ECE20016-03

자바 프로그래밍

[Lecture 1](#_1vx8z77ownkc)

[Who am I?](#_s2hchq1n2g2f)w

[Course outline and what to do?](#_qxo9j6mkozig)

[What is this course for?](#_mwpa7qm0q71s)

[Object oriented programming (OOP)](#_wabw4i7nwkjx)

[Java is everywhere!](#_jd2r29bxkpf5)

[Did you learn about tools for build tools or source code repositories?](#_7hbb4quodux2)

[Grading policy](#_b53izobo663q)

[숙제 기대치](#_10xqtiid7ua2)

[What is a program in a broad view?](#_yvahqa7bzkg9)

[What is input?](#_8dvc8zj7x61u)

[What is a program?](#_6ns3oop8phuk)

[What is output?](#_s7pt6zx003pg)

[‘Input, Output, Program에 어떤것이 있을까?’ 에 대한 학생들의 의견](#_npi88yuj9mps)

[Discussion](#_kzv28t8mb6ws)

[How Program works in the view of input and output?](#_kfs2prmqldvb)

[What do we need to write a program?](#_h5isn0cls1vq)

[Programing languages in different levels](#_4gzgku1jrpk)

[Feedback about Lecturing based on the google doc](#_yyqdlsaiasmq)

[Lecture 2](#_j6mgmfq1l42)

[Announcement](#_tyro0292v5im)

[How to generate an executable C program?](#_o3smyiqpzwzg)

[How to generate an executable Java program?](#_y7t0ytx7p6bs)

[What is a virtual machine?](#_9m6im94bgifq)

[What is Java Bytecode?](#_jbuk7n389li8)

[‘Virtual machine과 Bytecode가 무엇일까?’ 에 대한 학생들의 의견](#_tlvwjae0ge6v)

[Discussion](#_b79dvn49qs00)

[학생 질문](#_axahuh9rvdqa)

[Machine Language for JVM? JVM has a CPU?](#_vvxto2qp8qyy)

[일반적으로 프로그램이 돌아가는 환경](#_a34tutacvonv)

[자바가상머신(JVM)에서 프로그램이 돌아가는 모습](#_oj7cogez9l25)

[How Java program runs on a computer?](#_72dbngaqkx99)

[Can you find disadvantage of Java?](#_5jqyypbb31se)

[Can you find advantage of Java?](#_2vmsvh4onzi)

[Lecture 3](#_ocl0rjscfkqd)

[Announcement](#_2b7jufbhcrr6)

[Our First Java Program](#_s8243u2hpx5g)

[This Java code is consists of….](#_wy1vtqg6vx3t)

[What are they????????](#_dg46yaigr80n)

[What is Java programming philosophy?](#_fzse6lhojygh)

[Concepts of Object-oriented programming (OOP)](#_huyj2epfu42w)

[Object-Oriented Programming](#_l340mv9zccov)

[What are packages, Objects, and instances?](#_65gjv0wasci3)

[What is a Class?](#_3m9snvmc7kgq)

[What are Modifiers?](#_nwr9kk7rvy0w)

[Lecture 4](#_aligw3mvpq6b)

[Announcement](#_ci22bzucgaej)

[Typical Steps to write Java code in Eclipse](#_4pxwdwilmxam)

[5. Define a class name and its block](#_eb8ldwe50j99)

[Access levels (modifiers) of a class](#_3g1yxde0piuv)

[6. Implement a class](#_ynn417u59rr7)

[Add an instance variable](#_3y5no2vdqr68)

[Add a method](#_yhyat7vwux68)

[How to define a method?](#_2knzr5f7oi1f)

[Access levels of a method](#_sxwavd2txzi7)

[7. Create a main method in the class that starts our program](#_uqmzatsevb6h)

[8. Instantiate of an object](#_5e0zoffjndjq)

[9. Call a method of an instance (Method invocation)](#_5rubltak5ofb)

[Let’s outline for Smart Classroom](#_9gzw92ult0y0)

[Lecture 5: Language Basics](#_jst1dxic94gy)

[Announcement](#_eg6lfhs4mznk)

[Programming conventions we must/should follow (1)](#_fot2cpp4p11q)

[A magic of a static modifier](#_yk19hd6uasig)

[Variables and variable types](#_ko5sijxs6g90)

[The kinds of variables](#_n0utpukz1fec)

[Naming conventions](#_utcidn73yag)

[All related topics in the use of variables](#_g5k01nmn9nmg)

[Constants and named constants](#_q7yrnqj8ev3h)

[Levels of Definition](#_7t4m7b6x3u0m)

[Lecture 6: Language Basics (2)](#_8lxkn1yrwqc7)

[Announcement](#_olqq1nd6raxm)

[Keyboard and Screen I/O](#_ggnjvvsefw69)

[Documentation and Style](#_ji1frjetmct)

[Related topics in the defining classes and methods](#_nndbnfde0lk0)

[Design my object](#_gnh2shpih6qq)

[Information Hiding](#_ciranye90j1s)

[Encapsulation](#_h087e3e2kkx7)

[Lecture 7: Branching](#_1jofuwj5urm3)

[Flow of Control](#_sr7jdkk7umsf)

[Lecture 8: Branching and Language Basic (3)](#_hs0g1df8z7t9)

[Announcement](#_80fk3rqv6d0h)

[Type Boolean and Switch](#_n2phxjp6ewks)

[Object and References](#_huyv0hardz1)

[Software Repositories (Git and Github)](#_yzzu5shciii0)

[강의 7, 8 내용 요약](#_119icucyerst)

[Lecture 9: Build Tools and Loops](#_7xocbeqpqibi)

[Announcement](#_pvmyahst7xgm)

[Build tool: Gradle](#_gzy5hanqkz1w)

[How to use git in Eclipse](#_9lwl5j29k6fe)

[Flow of Control: Loops (1)](#_2tnh9pi4akne)

[Lecture 9 요약](#_ly3h874esu17)

[Lecture 10: Loops and More about Objects and Methods (1)](#_62zgh174we2v)

[To initiate your project using git, Github, Gradle and Eclipse](#_jhmeeyfa7k2j)

[Flow of Control: Loops (2)](#_zdf1bhuuk48i)

[More about Objects and Methods (1)](#_k7uwt9p430lw)

[Lab07-HW1-4](#_m6nx3d9ir5cm)

[Lecture 11: More about Objects and Methods (2) and Arrays](#_hq01fpmdvnm9)

[More about Objects and Methods (2)](#_p40yrozbdg6l)

[Overloading](#_li7zocevf2c7)

[Method signature?](#_yxicnh9k5lk8)

[Enumeration as a class](#_ezz9so2hkgfq)

[Packages](#_bl5qpbrfxf3o)

[Arrays](#_dn9evgswbiy8)

[Lab08](#_ogbzi86wzavd)

[Lecture 12: Arrays (2)](#_xe6ewuyguv30)

[Announcement](#_nedqb8jidob7)

[Arrays in Classes and Methods](#_c4fys3ob2nn6)

[Case Study: Sales Reporter](#_37ms3e7i5z9f)

[Lecture 13: Arrays (3)](#_hhot6iv0kyha)

[Announcement](#_aj0mdxybtf6d)

[Sort Arrays](#_sha10x4b1vxw)

[Multi-dimensional arrays](#_6ds5mgy6k021)

[Lab10](#_m88buw1uk0c2)

[Lecture 14: ArrayList and HashMap](#_i6usl5adu21v)

[Announcement](#_fur0zvcw7wdt)

[ArrayList](#_5o3gfoccpmuh)

[HashMap](#_nmuc0i7zns2u)

[Lab11 and HW1-6](#_boeld018trat)

[Lecture 15: Inheritance, Polymorphism, and Interfaces](#_2orm9hme7tvo)

[Announcement](#_h77zsborpnwg)

[Inheritance](#_s7azcxqcehhv)

[Extends](#_sza03fqc0xkd)

[Super](#_djkdw8y5vc4g)

[is-a vs. has-a relationships](#_xxpq65c6t8og)

[The ultimate ancestor, Object](#_s97zqmnxpuvk)

[Lecture 17: Inheritance, Polymorphism, and Interfaces (2)](#_5s4k5pnk3zob)

[Polymorphism](#_24081ee9wro)

[객체는 항상 어떤 method action이 이름에 따라 실행이 될 것인지 결정 한다](#_om8js2tpjfgz)

[Interfaces](#_m670kr7t0r66)

[Lecture 18: Interface (2)](#_cpbwv8vx0mr0)

[Announcement](#_ignx8nslfrnv)

[InstanceOf](#_bxmw8uhm958)

[Interfaces (2)](#_vc25ebqop785)

[Lecture 19: Abtract Class. Exception Handling](#_3vwnnbo5bidw)

[Announcement](#_1s0vngjy8z4y)

[Abstract Class](#_sdopk7qjiokn)

[Exception Handling](#_rans3gl8el7v)

[Lecture 20: File I/O](#_k0mvnpgkt598)

[Announcement](#_fccmmudarxqr)

[Text file을 읽어보자.](#_pdc86c4hkc7a)

[Lecture 21: File I/O (2)](#_p2dwrnu2lnol)

[Binary File I/O](#_h8gp2xktw7yq)

[Lecture 22: Dynamic data structures and generics (2)](#_zhign82cy06m)

[Dynamic Data structures](#_oy2w6asuox11)

[Hashset](#_ikpwuuoji223)

[Linked List](#_zfoiz8z3t163)

[프로그램에 대한 간단한 설명과 앞으로의 방향](#_tedlqmtac5e)

[다시 LinkedList](#_kycastue11od)

[HW3 Class diagram](#_jv8ojqcgbwnh)

[Lecture 23: Generics (3)](#_rmek39693t05)

[Generics](#_nesz4qapjqh4)

[Designing and Coding for HW3](#_brj5apucf506)

[Lecture 24: Practical topics for Java programming](#_i22pr1lr9uxu)

[Announcement](#_5ic0g9eppjlz)

[Gradle distZip](#_4bso1bqhmfbc)

[Import an external library via Gradle](#_1kyfezcmsaid)

[Apache Commons CLI](#_17efhpf33gya)

[Regular Expression](#_bo2we7e6ad55)

[Lecture 25: Recursion](#_kwt7odctvwzn)

[Announcement](#_k1x3yujetnxr)

[Recursion](#_6ticxa3cyac7)

[Lecture 26: Q&A and discussion for HW3](#_6t3kn08vm3jc)

[Announcement](#_e5skdmq8t6ji)

[Issues](#_ndbpa5en2ges)

[Lecture 27: Concurrency (1)](#_gotb464vaur0)

[Announcement](#_2akkmrvrxi0s)

[Lecture 28: Concurrency (2)](#_wu4rdzghxd)

[Announcement](#_erh8uxurdopw)

[Thread Pools](#_qft4y39qenyt)

[Other topics in threads](#_xlvgwxxogfqk)

[Synchronization](#_ejva2idh4wgo)

[Deadlock](#_6oe8kqg31pip)

[Lecture 29: Summary](#_6x6ssrq8dakt)

[Announcement](#_afrzzgwevo3d)

# Lecture 1

(Mon/Thu 8:30-10:15 only for the first lecture. NTH 413. Other lectures will begin at 9:30 and end at 11:15.)

Put your name in the box below: (you can do it by “Suggesting” option)



| 남재창 | **김대교** | **김재윤** | **최성현** | **유정연** | **이현지** |
| --- | --- | --- | --- | --- | --- |
| **이준범** | **송창우** | **김미소** | **이수아** | **배재호** | **김도영** |
| **정희석** | **임성빈** | **최동현** | **양수진** | **임예원** | **한정섭** |
| **강동인** | **김태현** | **채정영** | 김기범 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Q&A (Write anything you want to know before the first lecture)

1. 한동대 오시기 전에 무엇을 하셨나요?

캐나다 워터루 공대 포닥, 포항공대 연구교수 했어요.

2. 이 수업을 통해 학생들이 얼마만큼 성취하실거라 예상하시나요?

자바로 뭐든 만들 수 있다는 자신감 얻을 수 있으면 제일 좋겠어요. 그런데 학생들 마다 편차가 좀 있겠죠?

아아….. 네.

3.

질문이 너무 거창 했던것 같습니다. 자바는 배우기 수월한 언어인가요? 언어를 이런거 저런거 배워가고 있는데 나중에 ㅎ

자바가 C보단 쉬워요.ㅎㅎ 헷갈리는 건 걱정 안해도 되요. 컴파일 할 때 틀린거 다 잡아주니...

555! 네…….

감사합니다.

...

## Who am I?

남재창!!!! JC

POSTECH Research Assistant Professor

University of Waterloo, Canada. Postdoc

PhD., Hong Kong University of Science Technology, Hong Kong

MsC , Blekinge Institute of technology, Sweden

BM,BE, HGU, 98

<https://lifove.github.io>

Lifove Bible (my first Android app published)

## Course outline and what to do?

Check Slides: <https://docs.google.com/presentation/d/1ic4xV9nv4NNIcS9938ghZhKZHJK6OYnRrS6gDW-f5IU/edit#slide=id.p3>

## What is this course for?

Java

One of programing language?

List programming languages you know:

| C, C++, C#, Script R  Python, D |
| --- |

## Object oriented programming (OOP)

The most important concept you have to understand.

Java defines an Object first in the source code.

Object can be anythings in the class.

## Java is everywhere!

| 게임 (게임 메크로)  서버  웹/웹사이트 (JSP, Java servelet)  안드로이드 스마트폰 앱  위젯  ... |
| --- |



This image shows the backside of a bluray player! Many electronics such as refrigerators, washing machines, etc. implement firmwares in Java!!

## Did you learn about tools for build tools or source code repositories?

깃헙을 관련해서 따로 배우는 강의가 있어요

빌드는 안배운거 같아요

저요

회원가입만 하고 어려워서 안했어요.

C배웠던 기억이 거의 사라졌어요… 빌드를 배우긴 했는데

파이썬배울때 깔아봤어요

## Grading policy

출석 : 5%

숙제: 50%

중간: 20%

기말: 25%

Extra Credit: 3%

Google doc 강의노트 진행이 학기 계속 이루어 질 경우에 적용(몇 번 이렇게 강의해 보고 안좋으면 슬라이드 강의로 전환할 것 임.)

## 숙제 기대치

숙제 하나 당 평균 10시간 코딩 할 수 있는 Task 부과할 예정

문제 해결 능력이 커질 것으로 생각해 난이도와 요구 코딩량이 증가함.

## What is a program in a broad view?



-> 광범위한 의미로 프로그램의 역할은, 입력값을 원하는 결과의 출력값으로 바꿔주는 것이다

### What is input?

* Keyboard input
* Mouse input
* Numbers
* I7 login id and password
* …

### What is a program?

* A sequence of instructions!

### What is output?

* Text on screen
* Mouse move
* Numbers
* Move to the first page of i7
* …

### ‘Input, Output, Program에 어떤것이 있을까?’ 에 대한 학생들의 의견

| Input  -키보드  -입력장치  -스피커모니터  -마우스  Output  -프린터  -모니터  -스피커  Program  -데이터를 가공하는 코드들  -컴퓨터 cpu  -OS  -중앙처리장치  Etc  -소프트웨어와 하드웨어 |
| --- |

### Discussion

Q) 카카오톡 채팅방에서 사람들의 채팅 횟수를 카운트하고 알려주는 프로그램을 만든다면, Input은 무엇이고, Output은 무엇일까?

A) Input : 채팅한 사람들의 이름과 채팅 시간이 적힌 CSV파일

Output : 사람들의 이름과 채팅의 횟수가 기록된 CSV파일

## How Program works in the view of input and output?



1. A Program is loaded into memory
2. The loaded program is executed sequentially by CPU.
3. CPU moves data to other locations in memory, HDD, displays,...

## What do we need to write a program?

Any languages you know?

C

C++

Java

Python

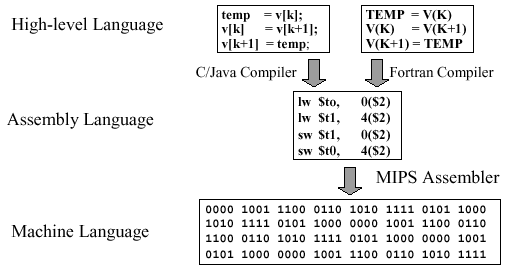
C#

…

⇒ High-level language

(High-level language란 무엇인가?)

## Programing languages in different levels



High-level Language == Human-readable language

Assembly Language (Low-level language) == less human readable

Machine Language == Original language for CPU

| Exam Question: Discuss about different levels of programing/computer languages  Answer:  High-level Language : 사람이 읽기 쉬운 고급 언어, 한 번의 명령으로 ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,CPU가 할 수 있는 여러가지 일이 가능  Assembly Language : 사람이 조금은 이해할 수 있으나 컴퓨터 친화적인 ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,낮은 단계의 언어, CPU 처리할 수 명령이 특정한 기호로 1대1 대응되어 표현 됨.  Machine Language : 0과 1, 2진법의 원초적 컴퓨터 언어 |
| --- |

## Feedback about Lecturing based on the google doc

| 교수님 글은 수정불가로 놓는게 좋을 거 같아요.  구글닥이 괜찮네요 생각보다  카카오톡이요.  말안하는거보다 써서 수업에 많이쓰면 시간이 많이걸릴듯한데, 질문이나 대답할때 쓰긴 참 좋은 것 같아요.  시간을 단축할 수 있는 방법을 조금만 더 고려해보면 괜찮은 수단이라고 생각해요!  아 오픈 카톡방도 괜찮을거 같아요!!  혹시 수요일 7시에 보강을 |
| --- |

# Lecture 2

## Announcement

1. 보강 7시 반은 안될까요ㅠㅠㅠ 보강을 놓치면 많이 힘들 거 같아요...ㅠㅠ

>> 다음 수업시간에 조정 해볼게요. 채플 7교시 학생들 채플 끝나자 마자 30분 이상 식사시간 가질 수 있을 것 같기도 하네요. 그런데 7교시 채플은 18:45에 끝나요? 19:00에 끝나요???

3/1 → **3/14** (Wed) 19:30

5/7 → 5/16(Wed) 19:30

~~5/10 → 5/23(Wed) 20:00~ >> can be changed in to 19:30~~~

축제 때 수업을 하면 좋을 것 같은 때, 보강 vs. 원래대로?? **축제 때 수업**

2. 각 강의 노트를 정리하고 Exam Question에 답변을 달면 Extra Point 3점 1점을 줄 예정. 한 학기에, 1~2번만 할 수 있음. 관심 있는 사람 어느날짜 강의 노트 정하고 싶은지 메일 [jcnam@handong.edu](mailto:jcnam@handong.edu) 보내기 바람. <https://goo.gl/51QGK8>

3. 오늘 처음 오는 사람. HisNet 공지사항에 강의노트, 슬라이드 자료, Kakao 채팅방 입장하는 정보 나와 있으니 참고 바람.

4. 10:45분 부터 실습하니 시간 되면 누가 좀 알려주길…

5. Kakao 채팅 방은 필수로 가입해야 하고, 굳이 메시지는 안보내도 됨. 수업중에 토론할 때 사용하면, 반응 수 계산해서 Extra Point 줄 예정임. 실명 안써도 됨. 대신 학번이랑 이름 연결하는 정보를 알려줘야 나중에 점수 계산 할 수 있음.  
연결 → <https://goo.gl/Z6rcK5>

Lecture Slides: <https://goo.gl/QdpNTe>

## How to generate an executable C program?

C Source code files

→ [Compile]

→ Object code files + C libraries (assembly + machine language)

→ [Linking]

→ An executable binary file

→ An executable binary file is run on a computer!

Slide: <https://docs.google.com/presentation/d/1d7jrg6dYWLeXOb3RasLw8yOsSLjGwEHNyD9BlpHoWsw/edit#slide=id.p19>

## How to generate an executable Java program?

Java source code files (\*.java)

→ [Compile]

→ Bytecode files (\*.class)

→ Bytecode files are run on a virtual machine!

→ Virtual machine runs the bytecode on a computer!

| Exam Question: Explain how Java source code file is finally executed in our computer? |
| --- |

Java 랭귀지의 생소한 용어

* Bytecode
* Virtual machine
* .class(바이트코드의 확장자명)

추가문제

Q. C언어와 자바언어는 interpret과정이 각각 몇 번 있는가?

1. C언어는 작성한 소스코드를 컴파일과 링크(빌드)를 통해 컴퓨터가 이해하도록 한다. 한 번의 통역과정이 있다.

반면 자바는 소스코드를 컴파일해서 바이트코드를 생성하고 그

바이트코드는 다시 JVM에 의해 컴퓨터가 이해하는 언어로

통역된다. 총 두 번의 통역과정이 있다.

## What is a virtual machine?

Software that implements virtual computer on a physical computer. For example, Oracle Virtualbox.

Java Virtual Machine (JVM) is a virtual machine for running Java programs.

| Exam Question: What is a virtual machine and a Java Virtual Machine? |
| --- |

## What is Java Bytecode?

Machine Language for the JVM.

Slide: <https://docs.google.com/presentation/d/1d7jrg6dYWLeXOb3RasLw8yOsSLjGwEHNyD9BlpHoWsw/edit#slide=id.p14>

## ‘Virtual machine과 Bytecode가 무엇일까?’ 에 대한 학생들의 의견

| Virtual machine  -가상기계  -임베디드 시스템이 들어간 기계  -물리적으로 존재하지 않지만 프로그램적으로  존재하는 기계  -에뮬레이터  -dev app  Bytecode  -소스코드  -기계언어  -1바이트 코드  -virtual machine을 위한 코드 |
| --- |

=> 바이트코드는 JVM만 알아들을 수 있는 기계언어, JVM이 바이트코드를 운영체제가 알 수 있도록 interpret해준다.

| Exam Question: What is Bytecode? |
| --- |

## 

## Discussion

Q1) Java에서 assembly code를 볼 수 있다?

A) 있다!

Q2) 그렇다면 assembly code를 보는 과정이 필수적인가?

A) 필수가 아니다. But, 프로페셔널들은 최적화(optimization) 위해 이용한다.

A2) 최적화는 컴파일러가 소스코드를 어셈블리어 또는 기계어로 바꾸어서 만들 프로그램을 사람이 직접 수정해, CPU실행과 메모리 사용을 효율적으로 하도록 하는 작업이다.

### 학생 질문

Q) 수정이 자주 필요한 프로그램(최적화를 요구하는)은

C보다 Java가 유용한가요?

A) 요즘같이 하드웨어가 좋은 환경에선 Java의 이식성과 간편성이라는 장점 때문에 C보다 사용함에 있어 낫다고 본다.

하지만 IoT 장비나 임베디드 장비같이 하드웨어가 좋지 않을 땐 C가 낫다. (참조 http://jejuq.tistory.com/30)

## Machine Language for JVM? JVM has a CPU?

### 일반적으로 프로그램이 돌아가는 환경



### 자바가상머신(JVM)에서 프로그램이 돌아가는 모습



## How Java program runs on a computer?

# 

| Exam Question: Discuss how Java program is executed on a computer (Draw the whole process as a figure and explain what each entity means)  Answer:  소스코드를 작성하고 자바를 이용해 컴파일을 한다. 소스코드는 바이트 코드로 변환된다. 그리고 JVM이 바이트 코드를 운영체제가 알 수 있도록 통역한다. 그 후 실행이 되어 자바 프로그램의 아웃풋이 나온다. |
| --- |

## Can you find disadvantage of Java?

Slow because of the Byte interpreter in JVM.

(Cf. C program: executable binary → [Execution])

Java: bytecode → [interpreter to machine code] → [Execution]

This disadvantage is not a big problem because H/W is fast enough.

## Can you find advantage of Java?

Bytecode is the same for all JVMs running on Windows, MacOS, Unix,...

This is PORTABILITY of Java programs.

Slide: <https://docs.google.com/presentation/d/1d7jrg6dYWLeXOb3RasLw8yOsSLjGwEHNyD9BlpHoWsw/edit#slide=id.p17>

| Exam Question: Discuss about advantage and disadvantage of Java in terms of using Java Virtual Machine.  Answer  장점 : JVM에서 바이트코드들이 실행되기 때문에 어느 OS에서든지 사용 가능하다(이식성이 좋다).  단점 : 컴파일 과정중 바이트코드로 만드는 것과 JVM을 거쳐 다시 통역되는 것 때문에 시간이 오래 걸린다. |
| --- |

| Exam Question: Discuss about portability of Java programs (how could Java programs have portability across various types of computers?)  Answer:  자바를 각 OS에 맞는 버전으로 다운받으면 자바로 작성한 프로그램은 수정없이 다른 OS에서 돌릴 수 있다. 왜냐하면 자바는 소스코드가 바이트코드로 컴파일만 되면, 전부 JVM을 이용해서 실행하기 때문이다. |
| --- |

# Lecture 3

## Announcement

* 앞으로 다룰 수업 내용 슬라이드: 이클립스에서
  + <https://goo.gl/S5QoYJ>
  + 구글 드라이브 공유 폴더 함에 자바 수업 관련 폴더 (Slides-Java Programming) 안에 03-04 How to write Java code in Eclipse로 열어서 볼 수도 있음.

## Our First Java Program

A source code file: OurFirstProgram.java

| // importing a package, java.util.Scanner. A package is a kind of external objects created by other developers. To build our program, we can reuse objects created by other developers. Technically, it is similar to ‘include’ in C.  import java.util.Scanner;  public class OurFirstProgram // define a class (a blueprint of an object)  {  static int howManyNumbers = 2; // An object has data and need to define.    // Each object has a method that defines an action of the object.  public static void main (String [] args)  {  // This is calling a method already defined in this object.  // We say calling a method as a method call or a method invocation.  printHello();  int n1, n2;  Scanner keyboard = new Scanner (System.in); //  n1 = keyboard.nextInt (); //  n2 = keyboard.nextInt (); //  System.out.println ("The sum of those two numbers is");  System.out.println (n1 + n2);  }  // this is my method I defined.  public static void printHello(){  // my system (computer) print OUT a line.  System.out.println ("Hello out there.");  System.out.println ("I will add " + howManyNumbers + " numbers for you.");  System.out.println ("Enter " + howManyNumbers + " whole numbers on a line:");  }  } |
| --- |

Output of OurFirstProgram.java

| C:>java OurFirstProgram  Hello out there;  I will add 2 numbers for you.  Enter 2 whole numbers  ... |
| --- |

## This Java code is consists of….

* A list of statements (like sentences)
  + A list of packages to be imported
  + A list of definitions of classes
  + A list of definitions of methods
  + A list of definitions of variables/constants
  + A list of statements ended with “;”
* A list of names
  + Class name
  + Method name
  + Variable name
  + ...
* A list of keywords
* Blocks

Full nodes for Java code:

<https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fjdt%2Fcore%2Fdom%2FASTNode.html>

⇒ If you can understand this link, you don’t need to take this class.

## What are they????????

(Don’t worry! We are going to learn one by one in 15 Weeks. We have enough time.)

## What is Java programming philosophy?

Java Programing Paradigm: Object-Oriented Programming

## Concepts of Object-oriented programming (OOP)

**Object:**

**Class (and its instance).**

**Package**

Inheritance: 나중에 배울 것임

Interface: 나중에 배울 것임

## Object-Oriented Programming

Programing paradigm based on objects:

***Everything is objects that interact with other objects.***

OBJECTS = their ACTIONS + STATES (Data)

A CLASS is a blueprint of an object!

A METHOD in a CLASS is a blueprint of each action of the object!

An ATTRIBUTE is a kind of variables that keep a particular state of an object! An attribute is also called as a member or a member variable.

(Slides: <https://goo.gl/jkGMnq>)

## What are packages, Objects, and instances?

## 



Object - Instance : Instantiate, Instantiation

## What is a Class?

* A design of an object
* A blueprint of an object

Further reading: <https://docs.oracle.com/javase/tutorial/java/concepts/class.html>



| Exam Question: Discuss about object-oriented paradigm. |
| --- |

| Exam Question: Discuss about an object,its class, and instances with real example. |
| --- |

## What are Modifiers?

내 패키지안에 있는 클래스와 클래스에 있는 메소드의 공개범위를 정하는 키워드

public-level: 클래스 정의할 때 public 키워드 사용 하면 됨.

package-level: 클래스 정의 할 때 그냥 class MyClass {} 이런식으만 정의하면 이 클래스는 내 패키지 안에서만 Instance만들어서 사용할 수 있음. 패키지 내부에서만 사용하는 클래스일 경우 이렇게 정의. 예를들어, 프로젝터 안에 렌즈, 팬 등을 package-level 클래스로 정의 하면 됨. public class로 정의하면 모든 사람들이 이 클래스를 사용할 수 있음.

| Exam Question: What is a modifier and what kinds of modifiers exist in Java? |
| --- |

## 

요약  
1. 주요개념  
(1) Object (객체): 현실 세계에 존재하는 모든 것들로 독립적으로 식별가능하며, 어떠한 상태를 나타내거나 행위를 하는 모든 실체  
ex) projector, screen, smart controller 등등

→ 데이터(상태 값 등), 액션(절차, 방법, 기능 등) 등으로 이루어짐

(2) Object Oriented Programming(객체지향 프로그래밍, OOP)  
: 프로그램을 객체들의 모임 또는 상호작용으로 생각하는 프로그래밍의 한 기법

(3) Class: 변수(Variable)과 메소드(Method)를 포함하고 있는 블록

(4) instance(인스턴스): 객체의 개념이 실제로 구현되어 특정 이름을 갖게 된 것(Object를 구체화 하는것)

Ex) NTH 413에 있는 projector

* instantiation: 인스턴스화 하는것

(5) Modifiers: 패키지 안에 있는 내 Class나 method의 공개범위를 설정하는 Keyword

# Lecture 4

## Announcement

* 보강일정 변경.
  + 3월 14일(수) 19:30 → 3월 21일(수) 19:30???
    - 카카오 챗에 변경 반대하는 사람 아직까지 없음.
    - 거의 확정...
* 기말고사 일정: 월 (6/11) or 목 (6/14)? >> 6월 11일 월요일 확정
  + <https://goo.gl/VRsXZ4>
* 오늘 강의 슬라이드: <https://goo.gl/feTYGd>
* 오늘 랩 문제: <https://goo.gl/YHqeHi>
* 모든 강의 및 랩 자료: drive.google.com → 공유문서함 or shared with me → Java programming folder → labs and slides.

## Typical Steps to write Java code in Eclipse

1. Create an Eclipse Java Project
2. Create packages in the Eclipse Java project
3. Add a java class (.java source file) in the project
4. Add comments and Javadoc comments frequently whiling coding
5. Define a class name and its block
6. Implement a class. (Define class members.)
7. Create a main method in the class that starts our program
8. Instantiate an object.
9. Call a method of an instance! (Method invocation)

## 5. Define a class name and its block

\* In an Eclipse project, create a java class file under the package folder in the 'src' folder. Then, Eclipse will automatically generate code for the class.

### Access levels (modifiers) of a class

Modifiers are keywords that show access levels between packages.

|  | **public** | **(default)** |
| --- | --- | --- |
| **Same Package** | O | O |
| **Others** | O | X |

\* Define a class and its block

| package edu.handong.csee.java.example;  public class Projector{  }  /\* This is package-level class.  \* We can put multiple package-level classes in one java file!  \* But we can also create a package level class in a separate java file.  \*/  class Lens {  } |
| --- |

Only one public class must be defined in a java file.

## 6. Implement a class

### Add an instance variable

= a member variable = a field = an attribute

### Add a method

Add an action of an object. It's like a function in C. A method can get any inputs and return an output.

### How to define a method?

### Access levels of a method

|  | **public** | **protected** | **(default)** | **private** |
| --- | --- | --- | --- | --- |
| **Same Class** | O | O | O | O |
| **Same Package** | O | O | O |  |
| **Sud(derived) Class** | O | O |  |  |
| **World** | O |  |  |  |

\* Define a method and its block

| public void myMethod(){  } |
| --- |

* void: do not return any output.

## 7. Create a main method in the class that starts our program

## 8. Instantiate of an object

## 9. Call a method of an instance (Method invocation)

## Let’s outline for Smart Classroom

Package: edu.handong.csee.java.example.projection

| Projector.java   | package edu.handong.csee.java.example.projection  /\*\*  \* This class implements a Projector object  \* @author JC  \*/  public class Projector {  private int mLampTemperature = 20; // this is an instance variable  /\*\*  \* This instance variable shows a description of this class.  \*/  public String mDescription = "This is a projector class";    /\*\*  \* This is a method for turning on a projector  \*/  public void turnOn() {    System.out.println("My project is turning on!!!");  getLampTemperature(); // call a private method in my current class    }    /\*\*  \* This is a method for turning off a projector  \*/  public void turnOff() {    System.out.println("My project is turning off!!!");    }    /\*\*  \* This is a method for getting the lamp temperature of the projector  \*/  private void getLampTemperature() {    System.out.println("My project tenperature is " + mLampTemperature);    }  } | | --- |   Screen.java   |  | | --- | |
| --- | --- | --- |

Package: edu.handong.csee.java.example.control

| Controller.java   | package edu.handong.csee.java.example.control;  import edu.handong.csee.java.example.projection.Projector;  public class Controller {  public static void main(String[] args) {    // Type these lines for you test  // These lines print out texts.  System.out.println("This is my controller!!");  System.out.println("My first word used as an input is " + args[0]);  System.out.println("My first word used as an input is " + args[1]);    // instantiate my controller  Controller nth413Controller = new Controller();    // a method call of an instance, nth413Controller, of the Controller class  nth413Controller.turnOnProjector();    }  public void turnOnProjector() {    // instantiate the Projector class  Projector nth413Projector = new Projector();    // a method call of an instance, nth413Projector, of the Projector class  nth413Projector.turnOn();    }  } | | --- | |
| --- | --- |

| Exam Question: What is instantiation? |
| --- |

| Exam Question: How instance variables (non-static field) are different from class variables (static field)? |
| --- |

| Exam Question: Draw a figure how the non-static methods in instances and static methods occupy memory (RAM). |
| --- |

## 

# Lecture 5: Language Basics

## Announcement

* 자바노트 정리 지원자 (Extra point 부여 할 예정.)
  + <https://goo.gl/51QGK8>
* 이번 수업, 다음 수업 랩 없음. 대신 Week 4,5에 확장된 랩 실시할 예정.
* TA Session
  + 강청웅 TA
    - 수요일 19:00~20:00
    - 장소: 협의
    - 연락방법: 010-6676-1925
  + 한보경 TA
    - 월/목 15:00~16:00
    - 장소: 오석관 상상랩 또는 코딩 스페이스 (협의)
    - 연락방법
      * 이메일: [21500777@handong.edu](mailto:21500777@handong.edu)
      * 카톡 p6002p

## Programming conventions we must/should follow (1)

1. MUST: A class name must be same as a java file name.
2. Should: The first character of a class name should always be a upper case.
3. MUST: A java source code file must have only one public class in it.
4. Should: Avoid defining multiple non-public classes in the same file because it is confusing. But a java source code file may have multiple non-public (package-level) classes.
5. Should: A package name can be any word but Java developers usually define package names with their domain names in a reverse order.
6. Should: The first character of a method name should be a lowercase to distinguish with a class name. But the first character of words in the method name should be a uppercase. e.g., turnOnMyProjector()
7. Should: Apply indentation in Eclipse (See slides: <https://goo.gl/zKfmWz>) \*indentation : 들여쓰기

Eclipse의 편리한 기능

* 소스코드들을 드래그하고 우클릭 한 후Source 란을 눌러 보면 Add Block Comment(주석 처리), Generate Element Comment(자바독 생성), Correct Indentation(자동 들여쓰기) 등 편리한 기능을 이용할 수 있다.
* 메소드 이름을 드래그하고 F3을 누르면 소스코드를 볼 수 있다. 원리는 바이트코드를 다시 eclipse가 하이레벨 랭귀지로 번역을 하는 것이다.

## A magic of a static modifier

Slides: <https://goo.gl/4VxCvj>

| class Controller{  public ***static*** void main(String[] agrs) {  String nameOfThisClass = "my name is Controller";  // Press F3 key on 'out' in Eclipse  System.out.println("Actually calling a method of a static instance.");  // Access mDescription without instantiatingController  System.out.println("Description: " + Controller.mDescription);  }  public ***static*** String mDescription = "This is a Controller class!";  public int mNumControlButtons = 10;  } |
| --- |

**Static** **modifier** makes variables (data of objects) and methods (actions of objects) exactly one copy of them in the computer memory

* Static으로 class 안에서 변수나 메소드를 정의하면 어디서든 연동이 된다.
* Instantiate가 필요없다. 왜냐하면 그 정보는 메모리에 올라갔기 때문이다.
* Util : static method로 자주쓰는 메소드들을 관리해주는 것
* 주의사항 : 메소드 안에서 static을 선언하면 에러가 뜬다. 메소드는 한 번 사용되고 사라지는 지역개념이기 때문

Q. 두 메소드가 한 변수의 결과를 공유하고 싶으면?

* Static은 사용이 안되므로 input과 output으로 변수를 서로 주고 받으면 된다.

## Variables and variable types

Slides: <https://goo.gl/eYTkXf>

변수란?

* 우리가 사용할 데이터(값)를 저장하는 곳

### The kinds of variables

* Instance Variables (Non-Static Fields)
* Class Variables (Static Fields)
* Local Variables
* Parameters

### Naming conventions

* Case-sensitive
  + a variable, 'myname', is not same as a variable, 'myName'.
* Start with letters
  + Then any combinations of letters, digits, $ and \_.
* All chracters in the first word should be lower cases. Then, the first chracters of other words should be upper cases.
  + e.g., currentTemperatureTooHigh

## All related topics in the use of variables

See Slides: <https://goo.gl/ghaxCp>

* 자바는 Unicode set을 사용한다. Unicode란 한글과 같은 각 나라의 언어들을 사용하기 위해 나온 코드 체계이다.

## 

## Constants and named constants

Slides: <https://goo.gl/wyxRGt>

Use the 'final' keyword

Use uppercases to name the constant.

‘Final’ keyword는 마치 C++에서의 const같이 변수를 상수화 할 때 사용된다.

## Levels of Definition



Varibles defined in second levels can be accessed in the third levels.

| Exam Question: What are naming conventions for a package, class, method, and variables? |
| --- |

# Lecture 6: Language Basics (2)

## Announcement

* 수업 적응도 설문조사 시행 시기 및 결과 조회 시기
  + (4주차~5주차) 2018.03.19(월)10:00 ~ 2018.03.31(일)23:00
* Recruiting S-lab members
  + <https://docs.google.com/document/d/1TotIlZh_k1b6nFI-zz7Kn7Lq3jRva3Q3plkrvqUDCHs/edit?usp=sharing>

### Keyboard and Screen I/O

Slides: <https://goo.gl/dSxExJ>

### Documentation and Style

Slides: <https://goo.gl/97oi1A>

## Related topics in the defining classes and methods

See Slides: <https://goo.gl/gbXUcL>

### Design my object

Class name: Ship

Data

Name

Purpose

weight

length

-width

-height

-maxWeightForShipping

Price

speed

fule

material

Tolerrence

maxSpeed

Year

Method (Actions)

+turnLeft

+turnRight

reverse

forward

getFuel

getCurrenPosition

getDepthToBottom

isThereAnyObstacles

### Information Hiding

### Encapsulation

Information hiding is realized by encapsulation.

# Lecture 7: Branching

* Lab04 and HW1-1: <https://docs.google.com/presentation/d/19xnMXd2i8eTJimJ7biMTtO_lenuE1xeFs34I0WIgH4A/edit#slide=id.p3>
  + HW1-1 Due: Mar 27 9:00AM (Late submission, zero mark, no mercy)
* Blockchain seminars: <https://drive.google.com/file/d/1zwSRhVyfx46af9LuJlMGJ0w9f2NLKjRk/view?usp=sharing>

## Flow of Control

Slides: <https://docs.google.com/presentation/d/1h15bWfN8djJBX27falwRA_6UJEIz_JiqL3XwbsHLxsM/edit#slide=id.p3>

# Lecture 8: Branching and Language Basic (3)

## Announcement

* Lab05 and HW1-2
  + <https://docs.google.com/presentation/d/1x9j7qurEY39U66ic5FDXO9ItoHTP5bL4v0oGS5YNeWI/edit#slide=id.p>

## Type Boolean and Switch

Slides: <https://docs.google.com/presentation/d/1h15bWfN8djJBX27falwRA_6UJEIz_JiqL3XwbsHLxsM/edit#slide=id.g360defc94e_0_6>

## Object and References

Slides: <https://docs.google.com/presentation/d/10xVyLvDgVyG2sTeIoq-oIU91Mqz8NiulC5LmGJVm9eA/edit#slide=id.p53>

## Software Repositories (Git and Github)

Slides: <https://docs.google.com/presentation/d/1gBkrNSednTSyCyy2qkJprPHoK9LXydoB2sAW9M_fGv4/edit#slide=id.p>

## 강의 7, 8 내용 요약

‘Flow of control’이란, 프로그램이 수행하는 순서라고 볼 수 있다. 기본적으로 순서는 순차적이다.

**Branch statement** : 2개 이상의 조건에서 선택하는 것

**Loop statement** : 종료조건때까지 반복하는 것

* **If else statement**

\*\* 연달아서 쓰면 2개 이상의 조건도 만들 수 있다. (if, else if …)

\*\*간단한 조건을 한줄로 나타내고 싶을 때는 조건연산자를 사용하자

<ex> max = (n1 > n2) **?** n1 **:** n2;

* **Boolean**

Boolean : 참, 거짓을 표현하는 값이다. 비교연산자와 논리연산자를 통해 참, 거짓을 판별할 수 있다.

* 이름 붙이기 : 이름을 의문문형식으로, 정확하게 지정하는게 가독성이 높다.
* 주의 : 아무리 같은 내용의 문자열(String)이더라도 두 문자열 간 ‘==’연산을 하면, (일반적인 new 선언을 할때,) 같지 않다고 나온다! 이는 두 문자열의 주소값을 비교하는 것이기 때문이다.
* **이번 강의에서 배운 메서드**
* String.equals(s2) : 두 문자열이 같은지 알려준다
* String.equalsIgnoreCase(s2) : 대소문자 상관없이 문자열 같은지 알려준다.
* String\_1.compareTo(String\_2) : 두 문자열을 비교해서 string1이 유니코드에서 먼저 나오면 음수, 같으면 0, string1이 유니코드에서 나중에 나오면 양수 (자바는 유니코드를 지원한다.)
* String1.toUpperCase() : String1을 모두 대문자로 바꾼다.
* String1.toLowerCase() : String1을 모두 소문자로 바꾼다.
* System.exit() : 프로그램을 강제 종료시킨다
* **연산의 우선순위 → ppt**
* **Short circuit evaluation :** Short circuit evaluation —> 앞의 조건에 따라 뒤가 실행된다. 앞 조건이 ‘false’면 바로 건너뛰어서 효율적이다.
* **Switch statement :** ‘if statement’와 비슷한 multi-branch다. 각 case마다 break;를 써줘야 하며, 나머지 조건의 경우, ‘default : ’로 처리한다.
* **This :** class내에서 자기 자신을 가리킬때 사용한다!
* **변수의 type :** primitive type(메모리에 값을 저장), class type(메모리에 주소값을 저장)
* **Class의 Instance 선언 및 생성 :** MyClass obj = new MyClass();
* 객체에서 같다는 의미는 **주소**의 관점으로만 보아야 하며, 숫자값을 갖는 Primitive type외 클래스 변수는 주소값을 가진다. ~~‘==’는 주소를 가리킨다!~~ 그래서 if 문 조건에서 같은 값을 가지는 클래스 변수끼리 비교할 때 ==를 절대 안된다. equals 메소드로 비교를 해야한다. Primitive type으로 선언된 변수들은 '=='로 서로 같은지 비교할 수 있다.

Discussion Q) 메서드에서 객체(주소)를 파라미터로 줄 수 있다. 그러나 메서드 내에서 그 객체(파라미터)에 다른 객체의 주소값을 넣어도, 메서드를 빠져나오면 아무런 변화가 없다. 만약 파라미터로 전해진 객체의 내용을 바꾸고 싶다면 어떻게 해야 하는가?

A) 파라미터로 넘겨오는 것은 로컬변수로 생각하면 된다. 그렇기에 오브젝트간 주소(주소는 정수)를 넘겨주는 것도 원래대로 돌아온다. 그러나 오브젝트의 각 구성요소에서 본다면 파라미터 오브젝트의 구성요소에 다른 데이터를 직접 넣는다면 메서드(함수)를 빠져나와도 유지된다!

| Exam Question: Discuss about difference between a == b and a.equals(b) in case a and b are variables defined as String class. |
| --- |

| Exam Question: What is a for-each statement and when for-each statement is useful? |
| --- |

## 

# Lecture 9: Build Tools and Loops

## Announcement

* Lab06 and HW1-3
  + <https://docs.google.com/presentation/d/1IbzqQlOZZHb8GIqEpT9HdbxxmMxw82rDpAiOH24P3LI/edit#slide=id.p>
* TA Session
  + 강청웅 TA (By appointment)
    - 수요일 19:00~20:00
    - 장소: 협의
    - 연락방법: 010-6676-1925
  + 한보경 TA (By appointment)
    - 월/목 15:00~16:00
    - 장소: 오석관 상상랩 또는 코딩 스페이스 (협의)
    - 연락방법
      * 이메일: [21500777@handong.edu](mailto:21500777@handong.edu)
      * 카톡 p6002p

## Build tool: Gradle

Sdlies: <https://docs.google.com/presentation/d/1v05iLFuV1utYnJk3SOdzLeOHv6Og96Cc19AEEHc6zaU/edit#slide=id.p>

A build tool is automating the process of compiling source code.

There are many build tools for different programming languages.

<https://en.wikipedia.org/wiki/List_of_build_automation_software>

A build tool also supports many useful features such as

* Automatically download and add external packages into my project.
  + E.g., <https://mvnrepository.com/artifact/com.google.guava/guava/24.1-jre>
* Create an execution scripts automatically for various operating systems.
* ...

For Java, there are many different kinds of build tools. We use Gradle in this course. Official Android App build tool is Gradle.

## How to use git in Eclipse

Slides: <https://docs.google.com/presentation/d/1dMNuQ_fL9YwxC7DGExaKG3_O5lTtm5f_qirig4Tho1Q/edit#slide=id.p>

## Flow of Control: Loops (1)

Slides: <https://docs.google.com/presentation/d/127CXSv0lNLolWcX4D7csqFKgEdm9G2W441seaJs0qPY/edit#slide=id.p3>

## Lecture 9 요약

* Gradle : Gradle은 Groovy를 이용한 [빌드 자동화](https://ko.wikipedia.org/wiki/%EB%B9%8C%EB%93%9C_%EC%9E%90%EB%8F%99%ED%99%94) 시스템이다.
* **Groovy** : 공개 소스 소프트웨어로 개발되는 최근의 객체 지향 언어로서, 기존 언어와는 달리 변수 타입이 동적(dynamical)이고 유연하며, 소스 파일을 컴파일하지 않고 바로 실행시키는 스크립팅 언어.
* Build tool : 빌드 툴은 소스 코드 (예 : 안드로이드 앱용 .apk)에서 실행 가능한 애플리케이션을 자동 생성하는 프로그램 (출처: <http://giraffeb.tistory.com/70>)
* 반복문 : while, do~whlie, for
* 반복문을 쓸 때, 주의할 점 : 종료조건을 반드시 걸어야 한다!(or Infinite loop
* ‘for’ 반복문에서.. 조건을 복합적으로 줄 수 있다 (ex)for (n = 1, product = 1; n <= 10; **product = product \* n, n++)**;

# 

# 

# Lecture 10: Loops and More about Objects and Methods (1)

## To initiate your project using git, Github, Gradle and Eclipse

Slides: <https://docs.google.com/presentation/d/1A_ozTjvcMzZM-0P9pXK3Ucyyi-LgYgpunwrZrhwwMHY/edit#slide=id.p>

## Flow of Control: Loops (2)

Slides: <https://docs.google.com/presentation/d/127CXSv0lNLolWcX4D7csqFKgEdm9G2W441seaJs0qPY/edit#slide=id.p24>

## More about Objects and Methods (1)

<https://docs.google.com/presentation/d/1Lib6bhgKcy5DNF8p0iGypPtulbHGDOyuO2A23LJQ2ug/edit#slide=id.p3>

## Lab07-HW1-4

<https://docs.google.com/presentation/d/1vZojJZtXWL9NdYYd2fy8RvEegsDXZh7L3oKrwd2N6uI/edit#slide=id.p3>

1. **Git, Github, Gradle, Eclipse를 사용하여 Java Project를 initiate하기**
2. Java 프로젝트를 저장할 폴더 만들기
3. Local Git Repository 만들기
   1. Cmd로 자바 프로젝트가 있는 폴더 지정(cd.. / cd 사용)
   2. git init를 입력해 프로젝트 폴더 initiate (.git 폴더 생성)
   3. git remote add origin https://github.com/[YOUR]/[GIT\_ADDR].git 입력
4. Java Project에서 Gradle을 사용할 수 있도록 다음을 입력해서 초기화

gradle init --type java-library

1. Eclipse에서 Gradle 프로젝트를 input (이전에 New java porject를 대신 함)
2. Github에 Push할 파일을 Add index를 한 후 commit, 그리고 push를 함
3. 변경사항 생길 때 마다 Commit을 해 로컬에 저장한 후, Push는 나중에 한번에 해도 됨
4. **Loop Control 2**
5. Infinite Loops: 한 반복문이 탈출하지 않고 끝없이 반복되는 현상 → 주로 변수설정을 잘못하거나 해서 발생
6. Nested Loops: 한 Loop 안에 다른 Loop가 있는 것
7. for Statement: 변수 선언과, 조건문, 증가/감소 여부를 한 줄로 적어서 사용
   1. for(initialization; condition; Update){

Statements

}

1. **Constructors**

: "new"통해 어떤 class의 Instance를 생성되는 순간 실행이 되는 특별한 Method로 Parameter를 가질 수 있다.

Ex:

public Pet() // 파라미터가 없는 것들은 굳이 소스코드에 선언 안해줘도 된다.  
public Pet(String name, int age, double weight)

| Exam Question: Discuss about constructors. |
| --- |

## 

# Lecture 11: More about Objects and Methods (2) and Arrays

## More about Objects and Methods (2)

<https://docs.google.com/presentation/d/1Lib6bhgKcy5DNF8p0iGypPtulbHGDOyuO2A23LJQ2ug/edit#slide=id.p41>

### Overloading

#### Method signature?

| Exam Question: What is overloading? |
| --- |

## 

### Enumeration as a class

### Packages

## Arrays

<https://docs.google.com/presentation/d/1TIWKdngMjexhzMKj5a2rHxb_h0JV0cmZnQ9hJVsLWKU/edit#slide=id.p3>

## Lab08

<https://docs.google.com/presentation/d/1lK8NmfoFFxFWKf0iLMwa8XnN7wroUBsarg9Gvd1SWSE/edit#slide=id.p>

Lecture 11 요약

1. Overloading - 같은 이름의 함수를 여러개 정의하고, 1)매개변수의 유형과 2) 개수를 다르게 하여 다양한 유형의 호출에 응당하게 한다.
   1. Instantiation 이 될 때 주어진 매개변수에 따라 object가 다르게 instantiate 되게 하는 것이다. (member variable 이 지정된다던지..)
   2. Method Signature 란 method의 이름과 매개변수의 유형과 개수를 뜻한다.
      1. Signature 가 같은 method가 존재 할 경우 (즉 return 값만 다를 경우) 그 method는 정상적으로 작동하지 않는다.
   3. 가독성을 높히는 장점이 있다.
2. Enumeration - 같은 유형의 data type을 모아둔 집합
   1. C에서의 enum과 비슷하다( 하지만 C.enum != java.enum)
   2. Java에서는 enum에 class 처럼 action하는 method도 넣을 수 있다.
3. Array 는 특별한 Object 다. Array를 선언할때는 다음과 같이 해야한다
   1. *Base\_Type*[ ] *Array\_Name* = new *Base\_Type*[*Length*];
   2. (*Base\_Type* *Array\_Name*[ ] = new *Base\_Type*[*Length*];
   3. Array 안의 원소를 얻기 위해서는 C 와 같이 Base\_Type[Index] 로 얻을 수 있다.

(index 는 순서대로 0 부터 *Array\_Name.length - 1 까지있다.)*

* 1. Array는 object 이기 때문에 Array1 == Array2 는 다음 배열들이 같은 객체인지를 비교한다. 반면 array 안의 내용물이 같은 지를 확인하려면, Arrays.equals(Array1, Array2)를 사용하여 비교해야한다.

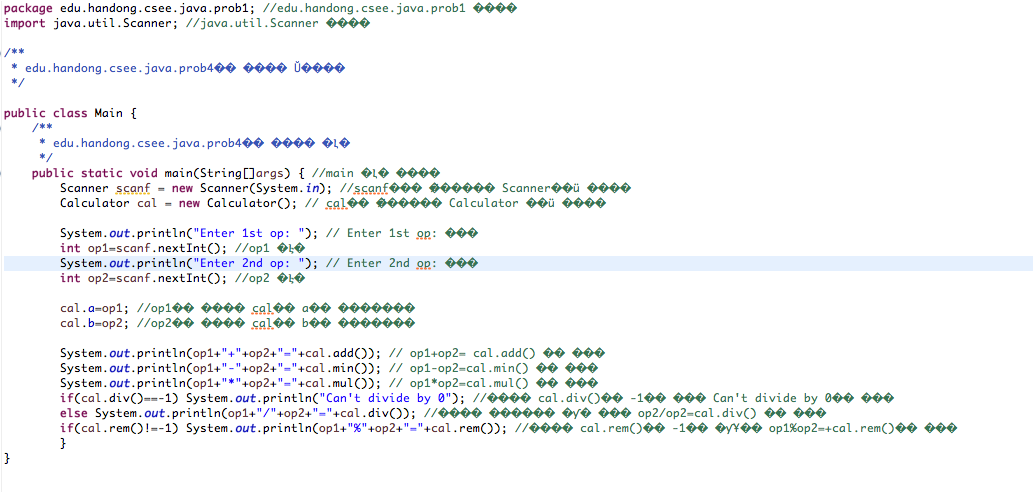
*Ps. Array1.equals(Array2) 는 Array1 == Array2 와 같은 일을 하기에 헷갈리지 않도록 해야한다.*

EXTRA: Eclipse를 사용할때 빠른 코딩을 위해 유용한 핫키를 찾아보니 이런 것들이 있더군요! 참고하시면 좋을것 같습니다.

[*http://blog.insightbook.co.kr/wp-content/uploads/1/lk310000000000.pdf*](http://blog.insightbook.co.kr/wp-content/uploads/1/lk310000000000.pdf)

# Lecture 12: Arrays (2)

## Announcement

* Do not use Korean for Javadoc and single line comments in your code.
* Lab09 and HW1-5
  + <https://docs.google.com/presentation/d/1Cq7wLIvtfdUQb9wOanWg7JR1807xirg5m421YTbmolo/edit#slide=id.p>
* Lecture slides
  + <https://docs.google.com/presentation/d/1TIWKdngMjexhzMKj5a2rHxb_h0JV0cmZnQ9hJVsLWKU/edit#slide=id.p16>

## Arrays in Classes and Methods

### Case Study: Sales Reporter

[We are coding!]

Lecture 12 요약 :

**수업 중 질문내역**

-Gradle을 사용하는 이유가 무엇인가?

Gradle의 장점때문이다.

1. 다른 사용자가 내 project를 import하기만 하면 나머지 setting을 알아서 해준다. -> 협업에 유리
2. Package 작업을 하는데 편리하다.(git의 외부 package를 불러들일 때 주소만 쓰면 된다.)

-Add to index가 무엇인가?

Git에서 change를 관리하는 목록에 포함시키다는 뜻이다.

-src/test 는 무엇에다가 쓰는 것인가?

Test는 import와 export를 report해서 보여주는 기능을 가지고 있다. 여러사람들과 협업을 할 때 build하면서 test를 같이하는데 이때 잘못된 부분을 분석하는데 편리하다.

\*commit 시간으로 과제제출 마감을 체크하기 때문에, push시간이 늦어도 commit시간이 pass면 인정

**Tip**

Array를 선언할 때 Arraysize를 같이 선언해도 되고 나누어서 나중에 선언해도 된다. (선언시 index에 변수가 들어가도 된다. cp. C언어)

Ex. Salesman[] team = new Salesman[length]; //-> 이것을

Salesman[] team;

team = new Salesman[length]; //-> 이렇게 나누어 표현할 수 있다.

# Lecture 13: Arrays (3)

## Announcement

* Midterm exam
  + 09:30 ~ 11:10 (100 mins) April 19 (Thu), 2018
    - Study candidate exam questions (which will be added gradually)!
* Check your scores
  + Shared a google sheet for you (Check your Handong mailbox.)
  + If you think your score is better or worse than you expected, leave comments in the google sheet.

## Sort Arrays

Slides: <https://docs.google.com/presentation/d/1TIWKdngMjexhzMKj5a2rHxb_h0JV0cmZnQ9hJVsLWKU/edit#slide=id.p36>

You may fork my code from GitHub: <https://github.com/lifove/SelectionSorting>

## Multi-dimensional arrays

Slides: <https://docs.google.com/presentation/d/1TIWKdngMjexhzMKj5a2rHxb_h0JV0cmZnQ9hJVsLWKU/edit#slide=id.p44>

## Lab10

<https://docs.google.com/presentation/d/1qxUxIxNce1N-MOEKWmCVGf6lv6Gvuu7vJNo-VodjE0Y/edit#slide=id.p>

Lecture 13 요약 :

**-Arrays를 선언할 때 private type으로 하는 것이 좋다.**(Arrays를 잘못 수정하면 오류가 뜨면서 피곤해지기 때문이다.)

**-Arrays sorting 방법** [**https://medium.com/@fiv3star/%EC%A0%95%EB%A0%AC%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-sorting-algorithm-%EC%A0%95%EB%A6%AC-8ca307269dc7**](https://medium.com/@fiv3star/%EC%A0%95%EB%A0%AC%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-sorting-algorithm-%EC%A0%95%EB%A6%AC-8ca307269dc7) **-(> 여기 설명 너무 잘 되어있음)**

**1.삽입(Insertion) 정렬**

**2.버블(Bubble)정렬**

**3.선택(selection)정렬 => 프로그램 구현이 쉽다**

**4.퀵(Quick)정렬 => 일반적으로 제일 빠른 sort**

**5.Radix sort => 교수님께서 수업시간에 찾아보라고 언급하셨던 Quick sort보다 빠른 정렬방법. 시간복잡도 O(n) = n으로 가장 빠르지만 제약이 따른다.**

**-Selection sort**

처음에 있는 Index를 가지고 나머지 Index와 비교하여 제일 작은 값을 첫 번째 위치에 놓고 비교하는 동안 첫 번째 값보다 더 작은 값이 발견되면 항상 첫 번째 값과 그 작은 값을 서로 교환한다. 다음에 두 번째 작은 값을 찾고 이를 두 번째 위치에 놓는다. 이런 과정을 반복ㆍ실행하면 정렬이 된다.

**-java.util.Arrays**

Arrays class: Arrays의 비교, Array의 정렬 또는 Array의 내용을 특정 값으로 채우고자 하는 등의 일을 수행하는 도구성 class

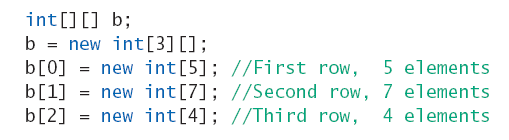
-> Arrays class는 sort함수를 가지고있다.

Ex) Arrays.sort(int[] a), Arrays.sort(double[] a), …

-**Multidimensional-Array Basics(다차원배열)**

선언방법: int [ ][ ] table = new int [10][6];

OR

**-for each statement**

사용방법: for(변수타입 변수이름 : 배열이름){반복내용}

필요성: 반복문과 배열이 결합될 때 사용하기 용이

주의점: Index를 바꾸는 것에는 불가능하다. Print할 때 좋다.

<http://library1008.tistory.com/66> ->자세한설명

# Lecture 14: ArrayList and HashMap

## Announcement

* Lecture Slides
  + <https://docs.google.com/presentation/d/1JZFEG4B8XuwXqTLgYhOvcLCgFkgBhS8uiNQRU_gyGgg/edit#slide=id.p3>
* Lab11 and HW1-6
  + <https://docs.google.com/presentation/d/1XXfIOrRzTLFleKutxmycjCz0mJRiZ9QBMWrI_Vcw8rc/edit#slide=id.p>
* Midterm exam questions will be added here.
  + <https://docs.google.com/document/d/13IO5K_3DCzCGuFX3uZNTrSihhZVSOdh1ug_Rqc3ggxA/edit>

## ArrayList

## HashMap

## Lab11 and HW1-6

<https://docs.google.com/presentation/d/1XXfIOrRzTLFleKutxmycjCz0mJRiZ9QBMWrI_Vcw8rc/edit#slide=id.p>

| Exam Question: Write single line comments for all lines of the following java source code. (One source code file from your labs or HWs will be given.) |
| --- |

# 

# Lecture 15: Inheritance, Polymorphism, and Interfaces

## Announcement

* Midterm exam
  + 09:30 ~ 11:10 (100 mins) April 19 (Thu), 2018
    - Study all candidate exam questions! Write answers in your own language although you memorize the same answers from the candidate exam questions.
* No lab today but discuss about exam questions
* Slides: <https://docs.google.com/presentation/d/1SfuTCdlt1IVoGbzhlmNwUtw8DsxCEXq3O0mKJJIWc1o/edit#slide=id.p3>

## Inheritance

### Extends

Inheritance basics = inheritance allows programmer to define a general class.

When we define subclass and inherit parent class, the way of writing subclass is that

(modifier) + class + extends + (parent class name)

In this way, we define a more specific class.

* We can add new details to general definition.
* New sub-class inherits all properties of initial, general class.

What is overriding?

* Overriding takes place in subclass. It’s a new method with same signature. Method in subclass with same signature overrides method from base class.

Overriding method is the one used for objects of the derived class.

Overriding method must return same type of value

### Super

The parent class (upper notion) is super class.

The child class ( lower notion) is sub class.

Sub class inherits all methods, etc.

In the constructor, super() brings the constructor in parent class.

super() needs to be declared in the first row always.

You must type the letters super() and cannot type the name of parent class.

ex) Person(); -> wrong. super(); -> correct

### is-a vs. has-a relationships

Is-a is the relationship that we can define sub class is a parent class. However, vice versa is wrong. ex) student is subclass. So we can say a student is a person. However, it is the wrong statement that a person ia a student.

Has-a is the relationship that a class can contain an object of another type.

### The ultimate ancestor, Object

It is possible, to write a method with parameter of type object.

Actual parameter in the call can be an object of any type.

* The words, “extends object” are omitted from every classes behind.

Therefore, when we declare objects, we can say the methods in object such as equals, toString, etc.

| Exam Question: Discuss about overriding. |
| --- |

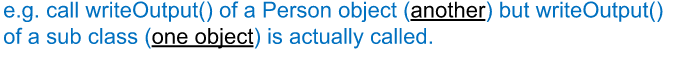
# 

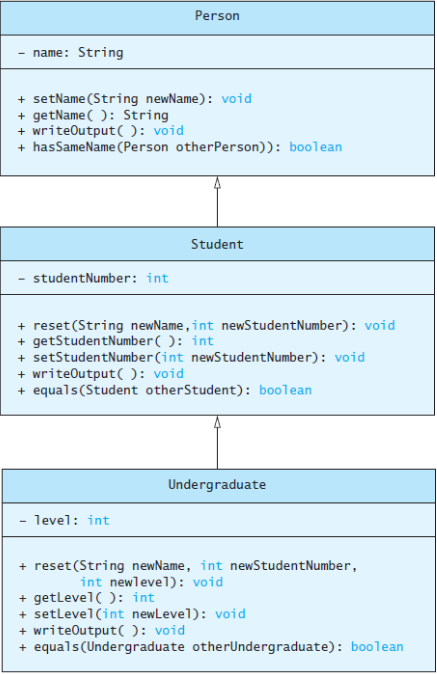
# Lecture 17: Inheritance, Polymorphism, and Interfaces (2)

## Polymorphism

<https://docs.google.com/presentation/d/1SfuTCdlt1IVoGbzhlmNwUtw8DsxCEXq3O0mKJJIWc1o/edit#slide=id.p27>

**polymorphism:**다형성

객체가 다양한 존재로 만들어 주는 것이다. 



student의 write output을 부른다. : overriding

특정 object가 있는데 다른 것으로 대체가 된다.

person은 실제로 student instance이다. 그래서 student object로 대체되어 write output을 출력한다.

정의는 person으로 했지만 프로그램을 진행할 때 프로그램을 부르기 때문에 student output이 불리게 된다.

**dynamic binding**: 우리가 person을 불러들었지만 프로그램이 실행 될 때 어떤 것을 불러들일지는 결정 나기 때문에 student의 method 대체가 된다.

**overridden 과 dynamic binding의 차이**

polymorphism 이라는 상황 안에서 overridden의 개념이 실행되는 상황

어떻게 돌아가는지 개념은 알지만 프로그램이 실제로 돌아갈 그 시점 실제 instance가 가르키는 people output을 실행한다.

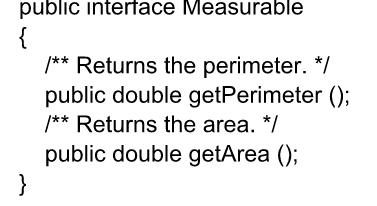
가변적인 상황이 생긴다면 무엇을 불러들일지 모른다. 그래서 실제로 실행될 때 나오는 output이 dynamic 상황에 결정이 되고 출력이 된다.

**질문: 위의 instance를 불러들이려면 어떻게 해야되나?**

만약 허락을 해준다면 inheritance의 개념이 깨져버린다.

디자인이 복잡하게 된다.

**overridden method로 실행될 때**



new 라는 실제 method 가 실행된다.

type으로 실행되는 것이 아니라 instance에서 결정 되어 실행

## 객체는 항상 어떤 method action이 이름에 따라 실행이 될 것인지 결정 한다

## 

## Interfaces

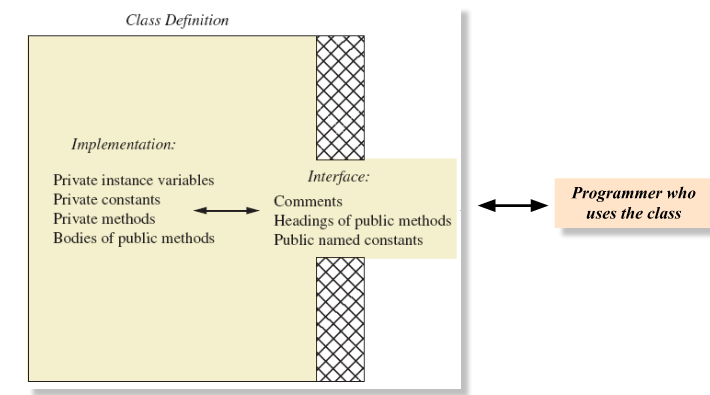
<https://docs.google.com/presentation/d/1SfuTCdlt1IVoGbzhlmNwUtw8DsxCEXq3O0mKJJIWc1o/edit#slide=id.p35>

**interface**

우리는 action의 head를 정하는 작업을 한다.

method heading이 interface이다.

예제에 나온 public method 가 interface이다.



**Interface가 필요한 이유**

개발자들의 소프트웨어를 잘 상호작용을 해주는 일종의 contract이다.

interface는 protocol이다. 꼭 지켜야 할 부분들이다.

**순서:** class는 이 interface를 구현한다.

interface를 먼저 선언하고 interface를 구현한다.

왜냐하면 interface안에는 method heading만 선언하기 때문

모든 action이 똑같고 구현이 다른 것이다.

interface heading의 기반으로 해서 구체적으로 설계를 하는 것이 class를 구현하는 것이다.

같은 behavior method 이름을 가지게 되는데 실제로 실행되는 method와 다르다.

**ex) 자동차 소프트웨어 만드는 회사**

navigation과 gps를 합치는 하청 회사가 있다고 하자.

자동차 소프트웨어 회사는 gps의 heading만을 필요로 한다.

그래서 하청회사에서 넘겨주는 자료에서 사용자가 필요로 하는 정보만 사용

-> 융통성을 가지고 서로 다른 회사 개발자들이 재사용 교류나 의사소통 가능

**특징:**

interface는 behavior heading만 선언하면 그 method가 실행

interface는 public name의 constant까지 선언할 수 있다.

interface의 이름은 uppercase interface의 형태는 interface.java로 저장가능

**장점:**

다수의 interface를 받아들일 수 있다. 여러 가지 관계들을 체계적으로 정리할 수 있는 장점을 가지고 있다.

interface는 수평적인 작업을 할 때 유용하다. (\*inheritance는 수직적인 작업을 할 때 유용)

interface method를 구현하기 위해선 implements Interface\_name형태로 선언

Lab12

<https://docs.google.com/presentation/d/1pZJJTtrLZ4Feelxg5OBIYDSjFRqXMaYtS4GoHEsCdlg/edit#slide=id.g374020a59d_0_49>

| Exam Question: what is interface and why need in java? |
| --- |

| Exam Question: what is correct answer to possible declaration in interface part? ( ) (1) Declaration of constructors  (2) Constant public name  (3) Method bodies |
| --- |

# 

# Lecture 18: Interface (2)

## Announcement

* 2018 Summer Camps
  + Check HisNet announcement of CSEE
    - 3116, 3117
* Lab13 and HW1-7
  + <https://docs.google.com/presentation/d/1LdUJGgeZaF_eyef4b0PPeJT4539qIuz7qIWsTqeWCM8/edit#slide=id.p3>

## InstanceOf

<https://docs.google.com/presentation/d/1SfuTCdlt1IVoGbzhlmNwUtw8DsxCEXq3O0mKJJIWc1o/edit#slide=id.g387f0f85f3_13_11>

Check a type of object

Instanceof is used to check if an object is an instance of a class, an instance of a subclass, or an instance of a class that implements a particular interface

Person p → we can call getLevel only in Undergraduate by casting p

Before casting, we need to check if p is actually Undergraduate type by using instanceof

Undergraduate studentOjb → so now, we can call getLevel.

공각기동대 애니메이션을 보고 일본 언어를 쉽게 배우는 한국인들이 있다. (문법이 유사하기 떄문에)

다른언어를 하나 배우면 다른 언어를 배우기가 쉽다.

The comparable interface

The comparable interface is used to impose an ordering upon the objects that implement it.

A class implementing comparable can be sorted by Arrays.sort() method.

Salesman 을 array.sort 를 이용해서 sort를 할 때 어떤식으로 sort를 해야할지 정의를 해줘야 실행이 가능하다. 그렇지 않으면 comparable이 안된다.

내부적으로 comparable에 따라 sorting을 하는데 구현이 되어 있지 않으면 sorting이 되지 않는다.

ex)

public class Salesman implements Comparable<Salesman>{

...

public int compareTo(Salesman anotherSalesman) {

return name.compareTo(anotherSalesman.name);

}

}

## Interfaces (2)

<https://docs.google.com/presentation/d/1SfuTCdlt1IVoGbzhlmNwUtw8DsxCEXq3O0mKJJIWc1o/edit#slide=id.p43>

# 

# Lecture 19: Abtract Class. Exception Handling

## Announcement

* 2차 수업 적응도 설문조사 시행 방법
  + HISNet 로그인 - 수업정보 - 설문평가 - 수업적응도 조사
* Lab14 and HW 2-1
  + <https://docs.google.com/presentation/d/1SgrXFkYvX6_Qq1aS3_GwN3d8s-XKUnmX9UF1suKzf0E/edit?usp=drive_web&ouid=108272865787539764399>
* 보강 (5월 7일 대체 공휴일)
  + 보강 없음
  + 6월 11일날 수업 있음. (기말고사 정리 및 한 학기 복습 예정)

## Abstract Class

<https://docs.google.com/presentation/d/1SfuTCdlt1IVoGbzhlmNwUtw8DsxCEXq3O0mKJJIWc1o/edit#slide=id.g387f0f85f3_13_0>

Abstract Class

클래스를 추상 클래스로 정의함으로써 슈퍼클래스에 method의 구현부(몸통)를 구현하지 않고 선언부만 추가 가능.

Abstract class의 효용을 비유로 알아보자

서로 10%씩 다른 복사본 3개를 만든다고 가정했을 때, 90%의 공통부분을 abstract class로 만들어 300% 노력을 90% + 10% + 10% + 10% = 120%로 줄일 수 있다. (수업 때 언급 된 건 아니고 자료를 찾다가 본 이 설명이 이해가 잘 됐어서 적어봤습니다...)

특징

1. 미완성 설계도로 완제품을 만들 수 없음, 몸통이 없는 ‘불완전’한 추상클래스로 instance 생성 불가능 = 조상 클래스로 사용 됨
2. Abstract method가 단 하나만 있더라도 abstract class !
3. ‘Dynamic binding’ 런타임에 무슨 method를 사용 할 지 결정된다

## Exception Handling

<https://docs.google.com/presentation/d/1t8mbTprCb07l7O6nJvQcG42gWCEcA1sXLXgObPPsmyo/edit>

exception이란? 내가 짠 프로그램에서 의도하지 않은 예외적인 상황이 벌어진 경우, 왜 error라고 부르지 않는가?

=> exception은 내가 수정할 수 있는 것. data와 action 가지고 있는 하나의 객체, 문제가 무엇인지 알고, 또 수정 할 수 있음!

에러 제시하는 것을 'throwing', 해결해 주는 것을 'handling' 이라고 함

Exception의 흐름

Try block-(throw), Catch block

Try 블록에서 문제 상황이 감지되어 throw(상황 인식) 되는 순간 catch 블록(해결요구)으로 들어옴.

ㄴ predefined exception in java - 미리 정의 해놔서 throw가 안 보이는 경우

Exception class를 상속 하게되면 개인의 exception을 만들 수 있다

특정한 예외 상황을 자신이 정의 할 수도 있음. 이것을 예외나 에러로 받아들여서 ~하게 수정하겠다 할 수도 있음 = 나만의 exception을 따로 정의 할 수 있음!

ex) 학번(ex 2nn000nnn)을 받는 경우 string 8개가 들어와야 되는데 7개가 들어왔다! - 원래 대로라면 문제없지만 error 취급해야 됨

미리 정의 된 Exception class의 파생 클래스 여야하며 객체의 유형은 파생 클래스의 이름이어야 한다.

exception class 더 알아보자.

exception 던지면 어디선가 분명히 받아야 한다. 이것을 제한해둬야 한다,

note distinction

Keyword throw used to throw exception

Keyword throws used in method heading to declare an exception

throw, throws 차이점 : 만약 throw에서 바로 해결을 안하고 다른method에 넘겨주고 싶을 경우 throws

ppt 19~20 참조

Exception의 종류

checked, unchecked => 꼭 처리를 해야 하는가의 차이

Multiple throws and catches

하나의 try 블록에서 여러가지 유형의 exception throw 가능 but catch는 하나의 throw에만 대응 가능

문제. Describe the flow of exception

Try 블록에서 문제 상황이 감지되어 throw(상황 인식) 되는 순간 catch 블록으로 들어와 수정 요구하게 됨.

# Lecture 20: File I/O

## Announcement

* 2차 수업 적응도 설문조사 시행 방법
  + HISNet 로그인 - 수업정보 - 설문평가 - 수업적응도 조사
* Lecture Slides
  + <https://docs.google.com/presentation/d/19_1kQalDTG8MEERIvj27VjJehhDjIZkTbYJfPWkiius/edit#slide=id.g37a0f8c2cb_0_10>
  + Example code
    - git clone <https://github.com/lifove/FileIO.git>
* Lab15
  + <https://docs.google.com/presentation/d/10VaUEyy9h4oDf7Ltq7DNl_-EDTF71zJ0s3lewV_lzDA/edit#slide=id.p>

How to show Java source code in Eclipse

<https://dzone.com/articles/attaching-java-source-eclipse>

Stream이란 무엇인가?

* A stream is a flow of input or output data

=> 데이터의 인풋/아웃풋의 흐름(또한 스트림의 실행을 위한 특정 클래스가 있다), 다른 표현으로는 자료의 입출력을 도와주는 중간 매개체. 혹은 중간 이동통로 정도로 비유 가능

데이터, 즉 파일은 두가지 종류가 있음.

Text file(treated as sequence of characters)

Binary file (All other files)

text file을 작성해 보자

PPT 5p~ , Class PrintWriter ! 이것을 만들기 위한 조건들이 있다

우선 java.io 패키지를 임폴트 해야함. 그 후 파일을 열기 위해서는

1. 스트림 변수를 선언해야함, 2. 생성자를 불러와서 pass file name 을 줘야함. 3. Try, catch 블록 사용해야함(에러가 생기는 경우를 대비해야 하기 때문 ex) 하드 꽉차는 경우...)

특징

파일 열기,쓰기는 기존의 파일을 덮어쓰기 함, 초기 파일은 비어있음, 데이터는 버퍼에서 메모리로 이동, 버퍼가 닫히면, 스트림은 종료 됨( ~.close 사용. 그런데 스트림의 생성에서 예외가 발생하는 경우 close를 사용하면 안되기 때문에 작성시 유의 해야 함)

<https://github.com/lifove/FileIO/blob/master/src/main/java/edu/handong/csee/java/example/TextFileOutputDemo.java>

ㄴ ppt의 코드

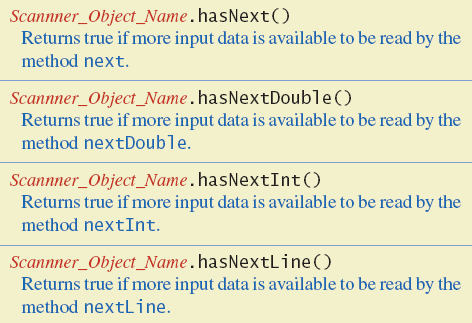
윗 단락에서 기존의 파일을 덮어쓰기 한다고 언급함. 이때 덮어쓰기 할 경우 new FileOutputstream(fileName, true) 사용해야 함. (true가 덮어쓰기 하겠다는 뜻)

# Text file을 읽어보자.

<https://github.com/lifove/FileIO/blob/master/src/main/java/edu/handong/csee/java/example/TextFileInputDemo.java> 파일을 읽고 화면에 출력하는 코드

이때는 Scanner object 사용해야 함, 부울 문 사용해서 판독 루프 종료해야 함.

기타 메소드들



Class File에 대해서 알아보자

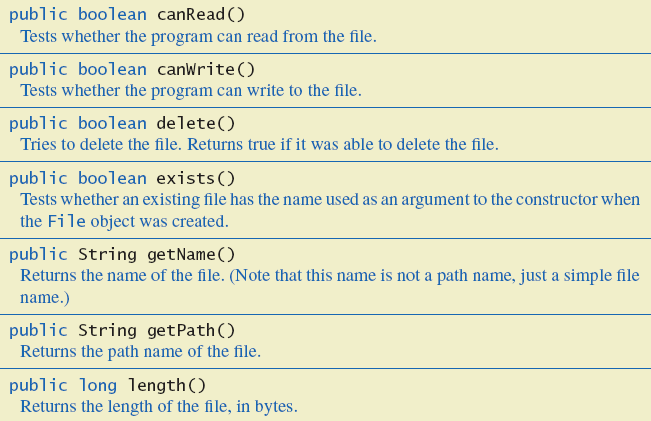
File 오브젝트는 파일의 이름을 나타냄. Ex) new File ("treasure.txt"); 이러한 형식으로 쓰는데 단순한 제목을 나타내는 것이 아니라, 사용가능한 이름임을 나타냄.

https://github.com/lifove/FileIO/blob/master/src/main/java/edu/handong/csee/java/example/TextFileInputWithUserFileDemo.java 교수님 예시

파일을 읽을때 경로(상대 경로, 절대 경로)를 지정하는 것이 가능! 단 os에 따라 주소방식이 다른 것은 유의

Recall that a File object is a system-independent abstraction of file's path name

File class의 메소드들



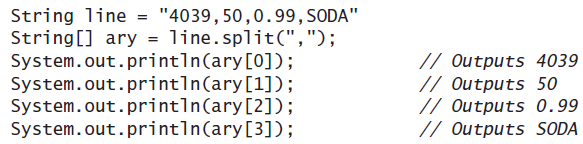
위의 메소드들은 string타입을 매개변수로 받는다. 스트림 오브젝트를 반환 할 것이고

예외로 If file not found, If some other I/O problem arises 두가지를 가질 것. = try catch block 사용해야 함.

<https://github.com/lifove/FileIO/blob/master/src/main/java/edu/handong/csee/java/example/TransactionReader.java>

예시, cvs file 받기.

Csv 파일은 ,로 정보가 끊겨서 표현이 되는데 이것을 split 메소드를 사용해서 적절하게 텍스트를 받아올 수 있다.



문제. Stream이 무엇인지 설명해라.

데이터의 인풋/아웃풋의 흐름(또한 스트림의 실행을 위한 특정 클래스가 있다), 다른 표현으로는 자료의 입출력을 도와주는 중간 매개체. 혹은 중간 이동통로 정도로 비유 가능

# 

# Lecture 21: File I/O (2)

* 블록체인 토론회
  + 오늘 오후 4시 코딩스페이스
  + 2nd course adjustment survey (2차 수업 적응도 설문조사 시행)
  + HisNet login - Course infor - Survey
  + (HISNet 로그인 - 수업정보 - 설문평가 - 수업적응도 조사)
* Slides
  + <https://docs.google.com/presentation/d/19_1kQalDTG8MEERIvj27VjJehhDjIZkTbYJfPWkiius/edit#slide=id.g39bb83db5d_0_15>
* Lab16
  + <https://docs.google.com/presentation/d/1qYFVkLzJ-lJtYWbOH7ZRmjtbPLyNiA0E6re59t0Y1hM/edit>

## Binary File I/O

# Lecture 22: Dynamic data structures and generics (2)

* HW3 (20 points) released!
  + <https://docs.google.com/presentation/d/1aw_Q2rOZUN0hIoiuE8QrXka8i7UjG-USB4ueT322AcE/edit>
* Lecture slides
  + <https://docs.google.com/presentation/d/1mHmJWRTMSQfkNctav_QrODF_FTi1TXPsEdll8xvtadI/edit#slide=id.p1>
* TAs
  + TA Session
    - 강청웅 TA
      * 수요일 19:00~20:00
      * 장소: 협의
      * 연락방법: 010-6676-1925
    - 한보경 TA
      * 월/목 15:00~16:00
      * 장소: 오석관 상상랩 또는 코딩 스페이스 (협의)
      * 연락방법
        + 이메일: [21500777@handong.edu](mailto:21500777@handong.edu)
        + 카톡 p6002p
    - 정찬미 TA
      * by appointment
      * 21500671@handong.edu
    - 최은서 TA
      * by appointment
      * [21400771@handong.edu](mailto:21400771@handong.edu)

## Dynamic Data structures

### Hashset

ArrayList처럼 객체들을 순차적으로 저장할수있다. Eg. Hashset<Integer> 는 integer 값을 받는 Hashset이다. Hashset 역시 method들이 다 미리 만들어 져있기때문에 ArrayList쓰는거 처럼 쓰면된다. Eg. Hashset.add(), Hashset.contains();

### Linked List

스택은 보통 LinkedList로 구현한다. LinkedList는 array와 같이 객체들을 저장하지만, 순차적으로 저장하는것이 아니라 pointer를 이용하여 객체를을 연결해준다. 그러므로 메모리가 복잡하게 섞여있어도, pointer로 시작점만 정해주면 객체를 찾아갈수있다. Java는 ArrayList와 Hashset과 같이 LinkedList 클래스가 이미있다.

### 프로그램에 대한 간단한 설명과 앞으로의 방향

컴퓨터는 무조건 Input을 받고 Output을 출력한다. Input 을 output으로 만들어주는과정을 Algorithm.

만드는 과정에서 데이터를 어떻게 체계적으로 다룰수있는지를 배우는것이 Data Structure.

프로그램에서 다루는 모든 Data를 저장하는 방식은 Database에서 결정.

A.I.가 다른 사물을 인식할뿐만아니라 자기자신을 인식하면 그때부터 Input은 필요없고 스스로 Output을 출력하게될것.

### 다시 LinkedList

Class ListNode

Private String data -String type인 데이터

Private ListNode link -다음노드의 주소값을 가지고있는 변수

자바는 따로 노드를 free 해줄 필요가없다(good)

Position.link는 c의 position->next 와 비슷한 개념이다.

LinkedList에 Inheritance를 적용할수있나? 없다 누구는 누구다했을때 상속이 가능한 관계이면 되는데 Node는 List다라는 상속관계과 성립되지않기 때문에 쓸수없다.

position.getlink()는 LinkNode class가 inner class가 아니고 독립된 private class로 존재할때 필요하다.

List를 traverse할때 쓰는 여러가지 Iterator method가 있다.

<https://github.com/lifove/CollectionsExample/blob/master/src/main/java/edu/handong/csee/java/exmple/collections/linkedlist/StringLinkedListWithIterator.java>

참조.

## HW3 Class diagram

List of classes

class FileLoader

class MessageParser

class PMCounter

class FileWriter

class RedundancyChecker

# Lecture 23: Generics (3)

Announcement

* Slides
  + <https://docs.google.com/presentation/d/1mHmJWRTMSQfkNctav_QrODF_FTi1TXPsEdll8xvtadI/edit#slide=id.p57>

## Generics

## Designing and Coding for HW3

class ChatCounter

class FileLoader

class MessageParser

class PMCounter

class FileWriter

class RedundancyChecker

class Message

| ChatCounter |
| --- |
|  |
| + main(String[] args): void |

| FileLoader |
| --- |
| - messages: ArrayList<String> |
| + readDirectory(String path): void  - getFileNames: ArrayList<File>  + getMessages(): void |

**Lecture Note(5/17)**

* **Generic(제너릭)**
* Java 5.0 때부터 클래스 내부에서 사용할 데이터타입을 외부에서 지정할 수 있도록 구현되었다.
* <> : type parameter 라고 부른다.
* <> 안에 파라미터 수는 여러개가 되어도 된다.
* Generic을 쓰는 이유는 안정성 때문이다. 컴파일에서는 문제가 없는데 런타임때 문제가 생길수 있기 때문이다.
* Generic은 기본 data type으로 설정 할 수 없다 (ex: 그래서 <Integer> <Double> 등의 wrapper class를 대입한다.)
* anArray = new T[20]; 처럼 선언할수 없다. (T는 class의 이름이여야 한다)
* <>안의 이름은 아무것이나 되어도 된다.
  + But Generally
  + E - Element (used extensively by the Java Collections Framework)
  + K - Key
  + N - Number
  + T - Type
  + V - Value
* 아직도 헷갈린다면 다음을 참고하면 좋을것 같다. <https://opentutorials.org/module/516/6237>
* **LinkedList**
* LinkedList 안에 inner class로 선언된 listNode를 linkedList class와 병렬로 선언해도 된다.

# Lecture 24: Practical topics for Java programming

## Announcement

* HW3
  + The final output file must have the counting information in descending order.  
    JC,243  
    HW,233  
    JS,212
  + If you are an Windows user, please change your encoding in Eclipse
    - <https://docs.google.com/presentation/d/1aw_Q2rOZUN0hIoiuE8QrXka8i7UjG-USB4ueT322AcE/edit#slide=id.g3b2af362d1_11_0>
  + Need to deal with following cases correctly!
    - [오전 05:22] and [9:40 AM]
    - CSV files have seconds but TXT files does not have seconds!
    - Messages that have a new line are saved differently in CSV and txt files.
  + To read a file, use BufferedReader with InputStreamReader and FileInputStream but not Scanner. Scanner may have some issues on Windows!
  + Due extended!!
    - 10PM May 27 (Sun)

## Gradle distZip

<https://docs.google.com/presentation/d/199XW7wcnFDf35K1wV9TryrK9brlDiemzVx31UpIyGEQ/edit#slide=id.p>

## Import an external library via Gradle

<https://docs.google.com/presentation/d/1SO8lu8s4371KT5Ow19eD4D5MA3IpWc5gH-za0_Ce_6g/edit#slide=id.p>

## Apache Commons CLI

<https://docs.google.com/presentation/d/11ZPK5NOpviyvsIVpCC6tYEqP3zKflSw-SoeaAjfqBro/edit#slide=id.p>

## Regular Expression

Our Example

<https://github.com/lifove/RegExpExample>

Regular Expression Cheat Sheets

<http://files.zeroturnaround.com/pdf/zt_regular-expressions-cheat-sheet.pdf>

Regular expression tester

<http://www.regexplanet.com/advanced/java/index.html>

Samples

<http://www.mkyong.com/tutorials/java-regular-expression-tutorials/>

# 

# 

# Lecture 25: Recursion

## Announcement

* One of students found bugs
  + My program did not deal with the following cases
    - --------------- Monday, March 12, 2018 ---------------
    - [남재창] [9:25 AM] Good morning everyone!
  + There is a file that does not have date. e.g., 자바-L7.txt
  + Bonus point 1 for the student who found this issue.

# Recursion

<https://docs.google.com/presentation/d/1UIFr-ZB9g3oRbG6VI9pg_p8F_ENUowWr4RQkLnaZ1Qk/edit>

public void myMethod(){

myMethod();

}

123

(1) n/10

12

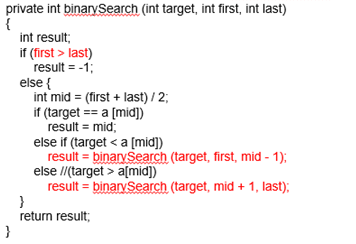
(2) 12/10

1

One Two Three

Recursion

어떤 사건이 자기 자신을 포함하고 다시 자기 자신을 사용하여 정의될 때 재귀적(recursive)이라고 한다.

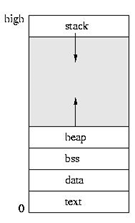
예를 들어, Binary search 코드를 살펴보자.

빨간 색 부분을 보면 다시 자신을 호출하고 있다. 이런 경우를 Recursion이라고 한다.

Recursion의 구조

1. Base case: stop condition

Recursion도 결국 반복이다. For문이나 while문에서 탈출 조건이 필요하듯이 Recursion도 탈출 조건이 필요하다. 그것을 Base case라고 한다. base case가 만약 없다면 무한히 자신을 호출 할 것이고 그 결과 Stack overflow가 발생한다.

cf> Stack overflow

메모리 구조는 오른쪽 그림처럼 생겼는데 heap부분에는 메모리 동적할당을 했을 때 그것들이 차곡차곡 쌓이고 Stack 부분에는 지역변수나 파라미터같이 일시적인 것들이 차곡차곡 내려온다. 이때 스택의 크기가 너무 커져서 범람하는 경우를 stack overflow라고 한다.

Recursion이 너무 많이 실행되거나 아니면 무한 recursion이 발생한다면 stack overflow가 발생할 수 있다.

2. General case: Handle larger cases

Base case가 아닌 실제로 반복하기 위한 부분이다.

이렇게 base case와 general case를 구별하는 것을 Branching statement라고 하고 recursion에서는 꼭 필요한 부분이다.

간단한 예제로 Factorial을 구성하는 코드를 보자

int factorial(int n) {

if (n == 1) // base case

return n;

return n \* factorial(n-1); // general case

}

Factorial의 구현은 굳이 Recursion을 사용하지 않고 for문(iterative)으로도 구현할 수 있다. 하지만 왜 Recursion을 사용하여 구현하는 것일까?

그 이유는 간단하다.

1. 코드의 명확성 확보.

2. 수학적으로 정의한 규칙을 그대로 코드로 옮기면 재귀함수의 형태가 된다.

하지만 단점도 있다.

1. Space complexity의 관점에서 기존 iterative statement보다 성능이 떨어진다.

2. Performance의 관점에서 기존 iterative statement보다 느리다.

# 

# 

# Lecture 26: Q&A and discussion for HW3

## Announcement

* HW3
  + 마감일 연장 5월 29일 화요일 22:00
  + 원래 마감일에 제출한 사람은 +2 보너스 점수
    - 반드시 메일로 알려줘야 점수가 나감.
  + gradle distZip에서 컴파일 에러 나는 사람들은
    - git bash 에서 gradle distZip 해야함
      * export GRADLE\_OPTS="-Dfile.encoding=utf-8"
      * gradle distZip
    - 또는 cmd 인코딩을 유니코드로 변경 해서 실행해야 함.
      * cmd /U 라고 친 후 하면 됨.
      * http://blog.daum.net/ipajama/143

## Issues

- 특정 라인만 중복확인 잘 안됨: JC 보강, 도인

- 시간이 같고 메시지 같은거 같은 걸로 친다.

# Lecture 27: Concurrency (1)

## Announcement

* HW4
  + <https://docs.google.com/presentation/d/1X1N5YKUTNsk8lB-dYyYd8nFpL6wAstYTTWdcgqPvvlY/edit#slide=id.p>
* Lecture slides:
  + <https://docs.google.com/presentation/d/1NFmRs2OXPHb_z6Ql-p0WauvHFXzCSv757ZxzXdeYZWw/edit#slide=id.p>
* Lab17
  + <https://docs.google.com/presentation/d/1lkJfU-NahiHIDYMr2RDAfWTAwGZR_wnlhU2D4XgeAJY/edit#slide=id.p>

# Lecture 28: Concurrency (2)

## Announcement

* 진도 나가는 마지막 수업
* 다음주 월요일은 기말고사 요약 및 전체 복습 (1시간 정도 예상)
* 기말고사
  + 다음주 목요일 9:30~11:15
* 5점짜리 보너스 HW??

## Thread Pools

Fundj4core

(1) runs 1000 threads at the same time

cpu0:1/2/13/14/

cpu1:15/17/999

cpu2:

cpu3:

(2) runs 1000 threads based on the number of cores.

queue 1,2, 3, 4,5, …..1000

1,2,3,4

1,2,5,4

Counter c = new Counter();

AThres

c

c.increase();

B thread

c

c.increase();

## Other topics in threads

### Synchronization

### Deadlock

# Lecture 29: Summary

## Announcement

* Candidate Final Exam Questions
  + <https://docs.google.com/document/d/168PnTDytOZSNrzqtMKnS9EbklDC9LkqHyrxe_LzTr_A/edit#>
* Mid-term evaluation scheme
  + <https://docs.google.com/document/d/1TkHWtERD3rP_WxW87wIs49_0iDe0GCc5NYTn2fSJgVw/edit>

## 