UI쪽(화면개발)- 안뽑음.

백엔드를 더 뽑음.

회사

메가존

인라이플

192.168.0.53

공유폴더.

\\192.168.0.53

SEOUL IT

0703

JDK = Java Development Kit

-> JRE + Standard Library + Tools

JRE = Java Runtime environmnet

-> JVM + Standard Library

JVM = Java Virtual Machine

-> Class Loader ++++++ … +++

lombok

(1) Java Agent

(2) Java Library

변수 선언 = 타입 + 변수명;

변수 정의 = 타입 + 변수명 + 초기값의 할당;

JDK tools

(1) jconsole

(2) jvisualvm

\* &, && 차이 \*

Short-circuiting

- &는 첫번째든 몇번째든 false가 나오든 말든 싹다 확인하고

- &&는 첫번째든 두번째든 false가 하나라도 나오면 나머지는 확인도 안하고 false로 리턴.

\* 논리연산자 \*

(1) 논리곱(AND) : &, &&

(2) 논리합(OR) : |, ||

(3) 부정(NOT) : !

\* new 연산자 (객체생성연산자) \*

(1) 무조건 해당 클래스로부터 객체를 생성한다.(힙에)

(2) 뒤에 나오는 클래스의 생성자를 호출한다.

(3) 힙에 생성한 객체의 주소를 반환한다.

\* 생성자 \* new 연산자에 의해 호출될 때,

힙에 생성된 객체의 인스턴스 필드의 초기화.

객체가 갖고있는 필드의 초기화.

디버깅을 하려면 break point가 필요하다.

break point 란 실행을 잠시 중단 시키는 위치.

Thread - 실오라기 하나.

-

클래스 - 실행 클래스 (메인 메소드 유무 차이)

- 라이브러리 클래스

2020.08.21(금)

자바를 실행 시키면, run 하면, JVM이 실행 클래스에 메인쓰레드가 메인메소드를 실행시킨다.

클래스의 인스턴스

1. 클래스의 인스턴스화 : 클래스란 틀에서, 객체를 생성하는 과정.
2. 클래스의 인스턴스화를 통해, 생성된 객체를 해당 클래스의 “인스턴스”라고 부른다.

객체

해당 클래스의 인스턴스.

JVM이 사용하는 메모리 영역 = RDA

1. 메소드 영역 - 컴파일이 완료된 .class파일의 바이트 코드를 그대로 읽어다가   
   저장하는 메모리 영역 (런타임 상수풀, 필드 데이터, 메소드 코드, 생성자 코드)  
   메소드에 들어가있는 클래스들도 객체로서 들어가는데, 그 객체의 타입을 Java.lang.Class로 정했다. 그래서, 메소드 영역에 들어가있는 클래스들을 클래스타입의 클래스라고 안하고, 클래스 타입의 클래쯔(clazz)라고 한다.
2. 힙 영역 – new 연산자를 통해 만들어낸 객체들의 주소를 갖는다.
3. 쓰레드 영역(JVM 스택 영역)

각자 독립된 체계를 갖고, 각각의 쓰레드 마다 스택이라는 영역을 가진다. 그러나 스택영역이 가득 차서, 더 이상 값을 넣을수 없을 때, 오버플로우라고 한다.

그러나, 값이 넣을게 부족하거나, 없다면 언더 플로우라고한다.

쓰레드의 특이한 성질 – 무작위성. 각자 독립된 체계.

하지만 메인 쓰레드가 실행되면 다같이 실행되고 각자 해야 할 일들을 수행하고,

, 메인쓰레드가 끝나면 다같이 끝난다.

메인 쓰레드가 끝나면 JVM이 종료되기 때문에.

쓰레드 마다 스택을 갖고있다.

스택은 Last In First Out

쓰레드가 메소드를 호출할때마다(메인도 포함), 스택에 각각의 Frame이 생성된다.

Frame에는 메소드를 호출할때, 호출된 변수와 메소드 시그니쳐 가 들어가있다.

메소드 시그니쳐 -> Public static void abc()

{ int a = 0; }

스택에 메소드를 push 하고 pop 하며, 메소드가 호출되면 스택에 넣고(push), 메소드가 끝나면 스택에서 꺼낸다.(pop). 스택에서 꺼내진 이후 그 메소드와 그 메소드에서 사용된 변수와 시그니쳐가 다 사라진다.

스택이 비어버리면 JVM 종료.

Null은 참조할 객체가 없을 때! 넣을 수 있다. -> 참조변수만.

\* 배열 \*

배열은 한번 생성하면, 그 크기와 타입을 바꿀 수 없다.

New -> new 연산자 -> 객체 생성 연산자

. -> dot 연산자 -> 객체 멤버(필드/메소드) 참조 연산자

Traverse(순회)

\*기본타입마다 소위 Wrapper Type Class(or Boxed Type Class)가 존재.\*

1. int -> Integer
2. byte -> Byte
3. short -> Short
4. char -> Character
5. float -> Float
6. double -> Double
7. boolean -> Boolean

\*Auto-Boxing/Unboxing 기법\*

-ASDF 참고

클래스 뒤에 s가 붙으면, helper 클래스이다.

예를 들어, Array에서 Arrays가 있으면, Array를 도와주는 클래스이다.

JCF -Java Collection Framework

\* OOP – Object Oriented Programming – 객체 지향 프로그램. – 부품을 모아 조립해 완성하자\*

(1) 클래스 캡슐화

(2) 상속 다형성1

(3) 인터페이스 다형성2

OOP특징 세가지. 캡슐화, 다형성(다형성1, 다형성2)

1. 캡슐화 : 객체의 필드 메소드를 하나로 묶고, 실제 구현 내용을 감추는 것. 필드와 메소드를 캡슐화하여 보호하는 이유는 외부의 잘못된 사용으로 인해 객체가 손상되지 않도록 하기 위해서다.
2. 다형성 : 같은 타입이지만 실행 결과가 다양한 객체를 대입할 수 있는 성질   
   예를 들어, 타이어를 봤을 때, 금호타이어, 한국타이어가 있을 때, 동일한 규격과 크기여도, 성능이 다르다. 하나의 변수지만, 그변수 안에 넣는 값에 따라 결과가 다르다.  
   규격을 만든다 – 다형성1, 그 규격에 맞게만들고, 결과값이 다르다 -다형성2

Checked Exception (일반예외) -> 실행 전 자바가 오류를 찾아내는 것.

Runtime Exception (실행예외) -> 실행 중 오류가 뜨는 것.

* Jar 파일이란?
* 자바 언어로 만든 압축/ 해체 프로그램
* JDK의 bin 폴더 안에 들어있는 명령어
* xxx.jar파일이 있다면, 아래와 같이 관리한다.

1. Xxx.jar파일의 내용보기
   1. C:\temp> jar tvf xxx.jar
2. Xxx.jar파일의 해체(압축풀기)
   1. C:\temp> jar xvf xxx.jar
3. Xxx.jar파일의 생성(압축생성)
   1. C:\temp> jar cvf xxx.jar 파일1 파일2 디렉토리1 디렉토리2

필드의 내용(예시 차.)

* 객체의 고유데이터
  + 모델, 색깔, 제작회사, 최고속도
* 객체의 현재데이터
  + 현재속도, 엔진회전수
* 객체의 부품데이터(다른클래스에서 객체 생성)
  + 차체, 엔진, 타이어

매개변수(parameter) -중개 역할 –method가 호출될 때, 안에 코드안에 중개 해준다.

전달인자(argument) -매개변수에 실제 넣어주는 값.

어떠한 매개변수에 10이란 전달인자를 준다.

Return 키워드의 역할

1. 리턴이 수행되는 순간. 무조건 실행을 종료 시킨다.
2. 리턴 뒤에 나오는 결과값을 호출자에게 준다.