJ에서 shift + F10 단축키를 통해 소스코드를 실행하면 위와 같은 과정이 진행된다.

```
3 public class Hello {
4     public static void main(String[] args) {
5         System.out.println("Hello, Java");
6     }
7 }
```

- 모든 소스코드는 class 선언부 안의 main() 메서드 선언부 안에 작성한다.
- 클래스 선언부와 메서드 선언부의 시작은 '{', 끝부분은 '}'이다.
- 열고 닫는 중괄호의 한 쌍을 잘 맞추어야 한다.
- 들여쓰기를 반드시 한다.(Tab – 들여쓰기, Shift + Tab – 내어쓰기)
- 문법에 맞게 코드를 작성하고 있다면 들여쓰기는 대부분 자동으로 된다.
- 실행문의 끝에는 반드시 세미콜론(';')을 붙인다.
- 모든 소스는 main메서드의 첫줄부터 마지막 줄까지 차례대로 실행된다.
- 자바 언어는 대소문자 구분을 명확히 해야한다.
- 작성한 코드의 실행 단축키는 Shift + F10이다.

출력문

- 실행 결과 혹은 값을 출력하여 결과를 확인할 수 있는 명령어
- 자바의 출력문은 System.out.print()와 System.out.println() 명령어 두 가지가 있다.
- 두 명령어의 차이는 출력 후 한 줄 개행 여부의 차이뿐이다.
- System.out.print() : 소괄호 안의 내용을 출력한다.
- System.out.println() : 소괄호 안의 내용을 출력하고 한 줄 개행한다.

출력문 상세 내용

- 숫자는 쌍따옴표에 감싸지 않고, 문자는 반드시 쌍따옴표에 감싼다.
- 소괄호 안의 내용이 연산 가능하다면 연산 결과를 출력한다.
- 자바는 문자를 더하는 연산이 가능하다. 문자끼리의 더하기 연산의 결과는 문자 나열이다.
- 숫자와 문자의 더하기 연산은 문자끼리의 더하기 연산 결과와 동일하다.
- 문자와 문자, 문자와 숫자끼리는 더하기 연산은 가능하지만, 더하기를 제외한 사칙연산(-, *, /)은 존재하지 않는다.
- 더하기 연산이 여러개 존재한다면 왼쪽부터 차례대로 더해지며, 괄호가 존재한다면 괄호부터 더하기 연산을 진행한다.

1-2. 변수(Variable)

변수(Variable)

- 변수란 하나의 데이터를 저장할 수 있는 공간, 혹은 그 공간의 이름을 말한다.
- 하나의 데이터를 저장할 수 있는 상자라고 생각하자.
- 변수는 필요한만큼 다수 사용할 수 있다.
- 변수를 사용하기 위해서는 1) 변수의 선언과 2) 변수의 초기화 과정이 진행되어야 한다.

변수의 선언

- 변수 선언 문법 : 자료형 변수명;
- 예시) int num; int age; int score;
- 자료형(Data Type)이란 데이터의 형태를 말한다.(숫자, 문자 등)
- num, age, score 등은 변수명이라 하며 변수명은 데이터를 저장할 수 있는 상자의 이름이라 생각하면 된다.
- 변수명은 마음대로 정할 수 있다. 단, 영어로 작성하도록 한다.
- 변수명은 중복으로 선언할 수 없다.
- 자바는 대소문자 구분이 확실하다. 변수도 마찬가지다.
- 변수는 소문자로 작성하는 것이 관례이다. 합성어의 경우 카멜케이스(CamelCase) 기법으로 작성하는 것이 관례이다.

변수의 초기화

- 변수의 초기화란 변수 선언 후 최초 값을 저장, 할당하는 과정을 말한다.

다음은 변수의 선언 및 초기화를 진행하는 코드의 예시이다.

```
int num;  //정수 하나를 저장할 수 있는 변수 num을 선언. 선언만 하면 num 변수에는 어떤 데이터도 저장되어 있지 않는 상태다.
num = 5;  //num 변수에 정수 5를 저장한다.
```

프로그래밍 언어에서 '=' 는 '같다'의 의미가 아니다! '우측 값을 왼쪽에 저장한다'로 해석하도록 한다.

아래와 같은 코드가 실행되었을 때 변수 age에는 어떤 데이터가 저장되어 있을까?

```
int age;
age = 20;
age = 30;
```

아래의 코드 실행 결과를 생각해보자.

```
int count;
count = 10;


System.out.println("count");
System.out.println(count);
System.out.println("count = " + count);
System.out.println("5" + count);
System.out.println(5 + count);
System.out.println("변수 count의 값은 " + count + "입니다");
```

아래의 코드 실행 결과를 생각해보자.

```
int money;
money = 100;
System.out.println(“money = ” + money);

money = 1000;
System.out.println(“money = ” + money);

money = 50 + 100;
System.out.println(“money = ” + money);

monry = money - 50;
System.out.println(“money = ” + money);
```

아래의 코드 실행 결과를 생각해보자.

```
int a;
a = 10;
int b;
b = 20;
int c;
c = a + b;


System.out.println(c);
System.out.println(“a + b = ” + c );
System.out.println(a + “ + ” + b + “ = ” + c);
```

변수를 사용하기 위해서는 변수의 선언 및 초기화가 진행되어야 한다고 했다. 변수의 선언과 초기화는 지금까지와 같이 각각 진행할 수 있지만, 동시에 진행해도 된다.

```
//변수의 선언과 초기화를 각각 진행
int a; //변수의 선언
a = 10; //변수의 초기화


//변수의 선언 및 초기화 동시 진행
int a = 10;
int b = 20;
```

1-3. 자료형(Data Type)

자바는 데이터를 형태(정수, 실수, 문자...)에 따라 구분하여 사용한다. 이러한 데이터의 형태를 데이터타입 혹은 자료형이라 한다.

자바는 다양한 자료형을 제공하며 크게 기본자료형(Primivite Type)과 참조자료형(Reference Type)으로 나뉜다.

참조 자료형은 다음에 학습할 것이며, 이번 시간에는 기본 자료형에 대해서만 알아보자. 기본 자료형은 다음과 같이 총 8가지를 제공한다.

자료형	데이터	크 기	표현 가능 범위
boolean	참과 거짓	1 바이트	true, false
char	문자	2 바이트	유니코드 문자
byte	정수	1 바이트	−128 ~ 127
short		2 바이트	−32,768 ~ 32,767
int		4 바이트	−2,147,483,648 ~ 2,147,483,647
long		8 바이트	−9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
float	실수	4 바이트	$\pm(1.40\times10^{-45} \sim 3.40\times10^{38})$
double		8 바이트	$\pm(4.94\times10^{-324} \sim 1.79\times10^{308})$

다음은 기본자료형의 사용 예시이다.

```java
public class UseVariable {
    public static void main(String[] args) {
        boolean bool = false;
        System.out.println("bool = " + bool);

        byte num1 = 10;
        System.out.println("num1 = " + num1);

        int num2 = 20;
        System.out.println("num2 = " + num2);

        double num3 = 10.5;
        System.out.println("num3 = " + num3);

        char word = '가';
        System.out.println("word = " + word);
    }
}
```

```
bool = false
num1 = 10
num2 = 20
num3 = 10.5
word = 가
```

자바는 문자를 문자와 문자열로 구분한다.

문자 : 홀따옴표로 감싼 한 글자.

문자열 : 쌍따옴표로 감싼 0개 이상의 글자.

자바의 기본 자료형에는 문자를 저장하는 자료형은 존재하지만 문자열을 저장하는 자료형은 없다.

문자열을 저장하는 자료형은 기본 자료형이 아닌, 참조 자료형에 포함되어 있다. 문자열은 많이 사용하기 때문에 기본자료형이 아니지만 소개를 하겠다.

String 자료형은 자바에서 문자열을 저장할 수 있는 자료형이다.

String str;　　String str = “자바”;

str = “자바”;

String의 앞글자(S)는 반드시 대문자!

String은 자바의 기본자료형이 아니다!

문자열 데이터이기 때문에 쌍따옴표(“”)로 감싼다!

아래 코드의 오류가 아는 이유가 뭘까?

```
3 public class UseVariable03 {
4     public static void main(String[] args) {
5         byte num1 = 128;
6
7         double num2 = 10.5;
8
9         float num3 = 10.5;
10
11        long num4 = 100000000000;
12
13        char var1 = 가;
14    }
15 }
```

- float num = 10.5; -> float num = 10.5f;
  float num = 10.5F;
- long num = 100000000; -> long num = 10000000L;
- char var1 = 가; -> char var1 = '가';

* 우리가 정수를 입력하면 자바는 기본적으로 int로 해석한다. 우리가 실수를 입력하면 자바는 기본적으로 double로 해석한다.

1-4. 형 변환(정수<->실수)

아래 코드는 오류가 없을까?

```
int num1 = 10;                  double num1 = 10.5;
double num2 =num1;              int num2 = num1;
```

자료형이 다른 두 값은 저장이 가능할 수 있지만, 그렇지 않을 수 도 있다.

범위가 상대적으로 작은 데이터 타입을 큰 범위의 데이터타입에 저장하는 것은 가능하다. 이때, 작은 범위의 데이터는 자동으로 큰 범위의 데이터타입으로 자료형이 변환되어 저장된다. 이렇게 작은 범위의 데이터 타입을 큰 범위의 데이터 타입에 저장할 때 일어나는 형 변환을 '자동 타입 변환' 또는 'promotion'이라 한다.

반대로 범위가 큰 데이터 타입을 범위가 작은 데이터 타입에 저장하는 것은 불가능하다. 하지만, 이런 불가능한 것을 강제를 자료형을 변환할 수 있다. 이를 '강제 타입 변환' 또는 'casting' 이라 한다.

casting의 예시는 다음과 같다.

```
int num = (int)1.5;
```

큰 범위의 데이터 타입을 작은 범위의 데이터 타입에 저장하고 싶을 때는 큰 범위의 데이터 앞에 '(작은 범위의 데이터 타입)'을 입력한다.

강제 타입 변환은 데이터의 손실을 유발한다.

타입이 다른 두 수의 연산은 큰 타입으로 자동변환되어 연산된다. 쉽게 말해서 정수끼리의 연산 결과는 정수, 실수끼리의 연산 결과는 실수, 정수와 실수의 연산 결과는 실수가 나온다는 말이다. 아래 예시의 실행 결과를 예측해보자.

```
int intValue = 10;
double doubleValue = 5.5;
double result = intValue + doubleValue;
```

```
int intValue = 10;
double doubleValue = 5.5;
int result = intValue + doubleValue;
```

```
int num1 = 5;
int num2 = 2;
double result = num1 / num2;
```

```
int num1 = 5;
double num2 = 2.0;
double result = num1 / num2;
```