Homework 1
Simple code generator

Aristeidis Mastoras

Compiler Design – SS18

(based on slides of Luca Della Toffola from Compiler Design – HS15)

Administrative issues

- Has everyone found a teammate?
- Mailing-list: **cd1@lists.inf.ethz.ch**
  - **Please subscribe if we forgot you**
- Assistants: **cd1-owner@lists.inf.ethz.ch**

Today

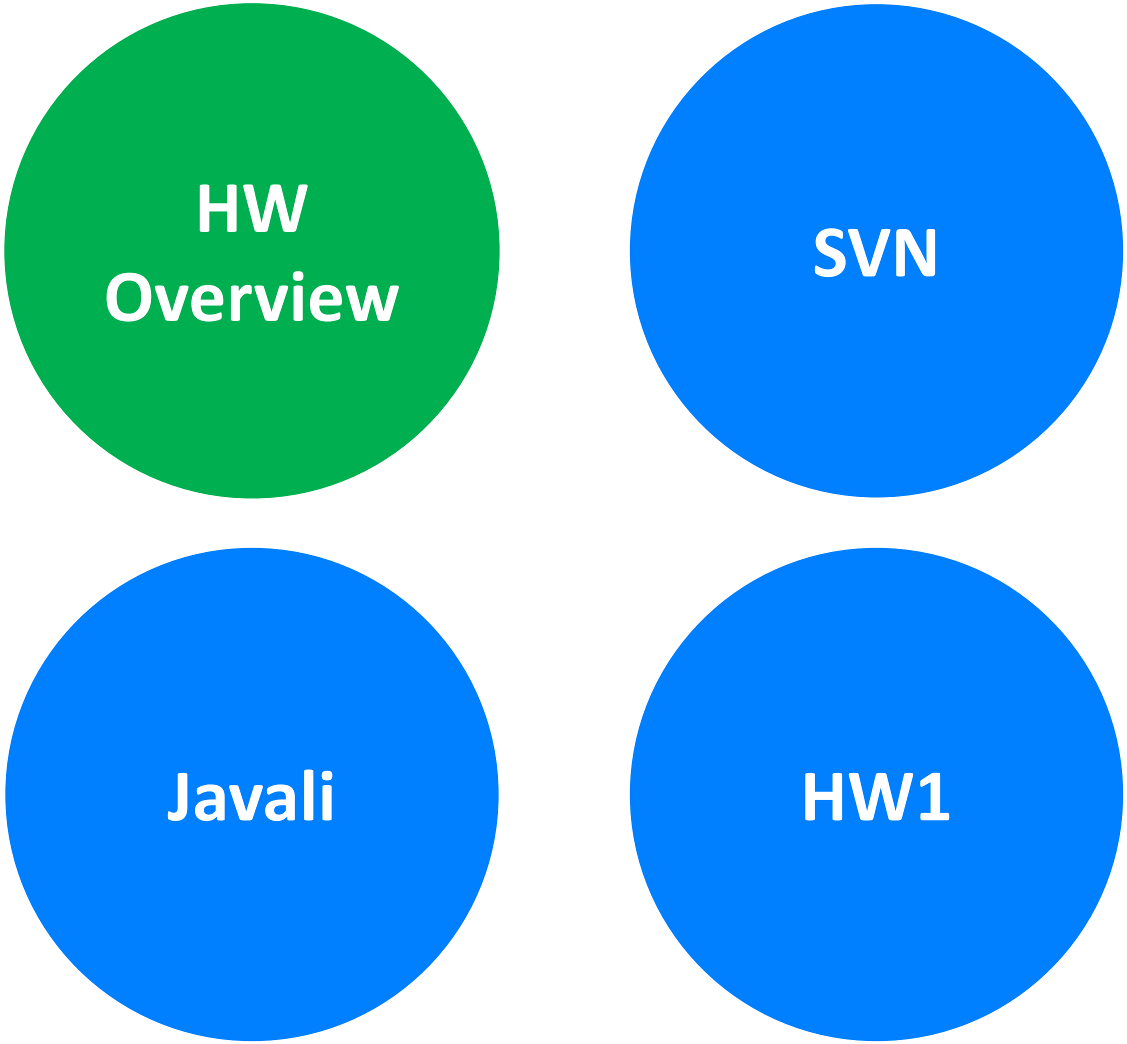
HW Overview

SVN

Javali

HW1

Today



HW Overview

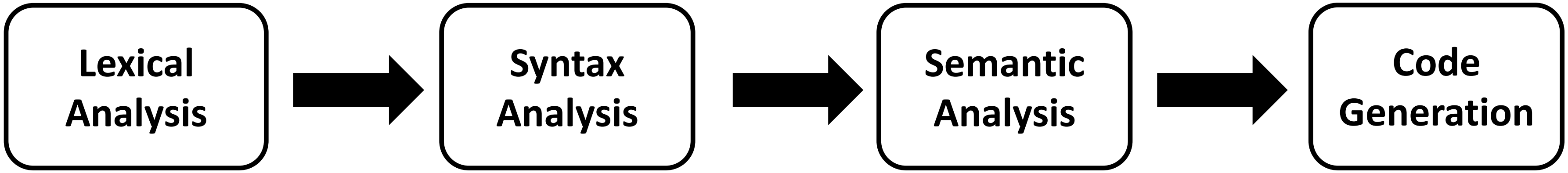
SVN

Javali

HW1

This course

Build a full Javali compiler



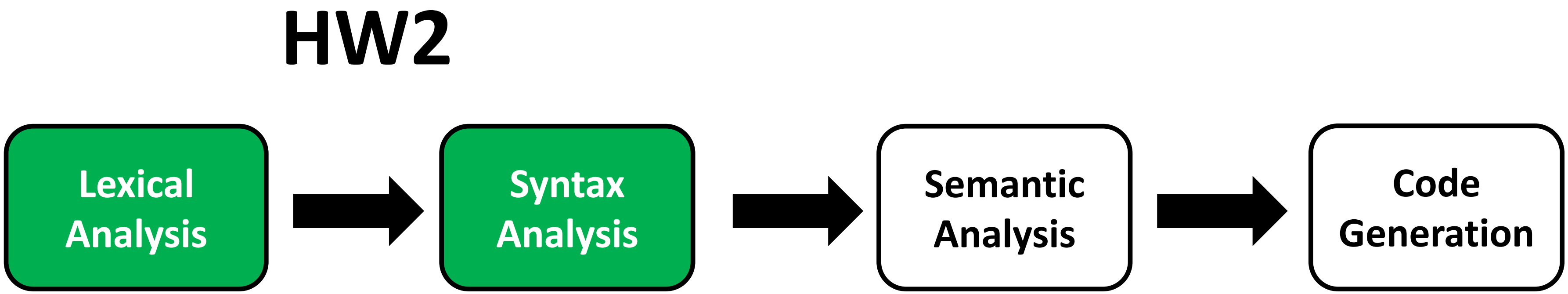
Lexical Analysis

Syntax Analysis

Semantic Analysis

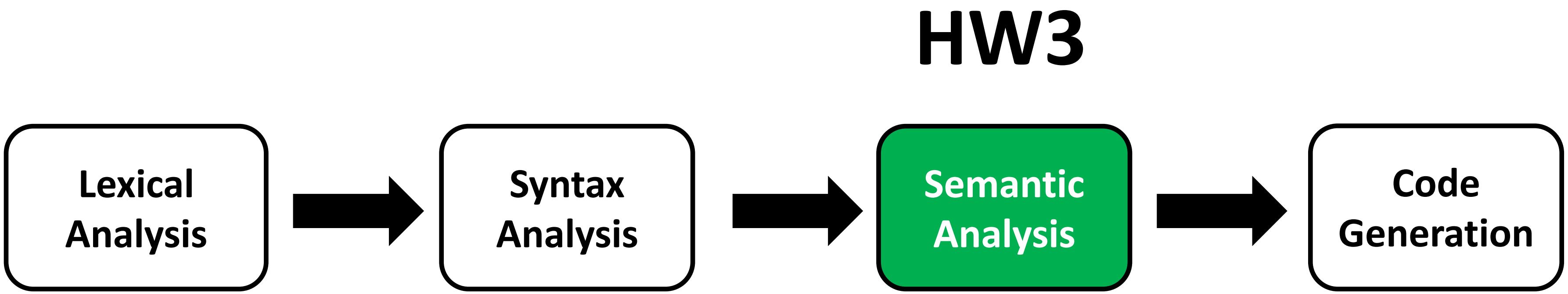
Code Generation

This course

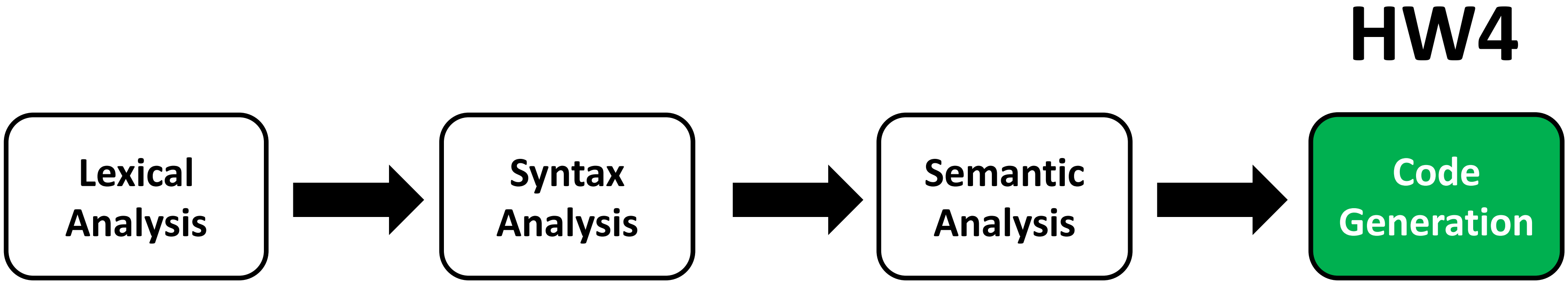


Parser

This course

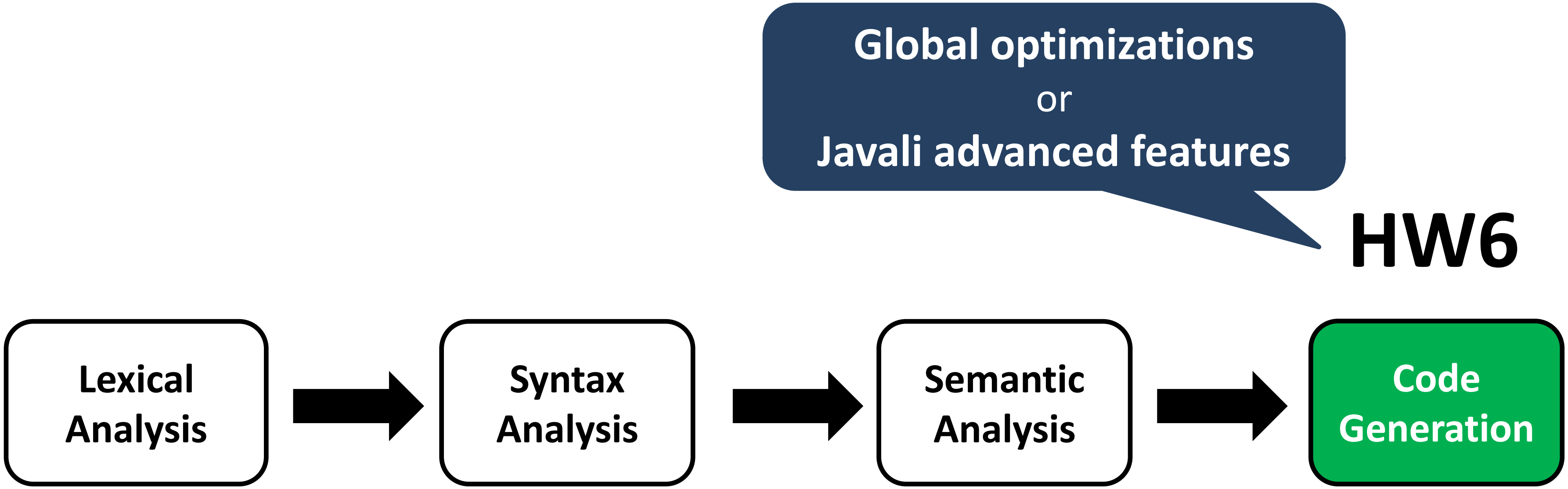

HW3
Lexical Analysis
Syntax Analysis
Semantic Analysis
Code Generation

This course

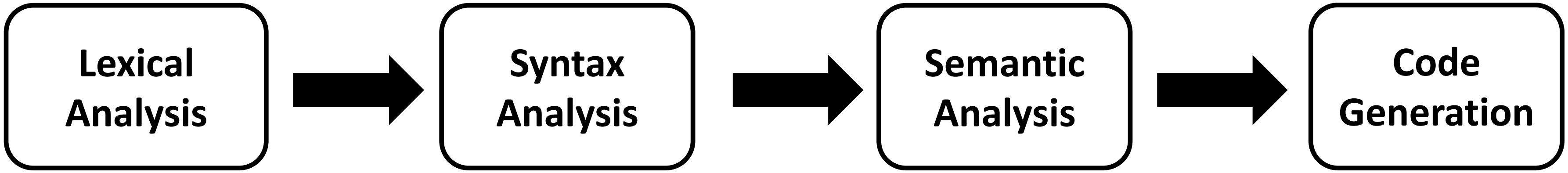


HW4
Lexical Analysis
Syntax Analysis
Semantic Analysis
Code Generation

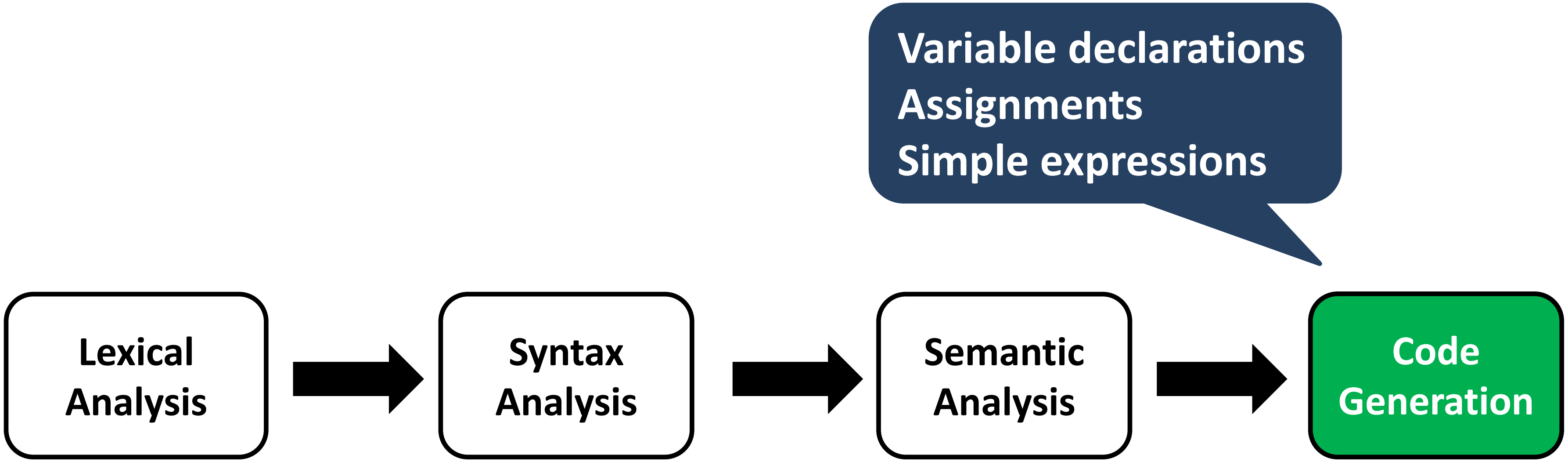
This course


Global optimizations
or
Javali advanced features
HW6
Lexical Analysis
Syntax Analysis
Semantic Analysis
Code Generation

Homework 1


Lexical Analysis
Syntax Analysis
Semantic Analysis
Code Generation

Homework 1



Variable declarations
Assignments
Simple expressions

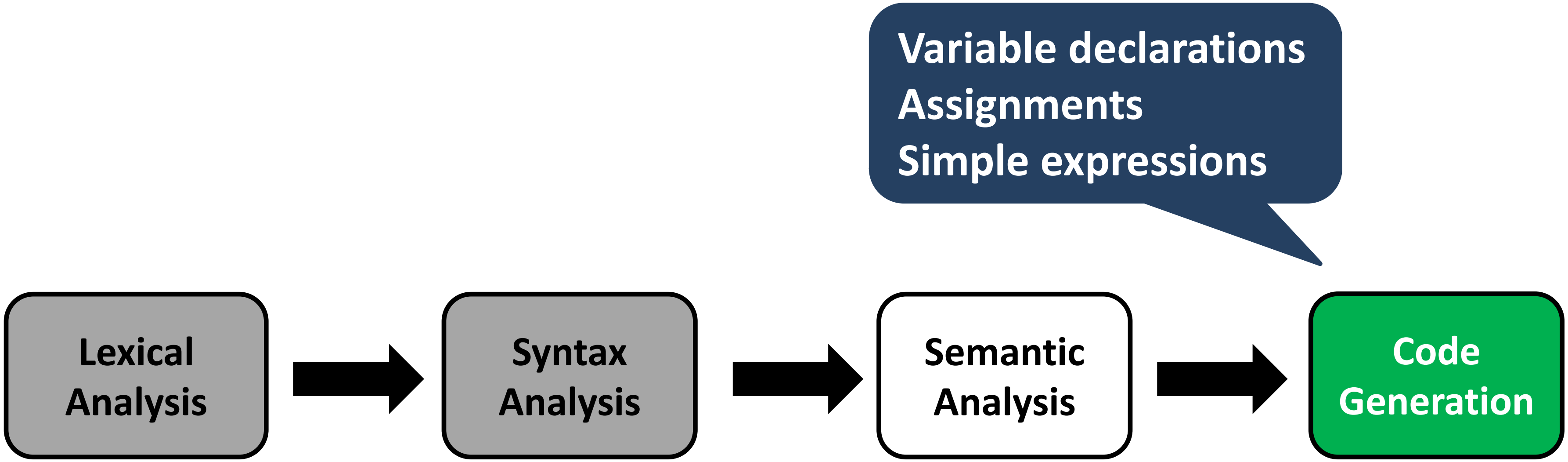
Lexical Analysis

Syntax Analysis

Semantic Analysis

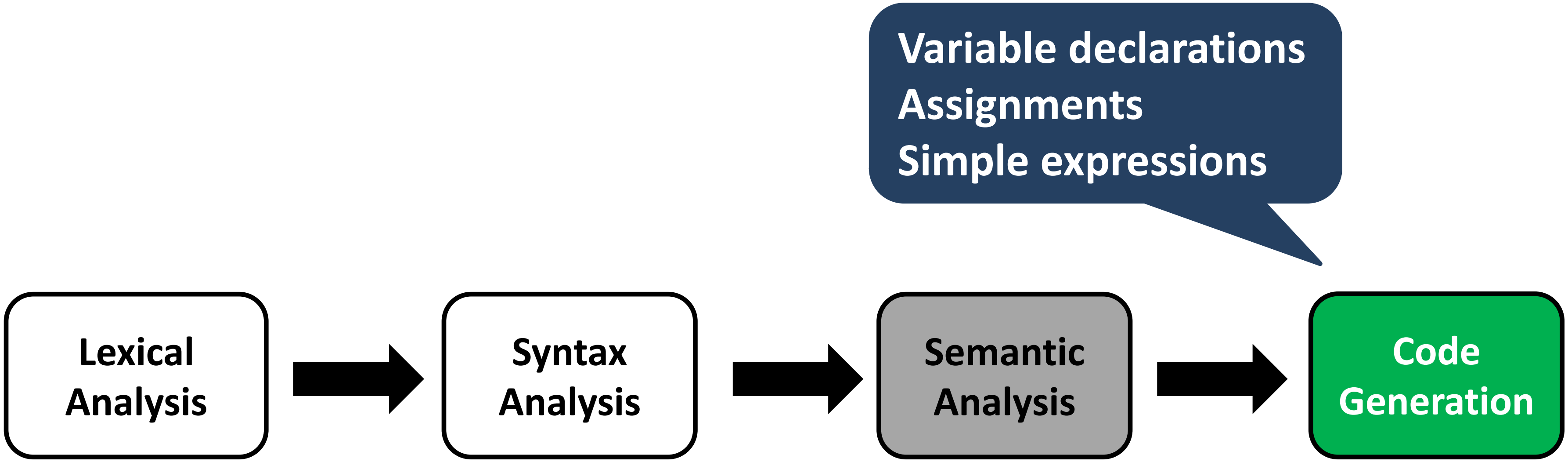
Code Generation

Homework 1



We give you the parser

Homework 1



Variable declarations
Assignments
Simple expressions

Lexical Analysis

Syntax Analysis

Semantic Analysis

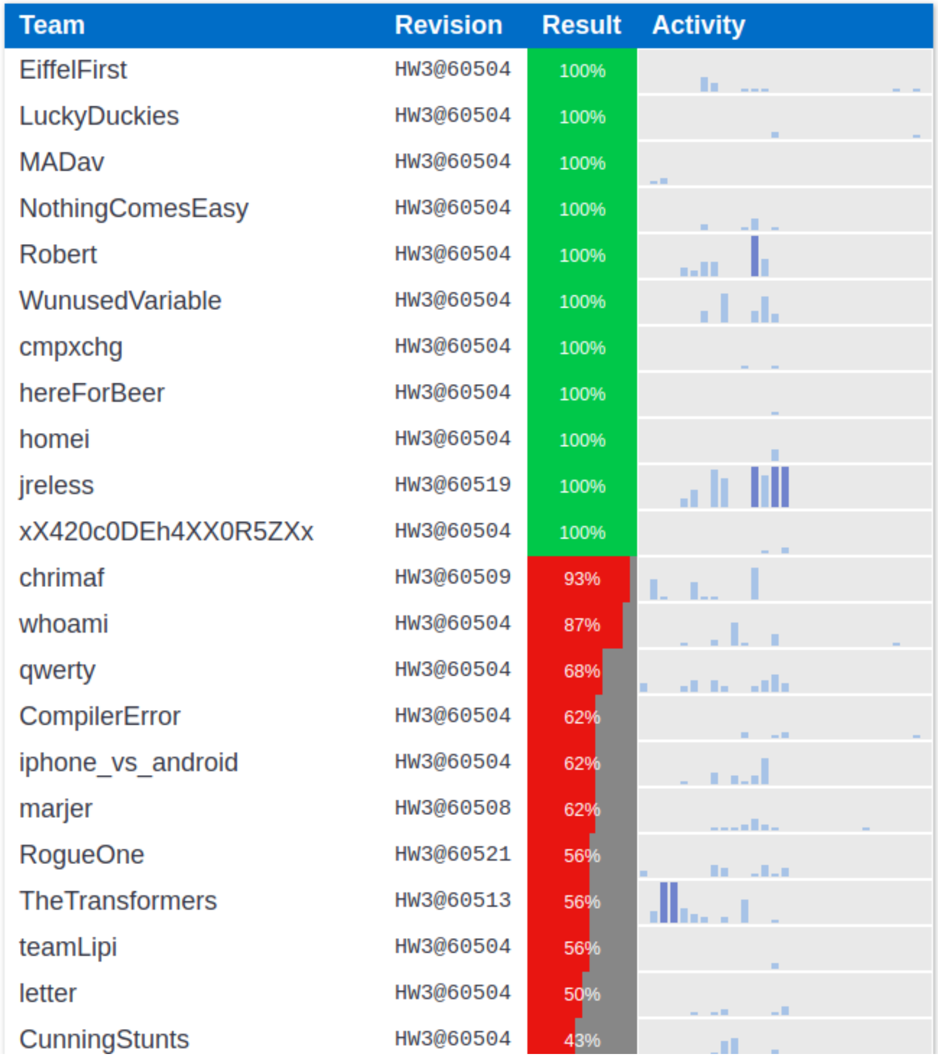
Code Generation

Not necessary for now

Grading

- Homework task
  - The compiler “works”
- Write your own tests
  - Test exhaustively that the compiler works
- Code quality
  - The code is readable

Grading



Team	Revision	Result	Activity
EiffelFirst	HW3@60504	100%	
LuckyDuckies	HW3@60504	100%	
MADav	HW3@60504	100%	
NothingComesEasy	HW3@60504	100%	
Robert	HW3@60504	100%	
WunusedVariable	HW3@60504	100%	
cmpxchg	HW3@60504	100%	
hereForBeer	HW3@60504	100%	
homei	HW3@60504	100%	
jreless	HW3@60519	100%	
xX420c0DEh4XX0R5ZXx	HW3@60504	100%	
chrimaf	HW3@60509	93%	
whoami	HW3@60504	87%	
qwerty	HW3@60504	68%	
CompilerError	HW3@60504	62%	
iphone_vs_android	HW3@60504	62%	
marjer	HW3@60508	62%	
RogueOne	HW3@60521	56%	
TheTransformers	HW3@60513	56%	
teamLipi	HW3@60504	56%	
letter	HW3@60504	50%	
CunningStunts	HW3@60504	43%	

- **Indication** of your grade
- Link will be announced

Good news!

- HW1 is independent of HW2
- The same applies to the next HW
- **You can still do the next HW even if you don’t manage to get this right**

Today

HW Overview

SVN

Javali

HW1

SVN

What is SVN?

- a version control system
- it is used to store current and previous versions of (not only) source code
- you can revert to a previous version

Javali fragment

We provide:

- a compiler skeleton for every HW
- stored in an SVN repository
- every team has a different SVN repository

Get your fragment

https://svn.inf.ethz.ch/svn/trg/cd_students/ss18/teams/<YourTeam>

Case-sensitive
Homework + grades
Your submission platform

SVN basics

```
svn checkout https://<your_SVN_repo>
```

Get the remote copy of the repository on your machine.

```
svn commit –m “Message about your changes”
```

Update remote copy of the repository with local changes.

```
svn update
```

Get remote changes of your repository if modified.

SVN basics

svn add <file-or-dir-name>

Add a file/directory to the local copy.

(It requires *commit* to update the remote copy.)

svn remove <file-or-dir-name>

Delete a file/directory from the local copy.

(It requires *commit* to update the remote copy.)

SVN basics

svn status

Report files that are different in the local copy from those in the remote copy.

svn diff –r <version-number> <file-name>

Report the differences between the local copy and a specific version.

SVN resources

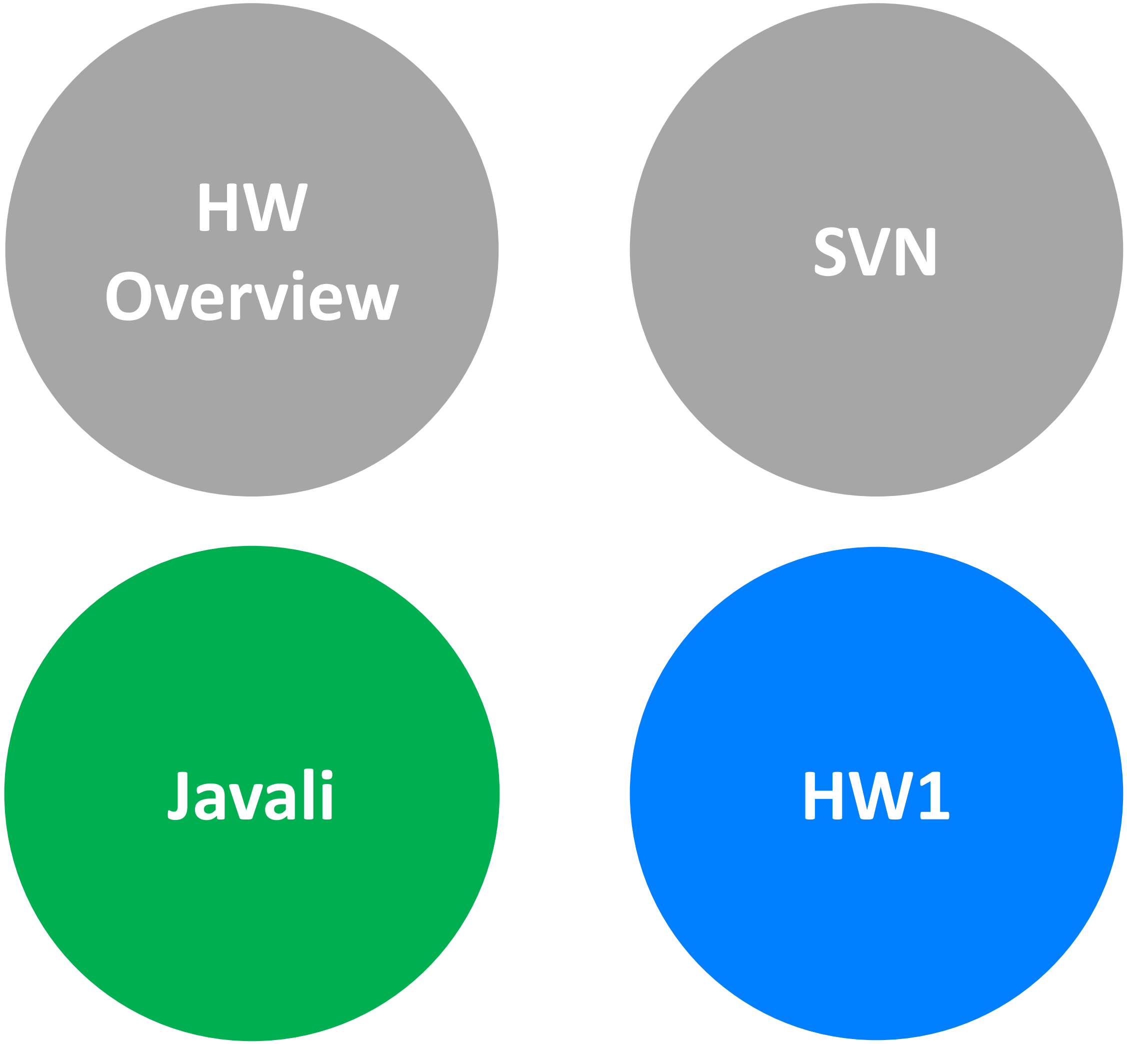
Links

- http://svnbook.red-bean.com
- https://www.google.ch/search?q=svn+tutorial

Software

- Eclipse Subversive
- Tortoise SVN
- Command-line

Today



HW Overview

SVN

Javali

HW1

Javali

- **Simple** OO programming language
  - Subset of Java
- **Javali specification** in the course web-site
  - **Updated** recently, subject to changes (and bugs)
- When the specification is incomplete
  - **Common sense** or **Java specification** apply
  - Use the **mailing-list** for **clarifications** or **questions**

Javali framework

- We provide a **framework skeleton**
  - To use for your homework
  - Utility classes and basic tasks
  - Free to modify or create your own
  - **Please comply to submission requirements**
- After each homework we provide a solution

Javali framework

src

Source of the compiler

test

Source files for testing your compiler

lib

Compiler dependencies as .JAR files

javali_tests

Unit-tests in form of .javali files. These are example programs to test

build.xml

Optional ANT script for command-line (can be used in Eclipse)

Compile the Javali framework

Fragment is an **Eclipse** project, it will **build automatically**

If you **don’t use Eclipse**, install **ANT** and type:

ant test

It compiles automatically too ☺

How to test your Javali compiler

- We provide a **JUnit-based** testing framework
- A test is a **Javali program** in the **javali_tests** directory
- The testing framework compares the output of your compiler against our reference solution

To test your compiler write more Javali programs that cover assignment tasks

You need to see green

How to test your Javali compiler

- Expected results are stored in **.javali.exec.ref** files
- **.javali.in** file determines the **standard input**
  - One line is equivalent to the result of a **read()** call.
- Run the tests using JUnit4
  - Eclipse provides a GUI to inspect results
- **.javali.err** file contains debugging output and error messages

DEMO

Javali framework changes

Files that may change per fragment

- **lib/frozenReference.jar** @ every fragment
- **build.xml** depending on the assignment
  - We will provide details in the recitation
  - Look for new targets
- **Javali specification**
  - As previously mentioned

Today

HW Overview

SVN

Javali

HW1

Simple code generator

Input: Javali Program

```
class Main {
        void main() {
                int a;
                a = 10;
                write(a);
        }
}
```

→

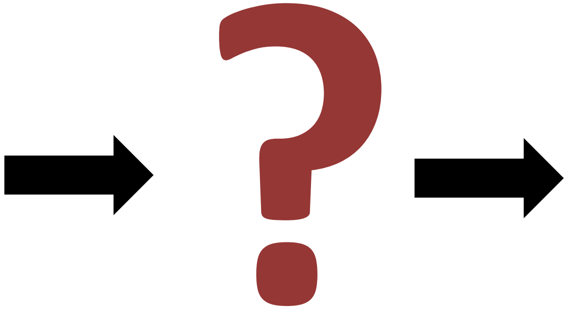
Output: x86 Assembly

```
        .section .data
STR_D:
        .string "%d"
        .section .data
var_a:
        .int 0
        .section .text
        .globl main
```

```
main:
        …
```

Simple code generator

```
class Main {
        void main() {
                int a;
                a = 10;
                write(a);
        }
}
```



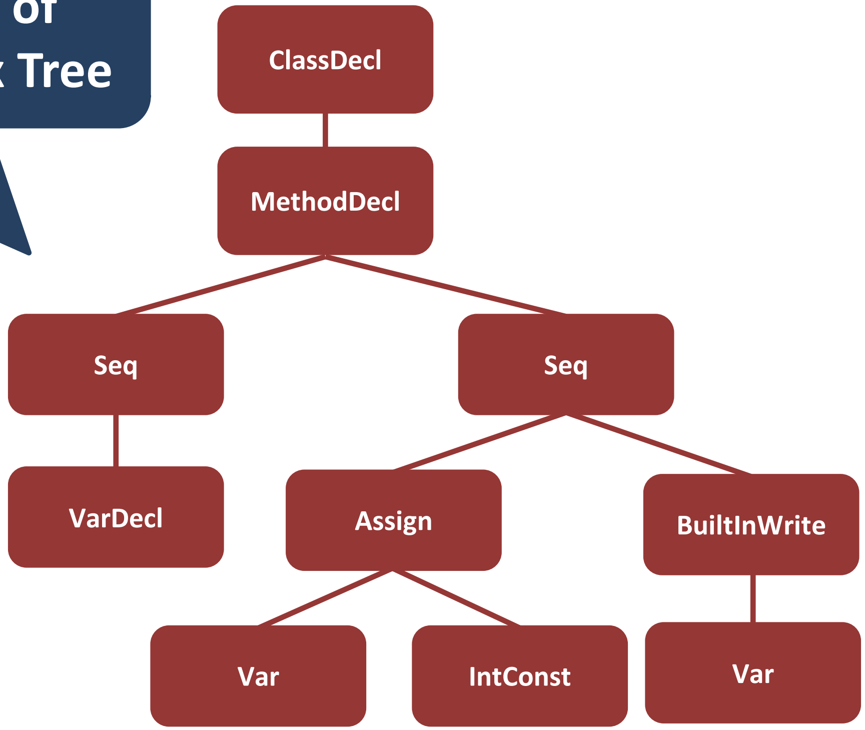
```
main:
        …
        # Emitting a = 10
                # Emitting 10
                movl $10, %edi
        movl %edi, var_a

        # Emitting write(a)
                # Emitting a
                movl var_a, %edi
        sub $16, %esp
        movl %edi, 4(%esp)
        movl $STR_D, 0(%esp)
        call printf
        add $16, %esp
        …
```

Javali program representation

The Intermediate Representation of Javali compiler is an Abstract Syntax Tree

```
class Main {
    void main() {
        int a;
        a = 10;
        write(a);
    }
}
```



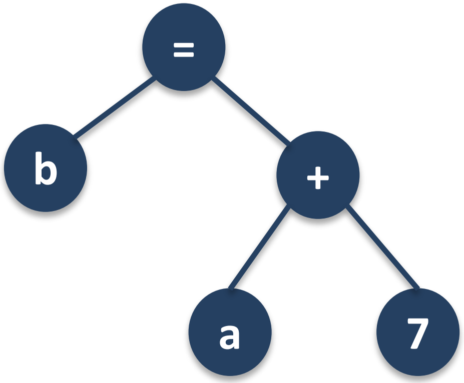
We give you the IR, i.e., the AST.

You need to generate the assembly code by using the AST.

Another example

Input: Javali Program

```
class Main {
	void main() {
		int a, b;
		a = 10;
		b = a + 7;
	}
}
```

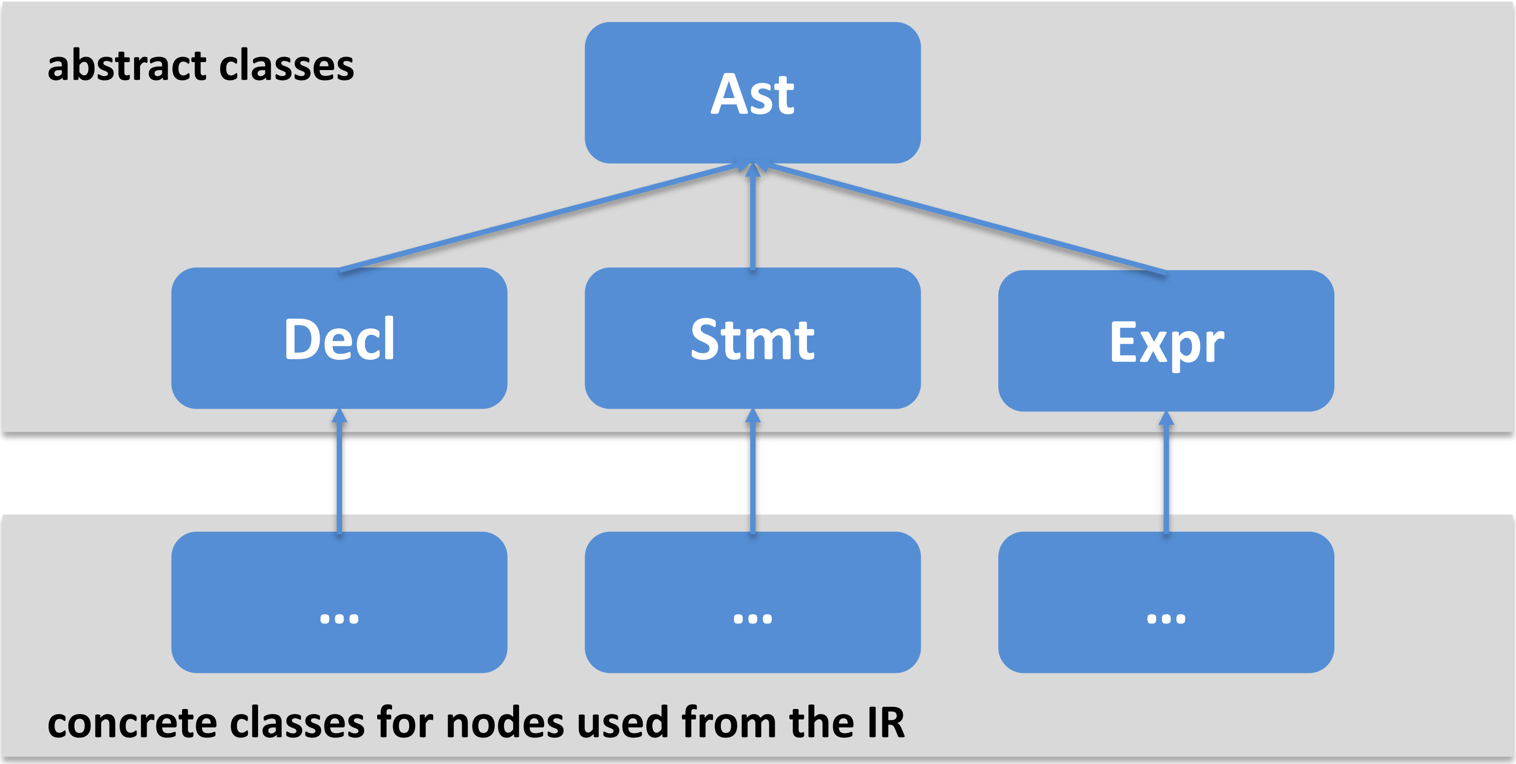


Output: x86 Assembly

```
main:
	...
	# Emitting a = 10
		# Emitting 10
		movl $10, %edi
	movl %edi, var_a

	# Emitting b = (a + 7)
		# Emitting (a + 7)
			# Emitting 7
			movl $7, %edi
			# Emitting a
			movl var_a, %esi
		add %edi, %esi
	movl %esi, var_b
	...
```

Javali Abstract Syntax Tree



Ast nodes declared in **cd/ir/Ast.java**

Javali Abstract Syntax Tree

Declarations

ClassDecl, MethodDecl, VarDecl

Statements

Assign, BuiltInWrite, BuiltInWriteln, IfElse, MethodCall, WhileLoop, Nop

Expressions

Var, IntConst, UnaryOp, BinaryOp, BuiltInRead, Index, NewArray, Field, Cast, NullConst, ThisRef, NewObject, BooleanConst

Javali Abstract Syntax Tree

Declarations

ClassDecl, MethodDecl, VarDecl

Only a subset of the AST nodes are used in Homework 1

Statements

Assign, BuiltInWrite, BuiltInWriteln, IfElse, MethodCall, WhileLoop, Nop

Expressions

Var, IntConst, UnaryOp, BinaryOp, BuiltInRead, Index, NewArray, Field, Cast, NullConst, ThisRef, NewObject, BooleanConst

Print the Abstract Syntax Tree

- We provide a utility class to print the AST
  - **cd/util/debug/AstDump.java**
- To check the AST for a test program
  - Examine the .parser.ref file, or the .err file.
- All AST nodes also have a toString() method

DEMO

Traverse the Abstract Syntax Tree

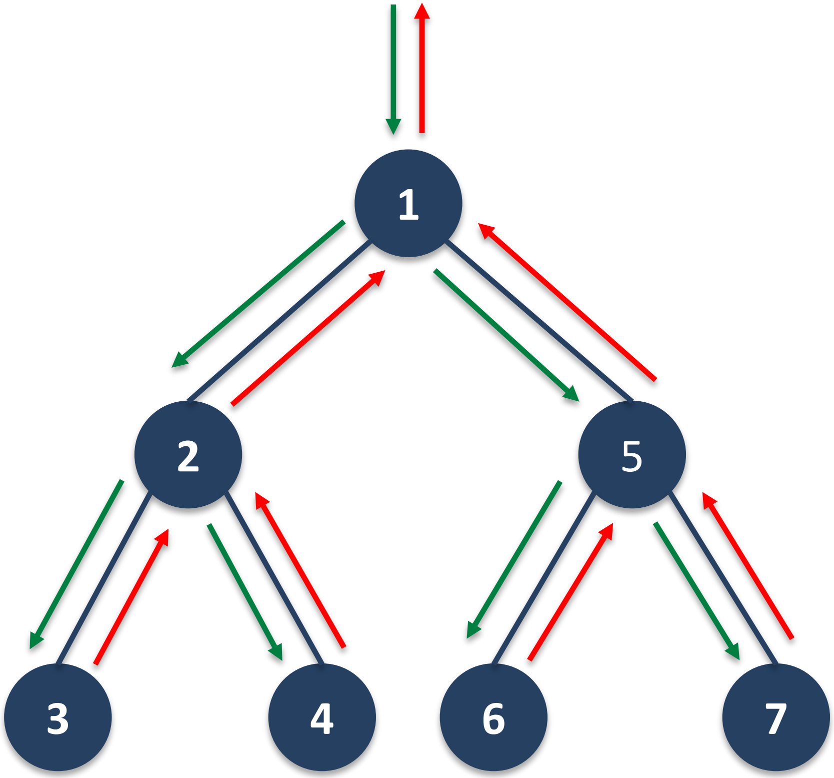
How can we traverse the AST?

↓

Similarly to a (binary) tree.

Traverse a binary tree

- How can we traverse a binary tree?

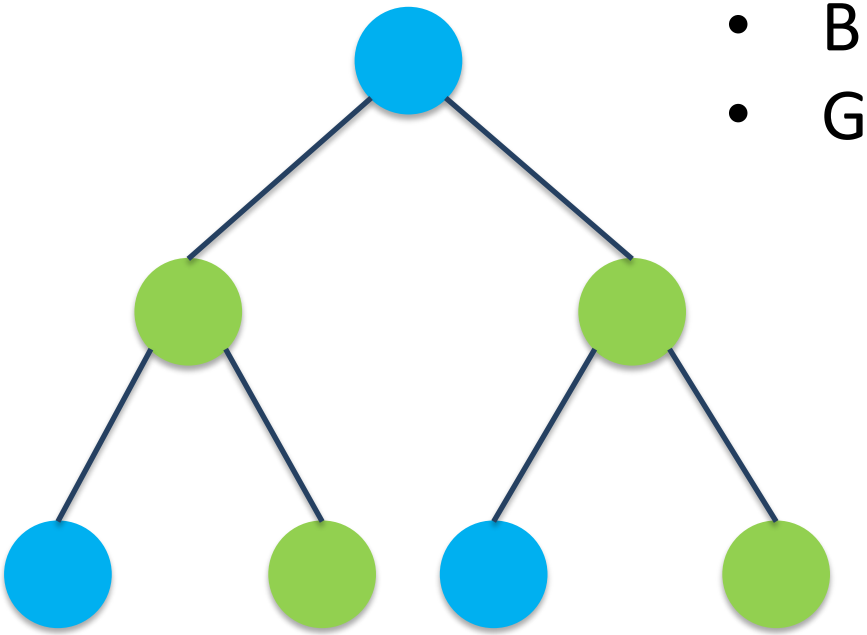


```
public class Visitor {
   void visit(TreeNode node) {
      if (node.leftchild != null)
         visit(node.leftchild);

      if (node.rightchild != null)
         visit(node.rightchild);
   }
}
```

Traverse a binary tree

- What if different nodes have different colors and different behavior?



- Blue nodes: print “blue”
- Green nodes: print “green”

Traverse a binary tree

```
abstract class TreeNode {
        public TreeNode leftchild;
        public TreeNode rightchild;
        …
}
```

```
class BlueNode extends TreeNode { … }

class GreenNode extends TreeNode { … }
```

Traverse a binary tree

```
public class Visitor {

        void visit(TreeNode node) {
                if (node instanceof BlueNode)
                        print "blue";
                else if (node instanceof GreenNode)
                        print "green";

                if (node.leftchild != null)  visit(node.leftchild);
                if (node.rightchild != null)  visit(node.rightchild);
        }
}
```

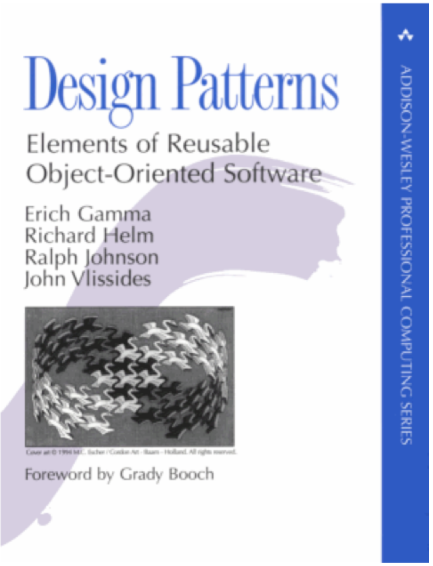
Add node-specific behavior

Traverse a binary tree

- This simple solution works well.
- But, there are more elegant solutions.
- **Usage of design patterns.**

Design patterns

… are descriptions of communicating objects and classes customized to solve a general design problem in a particular context



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Gamma et al.
Design Patterns
Elements of Reusable Object-Oriented Software

Design patterns

Wikipedia says:

- It is a general reusable solution to a commonly occurring problem within a given context in software design
- It is a description or template for how to solve a problem that can be used in many different situations

Design patterns

- There are many design patterns
- Patterns are related and can be combined
  - Design good software is an art
  - We may need multiple “tools” to solve a problem
- Examples of patterns that can be useful to build your compiler
  - *(and that you may find in our reference solution)*

Design patterns - Visitor

Intent

Defines an operation for an object structure.

Description

- Separates an algorithm from an object structure
- Does not change the structure
- Does not change class interface(s)
- Supports distinct unrelated operations

Visitor design pattern solution

```
class BlueNode extends TreeNode {
        void accept(Visitor v) {
                v.blueNode(this);
        }
 }

class GreenNode extends TreeNode {
        void accept(Visitor v) {
                v.greenNode(this);
        }
}
```

We define a method accept() for each node.

The accept method calls the proper method of the visitor class.

Visitor design pattern solution

```
public class Visitor {

    public void blueNode(BlueNode node) {
        print “blue”;
        if (node.leftchild != null) node.leftchild.accept(this);
        if (node.rightchild != null) node.rightchild.accept(this);
    }

    public void greenNode(GreenNode node) {
        print “green”;
        if (node.leftchild != null) node.leftchild.accept(this);
        if (node.rightchild != null) node.rightchild.accept(this);
    }
}
```

DEMO

Traverse the Abstract Syntax Tree

- Javali framework implements two main visitors to traverse the AST:
  - **ExprVisitor<R,A>** for Expressions
  - **AstVisitor<R,A>** for Statements and Declarations

Generic parameters

R = result
A = argument

Apply an operation for each AST node

- Avoid to modify AST class

Traverse the Abstract Syntax Tree

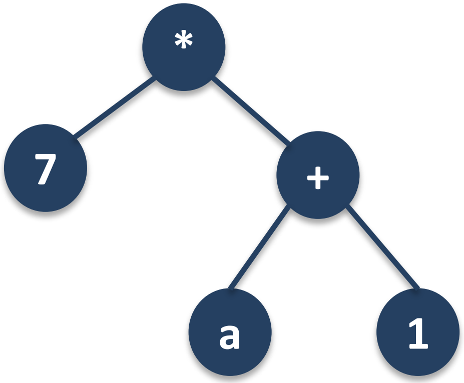
Two visitors for the **code generator**:

- **ExprGenerator** extends ExprVisitor<Register, Void>
- **StmtGenerator** extends AstVisitor<Register, Void>

Traverse the Abstract Syntax Tree

You can implement another Visitor to traverse the AST of an expression:

- calculate the required number of registers
- e.g., 7 * (a + 1)



HW1 summary

- Implementation of simple code generator
- No stack frame necessary
- Use **.data section** slots for each variable
- Look for throw new **ToDoException**()
- Use registers for intermediate results
- Use optimal number of registers