

# 1 과목소개

# 목표 및 사용언어

- 목표 : 프로그램 설계의 이해 및 숙달
  - 제어구조 (control structure) **CSE117 프로그래밍기초**
  - 자료구조 (data structure) **CSE210 자료구조론**
  - 부품구조 (**component structure**)
- Java를 쓰는 이유
  - 부품 구조 기반으로 설계된 언어이기 때문에
  - 널리 쓰는 언어이기 때문에

# 선수 및 병수 과목

- 선수과목: **CSE117 프로그래밍기초**
  - 제어구조의 기본기는 갖추고 있다고 가정
- 수강제한
  - **ELE213 프로그래밍방법론**을 수강한 학생
  - 3~4학년으로 Java 언어를 이미 잘 알고 있는 학생
  - 재수강은 예외

# 교재 및 슬라이드

## ○ 교재

- *Programming Principles in Java: Architectures and Interfaces*, David A. Schmidt (Kansas State University)
- <http://people.cis.ksu.edu/~schmidt/CIS200/home.html>
- 강의 홈페이지에서도 내려 받기 가능

## ○ 강의 홈페이지

- <https://cse2016hy.github.io>
- 강의 슬라이드 내려 받기 가능

## ○ 실습

- 코드 언 웹 (<https://erica.codeonweb.com>)
- 강의 홈페이지 참조

# 강의 주제

- 과목 소개
- 간단한 Java 응용 프로그램
- 산술연산과 변수
- 입출력과 상태
- 부품구조
- 제어구조
- 반복 패턴
- 자료구조
- 인터페이스를 활용한 프로그래밍
- 문서와 파일 처리

# 평가 및 연락처

## ○ 평가

- 중간고사 30%, 기말고사 30%, 실습 30%, 출석 10%

## ○ 연락처

- 교수

이우석 (woosuk@hanyang.ac.kr, x1008, 3공403호)

- 조교 1

박건우 ([parkgunwoo@gmail.com](mailto:parkgunwoo@gmail.com))

# 컴퓨터와 프로그래밍

# 컴퓨터 (Computer)?

## ○ 컴퓨터란?

- 명령을 수행하는 것
- 광의의 의미로 사람도 컴퓨터다.

## ○ 컴퓨터의 일반적인 구성

- 프로세서 (processor): 명령을 처리하는 것
- 주 저장소: 명령과 자료를 저장하는 것
- 그 외에
  - 부 저장소: 대용량 자료를 저장하는 것
  - 입출력 장치: 명령 수행 도중 사용자와 상호작용하도록 하는 것



# 프로그램 (Program)

## ○ 프로그램 또는 코드 (code)

- 컴퓨터가 수행할 수 있도록 충분히 구체적으로 작성된 명령 나열
- 이를 작성하는 행위를 프로그래밍 (programming) 또는 코딩(coding)이라고 한다.

```
sum = 0;
for(i=1; i<=100; i++) {
    sum += i;
}
```

## ○ 알고리즘 (algorithm)

- 목적을 달성하기 위한 명령의 나열
- 프로그램에 비해 덜 구체적
- **ELE334 알고리즘설계와 분석**

i를 1부터 100까지 증가시키면서 모두 더한다.

# 프로그래밍 언어

- 프로그램을 작성하기 위해 사용하는 언어
- 예전에는 기계어 사용
  - 기계가 빠른 속도로 수행 가능
  - 사람이 작성하기에는 매우 어려움
- 현재는 고급 언어 사용
  - 사람의 생각을 쉽게 프로그램으로 표현 가능
  - Fortran, Cobol, Lisp, Basic, Algol, Prolog, ML, C++, Java 등
  - **ENE414 프로그래밍언어론**

# 고급언어의 실행

- 기계는 고급언어를 이해하지 못한다.
  - 컴파일러 (compiler): 고급언어로 작성된 프로그램을 기계어로 작성된 코드로 변환하는 프로그램
  - 수행기 (interpreter): 고급언어를 가상으로 직접 수행하는 프로그램
  - **CSE309 컴파일러구조**
- Java의 두 단계 방법
  - Java로 작성된 프로그램을 중간언어인 Java 바이트코드로 컴파일
  - Java 바이트코드로 작성된 프로그램을 Java 가상기계(Java virtual machine, 수행기)를 통해 실행하거나 기계어로 컴파일하여 수행
  - 이식성(portability)을 높이기 위한 방법

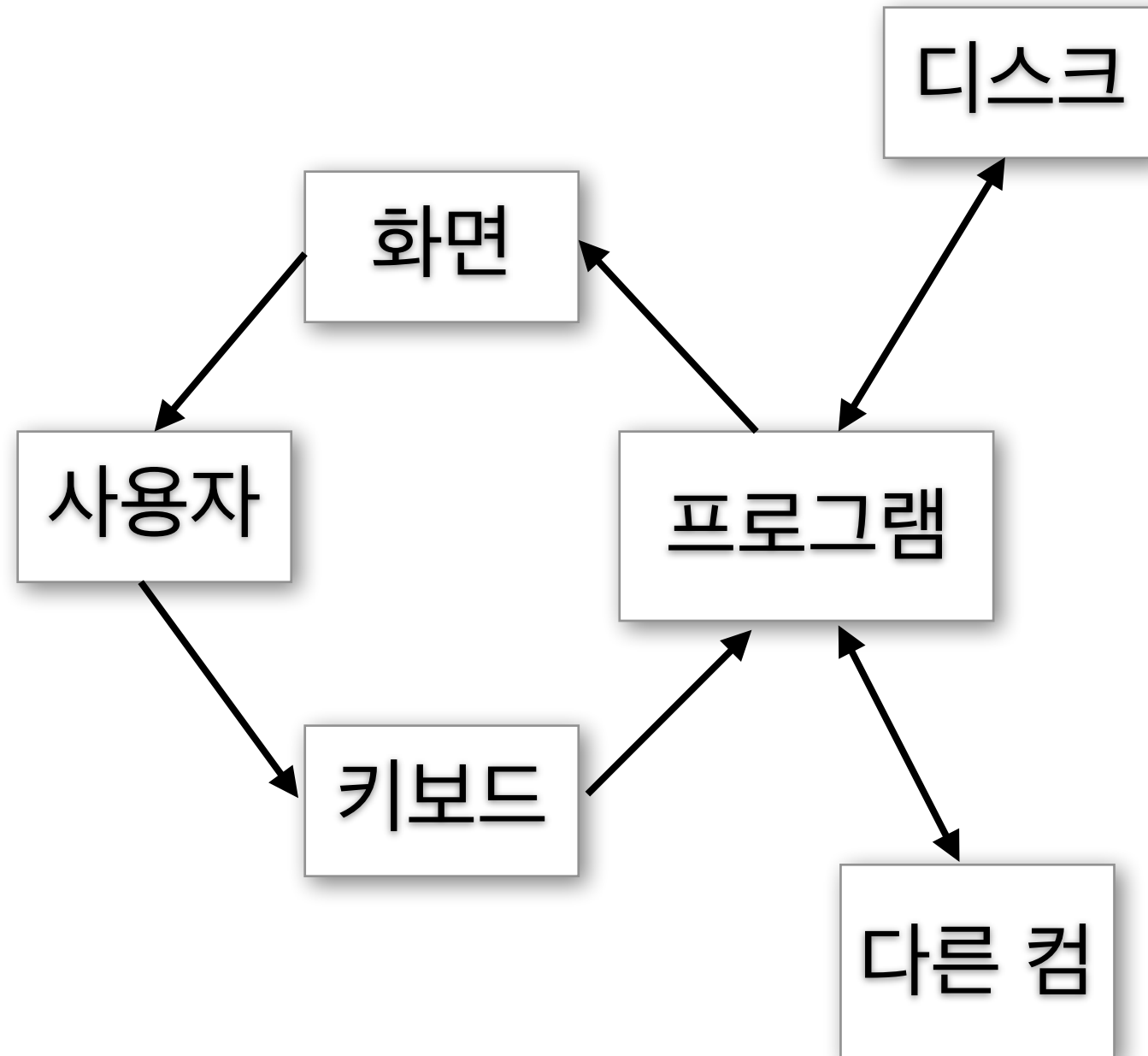
# 프로그램 설계

- 집짓기는 설계와 공사를 분리한다.
  - 건축가 (architect): 집을 어떻게 지을지 전체적인 구성을 설계하는 사람
  - 빌더 (builder): 집을 실제로 짓는 사람
- 프로그래머는 일반적으로 두 작업을 다 하지만, 좋은 프로그램 작성을 위해서는 설계와 구현을 분리해야 한다.
  - 설계 (design): 소프트웨어 구조 (software architecture) 구축 **CSE406**  
소프트웨어공학
  - 구현 (implementation): 프로그램 또는 코드 작성

# 객체 지향 설계 (Object-Oriented Design)

- 프로그램을 설계하는 한 방법
- 기본
  - 모든 것은 객체(object)이다.
  - 객체는 자신이 해야 할 일, 즉 메소드(method)를 갖는다.

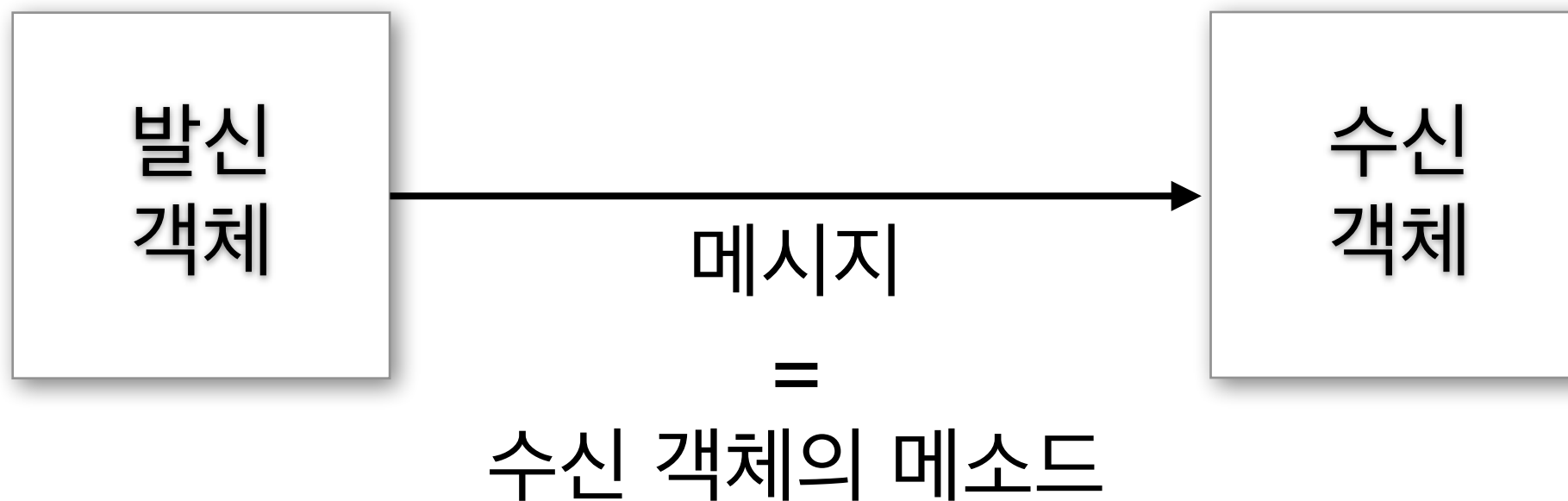
# 프로그램 수행을 객체로



- 객체: 키보드
  - 메소드: 문자 입력
- 객체: 화면
  - 메소드: 화면과 그림 출력
- 객체: 사용자
  - 메소드: 키보드로 문자열 입력, 화면에서 결과를 봄

# 계산 = 메시지 전달의 연속

- 객체 간에 메시지(message)를 전달하여 소통하는 것을 통해 계산이 수행된다.



# 예

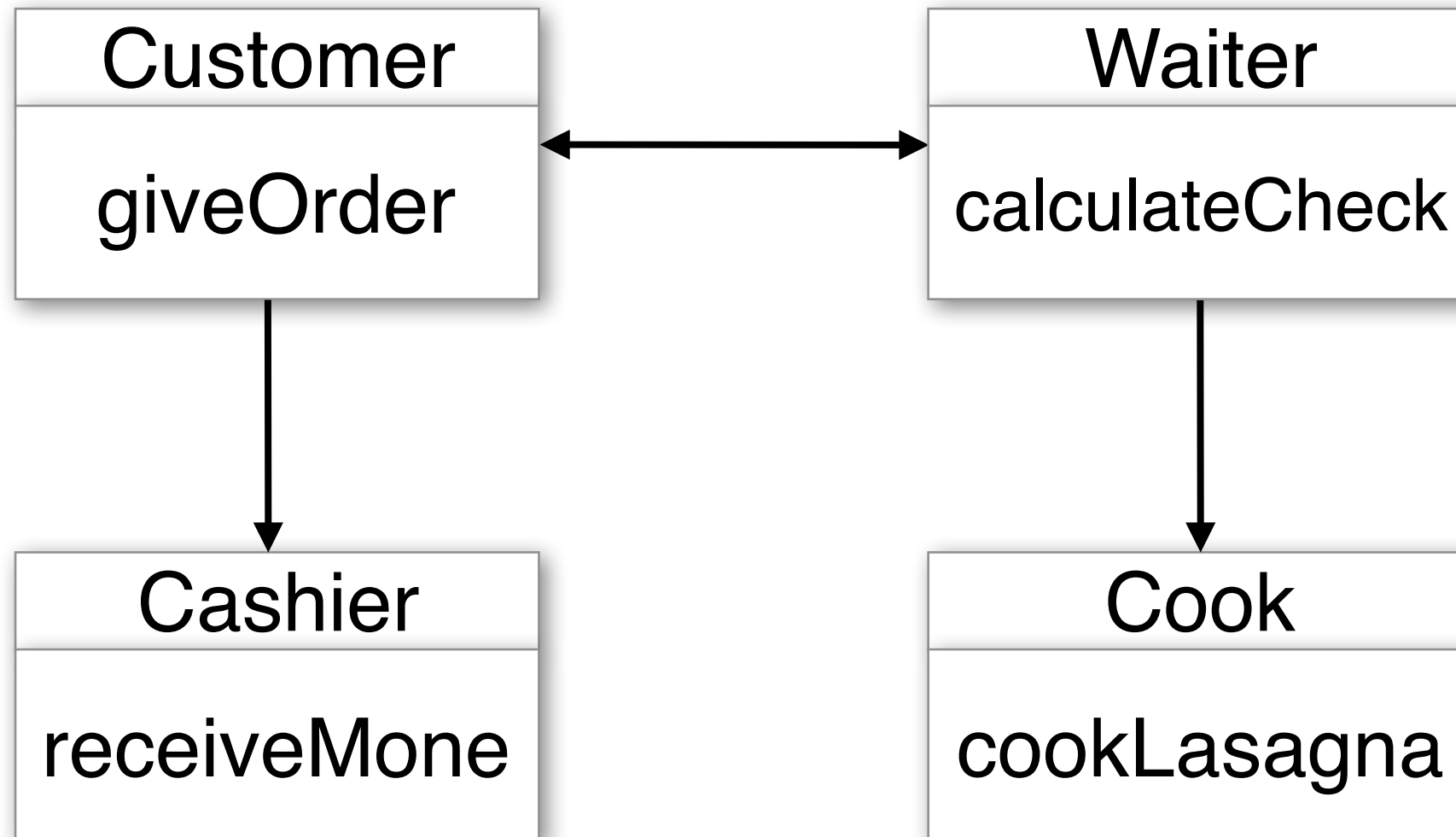
- <사용자>는 어떤 수의 제곱 값을 알고 싶어 <키보드>에 그 수에 해당하는 키를 입력한다.
- <키보드>는 눌러진 키를 수로 변환하여 <프로그램 수행>에 수를 전달한다.
- <프로그램 수행>은 수의 제곱 값을 구해 <화면>에 전달한다.
- <화면>은 수를 화면에 기호로 표시하여 <사용자>의 눈에 전달한다.



# 클래스 구조도 (Class Diagram)

- 객체 지향 설계 방법론에서의 설계도
- 클래스 (class)
  - 객체를 생성하기 위한 틀
  - 메소드를 갖고 있다.
  - 객체 vs 클래스: <화면> 클래스로부터 <화면1>, <화면2> 객체를 생성한다.
- 클래스 구조도
  - 클래스와 클래스간의 소통을 그려놓은 설계도

# 이태리 요리집의 클래스 구조도



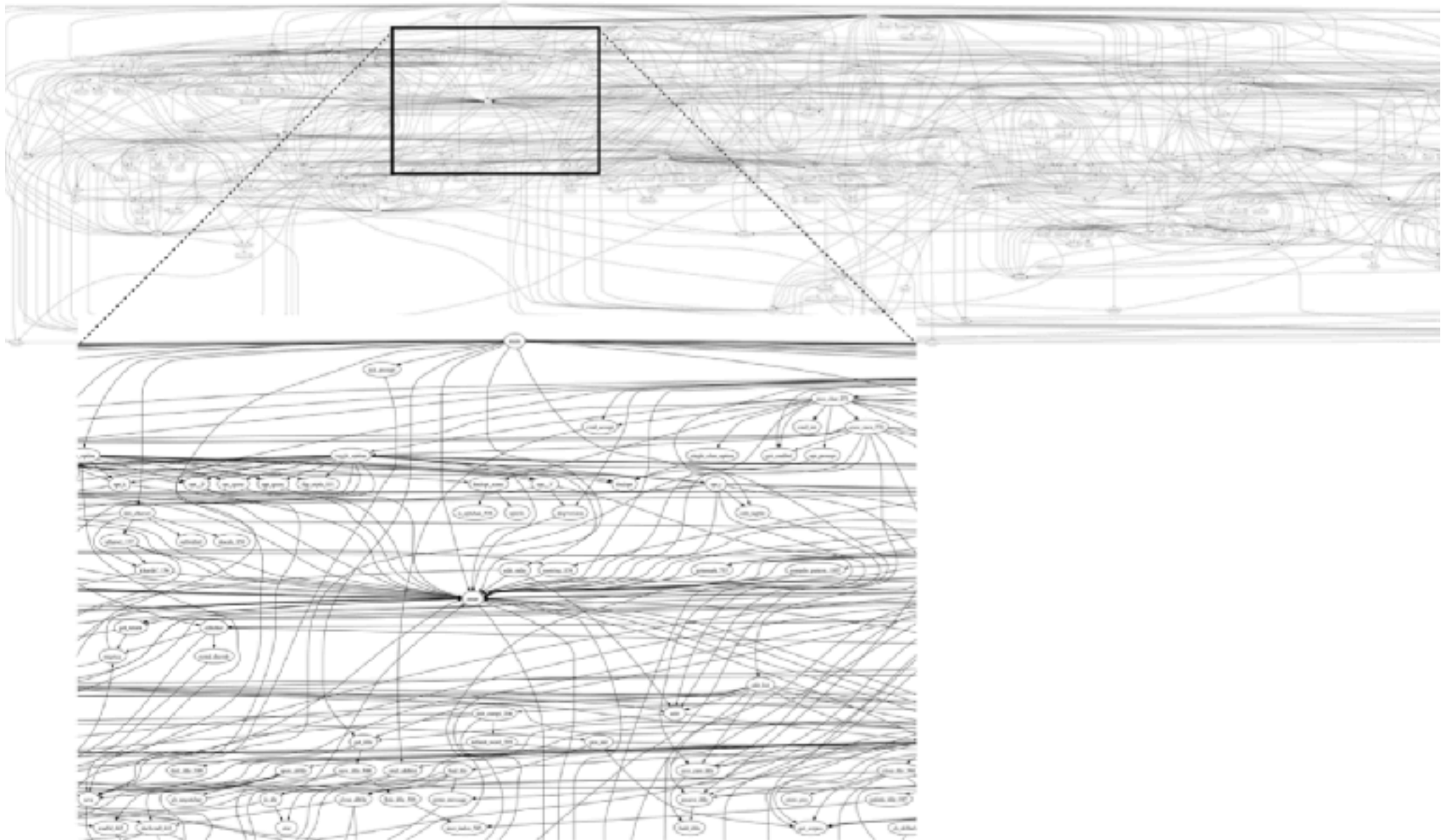
바람직한 클래스 구조도를 그리는 기술은 **ELE326 객체지향개발론**

# 프로그래밍의 어려움

- 프로그램의 규모와 복잡도가 점점 커짐
  - SW 복잡성의 증가속도  $\gg$  HW 성능의 성장 속도
  - “SW 는 가스다.” “Software is gas.”

# 프로그래밍의 어려움

- SW 복잡성의 예: less-382 (23,822 LoC)



# 프로그래밍의 어려움

- 과연 제가 짠 프로그램이 제 의도대로 돌아가요?
- 프로그램의 실행을 “미리 완벽히 알기”가 어려움
  - 자동으로 완벽히는 불가능: 1936년 Turing 에 의해 증명됨
  - 사람이 확인해야: 다양한 도구로 비용 절감
    - 구문 검사
    - 타입 검사
    - 의미 기반 검사 (정적 / 동적 분석)
- 프로그램 안전성 확보를 위한 기술은 **CSEXXX 프로그램 분석**

# 객체 지향 개발 방법론의 실제 적용 사례

- 개인적 경험: UC Berkeley Dawn Song 교수 연구팀 방문연구
- DroidBlaze: 임의의 안드로이드 앱에 대해서 나쁜 앱으로 의심되는 정도와 그 근거를 보고하는 시스템
- 앱이 하는 나쁜 짓의 예
  - 사용자 몰래 개인정보 누출
  - 유료 전화/문자 발송
  - 다른 나쁜 앱을 기존 앱의 업데이트를 가장하여 설치
- 앞서 말한 프로그램 분석 + 기계학습 기술 적용


# 객체 지향 개발 방법론의 실제 적용 사례

APK file: ADEB-4015.apk  
 Suspicious behaviors: 20  
 Using dynamic results: true  
 Threat level: 10/10

## Summary of behaviors

Category	Threat Score
CODE_SIMILARITY	10
SMS_SENDING	2
SENSITIVE_INFORMATION	10
OBFUSCATION	6
NETWORK	0
FILESYSTEM	6



## CODE\_SIMILARITY

Threat	Behavior	Stat.	Dyn.	Confidence
high	Match to malware family: ADR.D/Geinimi 	✓		<div><div></div></div>
(Category threat level: 10/10)				

## SMS\_SENDING

Threat	Behavior	Stat.	Dyn.	Confidence
low	SMS sending (dst: UNKNOWN, text: UNKNOWN) 	✓		<div><div></div></div>
(Category threat level: 2/10)				

## SENSITIVE\_INFORMATION

Threat	Behavior	Stat.	Dyn.	Confidence
high	Sensitive information updated (Preferred APN) 	✓		<div><div></div></div>
high	Sensitive information read (Preferred APN) 	✓		<div><div></div></div>

# 객체 지향 개발 방법론의 실제 적용 사례

## ○ 모듈화(modularization)

- 정적 분석 — 앱을 실행시키지 않고 분석
- 동적 분석 — 앱을 직접 실행시키며 분석
- 앞단(Frontend) — 안드로이드 앱 바이트코드를 분석하기 용이한 비교적 간단한 언어로 쓰인 프로그램으로 변환
- 뒷단(Backend) — 정적/동적 분석결과를 취합 후 최종 결과 보고서 출력

## ○ 인터페이스(interface) 를 활용한 프로그래밍

- 앞으로 새로 추가될 앱 분석 알고리즘이 구현해야할 명세

## ○ 상속(inheritance)

- 많은 분석 알고리즘들은 공통된 부분을 가지고 있음
- 공통된 부분이 쉽게 재사용되게 함

## ○ 메소드 오버라이딩(method overriding)

- 그럼에도 불구하고 분석 알고리즘 마다 약간씩의 차이 -- 필요한 경우 커스텀 화 가능케 함



# 객체 지향 개발 방법론의 실제 적용 사례

## ○ 앞선 방법론들 적용 결과

### ○ 인터페이스, 상속 및 오버라이딩:

기존 알고리즘으로 탐지 안되는 새로운 악성 앱 발견 시, 필요한 새로운 분석 알고리즘 추가가 매우 용이해 짐

(예: 1주일만에 개인정보 누출 여부 분석 알고리즘 구현 및 추가)

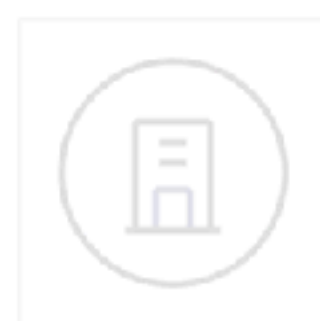
### ○ 모듈화:

- 분석 프로그램에 오류가 있을 때 어느 부분이 문제인지 쉽게 찾아낼 수 있고, 역할 분담이 용이.
- 부품을 갈아끼우기 쉬움.

# 객체 지향 개발 방법론의 실제 적용 사례

결과: 스타트업 Ensighta 창업 후 보안회사 FireEye 에 매각

## ENSIGHTA SECURITY



2011  
FOUNDED

M&A  
STATUS

1-10  
EMPLOYEES

M&A  
LATEST DEAL TYPE

\$3.2M  
LATEST DEAL AMOUNT

### Description

Provider of a malware detection platform. The company offers a software for the market-scale mobile malware static and dynamic analysis system.

Ownership Status  
Acquired/Merged

Primary Industry  
Social/Platform Software

Primary Office  
2003 Vine Street  
Berkeley, CA 94709  
United States

Financing Status  
Formerly Accelerator/Incubator...

Acquirer  
FireEye

## Ensighta Security Valuation and Funding

Deal Type	Date	Amount	Raised to Date	Post-Val	Status	Stage
3. Merger/Acquisition	20-Dec-2012	\$3.2M			Completed	Startup
2. Accelerator/Incubator					Completed	Startup
1. Grant	01-Jan-2012	\$100K			Completed	Startup

# 중요

- 좋은 프로그래머란
  - 좋은 건축가가 되는 것과 같은 것
- 좋은 프로그래머가 되려면
  - 표준 예제들을 익히고
  - 조합의 기본 기술들을 익히고
  - 연습하는 것

# 실습

- CodeOnWeb에서 수행
  - 온라인 사이트 가입 만으로 실제 코딩환경에서 코딩 실습
  - 인터넷이 되는 환경이면 어느 디바이스에서나 가능
  - codeonweb.com 에서 회원가입 후 이메일을 parkgunwoo@gmail.com 으로 전송