Lecture 2 - May 8

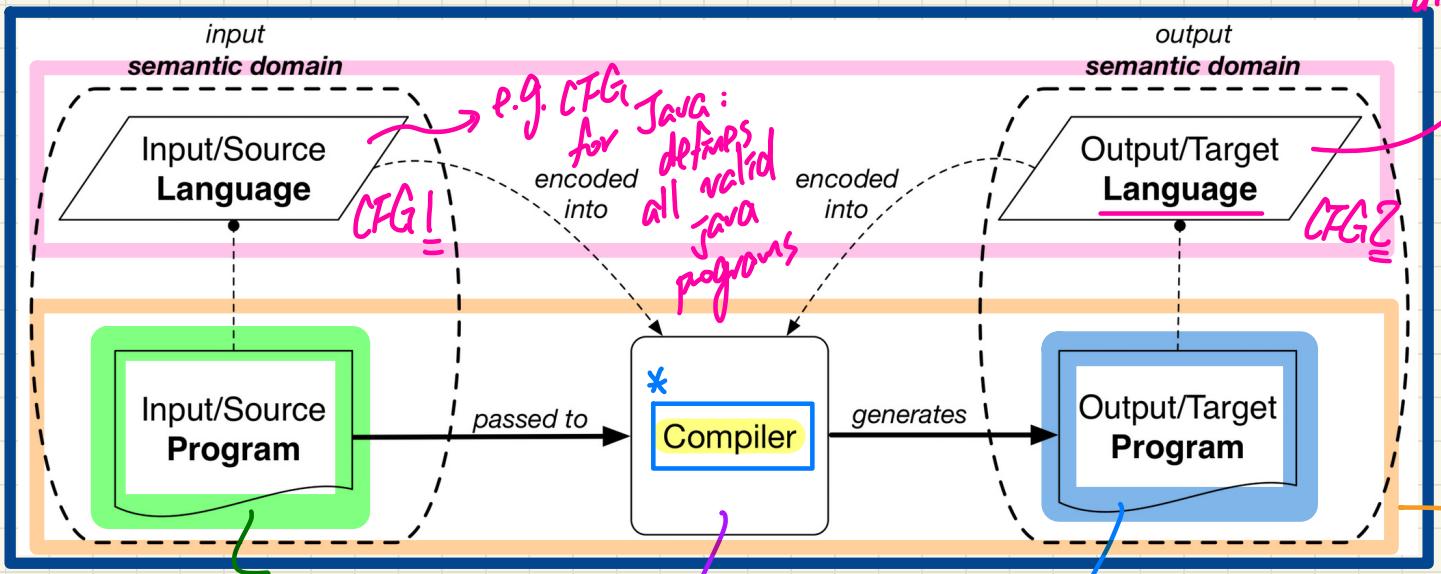
Overview of Compilation

What is a Compiler?
Inferring Dynamic Behavioiur
Design Principles
Intermediate Representation (IR)

* Implementation is defined at the CFG level (i.e., abstract syntax trees)

What is a Compiler?

e.g. object-relational bridge



e.g. CFG for Java: defines all valid Java programs

CFG1

CFG2

e.g. CFG for SQL defines all valid SQL programs

MyClass.java
↳ classes
↳ attributes
↳ methods (input instance)

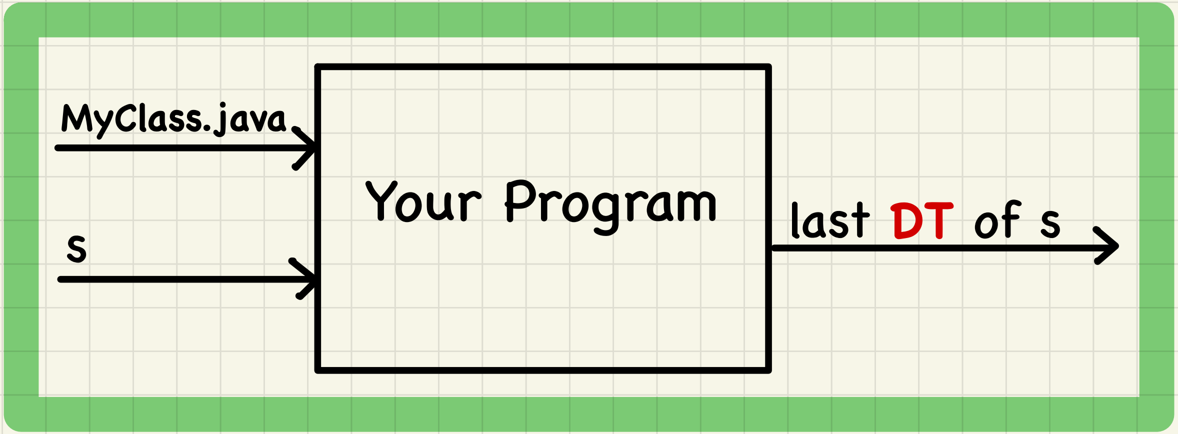
should preserve the meaning of the input OO program in the output RD program

MyDB.sql
↳ tables
↳ queries

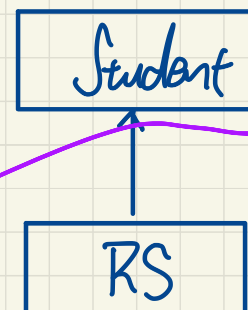
a single run of the compiler on specific input/output

** if the loop does not halt → your compiler also does not halt → unacceptable

An A+ Challenge: Inferring the DT of a Variable

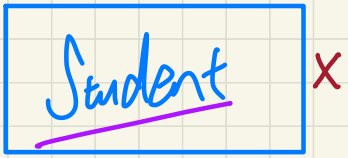



```
class MyClass {
  main (...)
    Student s = ...;
    ...
    s = new ResidentStudent(...);
  }
}
```

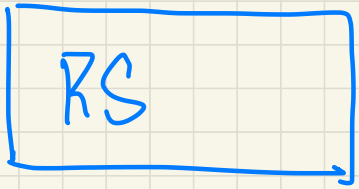


the best way for compiler to know if body of loop terminates is by ~~simulating~~ interpreting ** arbitrarily comp. cond.

while ( ) {
}



X



double getPremiumRate()

only exists in RS.

X

RS rs = new Student();

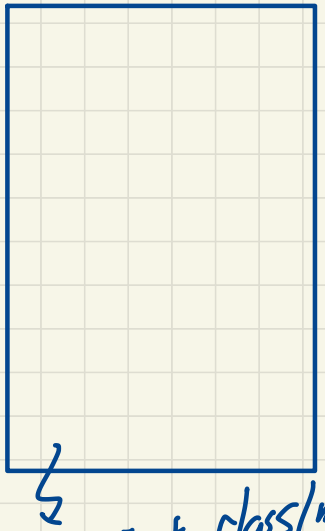
S → RS

Student S = new RS();

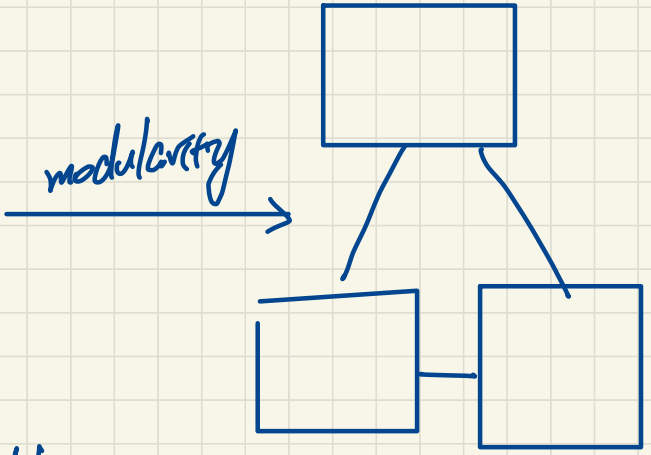
S.getPremiumRate()

a compiler's limitation is to infer the dynamic behaviour of a program.

"Superman" Class

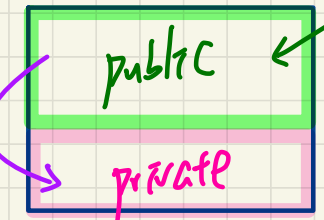


a giant class/module responsible for doing too many things



(separation of concerns)

Information Hiding



what external users depend on (e.g. ArrayList API page)

to implement the module, some public methods may be implemented using private attributes

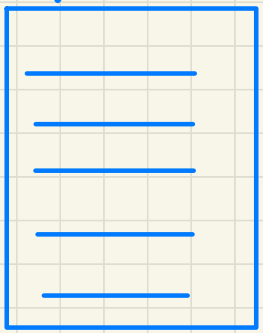
Principles

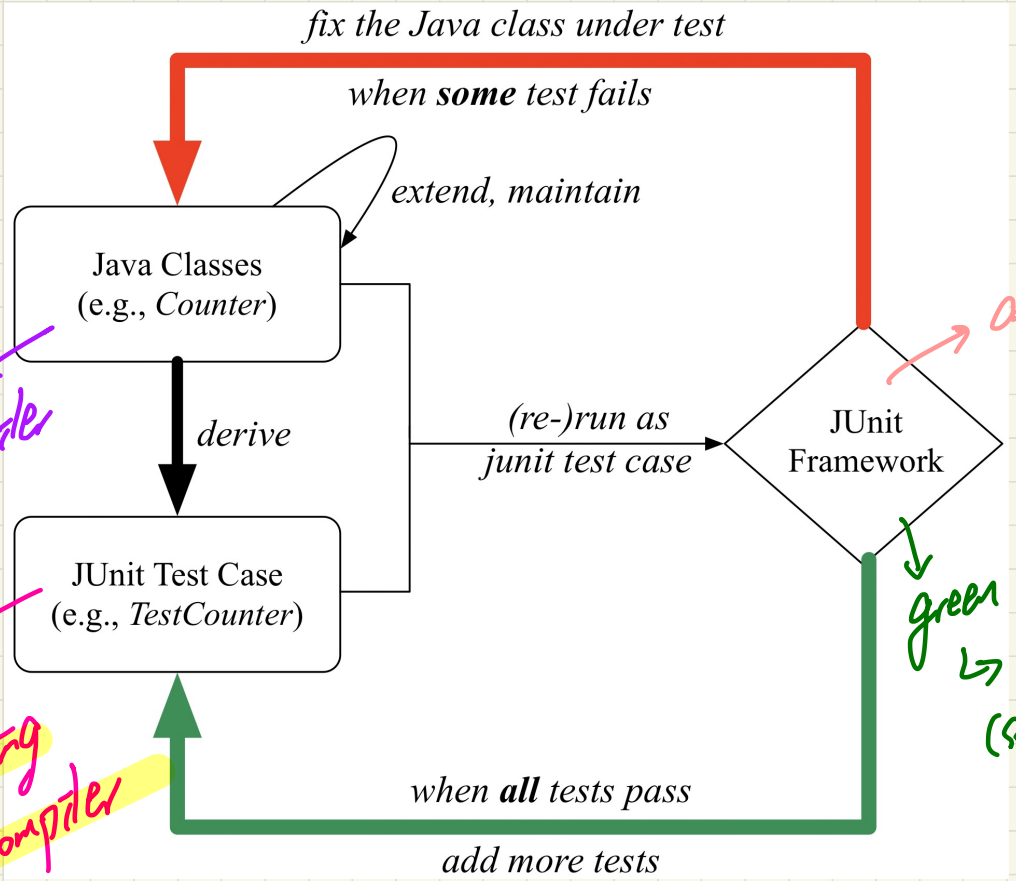
① changes to the modules should be done to the private part, without the user realizing it.

② if a part of module is changed often, it should be made private

Test-Driven Development (TDD): Regression Testing

readings





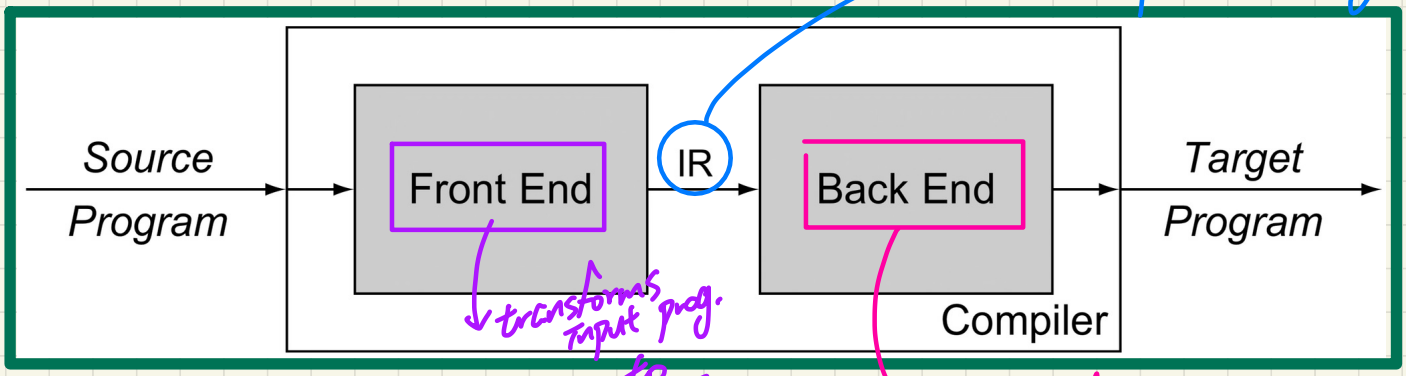
assume: Eclipse

imp. of compiler

you're expected to write Junit test cases while developing the compiler

green bar
↳ SUT is correct (software under test)

Compiler: Typical Infrastructure (1)



Intermediate Representation expressive enough to capture/encode ideas of any valid input

transforms input prog. to IR

transforms IR into output prog.

Concrete Syntax vs. Abstract Syntax

Q. How many IRs are necessary to build a number of compiler to transform between Java, C, C#, C++, and Python?

Hint: # of Permutations

How many IRs

Permutations

inputs (4): Java, C, C#, C++, Python

outputs: Java, C, C#, C++, Python

C-to-Java

* ~~Java-to-C~~ may not be feasible because Java can only be exp. using IR1 and IR3 (but IR2 is req. by C)

Total: 5 * 4 = 20 compilers

input. # output

Analysis of Semantic Domains