디지털컨버젼스 자바(JAVA)
웹앱 개발자 양성과정

2020.12.29 ~ 2021.05.27

자바 시작하기

- 자바 프로그램의 기본 구조를 학습한다.
- 자바 프로그램을 위한 개발 환경을 구축한다.

자바 프로그램의 기본 구조

예제 Hello Java를 출력하는 자바 애플리케이션
[파일 이름:Hello.java]

```
public class Hello {
 public static void main(String[] args){
   System.out.println("Hello Java");
 }
}
```

클래스정의

메소드 정의

(1) 클래스 정의

- 자바 프로그램은 확장자가 “java”인 소스 파일을 하나 만들어서 자바 문법에 맞는 내용을 기술해야 한다.
- 자바는 클래스를 하나의 단위로 프로그램을 작성하기에 자바 소스 파일 안에 클래스를 정의해야 한다.
- 제대로 동작하려면 자바 소스 파일명이 클래스명과 동일해야 한다.

(2) main 메소드 정의

- 프로그램의 진입점이 다.
- 프로그램을 실행시키면 main 함수 내부에 기술된 내용들을 순차적으로 수행한다.

(2) main 메소드 정의

- **public static void main(String[] args)**

❶ ❷ ❸ ❹ ❺

❶ public

- 누구나 접근 가능하도록 하기 위한 예약어로 접근 지정자의 일종이다.

❷ static

- static으로 선언된 메소드는 클래스만 존재하면 수행할 수 있도록 한다.

❸ void

- 값을 갖지 않는다는 의미를 갖는 자료형태이다.

(2) main 메소드 정의

- **public static void main(String[] args)**

  ❶ ❷ ❸ ❹ ❺

❹ **main**

- 자바는 JVM에 의해서 실행되는데 자바 애플리케이션을 실행시키면 JVM은 이름이 main 메소드를 찾아 이 내부에 기술된 내용들을 순차적으로 실행한다.
- 그래서 main 메소드를 프로그램의 진입점이라고 한다.

(2) main 메소드 정의

- **public static void main(String[] args)**

❶ ❷ ❸ ❹ ❺

❺ **String[] args**

- 메소드를 실행시키기 위한 재료가 될 만한 데이터를 전달받아야 할 경우 메소드_이름 다음에 기술하는 ( )을 사용한다.

- ( ) 안에 기술한 args가 메소드에 값을 전달했을 경우 이를 받아 올 수 있는 전달인자가 된다.

(3) 문장

- main 메소드까지 정의했다면 이 메소드 안에 수행할 내용을 기술할 차례이다.
- 자바는 문장 단위로 프로그램을 작성해야 한다.
- 메소드 내부에 기술할 문장으로는 변수의 선언문이나 다른 메소드를 호출하는 문장들이 있다.
- JAM은 세미콜론으로 끝나면 이를 하나의 문장으로 인식한다.
- 반드시 문장의 끝을 세미콜론으로 마감해야 한다.

```
System.out.println( "Hello Java" ) ;
```

문장의 끝을 세미콜론으로 마감

(4) 출력을 위한 문장

```
System.out.println( "Hello Java" ) ;
```

출력할 내용

- **위 문장은 화면에 "Hello Java"를 출력하라는 내용이다.**
- **System.out.println( ) 메소드는 ”(큰 따옴표)로 둘러싸인 문자열을 화면에 출력하는 역할을 한다.**

자바 프로그래밍하기 (1) 편집기로 소스 작성하기

- **메모장에 다음의 내용을 입력하고 C:₩에 Hello.java란 파일이름으로 저장한다.**

```
public class Hello {
  public static void main(String[] args) {
    System.out.println("Hello Java");
  }
}
```

자바 프로그래밍하기 (1) 편집기로 소스 작성하기

- **자바 프로그램을 작성할 때 유의할 점**

  1. 자바 소스 파일명이 클래스명과 동일한 “Hello”로 확장자는 “java”로 지정해야 한다.

  2. 클래스 내부에 메소드가 포함되어 있다는 것을 한눈에 불 수 있도록 들여쓰기를 한다. 이렇게 작성해 두어야만 프로그램을 분석하기가 쉽다.

  3. 반드시 대소문자를 구분해서 기술해야 한다.

자바 프로그래밍하기 (2) 컴파일

- 컴파일은 자바 컴파일러인 "javac.exe" 명령어로 수행해야 한다.
- javac 다음에 자바소스파일명과 함께 확장자 java를 반드시 입력해야 한다.

```
c:₩>javac    Hello.java
```

- 자바로 만들어진 소스(파일명.java)가 에러 없이 성공적으로 컴파일하면 결과물로 파일명.class 형태의 클래스 파일(바이트 코드)을 얻을 수 있다.

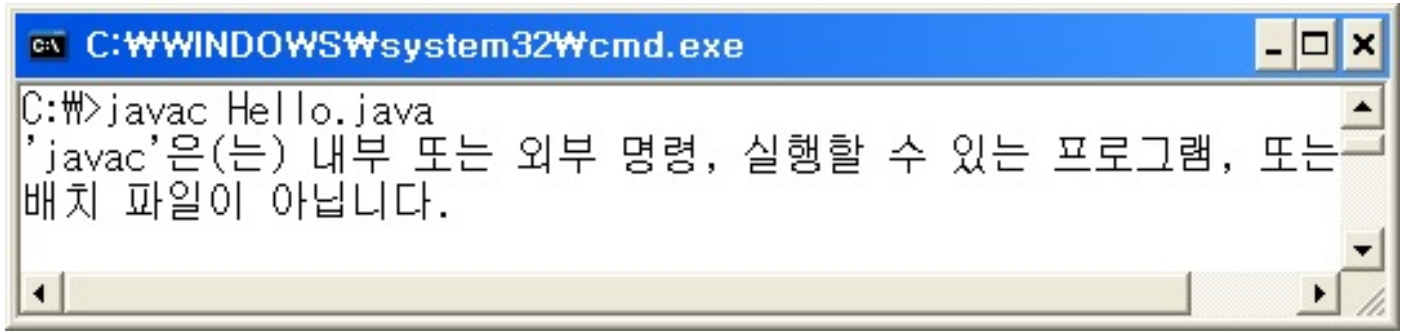
자바 프로그래밍하기 (3) 실행

- 컴파일의 결과로 생성된 클래스 파일을 실행하기 위한 도구가 바로 자바의 인터프리터(Interpreter)인 “java.exe" 명령어이다.
- 이 명령어는 컴파일러에 의해서 생성된 바이트 코드(파일명.class)를 자바가상머신(JRE)에서 실행하도록 해주는 개발도구이다.
- 이 명령어로 실행하려면 java 다음에 확장자는 생략하고 자바클래스파일명만 기술해야 하다.

```
c:₩>java Hello
```

자바 환경 설정하기

- [시작]-[실행] 메뉴를 선택한 후 cmd를 입력하여 [명령 프롬프트 창]에서 지금까지 설명한 대로 “javac Hello.java” 입력하면 제대로 실행되지 않고 오류 메시지가 출력된다.



- 오류 메시지를 살펴보면 javac라는 실행 파일을 찾지 못하기 때문에 발생하는 문제임을 알 수 있다.
- 확장자가 exe인 파일은 해당 파일이 존재하는 위치에서만 실행시킬 수 있다.

자바 환경 설정하기

- 지금 실행한 “javac.exe"는 “C:₩Program Files₩Java₩jdk1.6.0_05₩bin”에 위치해 있기에 자바 소스 파일이 존재하는 "C:₩"에서는 javac라는 실행 파일을 실행시킬 수 없다.

자바 환경 설정하기

- 폴더의 위치에 상관없이 javac.exe"와 “java.exe" 명령어를 실행시키려면 이 명령어가 존재하는 폴더 위치를 PATH 설정해 두어야 한다.

```
PATH = C:₩Program Files₩Java₩jdk1.6.0_05₩bin;
```

- 해당 폴더에 없는 명령어이면 이 PATH 에 설정된 폴더에서 찾아 실행시키기 때문이다.

실습 자바 환경 설정

[1] 소스 코드 (Hello.java)

```
public class Hello{
  public static void main(String[] args){
    System.out.println("Hello World");
  }
}
```

편집기에 소스 코드를 입력한다.

컴파일 과정 | javac Hello.java

javac를 실행시켜서 Hello.java를 컴파일하여 오류가 없으면 Hello.class라는 바이너리 파일이 생성

[2] 바이너리 코드 : (Hello.class)
컴파일된 결과물 Hello.class

실행과정 | java Hello

자바 가상 머신을 통해 Hello.class파일을 실행
JVM은 바이트코드를 해당 플랫폼에서 이해할 수 있는 형태로 해석하여 그 프로그램을 실행시켜줌

[3] 가상 머신
Hello Java

주석문

- 주석문은 프로그램에 대한 설명을 덧붙이므로 코드를 이해하기 쉽게 하기 위해서 사용한다.
- 주석문은 실행 중에 영향을 미치지 않는다

주석문

```
public class Hello {
  /**
  * @작성자 : 홍길동
  */
  public static void main(String[] args) {
    /* 모니터에 Hello Java를 출력하는
    간단한 예제이다. */
    System.out.println("Hello Java"); //출력을 위한 문장
  }
}
```

주석문

- **/* ~ */ 주석문**
  - /* 로 시작해 */ 이 나올 때까지 모든 내용이 주석 처리가 된다.
  - 여러 줄에 걸쳐 블록 단위로 주석 처리할 경우 사용한다.

- **// 주석문**
  - // 뒤에 있는 한 줄만 주석처리 된다.

- **/** ~ */ 주석문**
  - 여러 문장을 주석 처리할 수 있다는 면에서 /* ~ */ 와 유사한 기능을 갖는다.
  - 한 가지 유용한 기능이 있다면 이 주석은 javaDoc를 이용해 소스코드에 대한 도움말을 생성하는 기능을 갖고 있다.

자바의 기본 출력

- **System.out.println("Hello Java");**

필드	설명
in	키보드로부터 입력받기 위한 필드
out	콘솔로 출력하기 위한 필드
err	에러 메시지를 콘솔로 출력하기 위한 필드

자바의 기본 출력

메소드	설명
println(내용)	ln이 LiNe의 약자인 것을 보면 알 수 있듯이 이 메소드는 자동 개행하기에 내용을 출력하고 줄이 바뀐다.
print(내용)	내용을 출력한 후에 마지막 출력한 문자 뒤에 다음 내용을 출력하므로 줄이 바뀌지 않는다.
printf("형식지정자", 내용)	형식지정자에 맞게 내용을 출력한다.

감사합니다.

끝