

# Programming Languages and Compilers

Prof. Dr. Uwe Kastens

WS 2013 / 2014

## 0. Introduction

### Objectives

The participants are taught to

- understand properties and notions of programming languages
- understand **fundamental techniques** of language implementation, and to use **generating tools and standard solutions**,
- apply compiler techniques for design and implementation of **specification languages and domain specific languages**

Forms of teaching:

**Lectures**

**Tutorials**

**Homeworks**

**Exercises**

**Running project**

## Contents

Week	Chapter
1	0. Introduction
2	1. Language Properties and Compiler tasks
3 - 4	2. Symbol Specification and Lexical Analysis
5 - 7	3. Context-free Grammars and Syntactic Analysis
8 - 10	4. Attribute Grammars and Semantic Analysis
11	5. Binding of Names
12	6. Type Specification and Analysis
13	7. Specification of Dynamic Semantics
13	8. Source-to-Source Translation
	9. Domain Specific Languages
	Summary

## Prerequisites

from Lecture	Topic	here needed for
	Foundations of Programming Languages:	
	4 levels of language properties	Language specification, compiler tasks
	Context-free grammars	Grammar design, syntactic analysis
	Scope rules	Name analysis
	Data types	Type specification and analysis
	Modeling:	
	Finite automata	Lexical analysis
	Context-free grammars	Grammar design, syntactic analysis

## References

Material for this course **PLaC**: <http://ag-kastens.upb.de/lehre/material/plac>  
 for the Master course **Compilation Methods**: <http://ag-kastens.upb.de/lehre/material/compii>

**Modellierung:** <http://ag-kastens.upb.de/lehre/material/model>  
**Grundlagen der Programmiersprachen:** <http://ag-kastens.upb.de/lehre/material/gdp>

John C. Mitchell: **Concepts in Programming Languages**, Cambridge University Press, 2003

R. W. Sebesta: **Concepts of Programming Languages**, 4. Ed., Addison-Wesley, 1999

U. Kastens: **Übersetzerbau**, Handbuch der Informatik 3.3, Oldenbourg, 1990  
 (not available on the market anymore, available in the library of the University)

A. W. Appel: **Modern Compiler Implementation in Java**, Cambridge University Press,  
 2nd Edition, 2002 (available for C and for ML, too)

W. M. Waite, L. R. Carter: **An Introduction to Compiler Construction**,  
 Harper Collins, New York, 1993

U. Kastens, A. M. Sloane, W. M. Waite: **Generating Software from Specifications**,  
 Jones and Bartlett Publishers, 2007

## References for Reading

Week	Chapter	Kastens	Waite Carter	Eli Doc.
1	0. Introduction			
2	1. Language Properties and Compiler tasks	1, 2	1.1 - 2.1	
3 - 4	2. Symbol Specification and Lexical Analysis	3	2.4 3.1 - 3.3	+
5 - 7	3. Context-free Grammars and Syntactic Analysis	4	4, 5, 6	+
8 - 10	4. Attribute Grammars and Semantic Analysis	5		+
11	5. Binding of Names	6.2	7	+
12	6. Type Specification and Analysis	(6.1)		+
13	7. Specification of Dynamic Semantics			
13	8. Source-to-Source Translation			
	9. Domain Specific Languages			

## Course material in the Web

Lecture Programming Languages and Compilers WS 2013/14

UNIVERSITÄT PADERBORN  
Die Universität der Informationsgesellschaft

Fachgruppe Kastens > Lehre > Programming Languages and Compilers WS 2013/14

**Lecture Programming Languages and Compilers WS 2013/14**

Slides	Assignments
<ul style="list-style-type: none"> <li>Chapters</li> <li>Slides</li> <li>Printing</li> </ul>	<ul style="list-style-type: none"> <li>Assignments</li> <li>Printing</li> </ul>
Organization	Ressources
<ul style="list-style-type: none"> <li>General Information</li> <li>News</li> </ul> <p>04.10.2013 Lectures begin on Mo October 14 at 09:15, Room FD.530.</p>	<ul style="list-style-type: none"> <li>Objectives</li> <li>Prerequisites</li> <li>Literature</li> <li>Online Reading Material (Koala)</li> <li>Eli Online Documentation</li> </ul>

Veranstaltungs-Nummer: L.079.05505  
 Generiert mit Camelki | Probleme mit Camelki? | Geändert am: 06.10.2013

## Commented slide in the course material

### Programming Languages and Compilers WS 2012/13 - Slide 009

**What does a compiler compile?**

A compiler transforms correct sentences of its **source language** into sentences of its **target language** such that their **meaning is unchanged**. Examples:

Source language:	Target language:
Programming language C++	Machine language Sparc code
Programming language Java	Abstract machine Java Bytecode
Programming language C++	Programming language (source-to-source) C
Domain specific language LaTeX Data base language (SQL)	Application language HTML Data base system calls
Application generator: Domain specific language SIM Toolkit language	Programming language Java

Some languages are **interpreted** rather than compiled:  
 Lisp, Prolog, Script languages like PHP, JavaScript, Perl

## Organization of the course

### Programming Languages and Compilers WS 2013/14 - Organization

Lecturer
<b>Prof. Dr. Uwe Kastens:</b> <b>Office Hours</b> <ul style="list-style-type: none"> <li>• Wed 16.00 – 17.00 P2.308</li> <li>• Tue 11.00 – 12.00 P2.308</li> </ul>
Hours
<b>Lecture</b> <ul style="list-style-type: none"> <li>• V2 Mo 09.15 – 10.45, P0.530</li> </ul> <b>Start date: Oct 14, 2013</b>
<b>Exercises</b> <ul style="list-style-type: none"> <li>• U1 Mo 11.00 – 11.45, P0.530 / F1.520</li> </ul> <b>Start date: Oct 14, 2013</b>
<b>Examination</b> <p>Oral examinations of 20 to 30 min duration. Any topic of the lecture and of the tutorial may be subject of the exam. See also the sequence of questions in Chapter 10.</p> <p>Two time spans are offered for examinations:</p> <ol style="list-style-type: none"> <li>1. Feb 12 to 14 in 2014</li> <li>2. April 01 to 03 in 2014</li> </ol> <p>Register in FALL for the one or the other time span; then ask for an appointment by email to my secretary Mrs. Gundelach (tigi@upb.de).</p>
Assignments
<ul style="list-style-type: none"> <li>• Assignments will be published every week.</li> </ul>

## What does a compiler compile?

A **compiler** transforms correct sentences of its **source language** into sentences of its **target language** such that their **meaning is unchanged**. Examples:

<b>Source language:</b>	<b>Target language:</b>
<b>Programming language</b>	<b>Machine language</b>
C++	Sparc code
<b>Programming language</b>	<b>Abstract machine</b>
Java	Java Bytecode
<b>Programming language</b>	<b>Programming language (source-to-source)</b>
C++	C
<b>Domain specific language</b>	<b>Application language</b>
LaTeX	HTML
Data base language (SQL)	Data base system calls
<b>Application generator:</b>	<b>Programming language</b>
<b>Domain specific language</b>	Java
SIM Toolkit language	

Some languages are **interpreted** rather than compiled:  
Lisp, Prolog, Script languages like PHP, JavaScript, Perl

## What is compiled here?

```
class Average
{ private:
  int sum, count;
public:
  Average (void)
  { sum = 0; count = 0; }
  void Enter (int val)
  { sum = sum + val; count++; }
  float GetAverage (void)
  { return sum / count; }
};
```

-----  
\_Enter\_7AverageI:

```
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%edx
    movl 12(%ebp),%eax
    addl %eax,(%edx)
    incl 4(%edx)

L6:
    movl %ebp,%esp
    popl %ebp
    ret
```

```
class Average
{ private:
  int sum, count;
public:
  Average ()
  { sum = 0; count = 0; }
  void Enter (int val)
  { sum = sum + val; count++; }
  float GetAverage ()
  { return sum / count; }
};
```

1: Enter: (int) --> void

Access: []

Attribute 'Code' (Length 49)

```
Code: 21 Bytes Stackdepth: 3 Locals: 2
0:  aload_0
1:  aload_0
2:  getfield cp4
5:  iload_1
6:  iadd
7:  putfield cp4
10: aload_0
11: dup
12: getfield cp3
15: iconst_1
16: iadd
```

## What is compiled here?

```
program Average;
  var sum, count: integer;
      aver: integer;
  procedure Enter (val: integer);
    begin sum := sum + val;
          count := count + 1;
    end;
  begin
    sum := 0; count := 0;
    Enter (5); Enter (7);
    aver := sum div count;
  end.
```

```
-----
void ENTER_5 (char *slnk , int VAL_4)
{
  /* data definitions: */
  /* executable code: */
  {
    SUM_1 = (SUM_1)+(VAL_4);
    COUNT_2 = (COUNT_2)+(1);
    ;
  }
} /* ENTER_5 */
```

```
\documentstyle[12pt]{article}
\begin{document}
\section{Introduction}
This is a very short document.
It just shows
\begin{itemize}
\item an item, and
\item another item.
\end{itemize}
\end{document}
```

```
-----
%%Page: 1 1
1 0 bop 164 315 a Fc(1)81
b(In)n(tro)r(duction)
164 425 y Fb(This)16
b(is)g(a)h(v)o(ery)e(short)
i(do)q(cumen)o(t.)j(It)c(just)g
(sh)o(o)ws)237 527 y Fa(\017)24 b
Fb(an)17 b(item,)
c(and)237 628 y Fa(\017)24 b
Fb(another)17 b(item.)
961 2607 y(1)p
eop
```

## Languages for specification and modeling

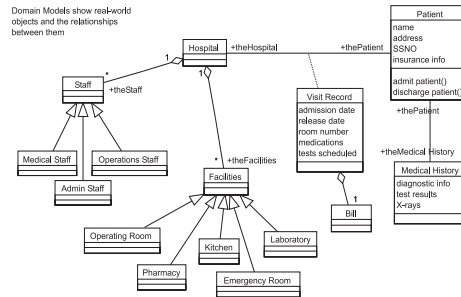
PLaC-0.12

SDL (CCITT)  
Specification and Description Language:

UML  
Unified Modeling Language:

```

block Dialogue;
  signal
    Money, Release, Change, Accept, Avail, Unavail, Price,
    Showtxt, Choice, Done, Flushed, Close, Filled;
  process Coins referenced;
  process Control referenced;
  process Viewpoint referenced;
  signalroute Plop
    from env to Coins
      with Coin_10, Coin_50, Coin_100, Coin_x;
  signalroute Pong
    from Coins to env
      with Coin_10, Coin_50, Coin_100, Coin_x;
  signalroute Cash
    from Coins to Control
      with Money, Avail, Unavail, Flushed, Filled;
    from Control to Coins
      with Accept, Release, Change, Close;
  ...
  connect Pay and Plop;
  connect Flush and Pong;
endblock Dialogue;
    
```



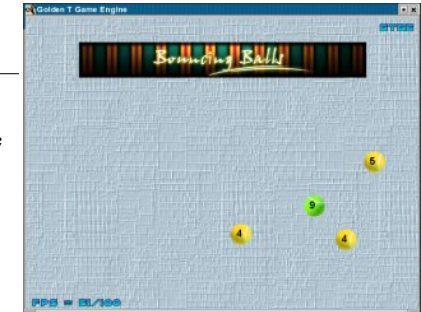
## Domain Specific Languages (DSL)

PLaC-0.13

A language designed for a **specific application domain**.  
**Application Generator:** Implementation of a DSL by a **program generator**

**Examples:**

- Simulation of mechatronic feedback systems
- Robot control
- Collecting data from instruments
- Testing car instruments
- **Game description language:**



```

game BBall
{
  size 640 480;
  background "pics/backgroundbb.png";
  Ball einball; int ballsize;

  initial {
    ballsize=36;
  }

  events {
    pressed SPACE:
      { einball = new Ball(<100,540>, <100,380>);
    }
  }
}
    
```

## Programming languages as source or target languages

PLaC-0.14

**Programming languages as source languages:**

- **Program analysis**  
call graphs, control-flow graph, data dependencies,  
e. g. for the year 2000 problem
- **Recognition of structures and patterns**  
e. g. for Reengineering

**Programming languages as target languages:**

- Specifications (SDL, OMT, UML)
- graphic modeling of structures
- DSL, Application generator

=> **Compiler task: Source-to-source compilation**

## Semester project as running example

PLaC-0.15

### SetLan: A Language for Set Computation

**SetLan** is a domain-specific language for **programming with sets**.  
Constructs of the the language are dedicated to describe sets and computations using sets. The language allows to define types for sets and variables and expressions of those types. Specific loop constructs allow to iterate through sets. These constructs are embedded in a simple imperative language.

A source-to-source translator **translates SetLan programs into Java** programs.

The SetLan translator is implemented using the methods and tools introduced in this course.

The participants of this course get an implementation of a **sub-language of SetLan as a starting point** for their work towards their individual extension of the language and the implementation.

```

{
  set a, b; int i;
  i = 1;
  a = [i, 3, 5];
  b = [3, 6, 8];
  print a+b; println;
  print a*b <= b;
  println;
}
    
```