Lecture 01
Monday January 06

What is a **Compiler**?

semantics-preserving

input
semantic domain

Input/Source
Language

rule for source (shape)

Input/Source
Program

e.g. Java

for (int i=0;...)

encoded into

passed to

Compiler

Software system

encoded into

generates

output
semantic domain

Output/Target
Language

meaning
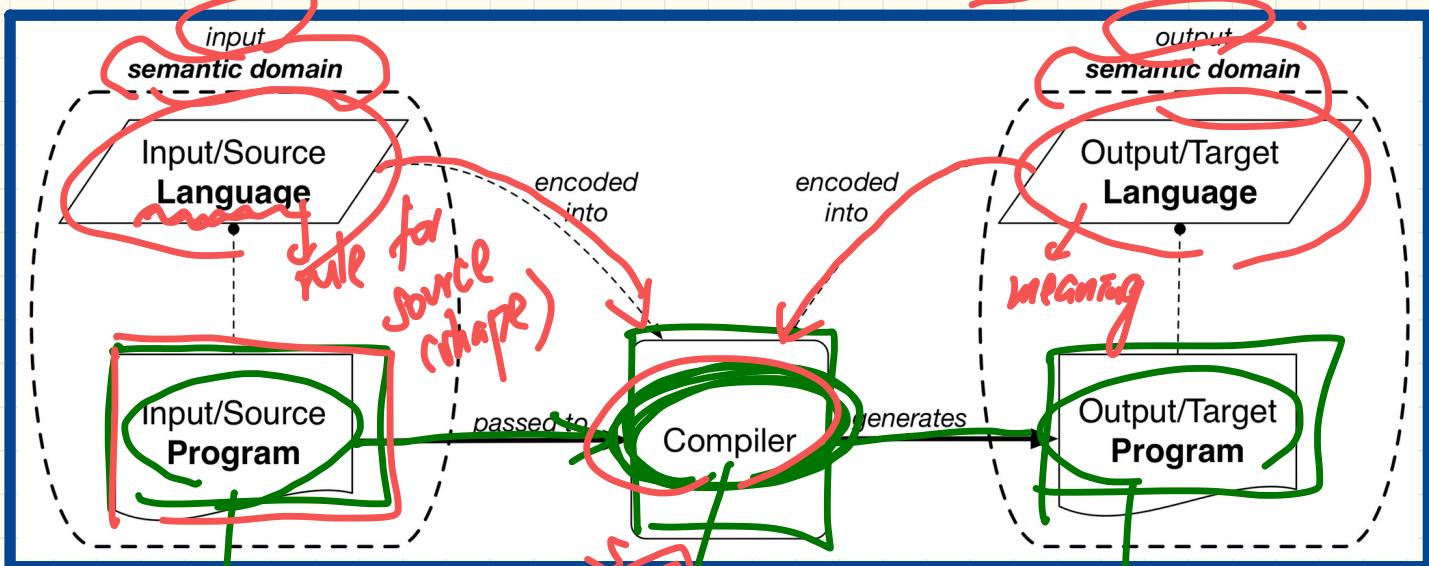
Output/Target
Program

if — byte code

SQL database

Compiler: Typical Infrastructure (1)
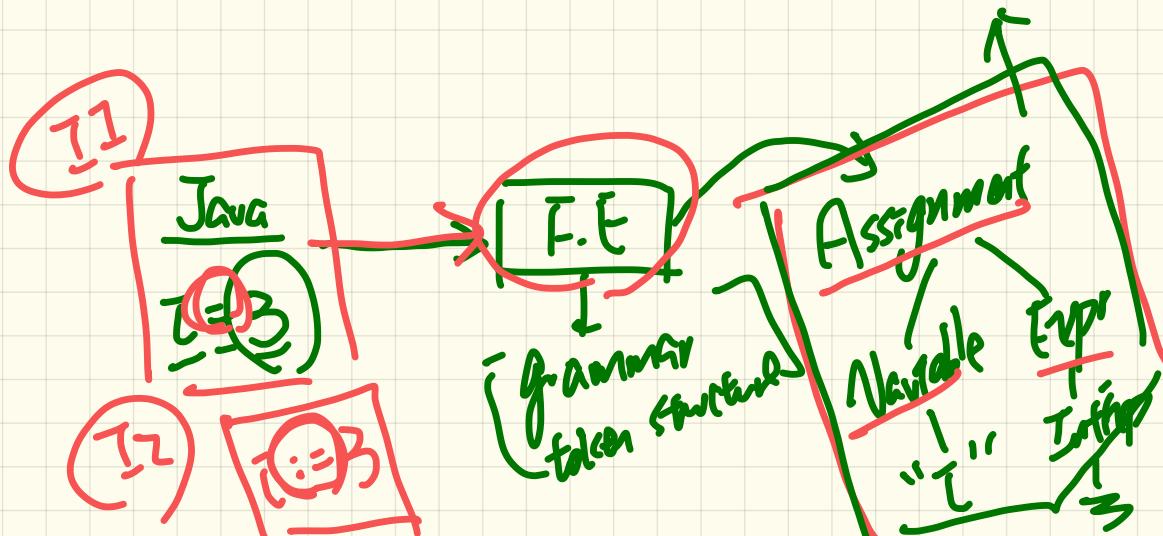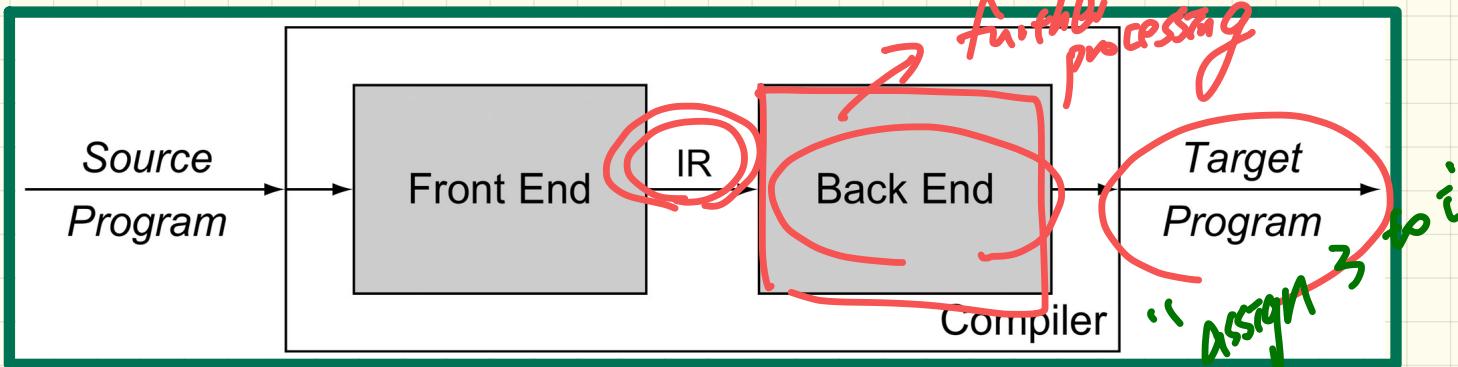
Source Program → Front End → IR → Back End → Target Program

Compiler

further processing

"assign 3 to i"

T1

Java

T2

F.E

grammar
token structure

Assignment

Variable   Expr

"i"   Integer

3

Q. How many **IRs** are necessary to build a number of compiler?

- **Java**-to-**C**
- **Eiffel**-to-**C**
- **Java**-to-**Python**
- **Eiffel**-to-**Python**

IR_J → how many Java features should be covered.

Java → IR_J → IR_C → C IR_E

Eiffel → IR_E → IR_C → C IR_C 1.5

IR_P

Q. How many **IRs** are necessary to build a number of compiler?

- **Java**-to-**C**
- **Eiffel**-to-**C**
- **Java**-to-**Python**
- **Eiffel**-to-**Python**

→ use Procedural

Procedural

OO

IR oo

Java

Eiffel

→ IR oo

→ C register work that's not just pretty printing
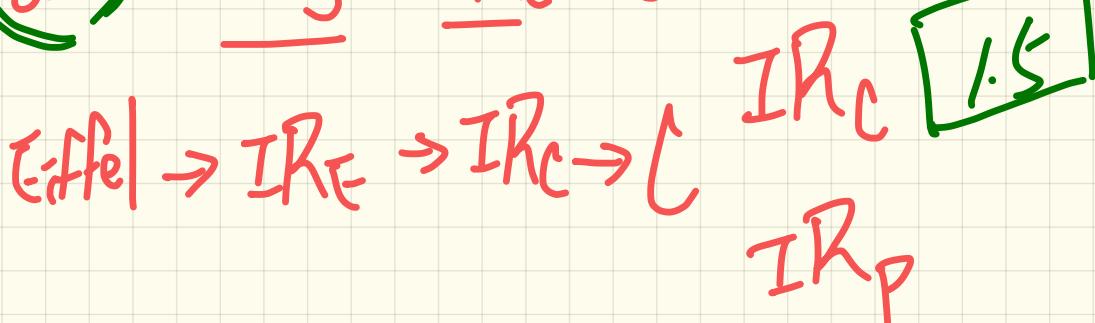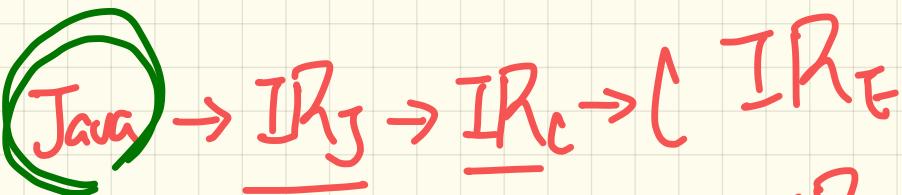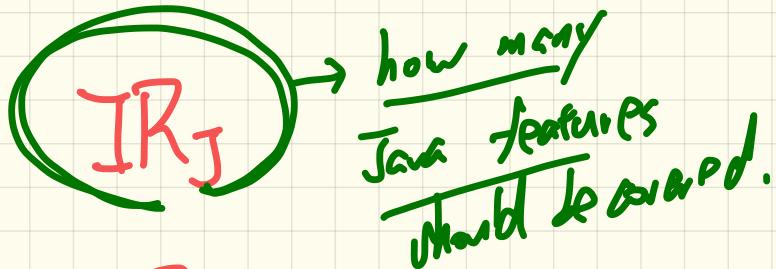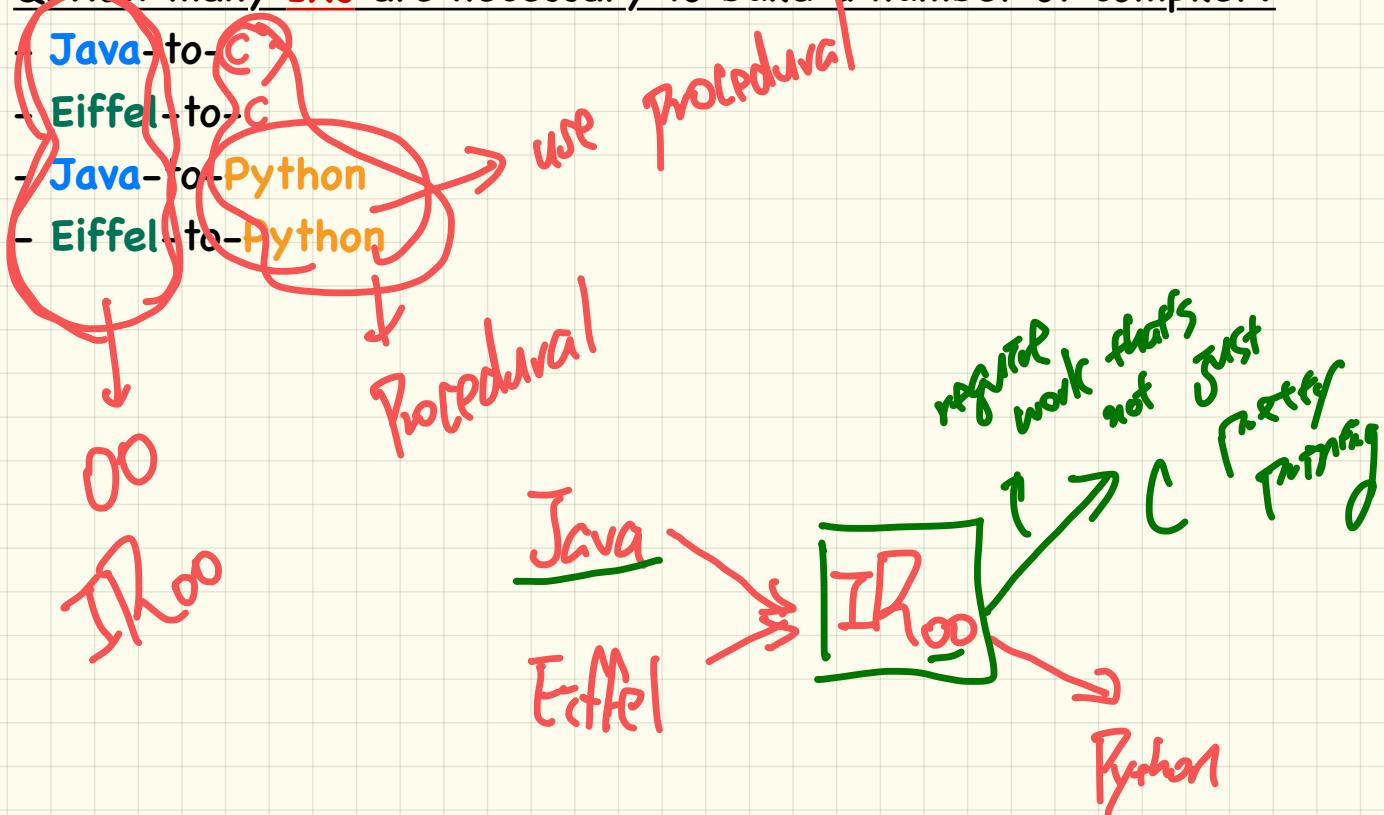
→ Python

One possible way.

Java

Eiffel

$Java \rightarrow IR_{OO} \rightarrow IR_{procedural}$

strategy of using procedural to implement OO

$IR_{procedural} \rightarrow$ C, Python, Java, Eiffel

Java - to - html compiler?

Java → [syntax tokens] --→ (中眼) --→ 眼眼 → *pretty printing* → html

Compiler: Typical Infrastructure (2)

USW

| Source Program | Front End | IR | Optimizer | IR | Back End | Target Program |

Compiler

C1
C2
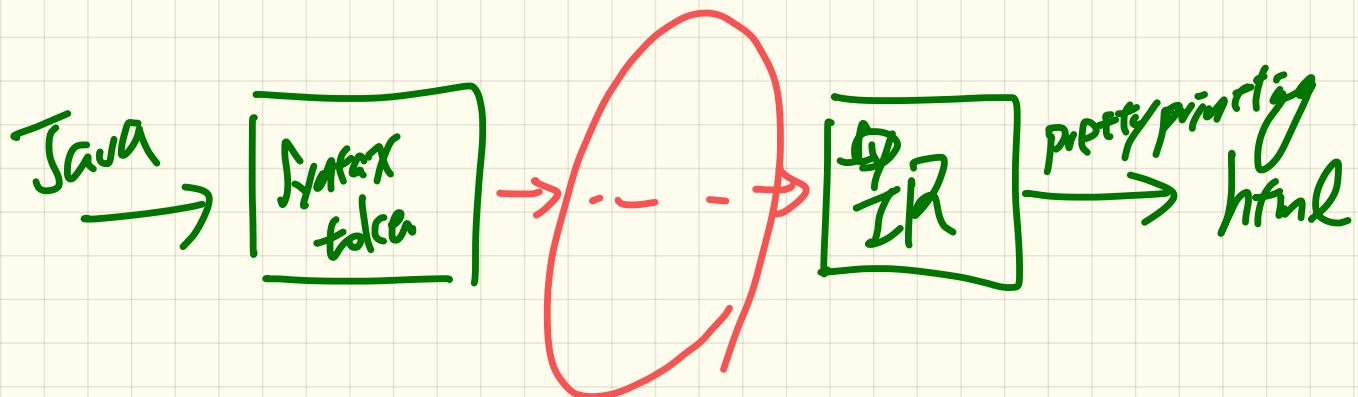
O1
O2
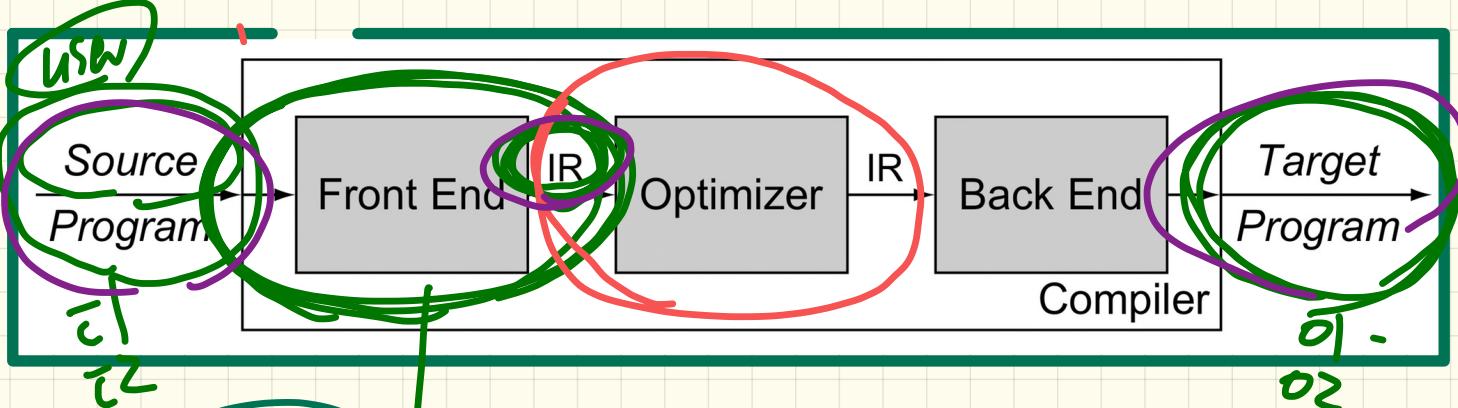
Q. What is the behaviour of the target program predicated upon?

1) Input-Output acceptance
Acceptance test.

1. Input to IR must be accurate

2) Unit testing (internal).

2. $IR_1 \Rightarrow IR_2 \Rightarrow IR_3 \leftarrow ? \rightarrow IR_n$

semantics preserving

3
$IR_n \rightarrow$ target

foo.java

Java M.

Java. A.

Java J.

foo! Cpp

fooI .Cpp

foo3 .Cpp

2. should be semantically same.
1. May be syntactically diffrent.
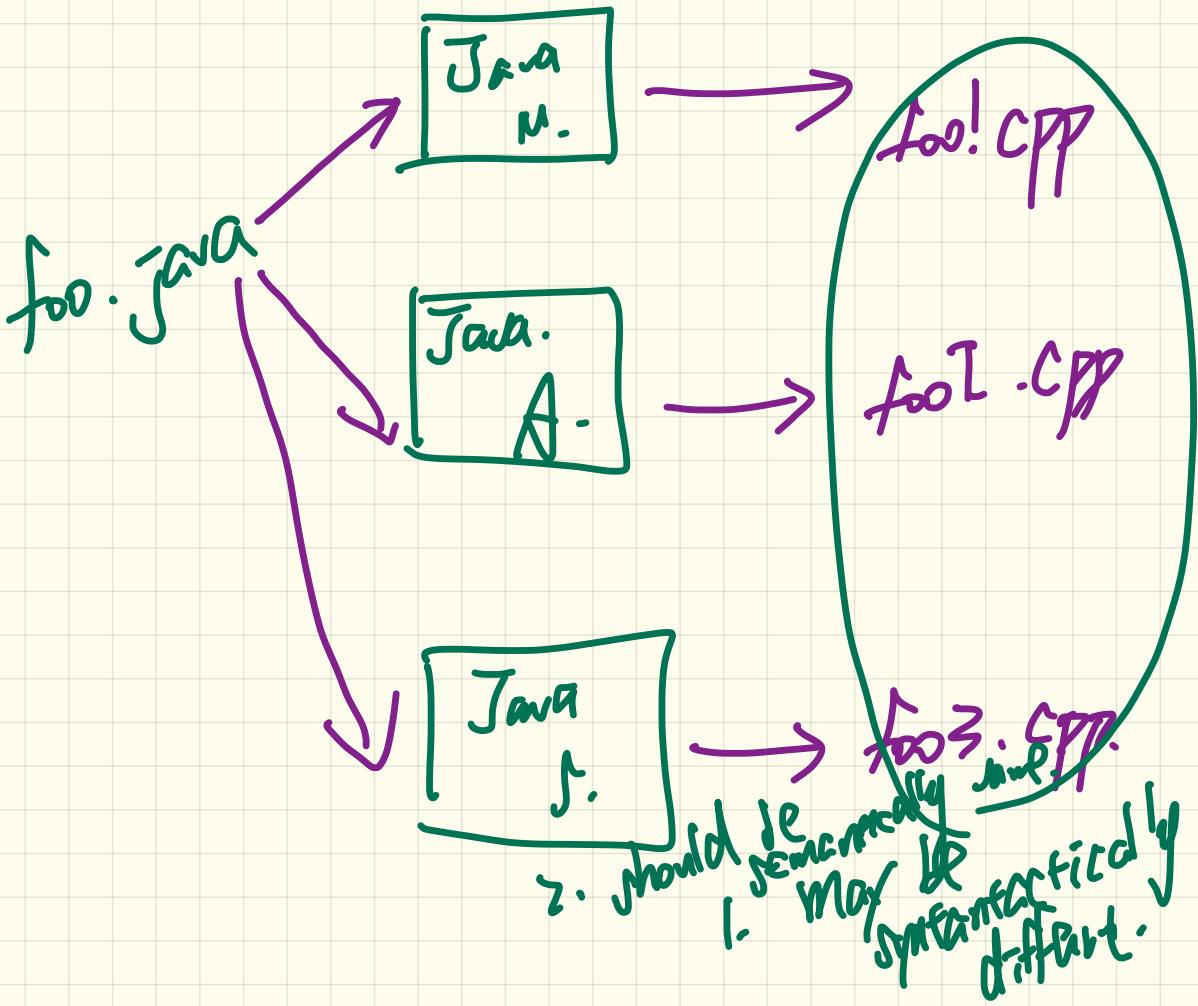
Example Compiler One: Infrastructure
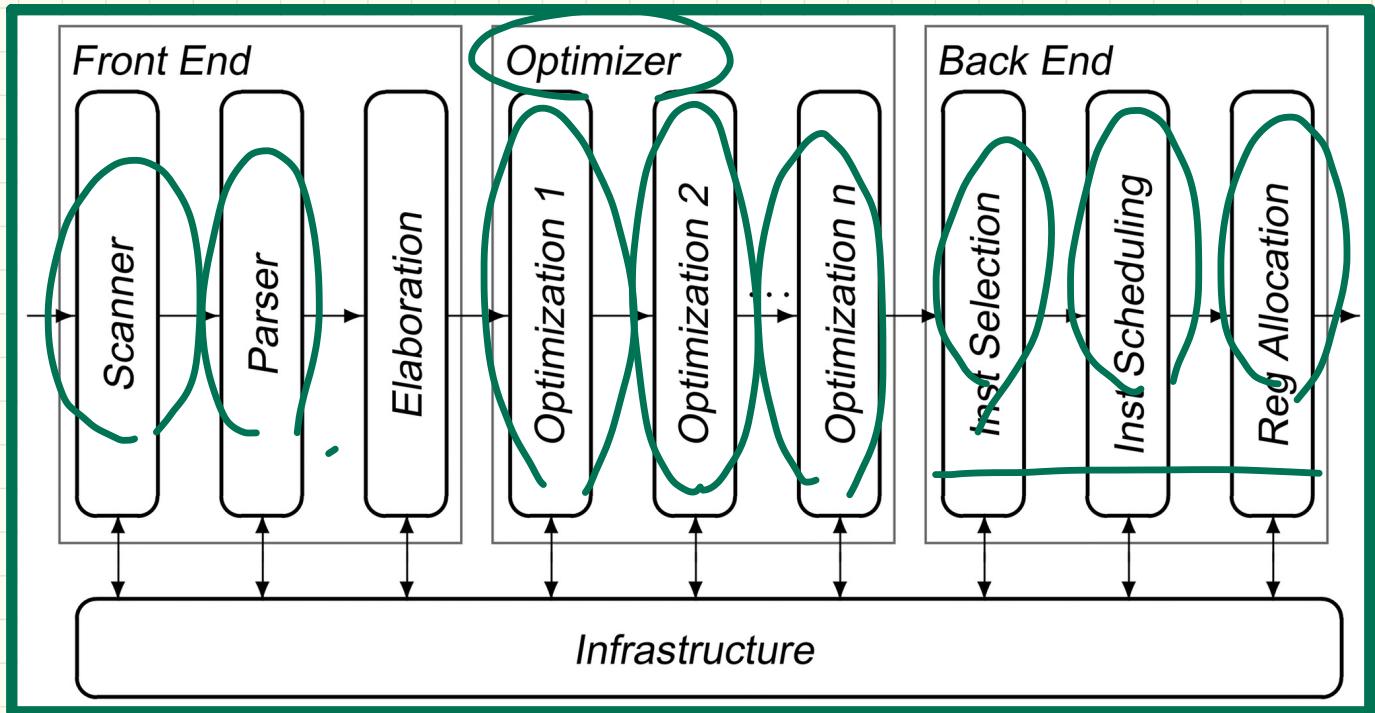
Example Compiler One: Scanner, Parser, Optimizer

Lexical Analysis

Syntactic Analysis

Semantic Analysis

Source Program
(seq. of **characters**) → Scanner → seq. of **tokens** → Parser → AST_1 → ... → AST_n → *pretty printed* → Target **Program**

characters
↓
words

words
↓
sentences

1. valid tokens
2. valid composition of tokens

subject   verb
(I) (play) (exam.)   noun

I eat exam.
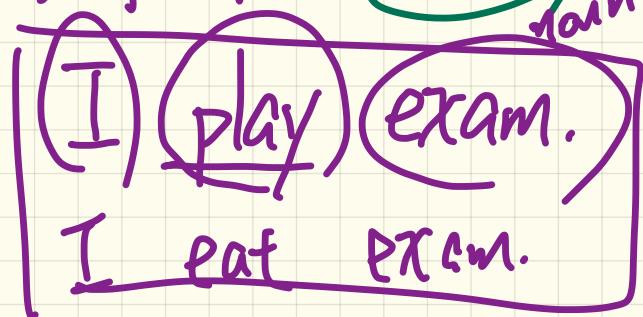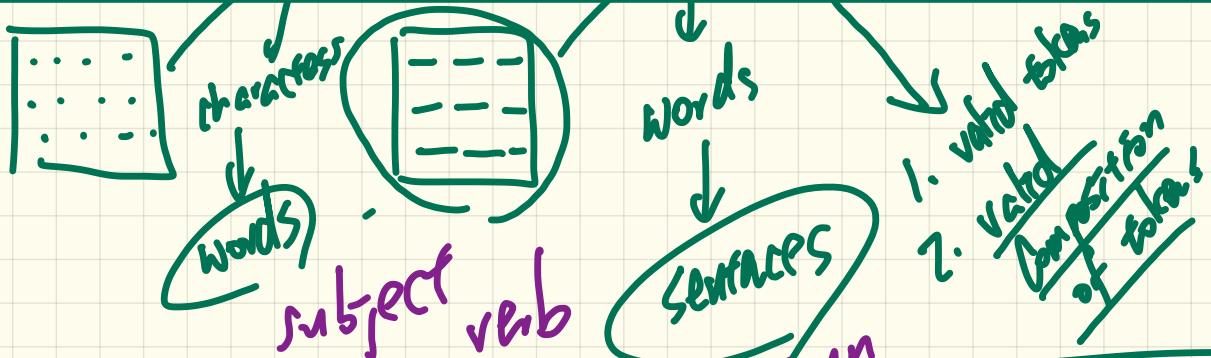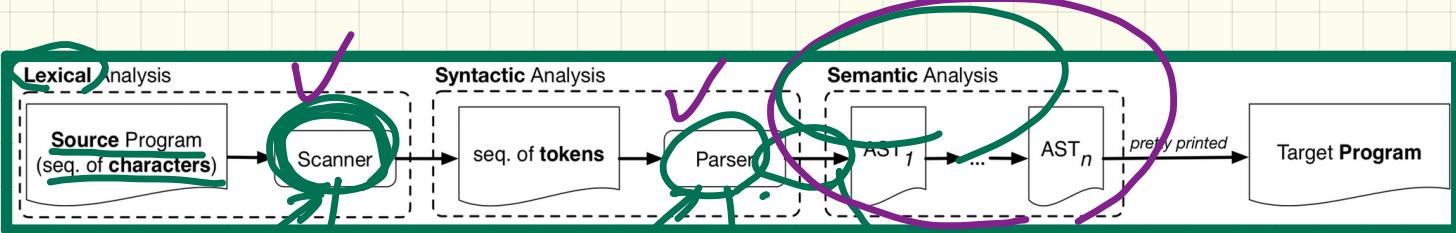
Class

Example   Java compiler   invalid

class @234Foo {

int a =

}

token
for identifier

Body of class   R BRAC

ID_Token   LBRAC

Non
TERMINALS

Token_CLASS

class

not matched:
parse
error.

pass scanner

class | Foo | {

int | a | =

}

```
class A {
    a := int
}
```
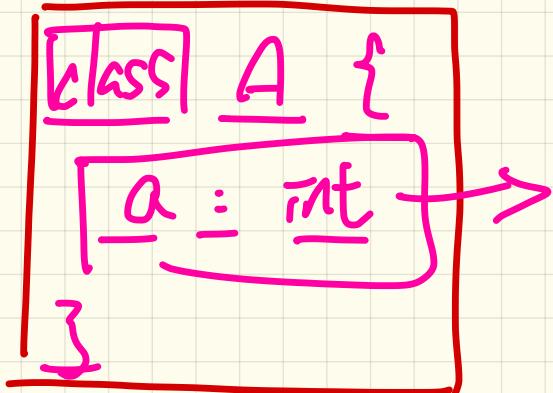
Composition of idioms for var. declaration in wrong order

```
class A {
    B b;
}
```
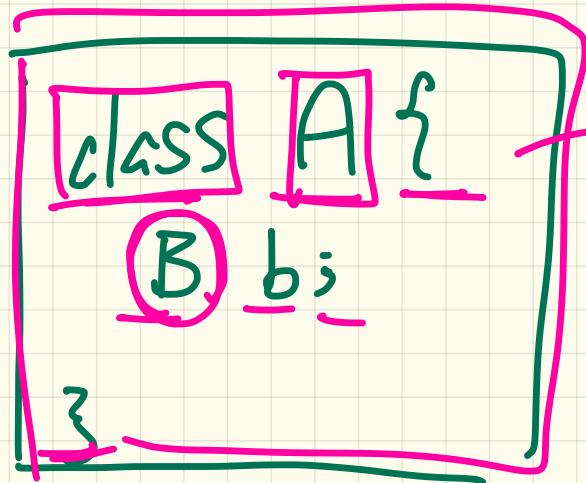
semantic error unknown class