

프로그램, 프로그래밍, 프로그래밍언어

2016.3.16

충남대학교 컴퓨터공학과

조은선

프로그램 개발 잘 하기

1. 프로그래밍 언어를 잘 알고 잘 선택하기
2. 프로그램을 잘 작성하기
3. 컴파일 잘 해서 돌리기

1. 프로그래밍 언어 선택하기

“C, C++, Java ...”

훌륭한 프로그래밍 언어들, 그러나

프로그래머는, 머리 아프다

GUI 코딩 할 때

lock 걸려 고생 해봤는가? 마우스/키보드 말고 다른 장비가 붙으면?
그 장비들이 자기들끼리 일을 한다면?

Concurrency 코딩, thread 코딩을 심하게 해봤는가?

yield join 쓰기 쉬운가?

장비, 안전성

robot.moveTo(x,y) ; // 언제까지 기다릴 것인가?
그 동안 로봇 두 대가 부딪히기도 한다!!

빅 데이터 실시간 filtering 간단한가?

naver, google...



프로그래머를 구해주자!

“복잡한 것들을 모두 **abstract** 시키자”

“편하게 코딩? 효율적인 처리?” -> 둘 다 필요



그래서, 프로그래밍언어

모델을 주고

문법과 철학 : 응용 domain 에 맞게, 그래서 개발 시간 단축

도구를 준다

컴파일러, 가상 머신 : 기반 구동 체계까지 잘 되어 있어야

프로그래밍 언어의 종류

General purpose languages

C, C++, Java, C# ...

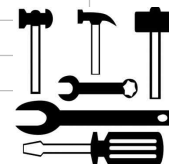
vs.

Domain specific languages

Domain에 따라 다름

그래서, 다양한 응용 Domain마다

| Domains | Domain Specific Approaches |
|---|---|
| Data processing | SQL |
| Web browsing | XML, Script languages |
| User centric interface | Graphical User Interface coding (MFC) |
| Games | Lua, ... other domain specific languages |
| Distributed programming | RMI, CORBA based languages |
| Web-based programming, frameworks | Ruby on Rails, Groovy.. |
| Big data, sensor processing | Map-reduce based programming, stream programming |
| Concurrent programming | Multi-thread programming |
| Asynchronous programming | Future based programming |
| Mobile programming, cyber physical system programming, IoT... | concurrency p.+ distributed p. + user centric + asynchronous p.+ data processing +... |





다른 뜻, 깊은 말..

Ubiquitous Computing

- ~ “Pervasive Computing”
- ~ “Context-Aware Systems”
- ~ “Cyber Physical Systems”
- ~ “Internet of Things”
- ~ “Smart Systems”
- ~ “Ambient Intelligence”
- ~ ...



여기서는 편의상 “Smart System” 이라고 하자

Device Example) Smartdust

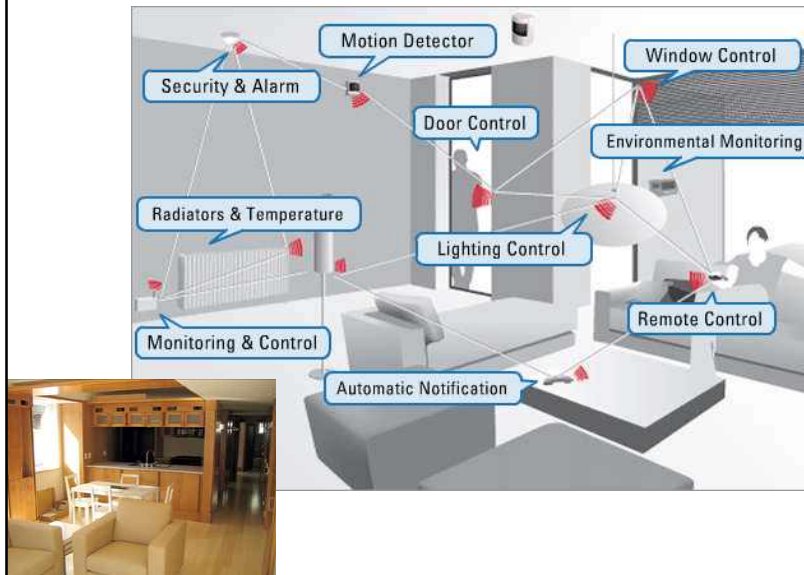
- A hypothetical system of many tiny microelectromechanical systems (MEMS) such as sensors, robots, or other devices, that can detect, for example, light, temperature, vibration, magnetism or chemicals; are usually networked wirelessly; and are distributed over some area to perform tasks, usually sensing.



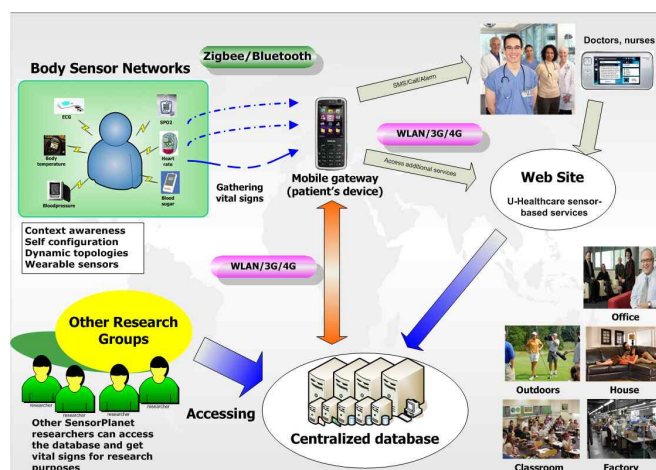
Application Example) Motion Detection



Application Example) Smart Home



Application Example) Smart-Health System



Smart Health System (conts')

- Smart sensors interacting with servers (or doctors)



Application Example) Smart Moving Objects (Robots)



다시, 프로그래밍 모델



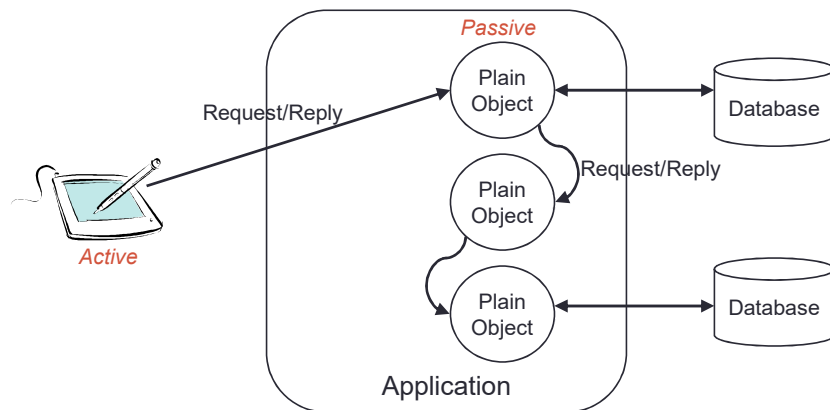
Smart Systems;

좀 프로그래밍 방법이 다르다.

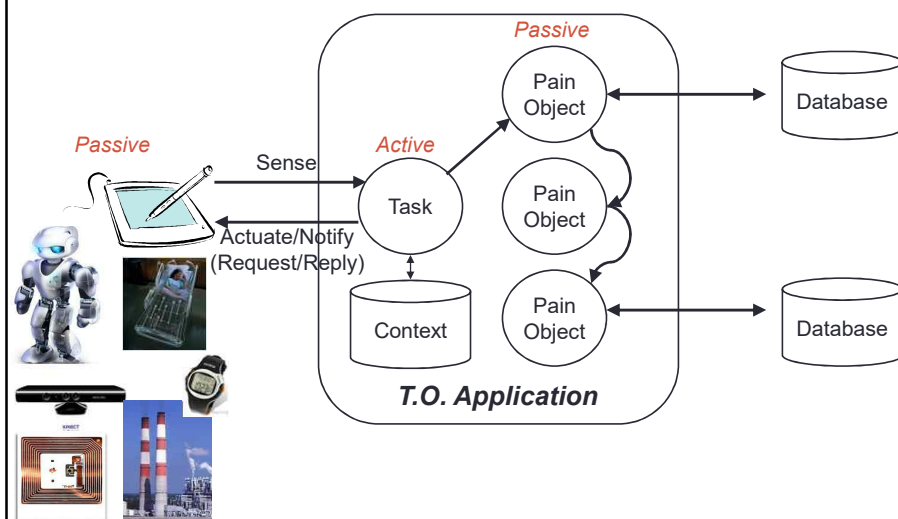
[1] Smart Computing needs....

Reactive Computing

기존의 응용 System: **Passive**



Context-Aware System: **Reactive**



참고 General GUI 코딩(Java)

```
public class AL extends Frame
    implements WindowListener, ActionListener {
    TextField text = new TextField(20);
    Button b;
    ...
    public AL(String title) {
        super(title);
        setLayout(new FlowLayout());
        addWindowListener(this);
        b = new Button("Click me"); add(b); add(text);
        b.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        numClicks++;
        text.setText("Button Clicked "
            + numClicks + " times");
    }
    ...
}
```

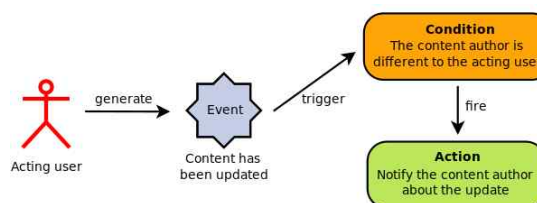


ECA Rules (based Programming)

“프로그램” == “규칙 집합”

Event-Condition-Action Rules

On event : 어떤 Event가 발생하고
 If condition 어떤 condition을 만족하면
 Then action 지정된 Action을 수행하라



Database에서 처음 사용됨 (“active database”)

GUI 프로그래밍, Game 프로그램에서도 유사 기능이 있음

예1

```
define event e1 = I48[0,100]
define event e2= C23(0)
```

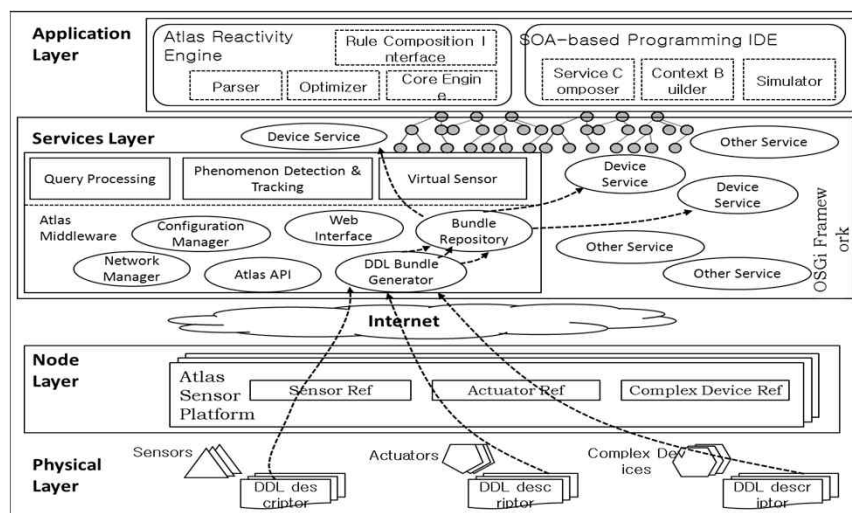
```
define condition c1 = true
define action a1 = B38(20)
define action a2 = B38(40)
```

```
config I48 1000
```

```
define event e3 = e1 * e2
define event e4 = e1 + e3
define action a3 = a1;a2
define rule r1 = e3, c1, a3
define rule r2 = e4, c1, a3
```

ATLAS

예1 ALTAS(Univ. of Florida)



예2 How About RxJava?



```
myObservable
    .map(s -> s + " -Dan")
    .map(s -> s.hashCode())
    .map(i -> Integer.toString(i))
    .subscribe(s -> System.out.println(s));
```

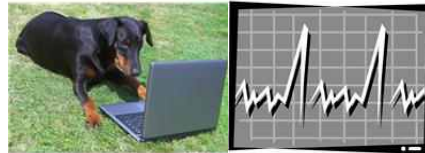
Naver DEVVIEW 동영상 <http://devview.kr/2014/session?seq=4>
Grokking RxJava: <http://blog.danlew.net/2014/09/15/grokking-rxjava-part-1/>

[2] Smart Computing needs....

**Event Stream Based
Programming**

E-C-A 중에서 Events의 감지 및 처리

- 다양한 발생 빈도
 - 한참 있다가 발생하기도 하고 eg. 사람이 방에 들어옴
 - 계속 발생하기도 함 eg. 주행 중 자동차의 위치 변화
- 가장 빈도가 큰 쪽을 고려해서 처리해야함
- 그러나 메모리에 많은 데이터를 넣어둘 수 없음



Event Stream Processing (ESP)

- 실시간성을 가지고 스트림으로 입력되는 이벤트를 처리하는 기법
- 또 다른 관점
 - 이벤트를 가공해서 실시간으로 새로운 복합이벤트를 만들어야 하는 mission
 - 예) “사람의 양 어깨의 위치가 빠른 속도로 내려가고 전체 적인 모양이 흔들렸다.” → “넘어졌다.”
 - $ESP \approx CEP$ (Complex Event Processing)

예1 ARENA: Exception Description for ATLAS

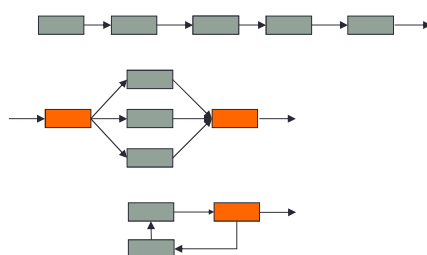
- Smart computing 응용에서의 **Exception**
 - 기술하기가 일반 프로그램보다 복잡하다.
 - 보통 프로그램 : “사람이 들어오면 불이 켜져라”
 - 예외 : “사람이 오늘 하루 동안 두 번 이상 들어오기만 하고 나가지는 않았으면 예외 상황을 알린다. 두 번 이상 들어온 간격이 30초 미만이면 1번으로 간주한다.” “불이 계속 깜박거리면 예외상황을 알린다”
- **Situation** based exception handling
 - 예외상황 표시를 편안하게
 - 복잡한 예외상황 처리를 빠르게



by PLASLab

예2 StreamIt

- Structured stream processing
 - Hierarchical structures:
 - Pipeline
 - SplitJoin
 - Feedback Loop
 - Basic programmable unit: Filter
 - Splits / Joins are compiler-defined



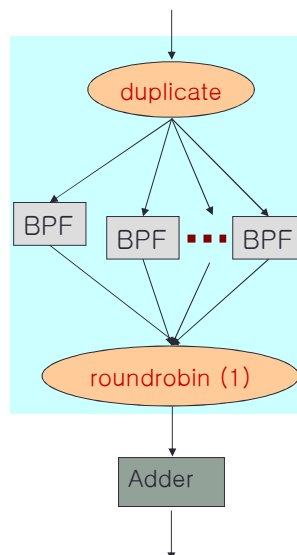
by MIT

SplitJoin Example: Equalizer

```

pipeline Equalizer (int N) {
  add splitjoin {
    split duplicate;
    float freq = 10000;
    for (int i = 0; i < N; i ++, freq*=2) {
      add BandPassFilter(freq, 2*freq);
    }
    split roundrobin;
  }
  add Adder(N);
}

```



참고: “나는 프로그래머다”

- <https://iamprogrammer.io/>
스트림처리 관련 클립
(핀테크, 금융데이터 분석 서비스편)



예3 Using Stream Queries

Stream SQL, CQL Continuous Query Languages

```
select * from TemperatureEvent
match_recognize (
  measures A as temp1, B as temp2, C as temp3, D as temp4
  pattern (A B C D)
  define A as A.temperature > 100,
         B as (A.temperature < B.value),
         C as (B.temperature < C.value),
         D as (C.temperature < D.value) and
         D.value > (A.value * 1.5))
```

more examples) Esper EPL, MS Stream Insight , Oracle Complex Event Processing(CEP) ...

예4 How About Stream in Java™ 8?

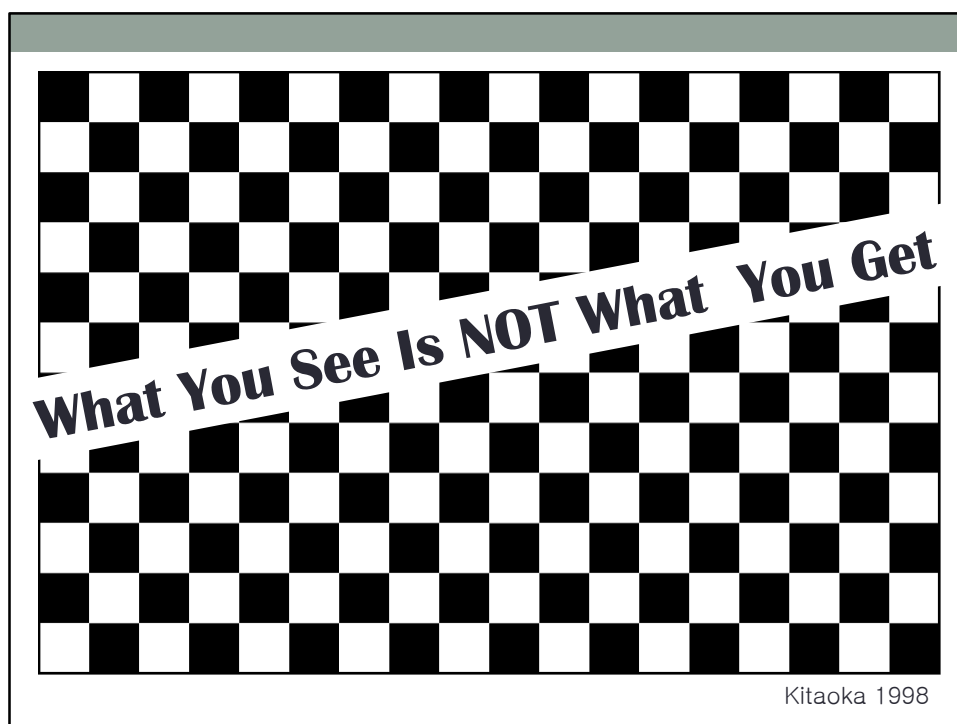
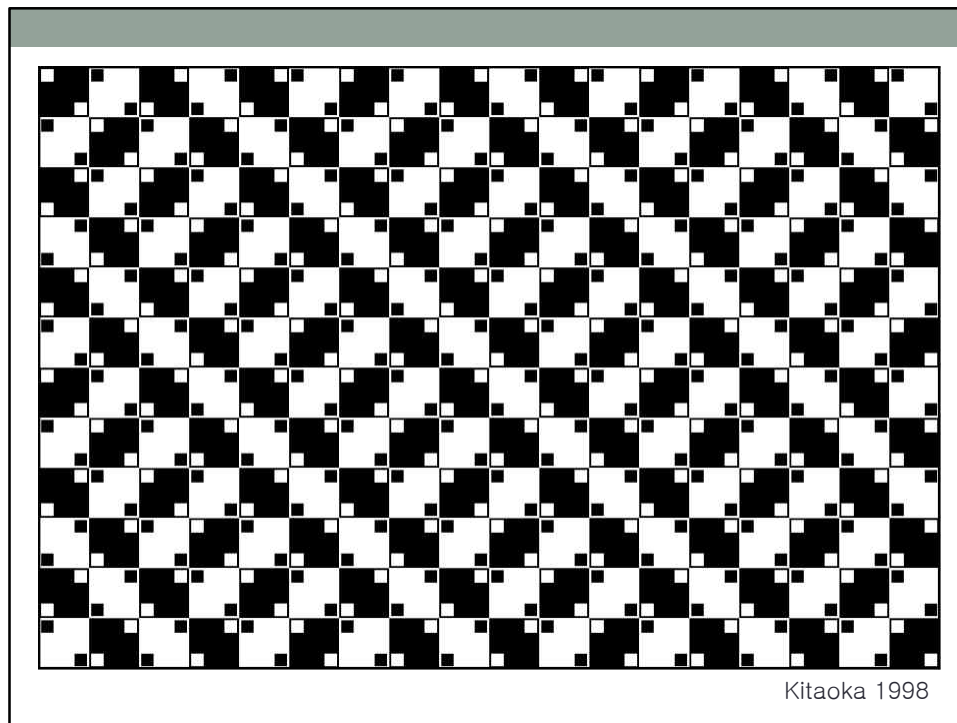
```
List<String> myList
    = Arrays.asList("a1", "a2", "b1", "c2", "c1");
myList
    .stream()
    .filter(s ->s.startsWith("c"))
    .map(String::toUpperCase)
    .sorted()
    .forEach(System.out::println);
```

2. 코딩 하기



네덜란드
De Graafschap
미드필더
(대략:1967~1981)

- 코딩을 잘 모르는 프로젝트 관리자가 관리하는 프로젝트?
- 코딩을 잘 모르는 설계자가 설계한 프로그램 설계?



1. Redefining Fields

```
class Super{
    public int a=1;
};

class Sub extends Super {
    public boolean a=true;
    public static void main(String[] args){
        Sub    s1 = new Sub();
        Super  s2 = s1;
        System.out.println(s1.a);    //?
        System.out.println(s2.a);    //?
    }
}
```

39

1. Redefining Fields conts'

```
class Super{
    public int a=1;
};

class Sub extends Super {
    public boolean a=true;
    public static void main(String[] args){
        Sub    s1 = new Sub();
        Super  s2 = s1;
        System.out.println(s1.a); //?
        System.out.println(s2.a); //? access hidden var
    }
}
```

Two different a's are coexist in Sub! ()

answer →

| |
|------|
| true |
| 1 |

similar result if both are int (or boolean)

| Name | Value |
|------|-------------------|
| args | String[0] (id=17) |
| s1 | Sub (id=18) |
| a | 1 |
| a | true |

40

1. Redefining Fields **conts'**

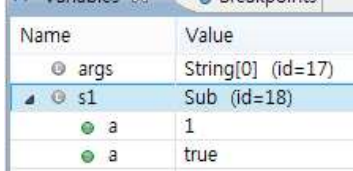
```

class Super{
    private int a=1;
};

class Sub extends Super {
    public boolean a=true;
}

public static void main(String[] args){
    Sub s1 = new Sub();
    Super s2 = s1;
    System.out.println(s1.a); // true
    System.out.println(s2.a); // compile error! (private field
    access)
}

```



Two different a's are coexist in Sub! ()

41

2. Overriding Methods

```

class Super{
    public int f() { return 1; }
};

class Sub extends Super {
    public boolean f() { return true; }
    public static void main(String[] args){
        Sub s1 = new Sub();
        Super s2 = s1;
        System.out.println(s1.f()); //?
        System.out.println(s2.f()); //?
    }
}

```

42

2. Overriding Methods conts'

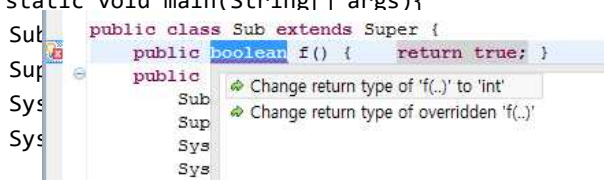
```

class Super{
    public int f() { return 1; }
};

class Sub extends Super {
    public boolean f() { return true; }
    public static void main(String[] args){
        Sub s1 = new Sub();
        Super s2 = s1;
        System.out.println(s1.f());
        System.out.println(s2.f());
    }
}

```

Compile error ! (Not allowed for overriding function)



43

3. More on Overriding Methods

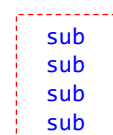
```

class Super{
    public Super g()
    { System.out.println("super"); return new Super(); }
};

class Sub extends Super {
    public Sub g()
    { System.out.println("sub"); return new Sub(); }
}

public static void main(String[] args){
    Sub s1 = new Sub();
    Super s2 = s1;
    Sub s3 = s1.g();
    Super s4 = s2.g();
    Sub s5 = s1.g();
    Super s6 = s2.g();
}

```



Overriding function! Why?

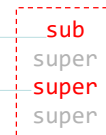
44

3. More on Overriding Methods conts'

```
class Super{
    public void h(Super s) {System.out.println("super");}
};

class Sub extends Super {
    public void h(Sub s)    {System.out.println("sub");}
}

public static void main(String[] args){
    Sub    s1 = new Sub();
    Super  s2 = s1;
    s1.h(new Sub());
    s2.h(new Super());
    s1.h(new Super());
    s2.h(new Sub());
}
```



Both coexists in Sub, Not Overriding! Why?

45

Quiz. 결과는?

```
class Super
{ protected void g() System.out.println("super"); }
class Sub extends Super
{ public void g() { System.out.println("sub"); //overriding? yes }

// in the same package
Sub    s1 = new Sub();        s1.g();
Super  s2 = s1;                s2.g();
Super  s3 = new Super();      s3.g();

// in a different package
Sub    s1 = new Sub();        s1.g();
Super  s2 = s1;                s2.g();
Super  s3 = new Super();      s3.g();
```

46

Quiz. 답

```

class Super
{ protected void g() System.out.println("super"); }
class Sub extends Super
{ public void g() { System.out.println("sub"); //overriding? yes }

// in the same package
Sub   s1 = new Sub();      s1.g();
Super s2 = s1;             s2.g();
Super s3 = new Super();    s3.g();

// in a different package
Sub   s1 = new Sub();      s1.g();
Super s2 = s1;             s2.g();
Super s3 = new Super();    s3.g();

```

sub
sub
super

// sub
// compile error
// compiler error

47

Liskov Substitution Principle (LSP)

“If S is a subtype of T , then objects of type T in a program may be **replaced with** objects of type S without altering any of the desirable properties of that program (e.g., correctness). 1988”

그래서, subtype할 때는 이것을 지켜라

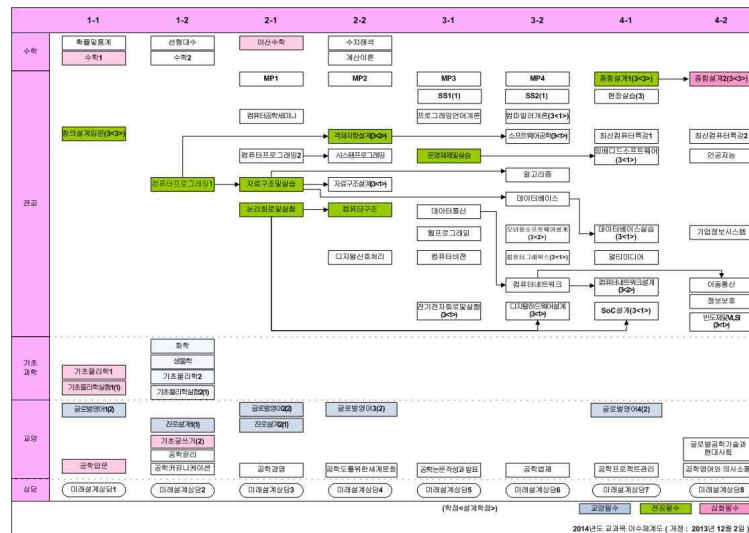
Contravariance
Covariance
No new exceptions

C에 포인터가 있다면,
Java에는 타입이 있다.
(**C++**은 둘 다 있다...)



48

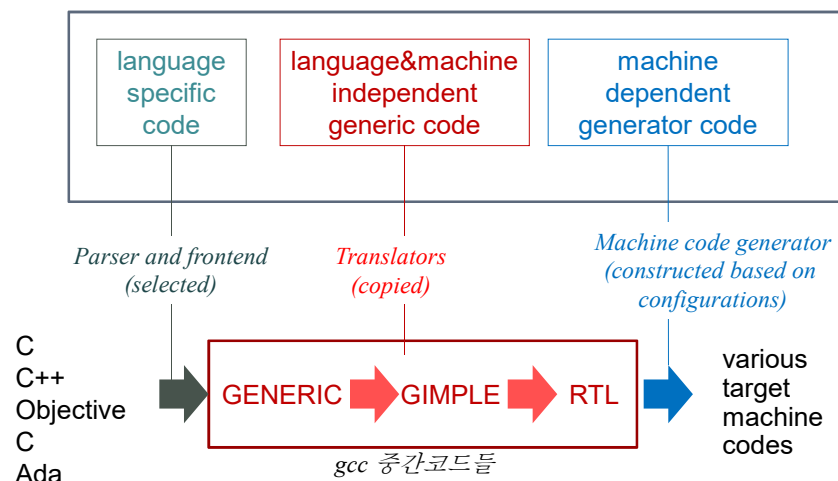
Solution: 많이 배우고 생각하기



3. 컴파일하기

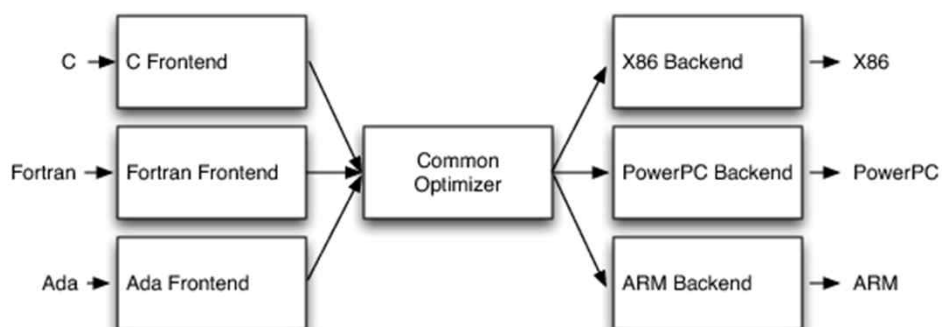
컴파일러를 잘 알고
컴파일러를 이용하기

GNU Compiler Construction Framework



LLVM

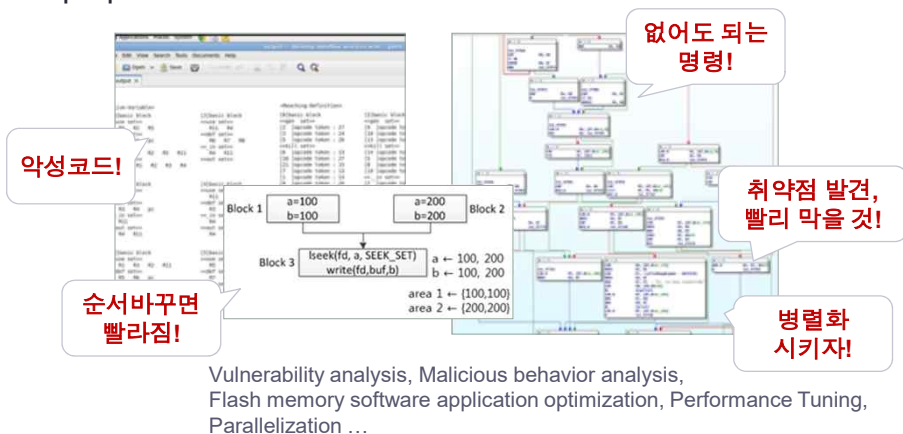
University of Illinois 에서 만든 컴파일러로 자체 IR을 보유
현재는 Apple에서 지원



Analysis (분석)

Dataflow analysis/ Control flow analysis

Loop optimization/ Instruction instrumentation / Parallelization...



프로그래밍 언어 및 시스템 연구실 PLAS Lab.

공 5502호, 구내7450

지도교수:

공5524호, 구내 6857

e-mail: eschough@cnu.ac.kr



주요 과제들

바이너리 코드에서 효율적인 역테인트 분석, 취약점 분석, 함수추출 기술 연구 : 미래부, 국가보안기술연구소

다중 장비 스마트 응용 프로그램을 위한 스트림 데이터 기반 예외처리 방법 :충남대학교, 미플로리다주립대, 한국연구재단, 한국전자통신연구소, (재)유비쿼터스사업단 등