[개요]

생코 강의를 들으면서 모든 내용을 워드에 정리하려고 하다보니 너무도 많은 시간이 걸리고, 수업에도 집중하기 힘들었다. 그리고 그렇게 고생해서 적은 내용들을 나중에 한 달, 두 달 뒤에 찾아보면 다시 이해하는데 또 시간이 걸릴뿐더러 텍스트 투성이의 공부 내용을 줄줄이 읽기가 참 힘들었다. 분명 읽기 쉽도록 구어체로, 나름 농담도 섞어가면서 적었음에도 불구하고 말이다.

이러한 문제를 인지하고 앞으로 수업을 들을 때에는 메모보다는 강의내용에 집중하고, 정말 강의에서 중요하다, 잊고 싶지 않다 싶은 내용들만 필기하기로 했다.

[Java SE]

Java Platform 이라고도 부른다.

Java Standard Edition, 자바의 표준안이다. 즉 자바의 명세서, 자바의 설명서다.

명세서는 JCP(Java Community Process, <http://jcp.org>)라는 조직이 관리한다.

[JVM]

자바 버추얼 머신. Java Virtual Machine.

자바가 실제 구동하는 환경이다. 운영체제와의 호환성을 위해서 가상 환경에서 자바를 굴린다고 생각하자.

Just In Time Compiler라고 하는데, JIT가 뭔지는 후에 찾아보자.

[Java EE]

기업용 자바다.

[Java Me]

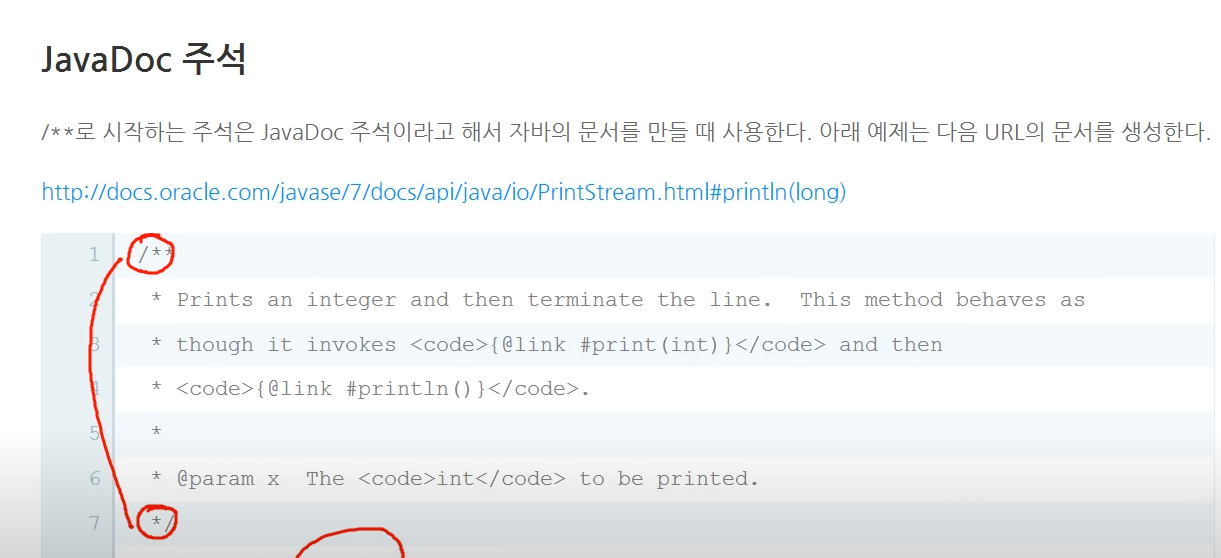
모바일 용 자바 버전이다.

[바이트 코드]

java 파일을 컴파일해서 나오는 class 파일들은 바이트 코드로 이루어져 있다.

어디가서 “바이트 코드가 어쩌구~” 소리 나오면 무슨 말일까 고민해보다가 이런 후보도 있다는 것을 머리속에서 꺼내서 추측해보자.

[JavaDoc 주석]



/\*\* 주석은 javadoc 문서를 만든다.

위는 println의 주석이다. 해당 주석은 자바 공식문서의 println에 대한 설명과 동일한 것을 알 수 있다.

공식 문서를 관리하기 위해서 쓰는 주석인 듯 하다.

아니면 일반인들도 쓸 수있을까?

[statement 스테이트먼트, 문장]

잘 들어본 적 없지만 코드 한 줄, 엄밀히 말하자면 코드의 시작과 끝(코드는 2줄이상 될 수도 있으므로)까지를 statement, 스테이트먼트라 한다.

그니까 세미콜론 하나로 끝나는 영역을 스테이트먼트라고 한다.

[세미콜론을 코드의 끝에만 쓰는 이유]

세미콜론을 보통 코드의 끝에만 붙이는데, {} 이와 같은 중괄호에는 안붙이는 이유는 곧 세미콜론은 문장의 끝을 의미하는데, 중괄호와 같은 괄호에는 이미 여는 곳과 닫힌 곳이 명시되어 있기 때문이다. 코드는 그런 것이 없으므로 세미콜론으로 끝내주고.

String a, b;

a = "Hello"; b = "World";

System.***out***.println(a+" "+b);

세미콜론이 코드의 끝을 의미하므로 이렇게 한 줄에 코드를 2개씩 적을 수 있게 되는 것이다.

가독성은 떨어지지만.

[상수와 데이터 타입]

int가 몇 바이트고 하는 건 대충 숙지하거나 아니면 나중에 찾아보기로 하자.

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

**float** a1 = 2.2; // X

**float** a2 = 2.2f; // O

**double** a3 = 2.2; // O

**long** b1 = 2147483648; // X

**long** b2 = 2147483648L; // O

**int** c1 = 100; // O

**byte** c2 = 100; // O가 나온다. why?

**byte** c3 = 100B; // X 이렇게 해줘야 하는 거 아닌가?

**short** c4 = 100S; // X 이렇게 해줘야 하는 거 아닌가?

**int** c5 = (**int**)100; // O

**byte** c6 = (**byte**)100; // O

**short** c7 = (**short**)100; // O

}

위 예시 코드를 보면 자바가 어떻게 상수를 다루는 지를 알 수 있다.

먼저 실수형을 살펴보면 기본적으로 2.2라는 상수는 자바에선 double형으로 인식한다.

따라서 float 형 변수에 상수 2.2를 저장하려고 하면 double 형 상수를 float에 저장할 수 없으므로 에러가 난다.

따라서 double형 상수 2.2를 float형으로 변환해줘야 하는데, 그 표현이 2.2F 가 된다.

정수형 타입 long도 그러하다. 2147483648이라는 수는 int형 타입이 저장할 수 있는 수의 범위를 1만큼 넘어선 수다. 따라서 범위가 더 넓은 long 타입을 쓰려고 했는데, 또 에러가 난다.

이유는 2147483648과 같은 정수형 상수들은 자바에서 기본적으로 int 타입이기 때문이다. 설령 그것이 int의 범위를 벗어났더라도 말이다.

따라서 또 long 타입으로 상수를 변환(casting)해줘야 한다. 그 표현은 2147483648L이 된다.

그럼 더 작은 범위는 어떨까? 결론적으로 말하면, 예외적으로 자바에서 int보다 더 작은 범위를 가진 데이터 타입의 경우 int 값을 넣는 것이 허용된다.

int형보다 더 좁은 범위를 가진 정수형 타입인 byte와 short에 int형 상수인 100을 넣기 위해서 각각 byte와 short로 형변환을 해줘야 할 것 같지만 자바에선 관용적으로 byte와 short에 int형 상수를 넣는 것이 허용되기 때문에 에러가 나지 않는다. 오히려 형변환을 해주려고 100B, 100S 따위의 표현을 쓰는 것은 에러를 불러온다.

굳이 형변환을 할 때에는 (int), (byte), (short)로 하는 게 더 나은 선택이다.

아 참 사실 100은 리터럴이다. 상수가 아니라. ㅎ; 엄밀히 따지려면 아래 주소를 확인하자.

https://mommoo.tistory.com/14

[암시적 형 변환(자동 형 변환)]

좁은 데이터 타입을 넓은 데이터 타입에 대입할 때에는 자동 형 변환이 일어난다.

**int** L1 = 123; // O

**long** l2 = (**int**)123; // O

**long** l3 = 123; // O

이런 코드가 가능하다는 것이다.

그런데 아까 상수와 데이터 타입 챕터에서 봤을 때

**long** b1 = 2147483648; // X

**long** b2 = 2147483648L; // O

위 코드는 안됐는데? 왜 위에 있는 건 되느냐?

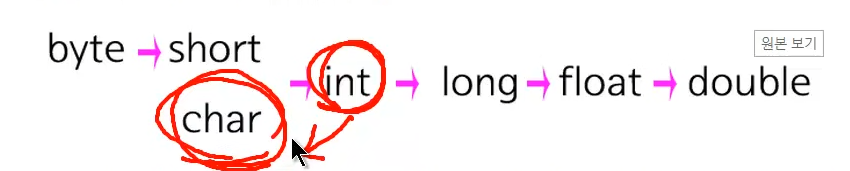
비직관적이라는 것을 나도 알지만, 2147483648이라는 수가 int의 범위를 벗어났기 때문이다.

근데 int의 범위를 벗어났다고 왜 안되느냐? 그것까진 모르겠다.

**double** d = 2.2F; // O

**float** f = 2.3d; // X

실수형에서는 위와 같이 더 큰 데이터 타입에 대입할 때 자동 형변환이 일어난다는 것을 알 수 있다.



위와 같이, byte는 short로 형변환 가능하지만 short는 byte로 형변환 할 수 없다.

그런 식으로 그림을 해석할 수 있다. long은 float으로 자동 변환될 수 있지만, float은 long으로 자동 형변환 할 수 없다.

---요점---

정보의 손실이 일어나지 않는다면 코딩의 편의를 위해서 암시적으로 형 변환을 해준다.

[나누기]

자바같이 정적타입언어에서 나누기는 참 귀찮다.

나눠지면 정수가 실수가 되는 경우가 허다하기 때문이다.

결론부터 말하자면 아까 말했던, 더 좁은 타입이 더 큰 데이터 타입으로 바뀐다는 것을 기억하자.

이를 테면

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

// **TODO** Auto-generated method stub

**int** a = 10;

**int** b = 3;

**int** c = a/b; // 3이 들어간다

**float** d = a/b; // 3.0이 들어간다

**int** f1 = 10;

**float** f2 = 3.0f;

**float** f3 = f1/f2; // 3.3333333이 들어간다

System.***out***.println(f3);

}

위와 같은 코드를 예시로 들 수 있다.

즉 나눌 때 더 큰 데이터 타입이 된다.

a와 b는 모두 int였으니 결과값도 int인 3이 된다.(0.3333… 은 소실됨)

f1와 f2는 각각 int와 float이니 f1/f2는 더 큰 데이터 타입인 float이 된다.

[문자열 비교]

자바에서 문자열 비교도 요상하게 구현되어있다.

String s1 = "Hello World";

String s2 = **new** String("Hello World");

System.***out***.println(s1 == s2); // false?

System.***out***.println(s1.equals(s2)); // true

System.***out***.println(s2.equals(s1)); // true

s1 == s2가 false가 뜬다. 골아프다.

String s1 = "Hello World";

String s2 = "Hello World";;

System.***out***.println(s1 == s2); // true

System.***out***.println(s1.equals(s2)); // true

System.***out***.println(s2.equals(s1)); // true

그런데 이렇게 하면 true가 뜨는 것으로 보아

new String으로 객체를 생성하는 것과 단순히 큰따옴표로 문자열을 묶는 것에는 내부적으로 차이가 있을 것임을 추측할 수 있다.

아직 생코에서 이 내용을 알려주진 않았으나, 나중에 알려준다고 하니 일단 강의를 듣도록 한다.

모르긴 몰라도 객체와 관련된 내용일 듯 싶다.

아니 그런데 나는 new String(“문자열”) 로 String 객체를 만드는거나

String a = “문자열” 로 만드는거나 똑같이 String 객체를 만든다고 생각했는데 아니었나???

<https://ict-nroo.tistory.com/18>

https://haenny.tistory.com/18

위 주소에 이걸 다뤘네. 근데 haenny가 더 깔끔담백하게 잘 설명했다.

[반복문 iteration]

반복문은 iteration이라고도 한다.

[자바에서 배열이란]

배열 또한 데이터 타입이다.

보통 변수엔 하나의 데이터만 들어갈 것이라고 생각하기 쉽지만, 자바에선 배열 타입의 변수를 선언해서 쓴다.

[인터페이스]

인터페이스란 무엇일까? 추상적으로는 알고 있겠지만, 말로 그것을 설명하는 것은 참 어렵다.

방금 이에 대한 인사이트를 얻었다. 인터페이스는 무엇인가와 또 다른 무엇인가, 즉 A와 B의 연결에서, A와 B가 연결되는 지점을 말한다.

UI도, 결국 유저와 APP 사이의 연결점을 말하는 것이다.