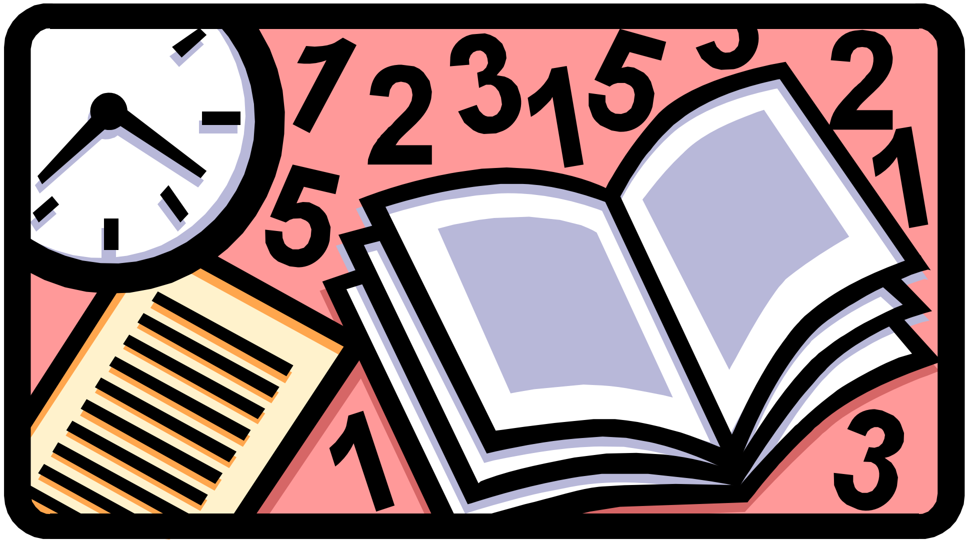
Software Tools
Type Systems

Part II - Lecture 9

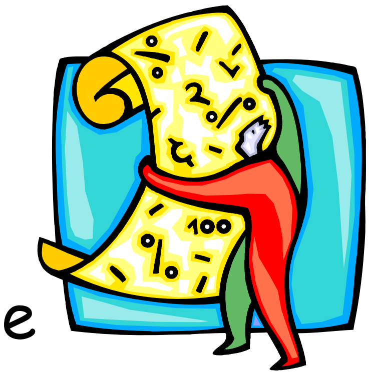
Today's Outline

- Introduction to Type Systems
- Simplified Java Type Rules
- Type Derivation

Assignment 2



Report Grading Schedule



Approx. **5 pages** (including figures) IEEE style

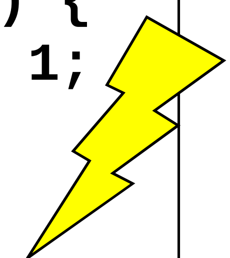
0. **IEEE style (5%), Abstract (5%)**
1. **Introduction (10%)**:
   Introduced & motivated the project and its aims?
2. **Related Work (20%)**:
   Cited & described academic related work (≥4)?
3. **Requirements (10%)**: What needed to be done & why?
4. **Design (20%)**: How did you design your solution?
   Why? Design alternatives? Strengths & weaknesses?
5. **Implementation (20%)**: How did you implement it?
   What did you contribute? The team work? Challenges?
6. **Conclusion (10%)**: Achievements? Conclusions?
   Lessons? Future/unfinished work?

Introduction to Type Systems

Type Systems

- Detect potential runtime errors in source code
- Some errors cannot be detected in general, e.g. division by zero, array boundary violations etc.
- Idea: only detect some errors ("forbidden errors")
- General type-checker algorithm:
  - Use type rules that define how elementary parts of the source code should look like
  - Type rules give program parts a type
  - If a type can be derived for a program, then it does not contain any forbidden errors

```
int m(String s) {
   int y = s + 1;
   m(y,3);
   return s;
}
```



The Environment Gamma Γ

- Γ is the scope at a particular place in the program
- It contains the signatures of the variables and methods that can be accessed there

```
class MyClass {
   int x;
   String y;
   int m1(int z) {
      int a = 0;
      return a + z;
   }
   void m2() {
      String a = "hello";
      System.out
         .println(a);
   }
}
```

Γ_2 = { `int x; String y; int m1(int z); void m2(); int z; int a;` }

Γ_3 = { `int x; String y; int m1(int z); void m2(); String a;` }

Γ_1 = { `int x; String y; int m1(int z); void m2();` }

Judgements

Statements about the correctness of program parts, e.g.

Symbols	Meaning
{ `int x;` } ⊢ ◊	"{ `int x;` } is a correct environment"
{ `int x;` } ⊢ `x+1`:`int`	"`x+1` is a correct expression of type `int` in environment { `int x;` }"
{ `int x;` } ⊢ `x=x+1;`	"`x=x+1` is a correct statement in environment { `int x;` }"
{ `int x;` } ⊢ `void m(){x=x+1;}`	"`void m(){x=x+1;}` is a correct method definition in env. { `int x;` }"
∅ ⊢ `class A {int x; void m(){x=x+1;} }`	"`A` is a correct class in an empty environment"

Type Rules

Rule [*expr* +] can be used to derive/check additions of integer expressions (e.g. 1+1)

Precondition (everything above the line)

Judgements

"is correct in environment"

Rule Name

"has type"

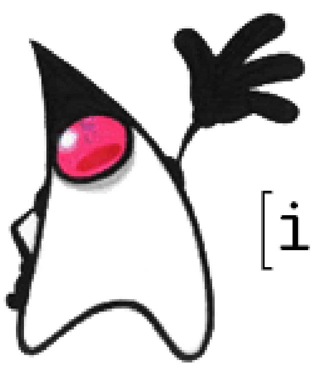
$$[\texttt{int}\ +]\quad \frac{\Gamma \vdash expr_1 : \texttt{int} \qquad \Gamma \vdash expr_2 : \texttt{int}}{\Gamma \vdash expr_1 + expr_2 : \texttt{int}}$$

Postcondition (everything below the line)

Judgement

"**If** $expr_1$ is a correct **int** expression in environment Γ and $expr_2$ is a correct **int** expression in environment Γ **then** $expr_1 + expr_2$ is also a correct **int** expression in environment Γ"

Simplified Java Type Rules



$$[\texttt{int +}]\ \frac{\Gamma \vdash expr_1 : \texttt{int} \qquad \Gamma \vdash expr_2 : \texttt{int}}{\Gamma \vdash expr_1 \texttt{+} expr_2 : \texttt{int}}$$

Type Derivation

Idea: derive smaller parts, combine them into big parts

From smallest to biggest:

1. Environments for methods
   (containing signatures for accessible methods and vars)
2. Expressions in methods
3. Statements in methods
4. The methods themselves
5. Member variables
6. The whole class

(Items 4–6: Not covered in 732)

The start rule (for creating environments):

$$[env] \quad \frac{sig_1, \ldots, sig_n \text{ are correct signatures}}{\{sig_1, \ldots, sig_n\} \vdash \diamond}$$

Expressions 1

Literals

$$[\texttt{boolean}\ lit]\ \frac{\Gamma \vdash \diamond \quad x \in \{\texttt{true}, \texttt{false}\}}{\Gamma \vdash x : \texttt{boolean}}$$

$$[\texttt{String}\ lit]\ \frac{\Gamma \vdash \diamond \quad x \in \texttt{String}}{\Gamma \vdash x : \texttt{String}} \qquad [\texttt{int}\ lit]\ \frac{\Gamma \vdash \diamond \quad x \in \texttt{int}}{\Gamma \vdash x : \texttt{int}}$$

Operators (*e.g.* **+** for **int**)

$$[\texttt{int +}]\ \frac{\Gamma \vdash expr_1 : \texttt{int} \quad \Gamma \vdash expr_2 : \texttt{int}}{\Gamma \vdash expr_1 \texttt{+} expr_2 : \texttt{int}}$$

$$[\texttt{String +}]\ \frac{\Gamma \vdash expr_1 : \texttt{String} \quad \Gamma \vdash expr_2 : \texttt{String}}{\Gamma \vdash expr_1 \texttt{+} expr_2 : \texttt{String}}$$

$$[\texttt{int ==}]\ \frac{\Gamma \vdash expr_1 : \texttt{int} \quad \Gamma \vdash expr_2 : \texttt{int}}{\Gamma \vdash expr_1 \texttt{==} expr_2 : \texttt{boolean}}$$

You can create analogous rules for other types, e.g. **double**

Expressions 2

Variable access $[var]\ \dfrac{\Gamma \vdash \diamond \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id : type}$

Pre: a correct environment with a variable signature

Post: an expression that accesses the variable

Method calls

$$[call]\ \frac{\Gamma \vdash expr_1 : type_1 \quad \ldots \quad \Gamma \vdash expr_n : type_n \qquad \{type_{ret}\ id(type_1\ id_1, \ldots, type_n\ id_n);\} \subseteq \Gamma}{\Gamma \vdash id(expr_1, \ldots, expr_n) : type_{ret}}$$

Pre: n correct expressions in an environment with a method signature (has n parameters with same types)

Post: method call using the expressions as arguments

Statements

Expressions as statements

$$[stat\ expr]\ \frac{\Gamma \vdash expr\colon type}{\Gamma \vdash expr\,;}$$

Assignments

$$[assign]\ \frac{\Gamma \vdash expr\colon type \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id\texttt{=}expr\,;}$$

Blocks of statements

$$[block]\ \frac{\Gamma \vdash stat_1 \quad \ldots \quad \Gamma \vdash stat_n}{\Gamma \vdash \{stat_1 \ldots stat_n\}}$$

If statement

$$[if]\ \frac{\Gamma \vdash expr\colon \texttt{boolean} \quad \Gamma \vdash stat}{\Gamma \vdash \texttt{if}\,(expr)\ stat}$$

If-else statement

$$[if\ else]\ \frac{\Gamma \vdash expr\colon \texttt{boolean} \quad \Gamma \vdash stat_1 \quad \Gamma \vdash stat_2}{\Gamma \vdash \texttt{if}\,(expr)\ stat_1\ \texttt{else}\ stat_2}$$

You can create analogous rules for **for**, **while**, …

Type Derivation

Derivation Example 1

Given the environment $\Gamma = \{$ **boolean x; int y;** $\}$ derive the following code:

```
if (x)  y = y + 1;
```

$$[var]\ \frac{\Gamma \vdash \diamond \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id\colon type} \qquad [\texttt{int}\ lit]\ \frac{\Gamma \vdash \diamond \quad x \in \texttt{int}}{\Gamma \vdash x\colon \texttt{int}}$$

1. Derive expression x

$$[var]\ \frac{\Gamma \vdash \diamond \quad \{\texttt{boolean x;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{x}\colon \texttt{boolean}}$$

2. Derive expression y

$$[var]\ \frac{\Gamma \vdash \diamond \quad \{\texttt{int y;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{y}\colon \texttt{int}}$$

3. Derive expression 1

$$[\texttt{int}\ lit]\ \frac{\Gamma \vdash \diamond \quad 1 \in \texttt{int}}{\Gamma \vdash 1\colon \texttt{int}}$$

Derivation Example 1 Cont.

Given the environment $\Gamma = \{$ **boolean x; int y;** $\}$
derive the following code:

```
if (x)  y = y + 1;
```

$$[assign]\ \frac{\Gamma \vdash expr\colon type \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id\texttt{=}expr\texttt{;}}$$

$$[if]\ \frac{\Gamma \vdash expr\colon \texttt{boolean} \quad \Gamma \vdash stat}{\Gamma \vdash \texttt{if}\,\texttt{(}expr\texttt{)}\ stat}$$

4. Derive **y+1**

$$[\texttt{int +}]\ \frac{\Gamma \vdash \texttt{y}\colon \texttt{int} \quad \Gamma \vdash \texttt{1}\colon \texttt{int}}{\Gamma \vdash \texttt{y+1}\colon \texttt{int}}$$

5. Derive **y=y+1**

$$[assign]\ \frac{\Gamma \vdash \texttt{y+1}\colon \texttt{int} \quad \{\texttt{int y;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{y=y+1;}}$$

6. Derive **if**

$$[if]\ \frac{\Gamma \vdash \texttt{x}\colon \texttt{boolean} \quad \Gamma \vdash \texttt{y=y+1;}}{\Gamma \vdash \texttt{if}\,\texttt{(x)}\ \texttt{y=y+1;}}$$

Derivation Example 2

Given the environment $\Gamma = \{$ **`int x; int m(String s);`** $\}$ derive the following code:

```
x = m("hello") + 7;
```

$$[call]\ \frac{\Gamma \vdash expr_1 : type_1 \quad \ldots \quad \Gamma \vdash expr_n : type_n \quad \{type_{ret}\ id(type_1\ id_1, \ldots, type_n\ id_n);\} \subseteq \Gamma}{\Gamma \vdash id(expr_1, \ldots, expr_n) : type_{ret}}$$

1. Derive **"hello"** $[\texttt{String}\ lit]\ \dfrac{\Gamma \vdash \diamond \quad \texttt{"hello"} \in \texttt{String}}{\Gamma \vdash \texttt{"hello"} : \texttt{String}}$

2. Derive **`m("hello")`**

$$[call]\ \frac{\Gamma \vdash \texttt{"hello"} : \texttt{String} \quad \{\texttt{int m(String s);}\} \subseteq \Gamma}{\Gamma \vdash \texttt{m("hello")} : \texttt{int}}$$

Derivation Example 2 Cont.

Given the environment $\Gamma = \{$**int x; int m(String s);**$\}$

derive the following code:

```
x = m("hello") + 7;
```

$$[assign]\ \frac{\Gamma \vdash expr\colon type \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id\texttt{=}expr\texttt{;}}$$

4. Derive 7

$$[\texttt{int}\ lit]\ \frac{\Gamma \vdash \diamond \quad 7 \in \texttt{int}}{\Gamma \vdash 7\colon \texttt{int}}$$

5. Derive addition

$$[\texttt{int +}]\ \frac{\Gamma \vdash \texttt{m("hello")}\colon \texttt{int} \quad \Gamma \vdash 7\colon \texttt{int}}{\Gamma \vdash \texttt{m("hello")+7}\colon \texttt{int}}$$

6. Derive assignment

$$[assign]\ \frac{\Gamma \vdash \texttt{m("hello")+7}\colon \texttt{int} \quad \{\texttt{int x;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{x=m("hello")+7;}}$$

Summary

Today's Summary

- **Type systems** detect potential runtime errors in code
- **Environment** Γ contains the signatures of the accessible variables and methods in a method
- **Type rules** with pre- and postcondition & judgements, e.g.

$$[\texttt{int}\ +]\ \frac{\Gamma \vdash expr_1 : \texttt{int} \quad \Gamma \vdash expr_2 : \texttt{int}}{\Gamma \vdash expr_1 + expr_2 : \texttt{int}}$$

- **Type derivation**: using the type rules, first derive smallest parts, then combine them into larger parts

Reference:

Luca Cardelli. Type Systems.

http://www.eecs.umich.edu/~bchandra/courses/papers/Cardelli_Types.pdf

Quiz

1. What is a type system?
2. What is an environment Γ? Why do we need it?
3. What is a judgement? Give examples.
4. Given the environment
   $\Gamma = \{$ **`String s; String m(int a, int b);`** $\}$
   Derive the following program:
   `if(s=="hello") s = m(1,2); else s = "abc";`