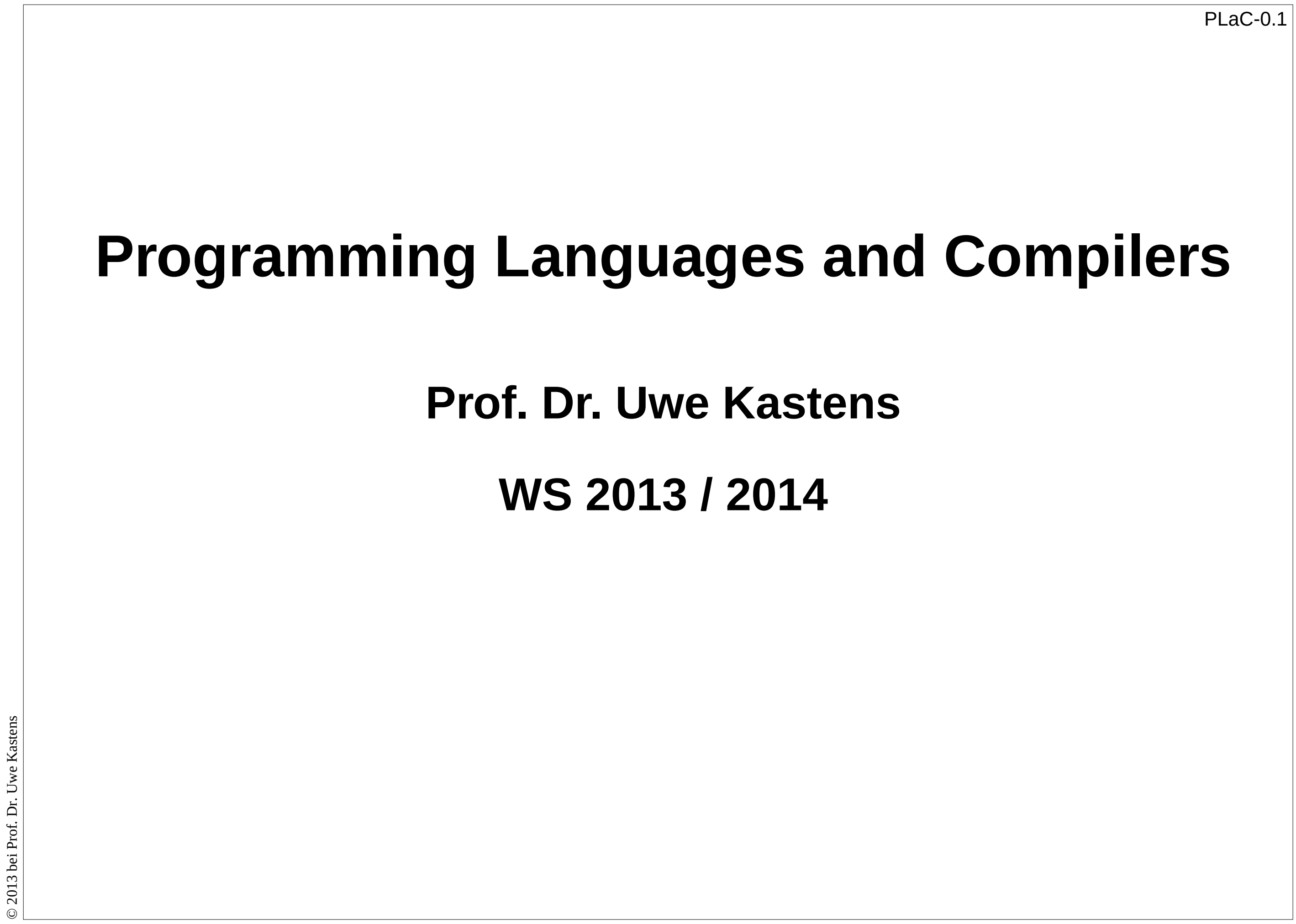


PLaC-0.1

# Programming Languages and Compilers

## Prof. Dr. Uwe Kastens

## WS 2013 / 2014

### Lecture Programming Languages and Compilers WS 2013/14 / Slide 001

**In the lecture:**

Welcome to the lecture!

# 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**

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 002

**Objectives:**

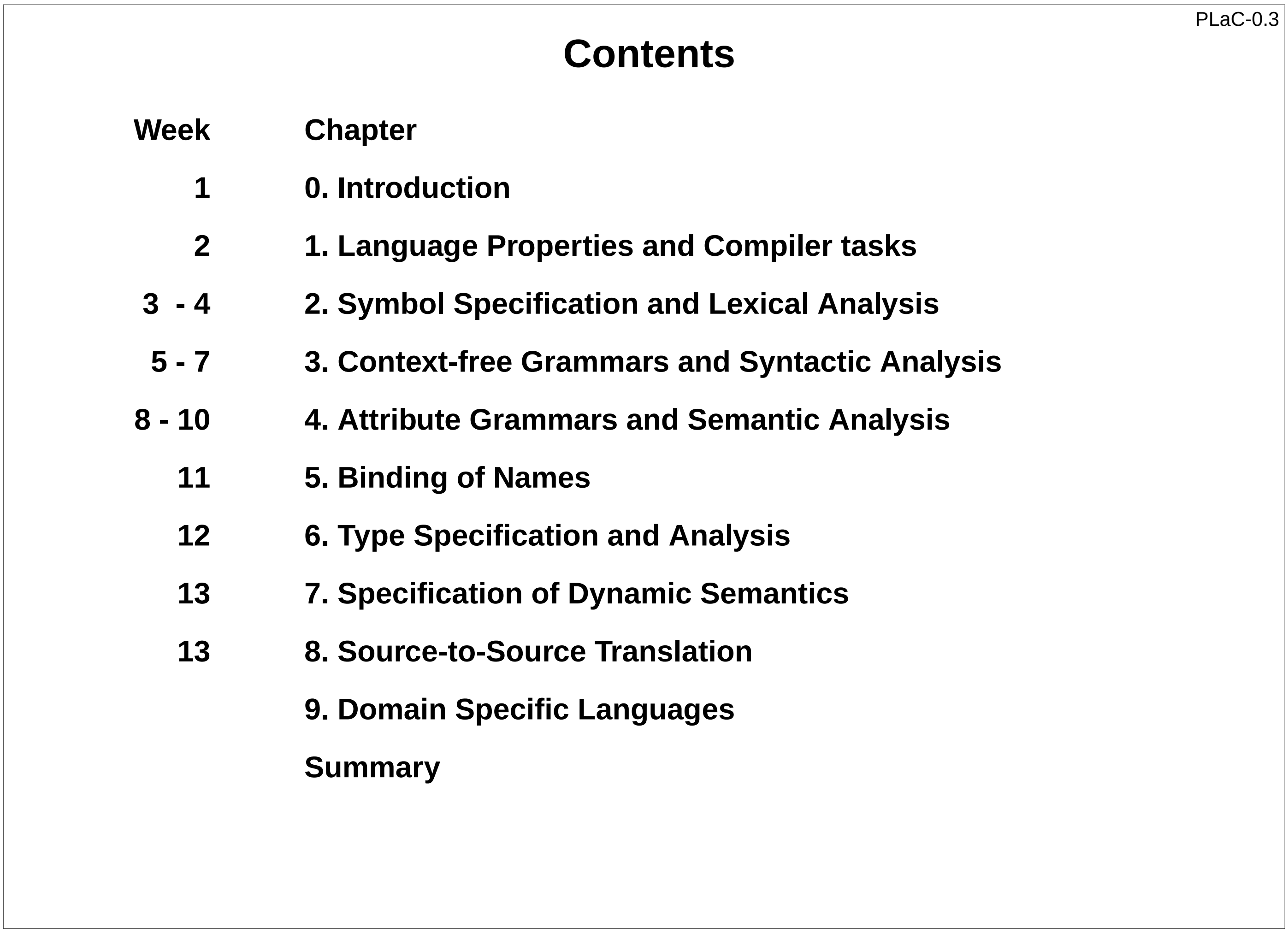
Understand the objectives of the course.

**In the lecture:**

The objectives are explained.

**Questions:**

- What are your objectives?
- Do they match with these?
- When did you last listen to a talk given in English?



PLaC-0.3

# 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 |

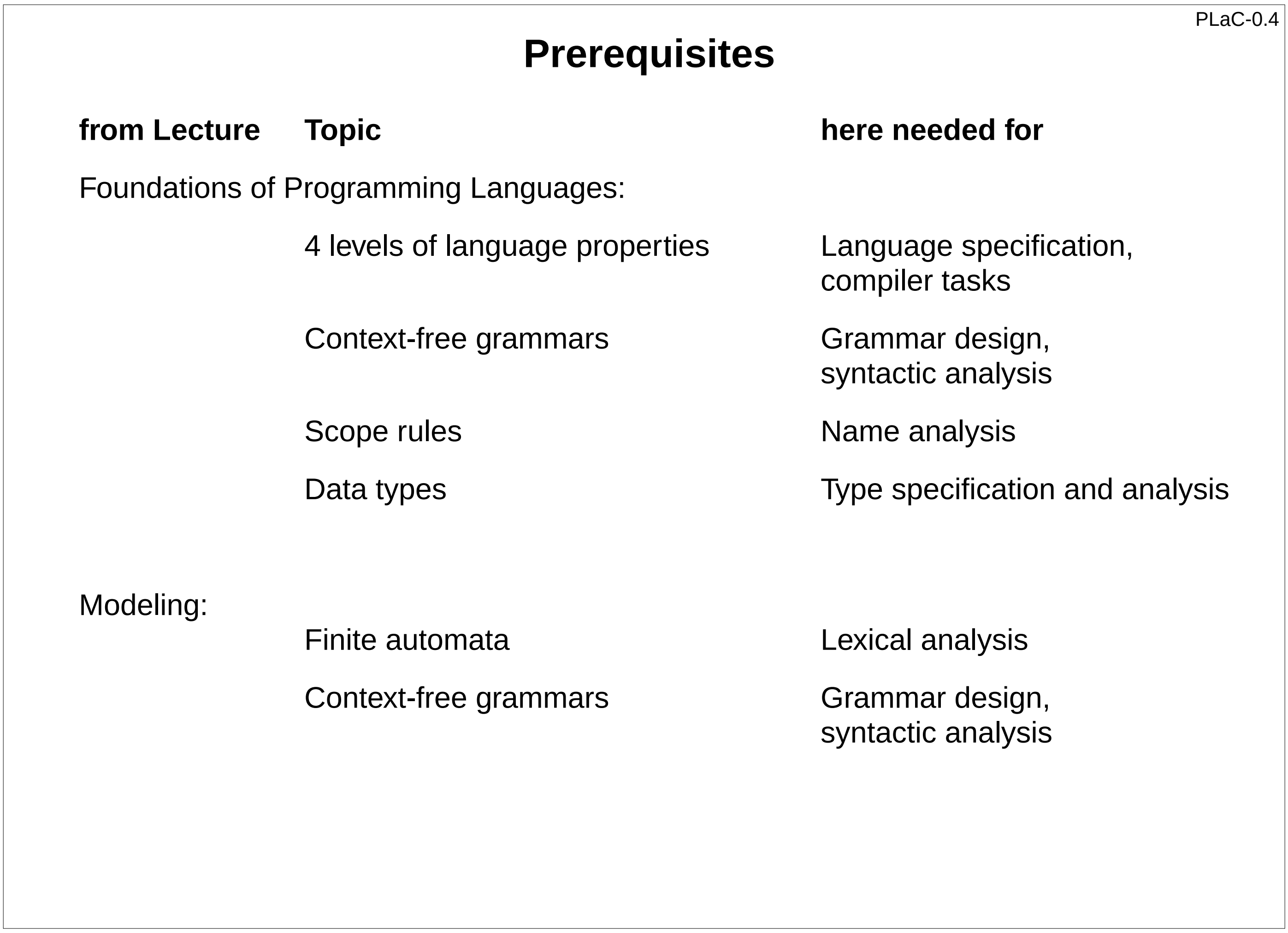
## Lecture Programming Languages and Compilers WS 2013/14 / Slide 003

**Objectives:**

Overview over the topics of the course

**In the lecture:**

Comments on the topics.



PLaC-0.4

# Prerequisites

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

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 004

**Objectives:**

Identify concrete topics of other courses

**In the lecture:**

Point to material to be used for repetition

**Suggested reading:**

- Course material for *Foundations of Programming Languages*
- Course material for *Modeling*

**Questions:**

- Do you have the prerequisites?
- Are you going to learn or to repeat that material?

# 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

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 005

**Objectives:**

Useful references for the course

**In the lecture:**

Comments of the course material and books

**Questions:**

- Find the material in the Web, get used to its structure, place suitable bookmarks.

# References forReading

| 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<br>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 | | | |

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 005a

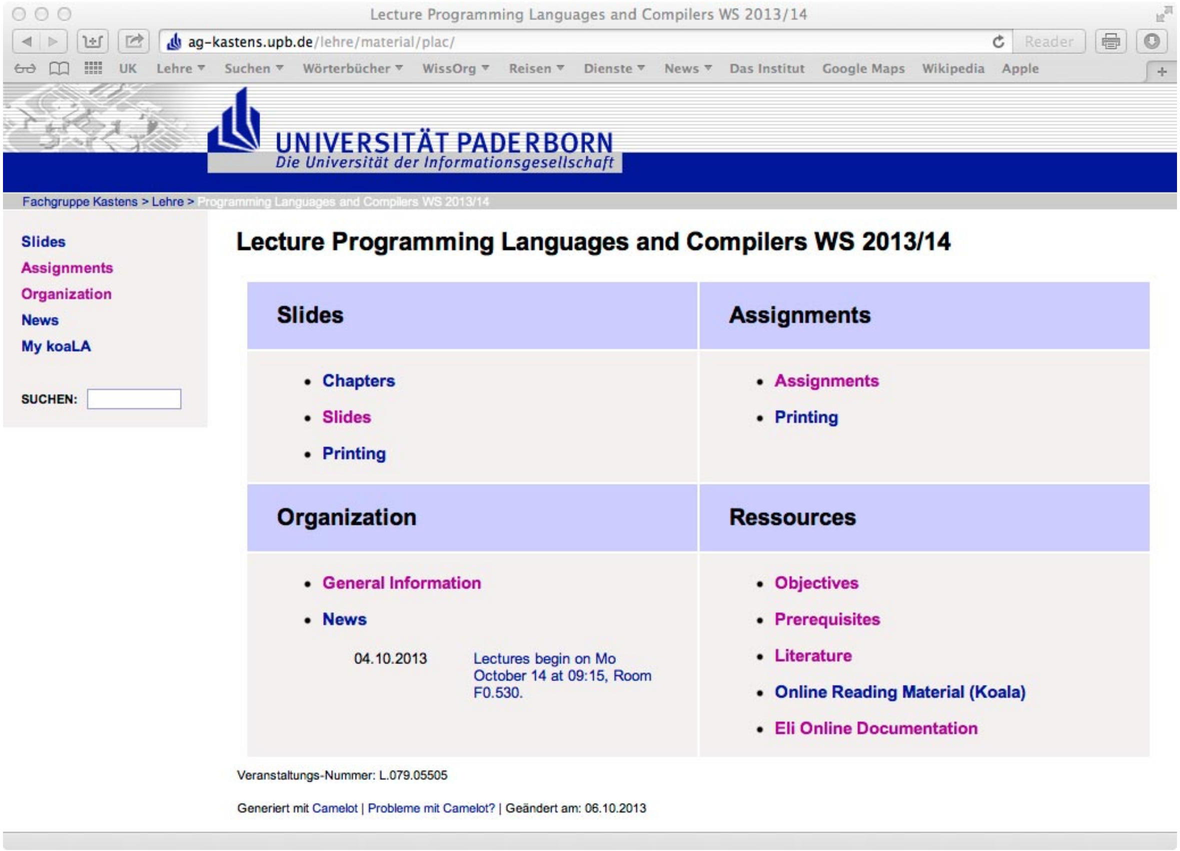
**Objectives:**

Associate reading material to course topics

**In the lecture:**

Explain the strategy for using the reading material

# Course material in the Web


Lecture Programming Languages and Compilers WS 2013/14
ag-kastens.upb.de/lehre/material/plac/
Reader
UK
Lehre ▾
Suchen ▾
Wörterbücher ▾
WissOrg ▾
Reisen ▾
Dienste ▾
News ▾
Das Institut
Google Maps
Wikipedia
Apple
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

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

- Slides
- Assignments
- Organization
- News
- My koaLA

SUCHEN:

## Lecture Programming Languages and Compilers WS 2013/14

### Slides

- Chapters
- Slides
- Printing

### Assignments

- Assignments
- Printing

### Organization

- General Information
- News

| | |
|---|---|
| 04.10.2013 | Lectures begin on Mo October 14 at 09:15, Room F0.530. |

### Ressources

- Objectives
- Prerequisites
- Literature
- Online Reading Material (Koala)
- Eli Online Documentation

Veranstaltungs-Nummer: L.079.05505

Generiert mit Camelot | Probleme mit Camelot? | Geändert am: 06.10.2013

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 006

**Objectives:**

The root page of the course material.

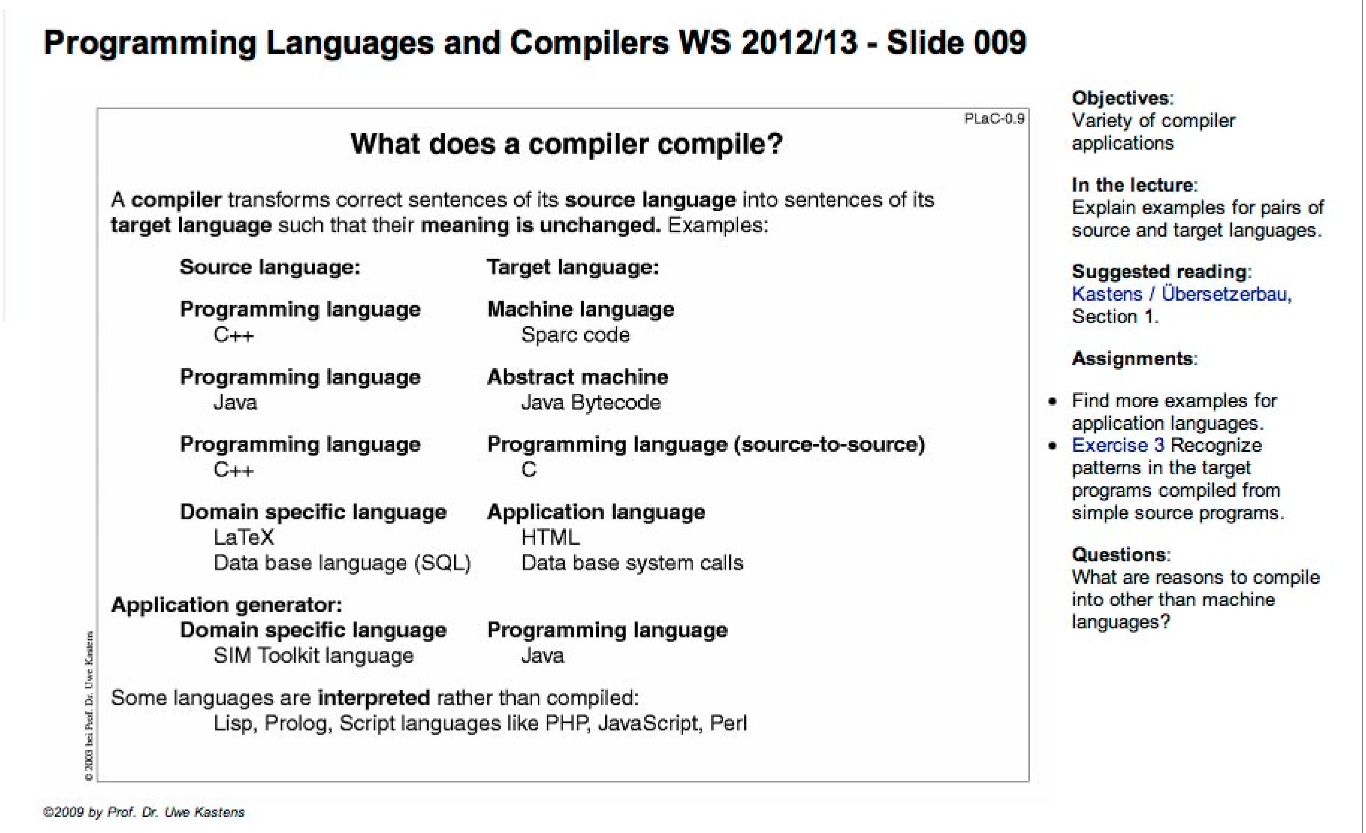
**In the lecture:**

The navigation structure is explained.

**Assignments:**

Explore the course material.

# Commented slide in the course material



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

PLaC-0.9

### 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**<br>C++ | **Machine language**<br>Sparc code |
| **Programming language**<br>Java | **Abstract machine**<br>Java Bytecode |
| **Programming language**<br>C++ | **Programming language (source-to-source)**<br>C |
| **Domain specific language**<br>LaTeX<br>Data base language (SQL) | **Application language**<br>HTML<br>Data base system calls |

**Application generator:**

| | |
|---|---|
| **Domain specific language**<br>SIM Toolkit language | **Programming language**<br>Java |

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

**Objectives**:
Variety of compiler applications

**In the lecture**:
Explain examples for pairs of source and target languages.

**Suggested reading**:
Kastens / Übersetzerbau, Section 1.

**Assignments**:

- Find more examples for application languages.
- Exercise 3 Recognize patterns in the target programs compiled from simple source programs.

**Questions**:
What are reasons to compile into other than machine languages?

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 007

**Objectives:**

A slide of the course material.

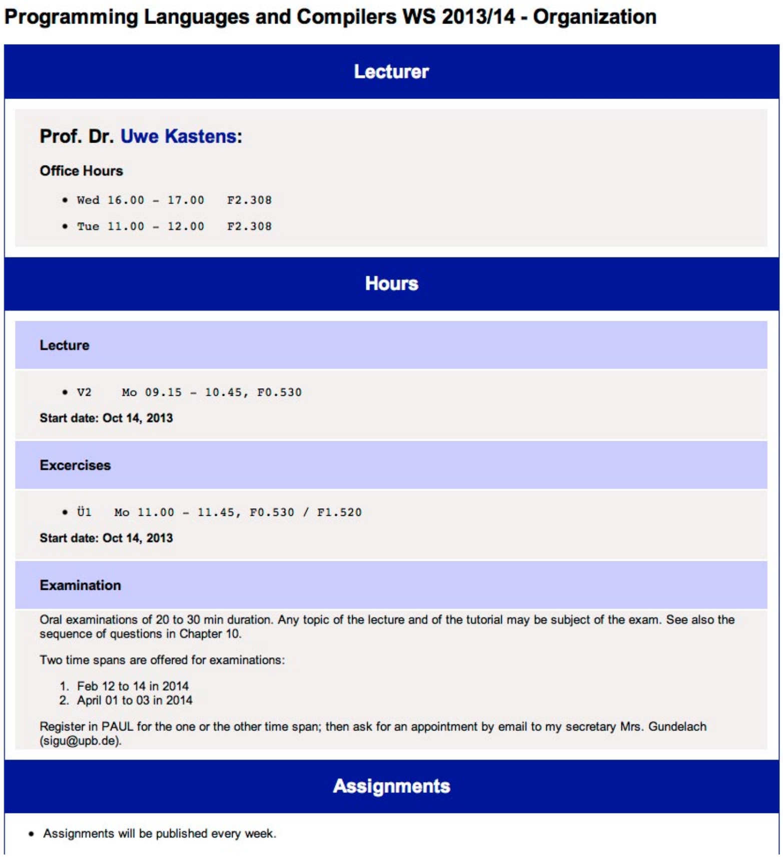
**In the lecture:**

The comments are explained.

**Assignments:**

Explore the course material.

# Organization of the course



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

**Lecturer**

**Prof. Dr. Uwe Kastens:**

**Office Hours**

- Wed 16.00 - 17.00 F2.308
- Tue 11.00 - 12.00 F2.308

**Hours**

**Lecture**

- V2 Mo 09.15 - 10.45, F0.530

**Start date: Oct 14, 2013**

**Excercises**

- Ü1 Mo 11.00 - 11.45, F0.530 / F1.520

**Start date: Oct 14, 2013**

**Examination**

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.

Two time spans are offered for examinations:

1. Feb 12 to 14 in 2014
2. April 01 to 03 in 2014

Register in PAUL for the one or the other time span; then ask for an appointment by email to my secretary Mrs. Gundelach (sigu@upb.de).

**Assignments**

- Assignments will be published every week.

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 008

**Objectives:**

Know how the course is organized

**In the lecture:**

Comments on exams and registration

**Assignments:**

Explore the course material.

# 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** | **Machine language** |
| C++ | Sparc code |
| **Programming language** | **Abstract machine** |
| Java | Java Bytecode |
| **Programming language** | **Programming language (source-to-source)** |
| C++ | C |
| **Domain specific language** | **Application language** |
| LaTeX | HTML |
| Data base language (SQL) | Data base system calls |

**Application generator:**

| | |
|---|---|
| **Domain specific language** | **Programming language** |
| SIM Toolkit language | Java |

Some languages are **interpreted** rather than compiled:

Lisp, Prolog, Script languages like PHP, JavaScript, Perl

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 009

**Objectives:**

Variety of compiler applications

**In the lecture:**

Explain examples for pairs of source and target languages.

**Suggested reading:**

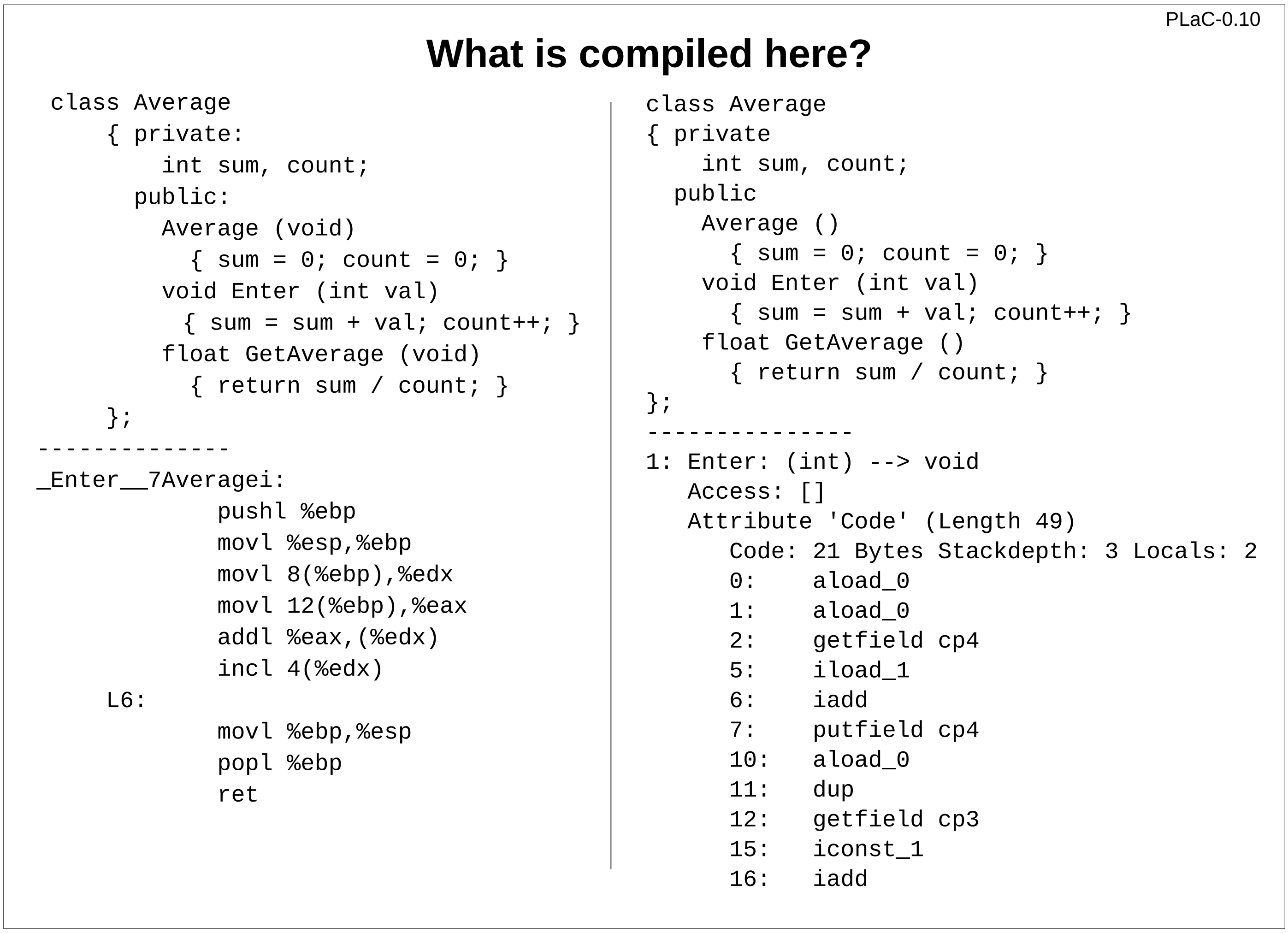
Kastens / Übersetzerbau, Section 1.

**Assignments:**

- Find more examples for application languages.
- Exercise 3 Recognize patterns in the target programs compiled from simple source programs.

**Questions:**

What are reasons to compile into other than machine languages?



# 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
```

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 010

**Objectives:**

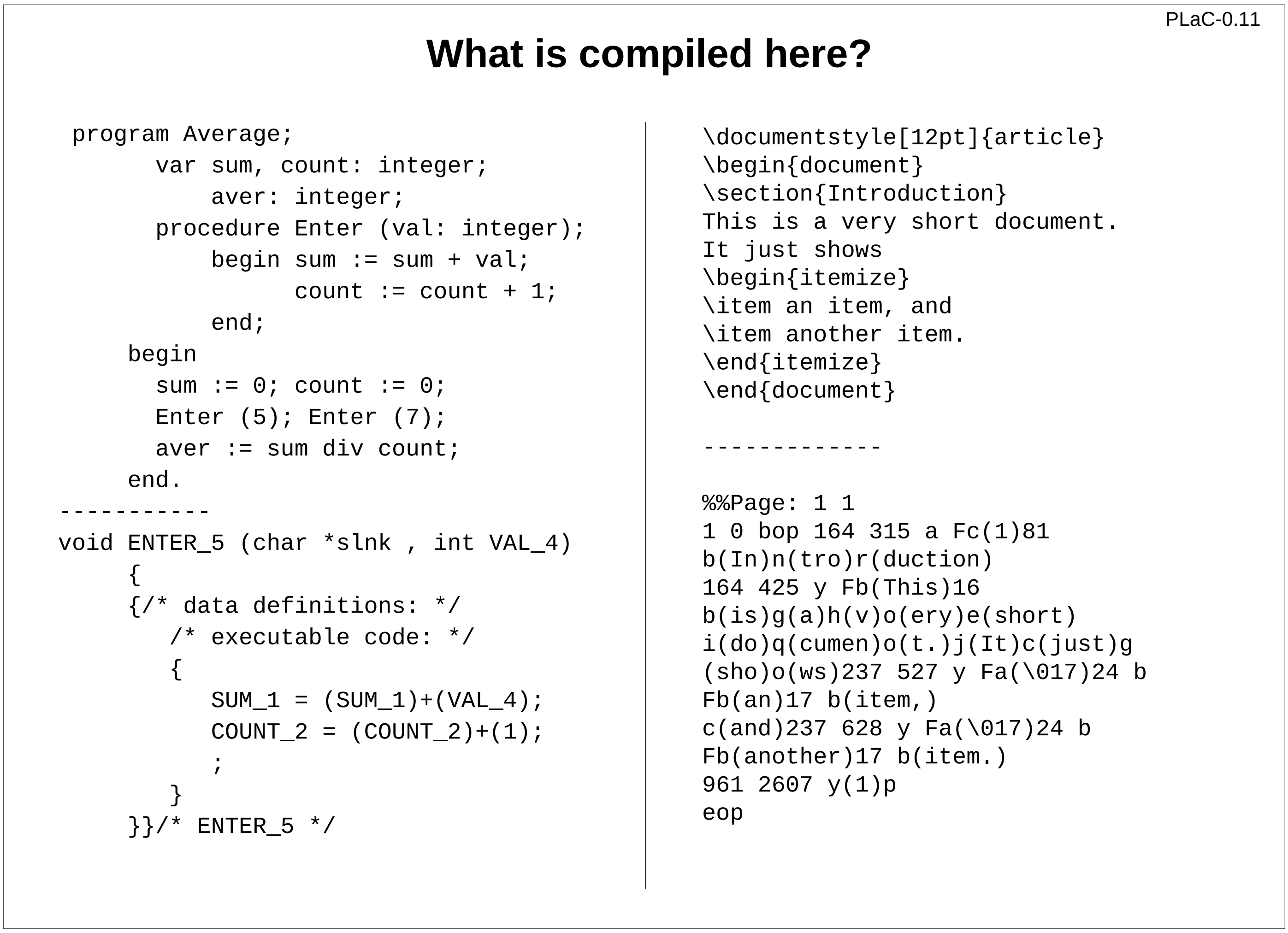
Recognize examples for compilations

**In the lecture:**

Anwer the questions below.

**Questions:**

- Which source and target language are shown here?
- How did you recognize them?



PLaC-0.11

# 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
(sho)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
```

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 011

**Objectives:**

Recognize examples for compilations

**In the lecture:**

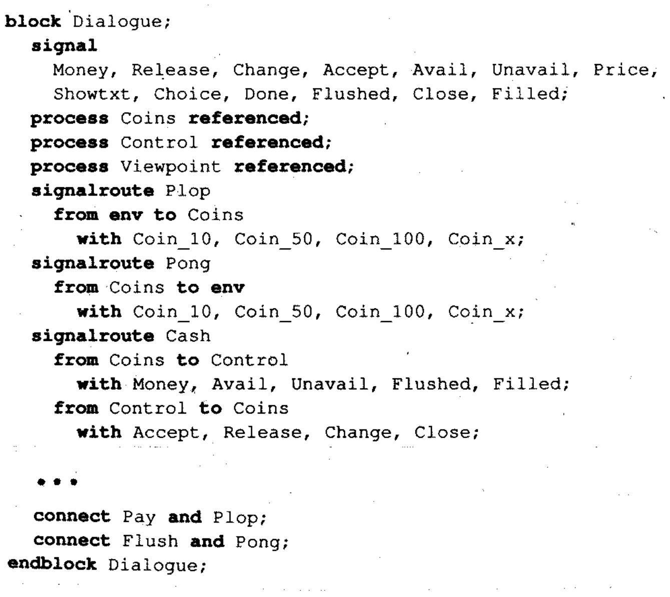
Anwer the questions below.

**Questions:**

- Which source and target language are shown here?
- How did you recognize them?

# Languages for specification and modeling

SDL (CCITT)
Specification and Description Language:

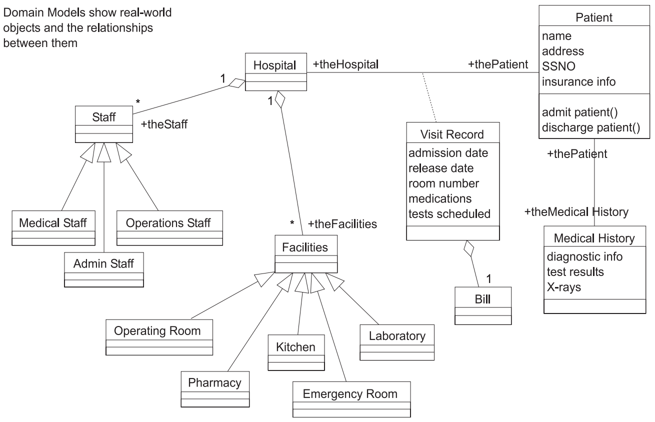


```
block Dialogue;
  signal
    Money, Release, Change, Accept, Avail, Unavail, Price,
    Showtxt, Choice, Done, Flushed, Close, Filled;
  process Coins referenced;
  process Control referenced;
  process Viewpoint referenced;
  signalroute Plop
    from env to Coins
      with Coin_10, Coin_50, Coin_100, 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;
```

UML
Unified Modeling Language:



Domain Models show real-world objects and the relationships between them

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 012

**Objectives:**

Be aware of specification languages

**In the lecture:**

Comments on SDL and UML

**Suggested reading:**

Text

**Questions:**

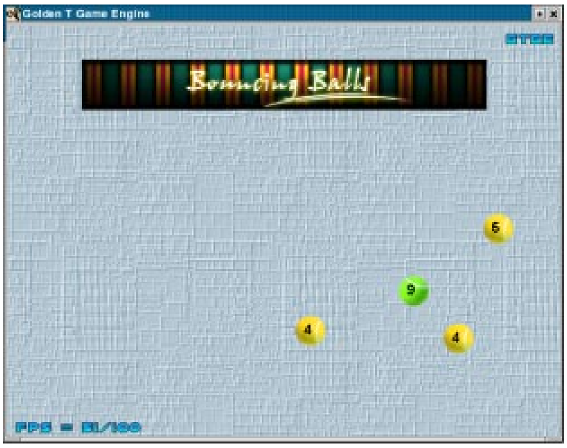
What kind of tools are needed for such specification languages?

# Domain Specific Languages (DSL)

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>);
```

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 013

**Objectives:**

Understand DSL by examples

**In the lecture:**

Explain the examples

**Suggested reading:**

- C.W. Krueger: Software Reuse, ACM Computing Surveys 24, June 1992
- Conference on DSL (USENIX), Santa Babara, Oct. 1997
- ACM SIGPLAN Workshop on DSL (POPL), Paris, Jan 1997

**Questions:**

Give examples for tools that can be used for such languages.

PLaC-0.14

# Programming languages as source or target languages

**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**

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 014

**Objectives:**

Understand programming languages in different roles

**In the lecture:**

- Comments on the examples
- Role of program analysis in software engineering
- Role of Source-to-source compilation in software engineering

**Questions:**

Give examples for the use of program analysis in software engineering.

# Semester project as running example

## 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;
}
```

## Lecture Programming Languages and Compilers WS 2013/14 / Slide 015

**Objectives:**

Get an idea of the task

**In the lecture:**

- Comment the task description.
- Explain the role of the running example.

**Assignments:**

In the tutorial

- Explain the application domain.
- What may a game description contain?
- Give examples for typical language constructs.
- Explore the language.
- Use the language.