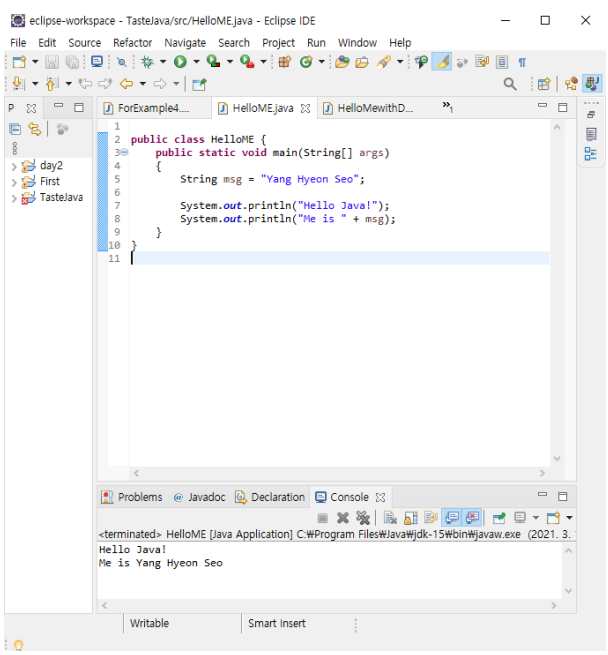
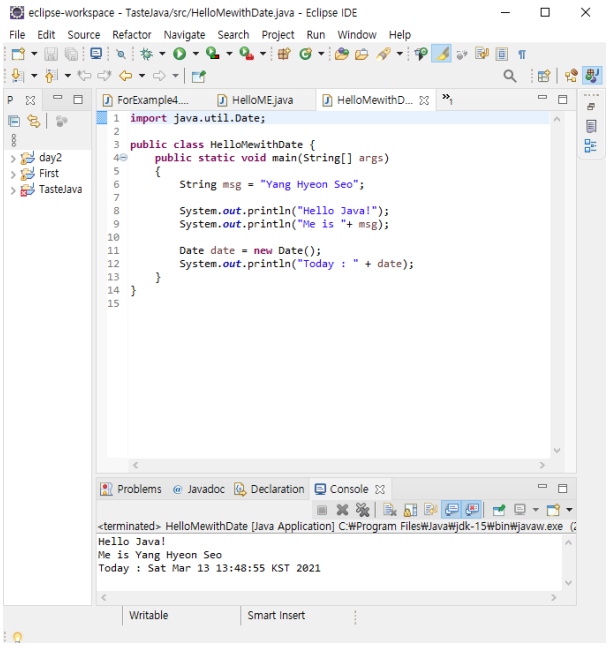
객체지향프로그래밍 2 주차 정리

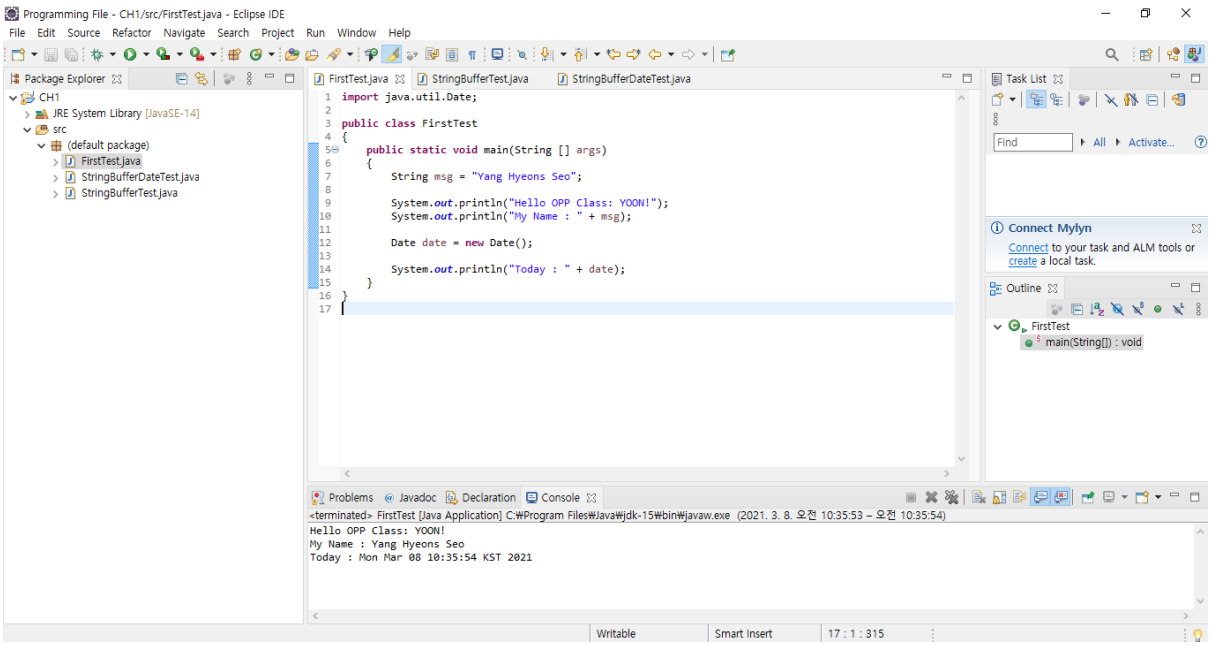
프로그래밍

1.7 taste Java : Lab1

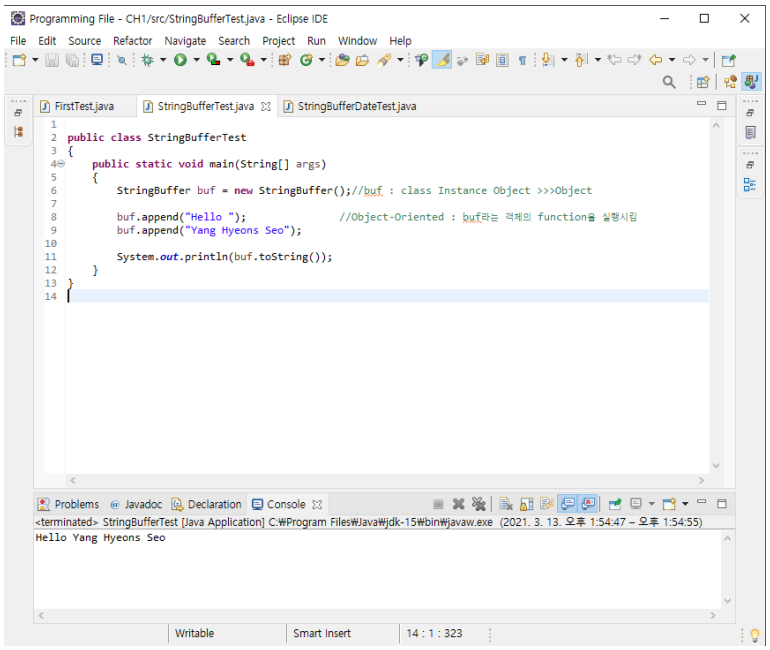




수업시간에 한 프로그래밍(FirstTest)



1.8 Java API : Lab2



buf :

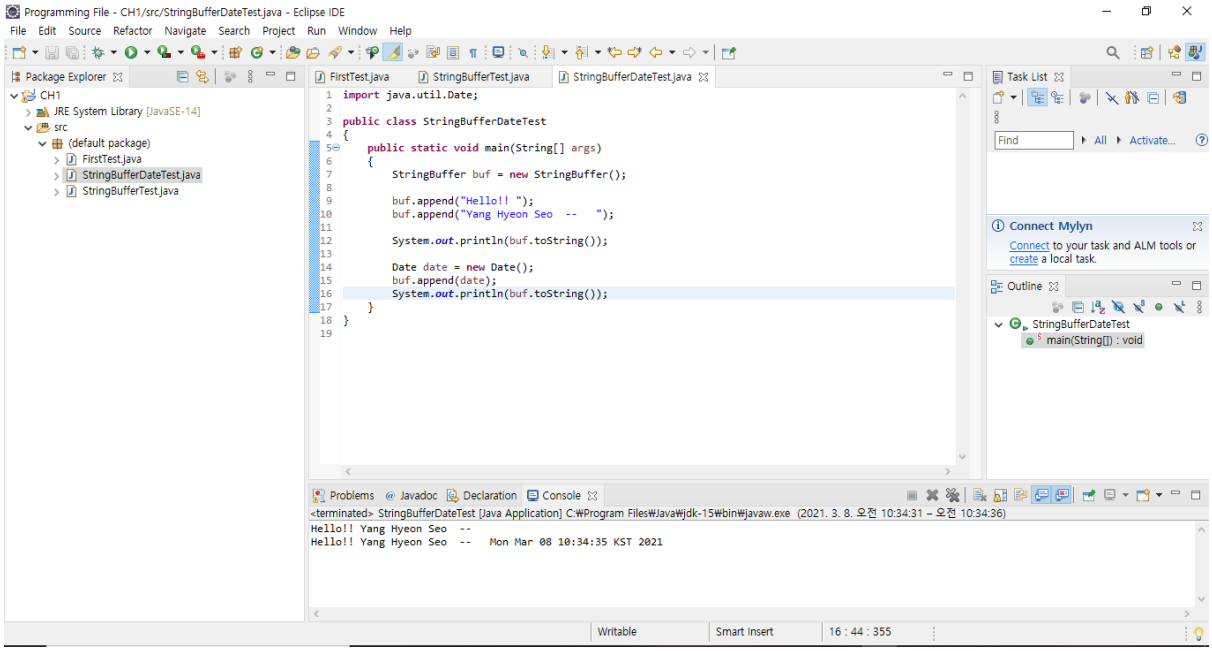
클래스 인스턴스 객체

>> 객체 생성

.append :

buf 라는 객체의 function 을 실행시킴

>>객체지향



Public : 공유할 때 필요

패키지를 사용할 때는 무조건 필요하다

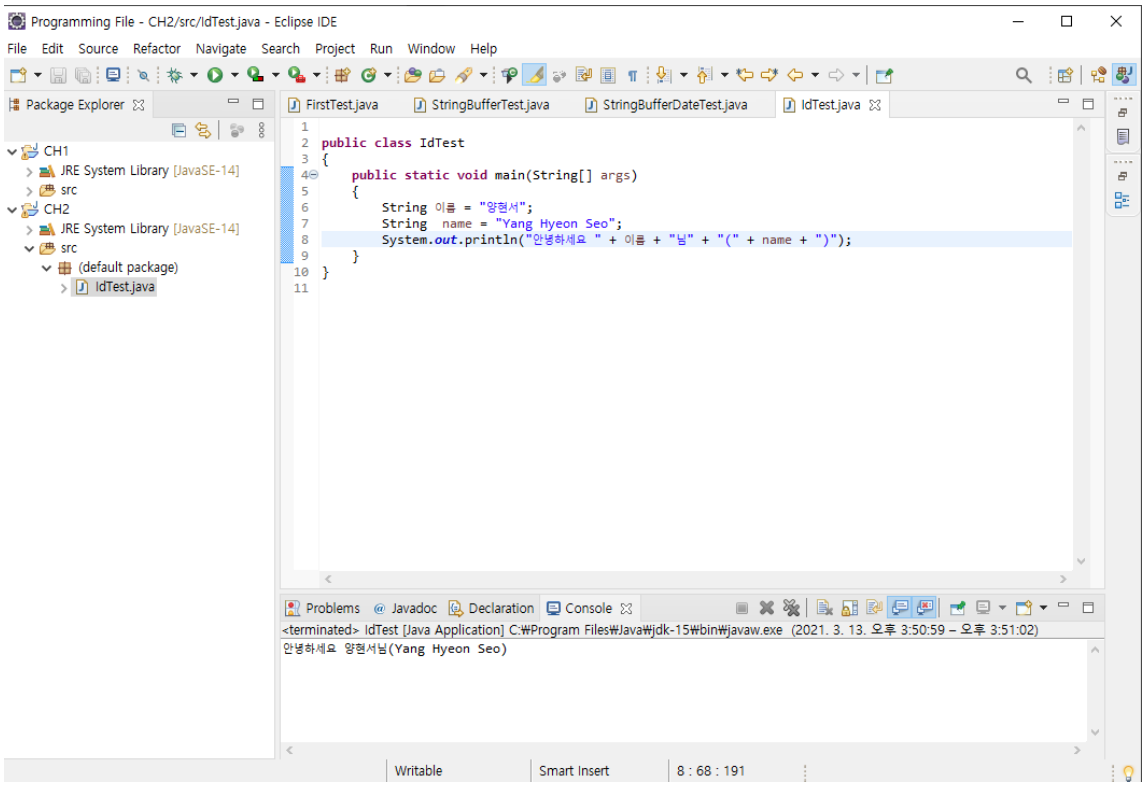
System > 화면 의미

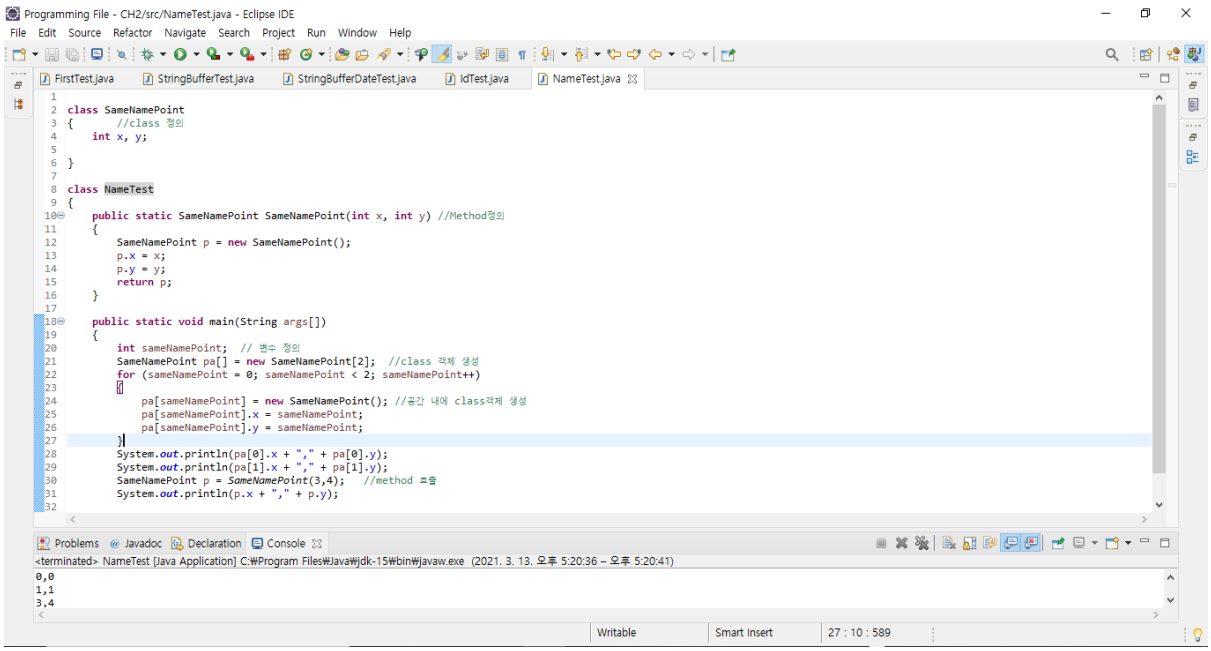
화면으로 받을 때 : System.in

화면으로 출력할 때 : System.out

프로그래밍

2.1 Java Identifier : Lab2-1





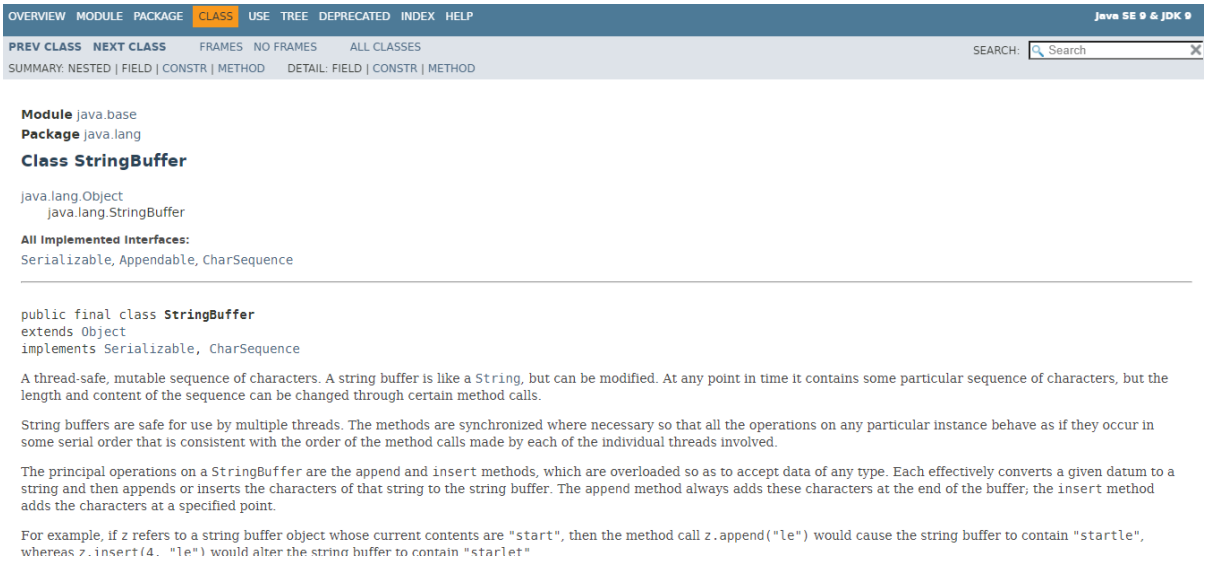
Java API

API : 자바 구현을 쉽게 하도록 한 클래스 라이브러리 집합

https://docs.oracle.com/javase/9/docs/api/java/lang/package-summary.html

패키지 안의 클래스와 메소드를 확인할 수 있고, 설명도 볼 수 있다.
객체지향에서는 클래스와 클래스의 위치를 아는 것이 중요한데 이는 프로그래밍을 하면서 쌓아갈 수 있다.

StringBuffer


OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP Java SE 9 & JDK 9

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES SEARCH: Search

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Module java.base
Package java.lang

Class StringBuffer

java.lang.Object
  java.lang.StringBuffer

All Implemented Interfaces:
Serializable, Appendable, CharSequence

public final class **StringBuffer**
extends Object
implements Serializable, CharSequence

A thread-safe, mutable sequence of characters. A string buffer is like a String, but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.

String buffers are safe for use by multiple threads. The methods are synchronized where necessary so that all the operations on any particular instance behave as if they occur in some serial order that is consistent with the order of the method calls made by each of the individual threads involved.

The principal operations on a StringBuffer are the append and insert methods, which are overloaded so as to accept data of any type. Each effectively converts a given datum to a string and then appends or inserts the characters of that string to the string buffer. The append method always adds these characters at the end of the buffer; the insert method adds the characters at a specified point.

For example, if z refers to a string buffer object whose current contents are "start", then the method call z.append("le") would cause the string buffer to contain "startle", whereas z.insert(4, "le") would alter the string buffer to contain "starlet".

쓰레드로부터 안전하고 변경 가능한 문자 배열
String과 유사하지만 수정할 수 있고 어느 시점에서든 배열의 길이와 내용을 특정 메서드를 호출하면서 변경할 수 있다.

소스 시퀀스와 관련된 작업이 발생할 때마다(예 : 소스 시퀀스에서 추가 또는 삽입) 이 클래스는 소스가 아닌 작업을 수행하는 문자열 버퍼에서만 동기화된다.
StringBuffer여러 스레드에서 동시에 사용하는 것이 안전하도록 설계되었지만 생성자 또는 append또는 insert작업이 쓰레드 간에 공유되는 소스 시퀀스를 전달하는 경우 호출 코드는 작업동안 일관되고 변경되지 않아야 한다.
호출 코드는 호출자가 작업 호출 중에 잠금을 유지하거나 변경 불가능한 소스 시퀀스를 사용하거나 쓰레드간에 소스 시퀀스를 공유하지 않음으로 유지할 수 있다.

모든 문자열 버퍼에는 용량이 있다.
문자열 버퍼에 포함된 문자 시퀀스의 길이가 용량을 초과하지 않는 한 새로운 내부 버퍼 배열을 할당할 필요가 없고, 내부 버퍼가 넘치면 자동으로 버퍼 용량이 커진다.
(출처 : J2SDK API Documents (SE14) – StringBuffer)

객체지향 프로그램의 기본 ★

Package 패키지 > import 할 때 ex)import 패키지이름

Class 클래스 > 선언

Constructor 생성자 > 생성

Method 메소드 > 실행

흔히 실수하는 부분(JAVA 의 특징)

1. **대소문자를 구분한다**

   클래스 이름

   대문자 명사로 시작한다

   '_' 사용하지 않는다

   동사를 사용한다면 동명사가 좋다

   메소드 이름

   소문자 동사로 시작하고 그 뒤로 명사(목적어)가 대문자로 붙는다

   Ex) getName

   변수 이름 (멤버 이름)

   소문자로 시작한다

2. **한글 이름 디렉토리를 인식할 때가 있고, 인식하지 못할 때가 있다**

   >> 오류 발생 가능성

3. **환경 변수를 잘못 설정**
4. **항상 main 이 존재**
5. **파일 이름 = 클래스 이름**
6. **파일 이름 나들 때 JDK 가 가지고 있는 클래스의 이름을 사용할 수 없다**

   >import 시에 충돌 발생

7. **클래스, 메소드, API 를 모두 기억하기는 힘들다**

>프로그래밍을 많이 하면서 기억해나가면 된다.

JAVA 의 기본 구조

package 패키지 이름;

>>package : 프로그래밍한 것들을 하나의 솔루션으로 묶음

import 다른 패키지나 클래스 이름;

public class 클래스이름{

>>public : 패키지에 들어가면 공유해야 하기 때문에 사용

실행할 문장들;

>> ClassName() : 생성자

ClassName 이라는 클래스의 인스턴스를 만든다

}

언어 동작 방식

컴파일러

소스파일 –(컴파일러)->오브젝트 파일(.obj)

오브젝트 파일(.obj) + 라이브러리 파일(.lib) –(linker)->실행파일(.exe)

인터프리터

한줄한줄 읽으면서 코드를 분석, 실행

시작 —>Get Instruction —> Tokenize(단계별로 쪼개고) & Parse(분석)

—>Execute Instruction (—>Get Instruction 으로 돌아간다)

자바

컴파일러 + 인터프리터

JAVA Program(.java) –(Javac)->클래스파일(.class)

바이트 코드 상태

-(Java)->JVM(Java Virtual Machine)

시작 —> Get Instruction —> Execute Instruction (—> Get Instruction 으로 돌아간다)

바이트 코드

버추얼머신이 실행하는 명령어의 현태

각 1byte 를 소비하지만 여러 개의 파라미터가 사용되는 경우가 있음

자바 바이트 코드를 이해하는 것은 C 개발자가 어셈블리어를 이해하는 것과 같다

Run-time Behavior 방식

실행 시 필요한 변수와 공간을 확보하는 방법

기본적으로 동적 메모리 할당 방식을 사용한다.

C

Compile-Behavior 방식

컴파일시 기본적으로 필요한 공간과 변수를 확보한다

malloc() : 특별한 경우 동적 메모리를 할당하기 위한 함수

Java Identifier(구별자)

클래스 이름, 메소드 이름, 변수 이름

이름 규칙

문자와 숫자의 연속

문자로 시작

키워드는 이름으로 사용할 수 없다

true, false, null 도 키워드 중 하나로 이름으로 사용할 수 없다

'_'와 '$' 사용 가능

대소문자를 구분한다

C 언어와 달리 길이 제한이 없다

얼마든지 길게 쓸 수 있다

한글 변수도 생성이 가능하나(한글 Id 가능) 컴퓨터마다 다르게 인식한다

>비추

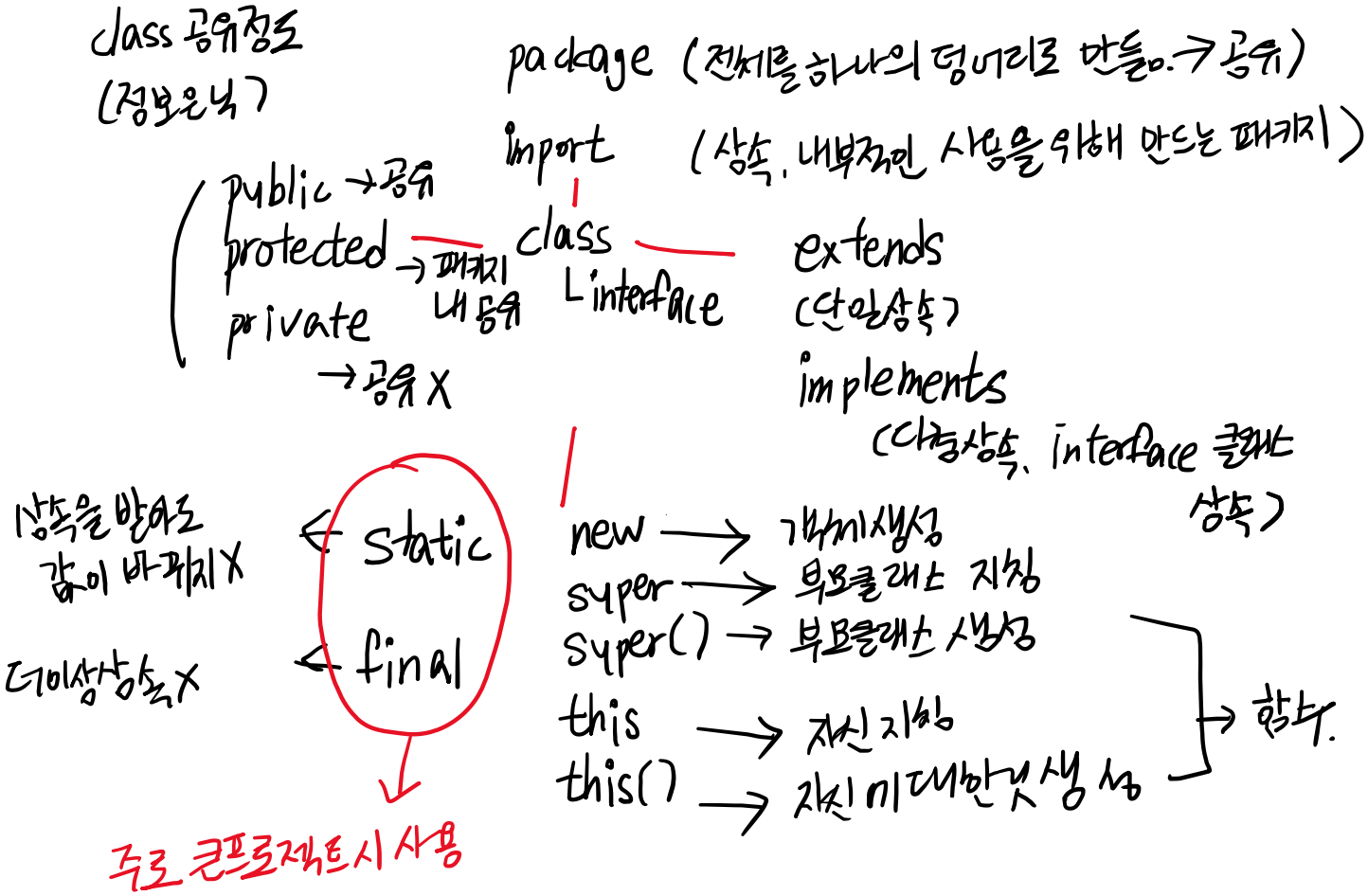
클래스의 이름, 메소드의 이름, 변수의 이름을 똑같이 만들면 안된다

프로그램을 작성하면서 작성자의 논리가 꼬일 가능성이 크다

JAVA 키워드

class

객체 지향은 class 로 시작해서 class 로 끝난다



Encapsultion :

Class ⊃mathod, field(변수)

캡슐에 묶어준 것, 클래스를 만드는 과정

Inheritance :

Extends, implements

상속받을 대상(클래스) 지정

super

상속받는 것 찾는 것, 상속 대상 지칭

static

final

상속을 계속할지(허용할지)말지

상속에 대한 권한

Information Hiding :

public, protected, private

현재 클래스에서 사용할 객체를 생성 : new

this

자기 자신을 지칭(현재 객체)

this()

자기 자신 생성(현재 객체)

예외처리

try, catch, finally (1 개의 패턴)

thorws

객체에서 예외 발생

예외 정의

Ex) throws ExceptionType

Throw

예외 발생

자바 cmd 로 실행하는 방법

1. **파일 위치를 이동한다**
2. **Javac 파일이름.java**
3. **Java 파일이름**