

자바 시작하기

#01. 프로그램이 만들어 지는 원리

사람이 이해할 수 있는 수준의 언어로 작성된 파일을 컴퓨터가 이해할 수 있는 명령어 형태로 변환하여 구성된다.

- 사람이 이해할 수 있는 수준의 언어 : **프로그래밍 언어** (Java, Python, C, C++ 등)
- 컴퓨터가 이해할 수 있는 명령어 : **2진수 집합**

1) 바이너리

- 컴퓨터가 이해할 수 있는 명령어들의 집합
- 명령어들은 이진수의 집합으로 구성된다. *.exe 등
- 일반적으로 **컴퓨터에서 실행 가능한 프로그램들을 바이너리**라고 한다.
- 사람이 직접적으로 내용을 이해할 수 없다.

2) 프로그래밍 언어

- 사람이 이해할 수 있는 수준의 언어
- C, C++, Java, C#, Python, R 등이 있다.
- 프로그래밍 언어마다 서로 다른 문법(작성규칙)을 갖고 있다.

3) 소스코드

- 프로그래밍 언어가 기록된 텍스트 파일
- 파일의 **확장자**를 프로그래밍 언어에 맞게 수정한다.
 - Java 언어의 확장자는 ***.java** 임.

파일 확장자


파일 확장자(영어: filename extension)는 컴퓨터 파일의 이름에서 파일의 종류와 그 역할을 표시하기 위해 사용하는 부분이다.

- helloworld.pptx : 파워포인트 파일
- helloworld.jpg : 사진
- helloworld.mp4 : 동영상
- helloworld.java : 자바 소스코드
- helloworld.py : 파이썬 소스코드

윈도우에서 파일 확장자 표시하는 방법

윈도우는 기본적으로 모든 파일의 확장자를 표시하지 않는다.

아래의 방법으로 확장자를 표시할 수 있다.

img/20201013130020677.jpg

파일 확장자 변경하기

파일 확장명에 체크가 된 상태에서 원하는 파일의 이름을 변경하면서 확장자도 함께 처리한다.

윈도우 폴더 창에서 파일을 선택한 상태로 **F2**버튼을 누르면 파일명을 변경할 수 있다.

4) 컴파일

텍스트 형태로 작성된 소스코드(*.java)를 작성중인 프로그래밍 언어를 의미하는 확장자로 저장하고 컴퓨터에서 실행할 수 있는 바이너리로 변환하는 작업.

컴파일을 수행하는 소프트웨어를 컴파일러라고 한다.

JAVA 소스코드는 컴파일 과정을 거쳐서 *.class 파일로 변환된다.

#02.컴퓨터를 사용하기 위한 환경(User Interface)의 종류

1) GUI

- Graphic User Interface
- 그래픽을 마우스로 클릭, 드래그 하여 사용한다.
- ex) 일반적인 바탕화면 상태.

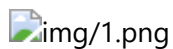
2) CLI

- Command Line Interface
- 명령어를 입력하는 환경을 의미한다.
- ex) 윈도우: 명령프롬프트(Commander), 파워셸(PowerShell)
- ex) 리눅스/맥: 터미널(Terminal)

3) 윈도우에서 명령 프롬프트 실행하기

기본 실행 방법

1. WinKey + R → cmd 입력 후 Enter
2. 관리자 권한으로 실행해야 할 경우 Shift+Enter



#03. JDK(Java Development Kit) 설치하기

🔗 [<https://jdk.java.net/archive/>](https://jdk.java.net/archive/) 에서 다운로드 받을 수 있다.

2024년 08월 기준 Spring3와 호환되는 버전은 17이므로 호환되는 버전(17.0.2)을 설치하도록 한다.

1) 현재 컴퓨터에 JDK 설치 여부 확인

```
$ javac -version
```

2) JDK가 설치되어 있는 경우

설치된 버전이 출력됨.

```
Microsoft Windows [Version 10.0.18362.175]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\leekh>javac -version
**javac 17.0.2**
```

3) JDK가 설치되어 있지 않은 경우

아래와 같이 표시됨. (자바 프로그래밍을 위해서는 설치 필요함.)

'javac'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는 배치 파일이 아닙니다.

4) JDK 설치하기

OpenJDK는 압축 파일 형태로 배포된다. 내려받은 파일을 적절한 위치에 압축 해제한다.

(여기서는 C\jdk-17.0.2)

5) JDK 환경 변수 등록

환경변수란 자주 사용되는 프로그램이나 파일들이 위치한 디렉토리(폴더)의 경로를 윈도우에 등록해 놓는 값이다.

환경변수를 설정해 놓으면 매번 그 위치에 찾아가지 않더라도 명령 프롬프트를 통해 프로그램을 즉시 실행시킬 수 있기 때문에 프로그래밍에 필요한 명령어들을 편리하게 사용할 수 있다.

JDK가 설치된 폴더 위치 확인

예) C:\jdk-17.0.2

해당 경로를 미리 복사해 놓으면 좋다.

시스템 속성 열기

바탕화면 혹은 탐색기 왼쪽의 트리에서 **내 PC**를 마우스로 우클릭 하고 **속성** 항목을 선택하고 **시스템** 창이 표시되면 오른쪽 메뉴에서 **고급 시스템 설정** 클릭한다.

시스템 속성창이 표시되면 **고급** 탭으로 이동 후 우측 하단의 **환경 변수** 버튼을 클릭한다.

JAVA_HOME 변수 등록하기

주의사항!!! : 시스템 속성창의 영역이 상/하단으로 나누어 지는데, 아래 영역만을 사용해야 한다.

화면 하단의 **새로 만들기** 버튼을 클릭한다.

다음의 항목을 입력하고 **확인**버튼을 누른다.

- 변수 이름 : JAVA_HOME

- 변수 값 : 앞에서 미리 복사해 둔 JDK 설치 폴더 경로

예) `C:\jdk-17.0.2`

PATH 변수 편집하기

전 단계에서 표시된 **환경 변수**창 아래 부분에서 **path**라는 항목을 찾아 선택한 상태로 하단의 **편집**버튼을 클릭한다.

환경 변수 편집화면 우측의 **새로 만들기** 버튼 클릭하고 새로 생성된 입력 칸에 `C:\jdk-17.0.2\bin`이라고 입력한다.

입력이 완료되면 모든 창들에 대해 **확인**버튼을 누르고 빠져나온다.

결과확인

열려 있는 명령프롬프트와 폴더 창들을 모두 닫는다.

다시 명령 프롬프트를 실행하여 `javac -version`명령어로 설치 결과 확인한다.

명령프롬프트를 반드시 새로 실행해야 한다.

#04. HelloWorld

- 개인용 작업 폴더 생성
 - 자신의 이름을 활용한 폴더로 구성하는 것을 권장
 - 프로그래밍에서 사용되는 파일명, 폴더명은 한글과 공백을 사용하지 않는 것이 원칙

예) `C:\Users\현재_로그인한_사용자이름\leekh-java`

- 생성한 폴더 안에서 빈 곳을 마우스 우클릭하고 **새로 만들기 > 새 텍스트 문서** 선택
- 생성된 파일의 이름을 `App.java`로 변경
- 메모장을 실행하여 생성한 파일을 메모장에 끌어 넣는다.

Winkey + R -> notepad (엔터)

1) 코드 작성하기

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

2) 컴파일

1. 소스파일이 있는 폴더의 주소 표시줄에서 `cmd` 입력 후 엔터
2. `javac` **파일이름** 형식으로 컴파일 명령어 수행

```
$ javac App.java
```

- 정상적인 경우 아무런 출력이 없다.
- 코드에 문제가 있는 경우 해당 파일의 이름과 위치(라인수)가 에러 메시지와 함께 출력된다.

3) 실행

java 확장자를 뺀_파일이름 형식으로 실행한다.

```
$ java App
```

- 출력결과

```
Hello World  
안녕하세요. 자바!!
```

4) 주석문

프로그램 소스코드 안에 개발자가 작성하는 설명문.

프로그램 컴파일에서 제외된다.

한 줄 짜리 주석문

문장 앞에 `//`를 적용한다.

```
// 이 부분은 주석입니다.  
// 여러 줄을 작성하기 위해서는 모든 라인 앞에 "//"를 추가해야 합니다.
```

두 줄 이상에 대한 주석문

`/* */` 형태로 블록을 구성한다.

이 영역안에서는 줄바꿈이 자유롭다.

```
/*  
    이 안에서는 자유롭게 여러 줄을 작성할 수 있습니다.  
    상당히 긴 내용을 정리할 때 사용한다.  
*/
```

3) 주석의 활용

특정 블록의 컴파일을 방지

아래 코드에서 `***안녕하세요. 자바!***`라는 문장은 컴파일 되지 않는다.

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

프로그램 설명문

프로그램 소스코드의 최상단에서 해당 소스파일의 작성자와 주요 기능을 설명하는 내용을 명시하는 형식.

특별히 정해진 형식은 없다.

개발자 개인마다 혹은 회사마다 형식을 정해놓고 사용한다.

🔗 미리 정의된 소스코드 작성 규칙을 **코딩 컨벤션**이라고 한다.

- **App.java (개선)**

```
/**
 * @filename      : HelloWorld.java
 * @description   : 자바 소스코드의 기본 구조를 파악하기 위한 예제
 * @author        : 이 광 호 (leekh4232@gmail.com)
 */
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
        System.out.println("안녕하세요. 자바!!");
    }
}
```

#05. HelloWorld 코드 분석

1) 기본 규칙

프로그램 소스코드는 줄바꿈이나 들여쓰기를 특별히 강조하지 않는다.

모든 줄바꿈과 들여쓰기는 개발자가 보기에 좋은 형식으로 정리하기 위한 용도이다.

대소문자를 엄격하게 구분한다.

프로그램 구문은 블록간의 중첩으로 구성된다. 블록은 **중괄호({})**로 표현한다.

서로 중첩된 블록간에는 들여쓰기로 블록의 깊이(depth)를 표현하기도 한다.

2) 예약어

프로그램 코드에서 특별한 기능을 갖는 키워드.

여기서 각 예약어들의 기능을 설명하기는 이르기 때문에 정해진 코드 양식정도로 여기고 외워두도록 하자.

아래는 이 예제에서 사용되었던 예약어들이다.

🔔 public, class, static, void, String[], main

3) 명령어

특정 기능을 수행하기 위한 문장.

하나의 명령어는 반드시 ****세미콜론(;)****으로 종료되어야 한다.

원래의 이름은 클래스, 객체, 메서드 등으로 불리지만 아직은 진도가 이르기 때문에 다음의 구문을 통째로 **출력을 위한 명령어**로 기억하자.

```
System.out.println( ... 출력할 내용 ... );
```

4) 문자열 (String)

쌍따옴표로 감싸진 문장.

예제에서는 출력할 내용으로 사용되었다.

5) 클래스 (매우중요)

자바 프로그램의 최소단위이자 최상위 블록.

1. 하나의 소스파일은 하나 이상의 클래스로 구성되어야 한다.
2. public은 상황에 따라 적용여부를 결정해야 하지만 public 이 적용된 클래스는 소스파일 내에서 반드시 단 하나밖에 존재하지 못한다.
3. public이 적용된 클래스의 이름은 소스파일의 이름과 반드시 동일해야 한다.
4. 클래스 이름은 개발자가 자유롭게 정의할 수 있다.

```
[public] class 클래스이름 {
    // ... 기능 구현 ...
}
```

🔔 당분간은 `public class`를 정규화 해서 무조건 적용하는 것으로 합니다.

6) 메서드 (매우중요)

클래스 안에서 기능을 구현하기 위한 블록 단위.

하나의 클래스 안에는 여러 개의 메서드가 존재할 수 있다.

메서드 이름은 개발자가 자유롭게 정의할 수 있다.

```
public class 클래스이름 {  
    public static void 메서드1() {  
        // ... 기능구현 ...  
    }  
  
    public static void 메서드2() {  
        // ... 기능구현 ...  
    }  
}
```

1. 다른 프로그래밍 언어에서는 메서드를 **함수**라고 부르기도 합니다.
2. 메서드 이름 앞에 명시되는 예약어들은 상황에 따라 다르지만 당분간은 `public static void`를 정규화해서 사용하도록 합니다.

7) main 메서드

프로그램의 시작점이 되는 사전에 미리 약속되어진 메서드

프로그램이 실행될 때 무조건 main 메서드의 블록이 실행된다.

main 메서드는 이름 오른쪽의 소괄호에 `String[] args`라는 구문을 명시하도록 약속되어 있다.

main 메서드를 포함하는 클래스를 특별히 main 클래스라고 부르기도 한다.

8) 결론 (자바 소스코드의 최소단위)

```
public class 클래스이름 {  
    public static void main(String[] args) {  
        // ... 기능구현 ...  
    }  
}
```