학교 JAVA

- 소스 파일 : 프로그래밍 언어로 작성된 텍스트 파일 ⇒ .java
- 컴파일 : 소스 파일(바이트 코드)을 컴퓨터가 이해할 수 있는 기계어로 만드는 과정 ⇒ .class
- .java파일을 실행하면 .class파일이 생성되며 .java안에 있는 모든 class 개수만큼 생김 → A class 안 B class는 A$B.class파일로 생성됨

- 자바 특징
  - 한 번 작성되어 컴파일 된 자바코드는 모든 플랫폼(HW+OS)에서 JVM만 있으면 바로 실행 가능
  - 플랫폼에 독립적인 언어 : JVM과 바이트 코드로 이뤄짐
  - 하나의 소스파일에 여러 클래스 작성 가능
  - 모든 변수와 메소드는 class 안에 선언되어야 함 → class 밖에 선언하면 오류남
  - public class는 .java파일에 하나만 존재해야 하며 public class 클래스명은 .java 파일 이름과 같아야 됨

    또한 public static void main(String[] args) {} 메소드가 존재해야 실행 가능하며 main() 메소드에서 시작됨

- JDK : 자바 개발 도구(컴파일러 등) + JRE(JVM + 자바API)
  - javac : 자바 소스를 바이트 코드로 변환하는 컴파일러
  - java : 자바 응용프로그램 실행기로 JVM를 작동시켜 실행함
- JRE : 자바 실행 환경으로 JVM + 자바API들이 들어있는 모듈 파일 등이 있음

- 식별자란 클래스, 변수, 상수, 메소드 등에 붙는 이름임

  <예시>

  `int name; // 이때 name이 식별자`

  사용 가능한 예

```
int    name;
char   student_ID;                    // '_' 사용 가능
void   $func() { }                    // '$' 사용 가능
class  Monster3 { }                   // 숫자 사용 가능
int    whatsyournamemynameiskitae;    // 길이 제한 없음
int    barChart;   int barchart;      // 대소문자 구분
                                      // barChart와 barchart는 다름
int    가격;                          // 한글 이름 사용 가능
```

- 기본 데이터 타입 : byte(1) / short(2) / int(4) / long(8) / float(4) / double(8) / char(2) / boolean(1bit)
- 참조 데이터 타입
  - 클래서(Class)에 대한 참조
  - 인터페이스(Interface)에 대한 참조
  - 배열(Array)에 대한 참조
  - 열거(Enum)에 대한 참조

- 리터럴이란 프로그램에서 직접 표현한 값(데이터 그 자체)으로 정수, 실수, 문자, 문자열, 논리 리터럴이 있음

  <예시>

```
int num = 10; // 10 리터럴
double db = 0.123; // 0.123 리터럴
char ch = 'A'; // 'A' 리터럴
String str = "ㅋ3ㅋ"; // "ㅋ3ㅋ" 리터럴
```

- 상수(final)

  <예시>

```
static final intNUM= 1;
// 상수는 보통 static과 같이 선언함
// 상수명은 대문자로 표시하며 선언과 동시에 초기화를 해줘야됨
// 실행중 값 변경 불가
```