



# Section

## Intro

Chapter 0 ""

Course "Compiler Construction"

Martin Steffen

Spring 2021

# Goal



INF5110 – Oblig 2

## Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in

1. **semantic analysis**, as far as
  - **typing** is concerned (“static semantics”)
  - other conditions (no duplicate declaration etc)
2. **code generation** for `compila21` (ish) programs

# Last time (O1)



INF5110 – Oblig 2

## Syntactic analysis

- lexer (scanner)
- parser
- abstract syntax tree

**this time:** continue with your previous deliv. (and repos)

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in

# Learning outcome



INF5110 – Oblig 2

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in

- understand type checking, implementing a simple variant
- understand (simple form of) bytecode and how to generate it from “source code” (as AST)
- extend an existing compiler code base with new functionality



# Section

## Semantic analysis

Chapter 0 ""

Course "Compiler Construction"

Martin Steffen

Spring 2021

# Semantic analysis & type checking



INF5110 – Oblig 2

- parser / context-free grammars
  - not powerful enough
  - cannot check all (static) properties of a language spec
- $\Rightarrow$  extend the front-end by a type checker
  - use the AST classes of last time
  - add type checking code
  - allowed to make **changes** or adaptations if advantageous.

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in

# Another glance at compila21



INF5110 – Oblig 2

Intro

Semantic analysis

Code generation

Testing

Starting point and hand in

```
program MyProgram begin
```

```
class Complex begin
  var Real : float;
  var Imag : float;
end;
```

Real and Imag are of the (built-in) float type.  
Complex defines a new (user-defined) type.

```
proc Add(a : Complex, b : Complex) : Complex
begin
  var retval : Complex;
  retval := new Complex;
  retval.Real := a.Real + b.Real;
  retval.Imag := a.Imag + b.Imag;
```

Check that the + operator is compatible with its operands' types, and that the assignment is legal.

```
  return retval;
end;
```

```
proc Main()
begin
  var c1 : Complex;
  var c2 : Complex;
  var result : Complex;
  ...
  result := Add ( c1, c2 );
  ...
  return;
end; end;
```

Check that the actual parameters to Add(...) are of the correct type, according to the formal parameters, and that the assignment to result is legal.

NB: 2021: structs, not classes

# Type checking for conditionals

- as “inspiration”, details may vary

---

```
class IfStatement extends Statement {  
    ...  
    public void typeCheck() {  
        String condType = condition.getType ();  
        if (condType != "bool") {  
            throw new TypeException("condition in an if  
                statement must be of type bool")  
        }  
    }  
}
```

---



# Type checking: assignments

---

```
class Assignment extends Statement {  
    ...  
    public void typeCheck() {  
        String varType = var.getType();  
        String expType = exp.getType();  
        if (varType != expType &&  
            !isAssignmentCompatible(varType, expType) {  
            throw new RuntimeException("Cannot assign  
                " from " + expType);  
        }  
    }  
}
```

---



# Section

## Code generation

Chapter 0 ""

Course "Compiler Construction"

Martin Steffen

Spring 2021

# Code generation



INF5110 – Oblig 2

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in

- byte code API and operations are described in the document “Interpreter and bytecode for INF5110”
- **Task:** add bytecode generation methods to your AST classes for instance

```
Ast.Node.GenerateCode(...)
```

- again: if adaptations of the AST are called for or useful, go for it...
- some people did visitors for ast-printing, one can also (re-)use the visitor pattern

# Code generation: limitations



INF5110 – Oblig 2

- interpreter and byte code library somewhat **limited**
  - cannot express full compila 21
  - no block structure
  - no reference types
- your delivery should support generating correct bytecode for the `compila 21` source code file `runme.cmp`

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in

# Code generation: creating a procedure

---

```
CodeFile codeFile = new CodeFile();  
// add the procedure by name first  
codeFile.addProcedure("Main")  
// then define it  
CodeProcedure main = new  
    CodeProcedure("Main", VoidType, TYPE, codeFile);  
main.addInstruction( new RETURN());  
//then update it in the code file  
codeFile.updateProcedure(main);
```

---

# Code generation: assignment



INF5110 – Oblig 2

```
//1: proc add(a: int, b : int ) : int {  
//2: var res : int;  
//3: res := a + b; // only bytecode for this line  
//4: return res;  
//5: }
```

```
// push a onto the stack  
proc.addInstruction(new LOADLOCAL(proc.variableNumber("a")));  
// push b onto the stack  
proc.addInstruction(new LOADLOCAL(proc.variableNumber("b")));  
// perform addition with arguments on the stack  
proc.addInstruction(new ADD());  
// pop result from stack, and store it in variable res  
proc.addInstruction(new  
    STORELOCAL(proc.variableNumber("res")));
```

**Intro**

**Semantic analysis**

**Code generation**

**Testing**

**Starting point and  
hand in**



# Section

## Testing

Chapter 0 ""  
Course "Compiler Construction"  
Martin Steffen  
Spring 2021



- bunch of test files, for testing the *type checker*
- preferable: make `ant test` workable
- test files inside  
`./tests/semanticanalysis/errors/` (and with `fail` in the filename) contain a syntactically correct but erroneous program (erroneous as the type system or generally the semantic phase is concerned)
- $\Rightarrow$  compiler returns error code 2 for semantic failure

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in





# Section

## Starting point and hand in

Chapter 0 ""

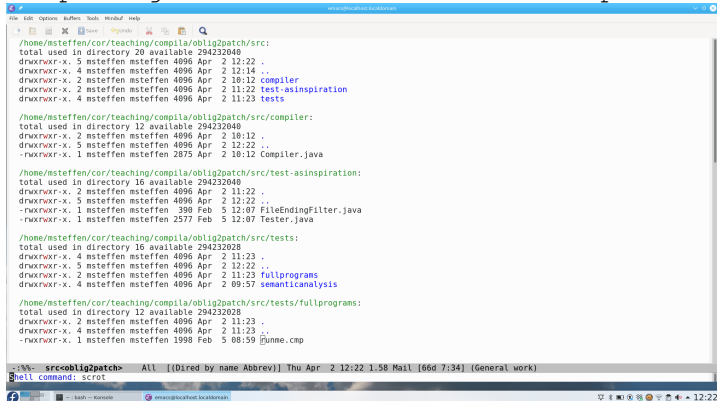
Course "Compiler Construction"

Martin Steffen

Spring 2021

# Provided source code (patch)

<https://github.uio.no/msteffen/compila>



```
du -x -h -l /home/msteffen/cor/teaching/compila/oblig2patch/src:
total used in directory 20 available 294232048
drwxrwxr-x. 5 msteffen msteffen 4096 Apr  2 12:22 .
drwxrwxr-x. 4 msteffen msteffen 4096 Apr  2 12:14 ..
drwxrwxr-x. 2 msteffen msteffen 4096 Apr  2 10:12 compiler
drwxrwxr-x. 2 msteffen msteffen 4096 Apr  2 11:22 test-asinspiration
drwxrwxr-x. 4 msteffen msteffen 4096 Apr  2 11:23 tests

/home/msteffen/cor/teaching/compila/oblig2patch/src/compiler:
total used in directory 12 available 294232048
drwxrwxr-x. 2 msteffen msteffen 4096 Apr  2 10:12 .
drwxrwxr-x. 5 msteffen msteffen 4096 Apr  2 12:22 ..
-rwxrwxr-x. 1 msteffen msteffen 2875 Apr  2 10:12 Compiler.java

/home/msteffen/cor/teaching/compila/oblig2patch/src/test-asinspiration:
total used in directory 16 available 294232048
drwxrwxr-x. 2 msteffen msteffen 4096 Apr  2 11:22 .
drwxrwxr-x. 5 msteffen msteffen 4096 Apr  2 12:22 ..
-rwxrwxr-x. 1 msteffen msteffen 390 Feb  5 12:07 FileEndingFilter.java
-rwxrwxr-x. 1 msteffen msteffen 2577 Feb  5 12:07 Tester.java

/home/msteffen/cor/teaching/compila/oblig2patch/src/tests:
total used in directory 16 available 294232028
drwxrwxr-x. 4 msteffen msteffen 4096 Apr  2 11:23 .
drwxrwxr-x. 5 msteffen msteffen 4096 Apr  2 12:22 ..
drwxrwxr-x. 2 msteffen msteffen 4096 Apr  2 11:23 fullprograms
drwxrwxr-x. 4 msteffen msteffen 4096 Apr  2 09:57 semanticanalysis

/home/msteffen/cor/teaching/compila/oblig2patch/src/tests/fullprograms:
total used in directory 12 available 294232028
drwxrwxr-x. 2 msteffen msteffen 4096 Apr  2 11:23 .
drwxrwxr-x. 4 msteffen msteffen 4096 Apr  2 11:23 ..
-rwxrwxr-x. 1 msteffen msteffen 1998 Feb  5 08:59 jrunme.cmp

-:~%_ src<oblig2patch> All [(Dired by name Abbrev)] Thu Apr  2 12:22 1.58 Mail [66d 7:34] (General work)
shell command: scrot
```

Tests: already included in the oblig1 checkout, so left out in the zip-patch this year.



INF5110 – Oblig 2

Intro

Semantic analysis

Code generation

Testing

Starting point and hand in

# Provided documentation (patch)



INF5110 – Oblig 2

Intro

Semantic analysis

Code generation

Testing

Starting point and hand in

```
ssh@ostrogotica.se
/home/esteffen/cor/teaching/compila/oblig2patch:
total used in directory 24 available 294235696
drwxr-xr-x. 4 esteffen esteffen 4096 Apr  2 11:21 ..
drwxr-xr-x. 13 esteffen esteffen 4096 Apr  2 12:18 .
-rw-rw-r--. 1 esteffen esteffen 4408 Apr  2 10:37 Readme.org
drwxr-xr-x. 3 esteffen esteffen 4096 Apr  2 12:03 doc
drwxr-xr-x. 4 esteffen esteffen 4096 Apr  2 11:22 src

/home/esteffen/cor/teaching/compila/oblig2patch/doc:
total used in directory 12 available 294235696
drwxr-xr-x. 3 esteffen esteffen 4096 Apr  2 12:03 .
drwxr-xr-x. 4 esteffen esteffen 4096 Apr  2 11:21 ..
drwxr-xr-x. 2 esteffen esteffen 4096 Apr  2 12:04 obligs

/home/esteffen/cor/teaching/compila/oblig2patch/doc/obligs:
total used in directory 1128 available 294235696
drwxr-xr-x. 2 esteffen esteffen 4096 Apr  2 12:04 .
drwxr-xr-x. 3 esteffen esteffen 4096 Apr  2 12:03 ..
-rw-rw-r--. 1 esteffen esteffen 212786 Mar 17 09:56 handout-oblig2.pdf
-rw-rw-r--. 1 esteffen esteffen 933947 Feb  5 08:59 slides-o2.pdf

--*-- oblig2patch All [(Diret by name Abbrev)] Thu Apr  2 12:12 8:92 Mail [66d 7:24] [General work]
Shell command: scroll
```

# Relevant directories



INF5110 – Oblig 2

- Java
  - `compiler`: updated compiler class (patch)
  - `test`: some code for performing tests (patch)
  - `bytecode`: classes for constructing bytecode (already there)
  - `runtime`: rte for executing the byte code (already there)
- Compila
  - `tests`: some test files (including `runme.cmp`)

**Intro**

**Semantic analysis**

**Code generation**

**Testing**

Starting point and  
hand in

# Deadline

## Deadline

14 May 2021

Note: end of semester, and I need to report the ones passing the oblig some time before the exam.

## delivs

- working type checker
- code generator (test with `runme.cmp`)
- report (including your name(s) etc.
  - discussion of your solution, choices you made, assumptions you rely on
  - printout of a test run (can be also checked in into the repos, but it needs to be mentioned where it is)
  - printout of the bytecode from `runme.cmp` (with a target like `ant list-runme`)
  - solution must “build” and be “testable” (typically via `ant`)



INF5110 – Oblig 2

Intro

Semantic analysis

Code generation

Testing

Starting point and  
hand in