09/22 금요일 토론 내용

정훈 : 병결 미참

철환 :
overflow

wrapper class

magic number

조건연산자 우선순위

predicates

null과 비교

조건연산자 우선순위, 연산 방식

JVM이 모든 배열 길이 별도로 관리해서 .length가 직접 상수값을 사영하는 것 보다 효율적
배열은 한번 생성하면 길이를 변경할 수 업슨 상수 = 수정할경우 new로 재선언해야함.

Arrays.fill(배열 변수. 데이터) // 기본적으로 초기화가 되어져 있어야 함.

int [] a; // namespace 생성;
new int [size]; -> heap 메모리를 가리키고 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0} 으로 initialization

(a, 10) -> {10, 10, 10, 10, 10, 10, 10, 10, 10, 10} 으로 값을 한번에 변경

윤호 : 2장

digit과 numeric차이
Integer(Wrapper class -> numeric)

- Unboxing하지 않을 시 산술 연산이 불가능
- null 값으로 처리 가능
- 저장공간이 큼
- null값으로 처리가 가능해 SQL에 용이하게 쓰인다.

int (primitive type -> digit)

- 산술 연산 가능
- null로 초기화 불가능
- 저장공간이 4Byte라고 작음

int x;

int y = 10;
y = 20;

initialization

reference type에서 reference type 의 경우 지정된 값으로 초기화를 하지 않아도 각 type마다 정해진 값으로 초기화 됨.
int -> 0
boolean -> false;

side effect

0915

procedure, -> side effect가 무조건 일어나야함.
function -> side effect가 무조건 일어나면 안됨(input - output은 필수로 존재해야 함.)
side effect가 발생하는것을 간과할 수 있음.
ex -> reference type에서의 field value는 메모리에 저장되어진 값이 변경되기에 side effect가 발생
method -> 객체 지향에서 상태 값을 변경시켜야 하는()

si
primitive type으로 확장해서 생각했을 경우, initialization을 한 variable의 값을 변경하는 경우 -> side effect;

ex) int x, y
x = y = 7 -> side effect 안일어남
int x = 10
x = 11 -> side effect 발생

아영 :

Definition과 Declaration의 차이

Declaration은 메모리를 차지 하지 않고, namespace만 만들어놓은 상태.
Definition은 namespace안에 실제 값 또는 statement가 저장됨. 메모리가 할당 된 상태.

Initialization과 Assignment 의 차이

initizlization, 변수 선언과 동시에 값을 넣는 경우 (Definition과 비슷함)
Assignment, 같은 주소에 값을 재정의 하는 것.

Statement, Expression

{}(code block)에서 단독으로 쓰일 수 있으면 Statement,
없으면 Expression
ex) a++ -> Statement
ex) 1+2 -> Expression

Array.length

arr.length 는 상수이므로, 다른 메서드처럼 length() 표현을 안함.
JVM에서 관리해줌

null비교
instance가 null인 경우, instance의 값이 주소값을 가지지 않고 있기에, 일반적으로 사용하는 equals를 사용하지 않고, = 연산자를 사용해야 한다.
A a;
a == null -> true;
a.equals(null) -> NPE발생

Predicate
boolean을 리턴하는 메서드를 칭함.

Magic Number
설명 없이 코드에 쓰는 숫자 literal. 많이 쓰면 프로그램 가독성 떨어지고 유지 관리도 어려워지니 constants(상수)
사용을 기피할 것.

Wrapper class
자바에서만 특이한 경우 (참고 : 객체지향에서는 해당 타입이 존재하는게 올바르지 않음)

조건 연산자 순서
not -> and -> or
(and의 경우, 앞이 false일 경우 뒤에 연산자 연산 안함)
(or의 경우도 마찬가지, 앞이 true일 경우, 뒤에 연산자 진행 안함.)

Overflow 메모리 변화 과정
011111111111111 + 1 (양수)
1000000000000 -> (음수) 첫번째 바이트가 부호를 나타내는 비트이기때문에 127 -> -128이 된다.

StackOverFlow
CallStack영역에 메모리가 가득 차면서 Heap영역을 침범하기 때문임.

++첨언
stack은 thread가 생성될때마다 하나씩 생성됨.