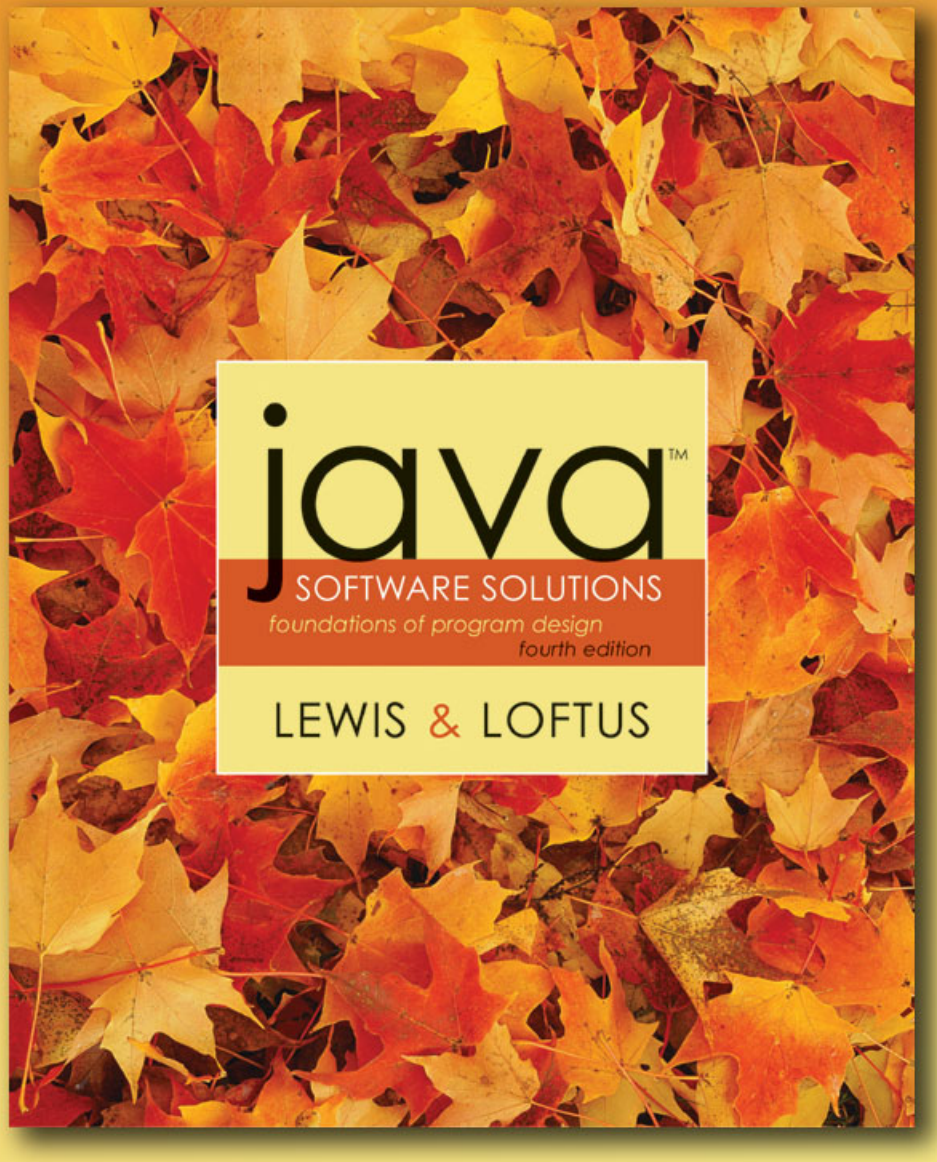
Lecture 1

Introduction



PEARSON Addison Wesley

Programming 1 (P1)

- A lecture in the Bachelor program Computer Science (1th Semester; 5 ECTS)
- Introduction to programming
- Notions and principles of object-oriented programming (objects, classes, inheritance etc.)
- Introduction to the Java programming language
- practical work with LINUX

Names and Addresses

Institut für Informatik (INF)

Institutssekretariat	D. Schroth	Neubrückstrasse 10 / 212	031 631 86 81
Prüfungssekretariat	B. Choffat	Neubrückstrasse 10 / 112	031 631 84 26

Peppo Brambilla Neubrückstrasse 10 / Büro 312
brambi@inf.unibe.ch 031 631 33 10

Ramona Beck
Lukas Jaun
Pascal Zaugg

} **p1-beratung@list.inf.unibe.ch**

Workstations **ExWi** **A93, A94, A95**
Systemadministration, LINUX, UNIX: *root@inf.unibe.ch*

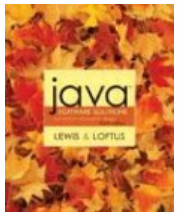
Course Home Page

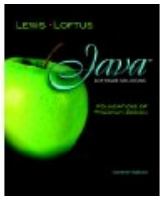
- The course home page contains lots of material and detailed information on the course
- Please visit the page regularly in order to be up-to-date on the P1 lecture
- The website is located at:

http://www.ilias.unibe.ch

- The most up-to-date version of the lecture slides will always be available on this website
- Register for the ILIAS page with the password announced in the course

Textbook





- **The textbook for this course is:**
  John Lewis, William Loftus
  Java Software Solutions
  4th or 5th or 6th or 7th or 8th edition, Pearson, Addison Wesley, 2005/2006/2008/2011/2014
- **The book can be bought at Studentische Buchgenossenschaft, Hauptgebäude, Uni Bern**
- **The etext of the book can be purchased here: http://www.vitalsource.com/**

Examples

- The Java program examples used during the lecture are from the Lewis/Loftus book
- All the examples are available on the P1 website in ILIAS
- The examples and other information on the book is also collected at the author‘s book website http://wps.prenhall.com/ecs_lewis_jss_8/
- The examples are linked from the lecture slides, but not part of the slides
- The students may want to print out the slides or examples before attending the lectures

Timetable

Event	Time	Location	Start
Lecture	Friday, 13-15	ExWi A6	Sep 23
Exercises	Friday, 15-16	ExWi A6	Sep 23
P1 Pool	Wednesday, 12-14	ExWi Pools	Sep 28
Consulting hour		ExWi Pools	By appoint-ment only

Practica Lessons

- Not compulsory, but recommended
- Possibility to solve exercises with the help of assistants
- Three assistants will be available
- Time: Wednesday, 12-14
- Start: Sep 28
- Location: ExWi pools A93, A94 and A95

Consulting Hour

- Not compulsory
- Individual consulting with an assistant
- All questions related to first steps in programming
- By appointment only

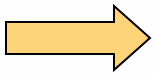
Focus of the Course

- **Object-Oriented Software Development**
  - **problem solving**
  - **program design, implementation, and testing**
  - **object-oriented concepts**
    - **classes**
    - **objects**
    - **encapsulation**
    - **inheritance**
    - **polymorphism**
  - **the Java programming language**

Today: Introduction

- The introduction focuses on:
  - programming and programming languages
  - an introduction to Java
  - an overview of object-oriented concepts
  - an introduction to LINUX

Outline



Organizational issues

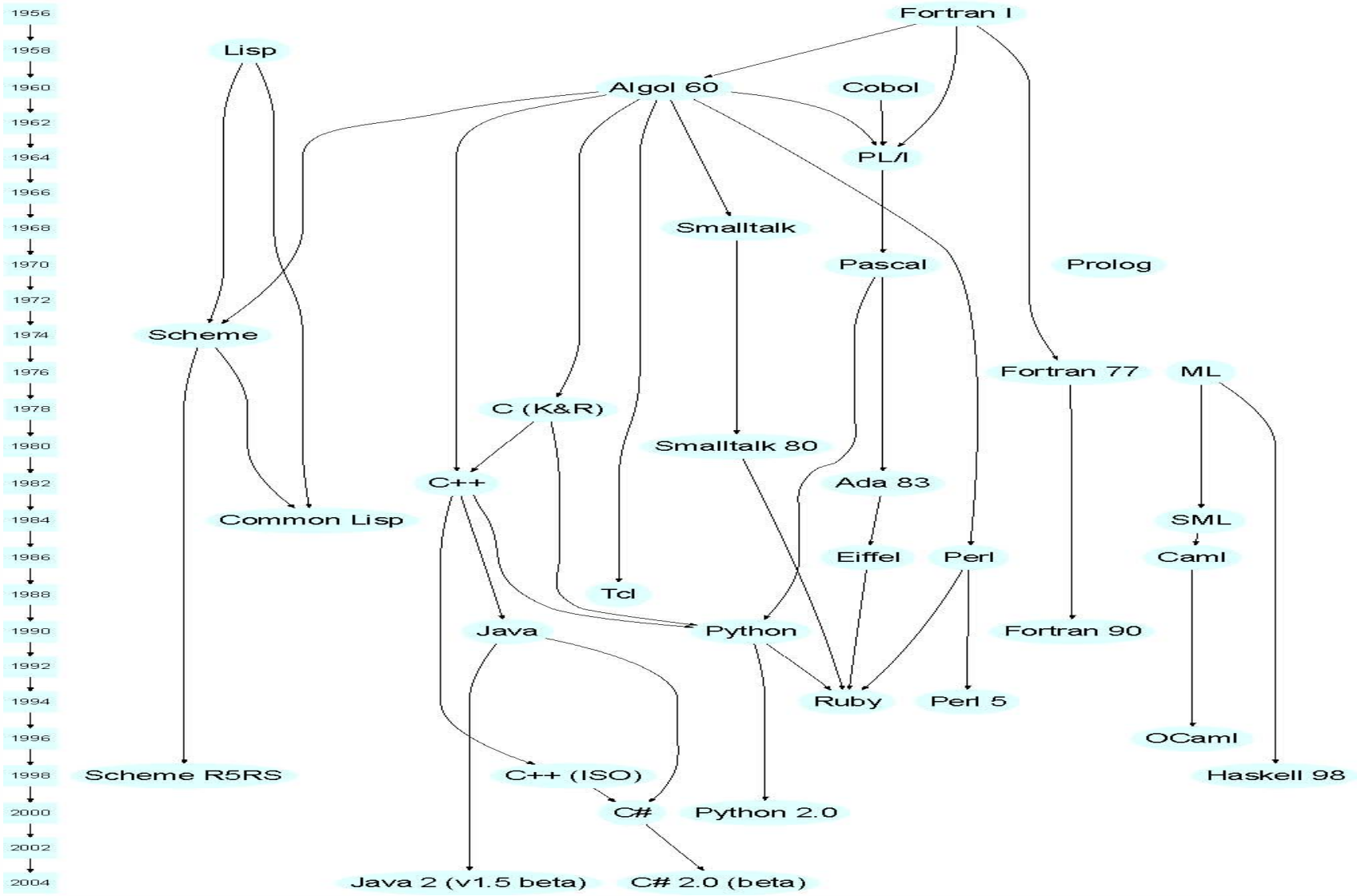
The Java Programming Language

Program Development

Object-Oriented Programming

UNIX/LINUX

Programming Language Landscape



Programming Paradigms

- procedural (“imperativ”)
  - Pascal, Fortran, Modula, C, ...
- functional (“applikativ”)
  - LISP, Haskell, …
- logic (“relational”)
  - PROLOG, …
- object-oriented (“direktiv”)
  - Simula, Smalltalk, C++, Eiffel, Java, …

Java characteristics

- object-oriented
- secure
- architecture-neutral
- parallel
- robust

Java

- A *programming language* specifies the words and symbols that we can use to write a program
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*
- The Java programming language was created by Sun Microsystems, Inc.
- It was introduced in 1995 and it's popularity has grown quickly since

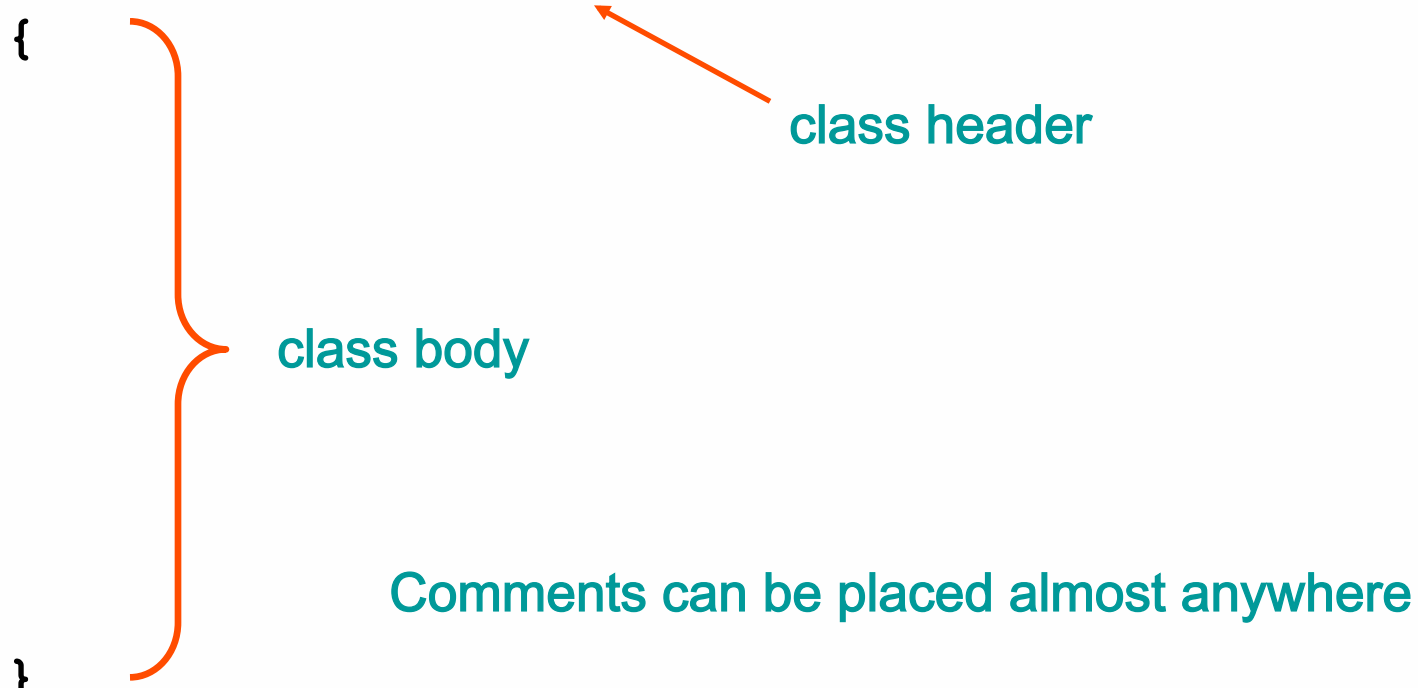
Java Program Structure

- In the Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements*
- These terms will be explored in detail throughout the course
- A Java application always contains a method called `main`
- See `Lincoln.java` (page 28)

Java Program Structure

```
//  comments about the class
public class MyProgram
{


}
```



Java Program Structure

```
//  comments about the class
public class MyProgram
{
   //  comments about the method
   public static void main (String[] args)
   {

   }

}
```

method body

method header

Comments

- Comments in a program are called *inline documentation*
- They should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Java comments can take three forms:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks       */

/** this is a javadoc comment   */
```

Identifiers

- *Identifiers* are the words a programmer uses in a program
- An identifier can be made up of letters, digits, the underscore character (_), and the dollar sign
- Identifiers cannot begin with a digit
- Java is *case sensitive* - `Total`, `total`, and `TOTAL` are different identifiers
- By convention, programmers use different case styles for different types of identifiers, such as
  - *title case* for class names - `Lincoln`
  - *upper case* for constants - `MAXIMUM`

Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as `Lincoln`)
- Sometimes we are using another programmer's code, so we use the identifiers that he or she chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
- A reserved word cannot be used in any other way

Reserved Words

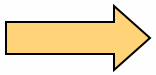
- **The Java reserved words:**

```
abstract    else        interface   switch
assert      enum        long        synchronized
boolean     extends     native      this
break       false       new         throw
byte        final       null        throws
case        finally     package     transient
catch       float       private     true
char        for         protected   try
class       goto        public      void
const       if          return      volatile
continue    implements  short       while
default     import      static
do          instanceof  strictfp
double      int         super
```

White Space

- Spaces, blank lines, and tabs are called *white space*
- White space is used to separate words and symbols in a program
- Extra white space is ignored
- A valid Java program can be formatted many ways
- Programs should be formatted to enhance readability, using consistent indentation
- See `Lincoln2.java` (page 34)
- See `Lincoln3.java` (page 35)

Outline



Organizational issues

The Java Programming Language

Program Development

Object-Oriented Programming

UNIX/LINUX

Program Development

- The mechanics of developing a program include several activities
  - writing the program in a specific programming language (such as Java)
  - translating the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process

Language Levels

- There are four programming language levels:
  - machine language
  - assembly language
  - high-level language
  - fourth-generation language
- Each type of CPU has its own specific *machine language*
- The other levels were created to make it easier for a human being to read and write programs

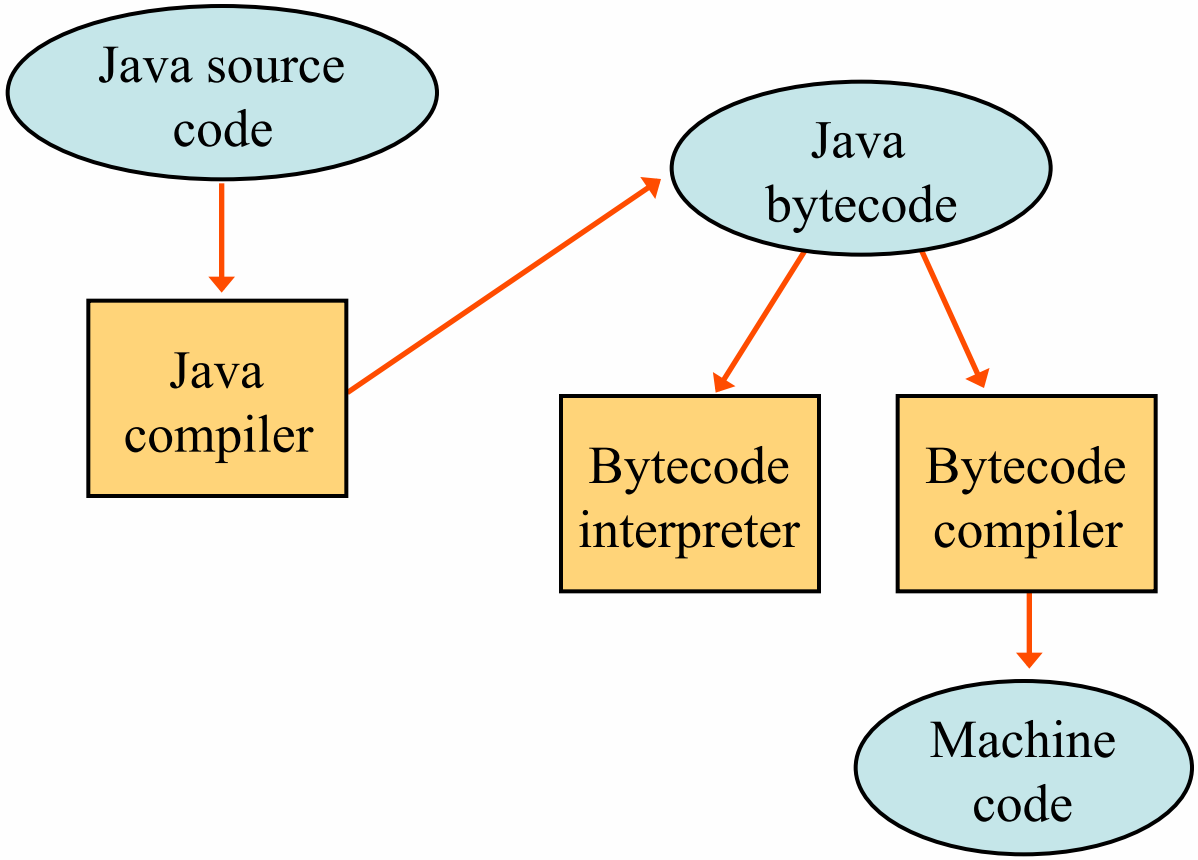
Programming Languages

- Each type of CPU executes only a particular *machine language*
- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates *source code* into a specific target language
- Often, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

Java Translation

- The Java compiler translates Java source code into a special representation called *bytecode*
- Java bytecode is not the machine language for any traditional CPU
- Another software tool, called an *interpreter*, translates bytecode into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine
- Java is considered to be *architecture-neutral*

Java Translation



Development Environments

- There are many programs that support the development of Java software, including:
  - Sun Java Development Kit (JDK)
  - Sun NetBeans
  - IBM Eclipse
  - Borland JBuilder
  - MetroWerks CodeWarrior
  - BlueJ
  - jGRASP
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

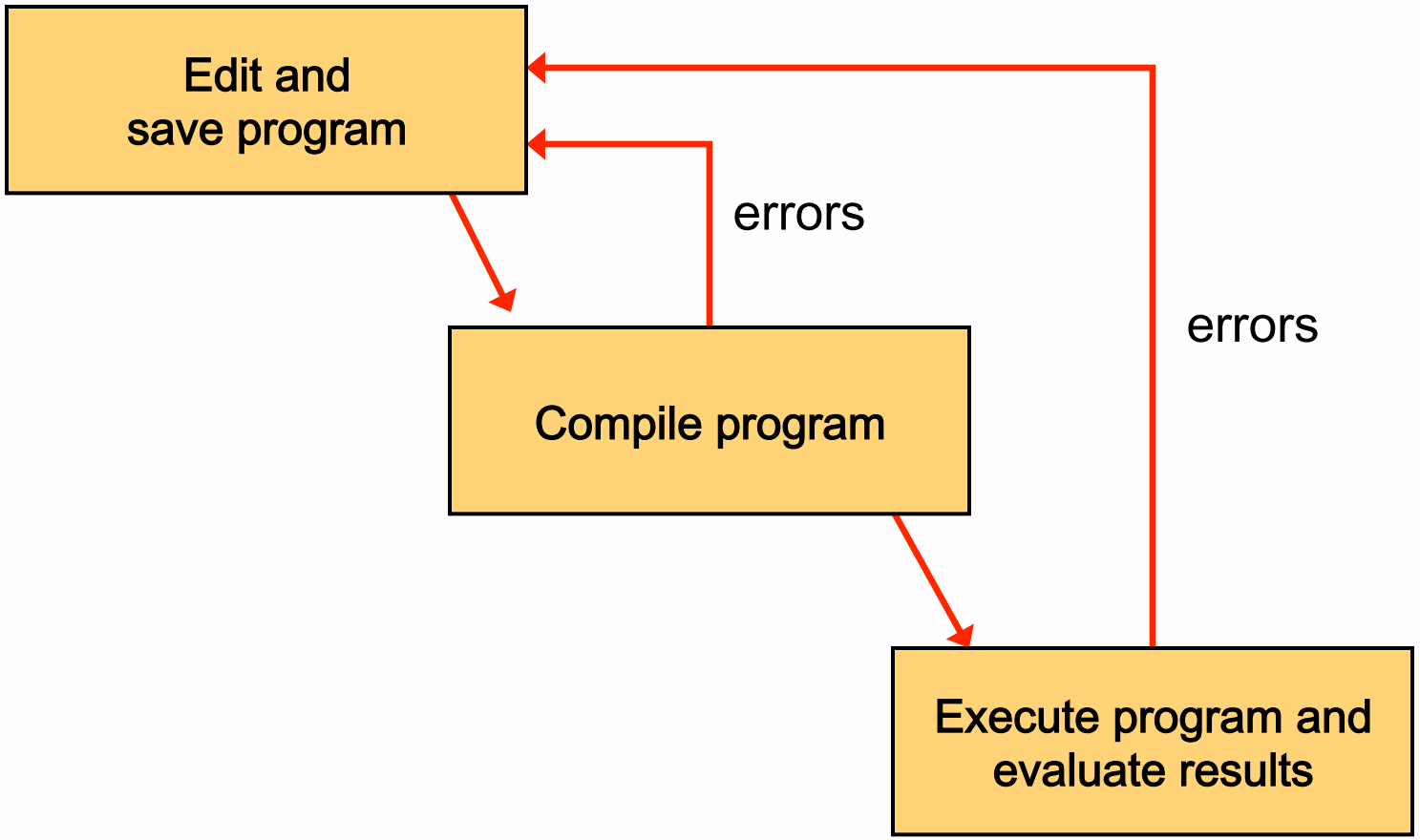
Syntax and Semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do

Errors

- A program can have three types of errors
- The compiler will find syntax errors and other basic problems (*compile-time errors*)
  - If compile-time errors exist, an executable version of the program is not created
- A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
- A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)

Basic Program Development

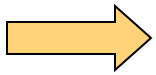


Outline

Organizational issues

The Java Programming Language

Program Development



Object-Oriented Programming

UNIX/LINUX

Problem Solving

- The purpose of writing a program is to solve a problem
- Solving a problem consists of multiple activities:
  - Understand the problem
  - Design a solution
  - Consider alternatives and refine the solution
  - Implement the solution
  - Test the solution
- These activities are not purely linear – they overlap and interact

Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An *object-oriented approach* lends itself to this kind of solution decomposition
- We will dissect our solutions into pieces called objects and classes

Object-Oriented Programming

- Java is an object-oriented programming language
- As the term implies, an object is a fundamental entity in a Java program
- Objects can be used effectively to represent real-world entities
- For instance, an object might represent a particular employee in a company
- Each employee object handles the processing and data management related to that employee

Objects

- An object has:
  - *state* - descriptive characteristics
  - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

Classes

- An object is defined by a *class*
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment of that concept
- Multiple objects can be created from the same class

Objects and Classes

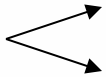
A class
(the concept)

An object
(the realization)

John's Bank Account
Balance: $5,257

Bill's Bank Account
Balance: $1,245,069

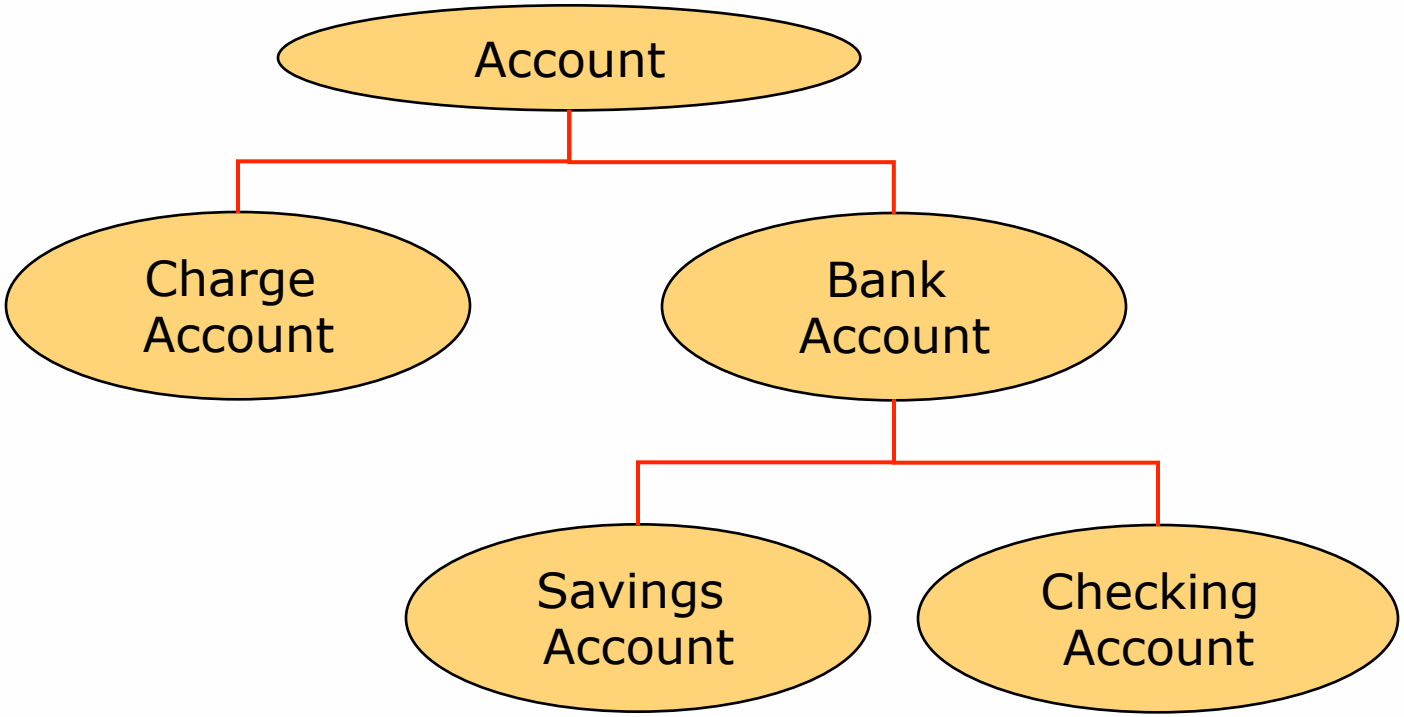
Multiple objects
from the same class



Mary's Bank Account
Balance: $16,833

Inheritance

- **One class can be used to derive another via *inheritance***
- **Classes can be organized into hierarchies**



Summary

- **Lecture 1 focused on:**
  - **organizational issues**
  - **programming and programming languages**
  - **an introduction to Java**
  - **an overview of object-oriented concepts**
- **After the break:**
  - **remarks on the exercises**
  - **introduction to UNIX/LINUX**

Further References

- **K. Arnold, J. Gosling, D. Holmes**
  The Java Programming Language, Addison Wesley
- **K. Arnold, J. Gosling, D. Holmes**
  Die Programmiersprache Java (deutsche Übersetzung)
- **M. Campione, K. Walrath, A. Huml**
  The Java Tutorial, Addison Wesley
  (cf. http://java.sun.com/docs/books/tutorial)
- **D. Flanagan**
  Java in a Nutshell, O'Reilly
- **J. Goll, C. Weiss, F. Müller**
  Java als erste Programmiersprache, Teubner

Further References

- **H.-P. Gumm, M. Sommer**
  Einführung in die Informatik, Oldenbourg
- **C. Horstmann**
  Java Essentials, John Wiley
- **C. Horstmann**
  Big Java, John Wiley
- **W. Küchlin, A. Weber**
  Einführung in die Informatik, Springer
- **G. Krüger**
  Handbuch der Java-Programmierung, Addison Wesley (cf. http://www.javabuch.de)

Further References

- **A. Plüss**
  Java exemplarisch, Oldenbourg
- **W. Savitch**
  Java: An introduction to computer science and programming, Addison Wesley
- **W. Savitch**
  Absolute Java, Addision Wesley

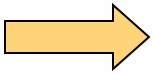
Outline

Organizational issues

The Java Programming Language

Program Development

Object-Oriented Programming



UNIX / LINUX

Today in the Exercise Hour

A very short introduction to

LINUX

– the operating system of the ExWi Pools

LINUX / UNIX Commands (1)

Name (UID) *Password*	• begin session
passwd	• change password
quota –v	• check disk space
mkdir [*options*] *dir-name ...*	• create directory
cd [*dir-name*]	• change current working directory
rmdir [*options*] *dir-name ...*	• delete directories
mv [*options*] *from-dir-name to-dir-name*	• rename directories
ls [**-l**] [*name*]	• Show information on directories and files
ls [**-la**] [*name*]	

LINUX / UNIX Commands (2)

rm [*options*] *file-name*	• delete files
mv [*options*] *from-file-name to-file-name*	• rename / remove files
cp [*options*] *from-file-name to-file-name*	• copy files
chmod *modus file-name ...*	• change file access rights
lpr *file-name*	• print file
nenscript [*options*] *file-name*	• print file in Postscript format (many options)
lpq	• show jobs in printer queue
lprm *job-number*	• remove job from printer queue
man [*options*] *cmd-name*	• help on command
apropos *keyword ?*	• help on keyword
javac [*options*] *prog-name*.**java**	• Java compilation
java *prog-name*	• Java program execution

LINUX / UNIX: Tips and Tricks

- joker symbols in file and directory names:
  `?` for a single character `*` für arbitrary many characters
- ⇥ (Tab) completion of names
- ↑ and ↓ (cursor keys) repeat preceeding commands
- elements of job/process management:
  &, **ps** , <ctrl>+c , **kill –9** *PID*
- input or/and output in files with `<` bzw. `>` or `>>` Examples:

```
java T1 < T1.in
java T1 > T1.out
java T1 < T1N.in >> T1.out
```

Ad **ls** and **chmod**:

```
-rw-rw-r--  1  muster  studi  1284  Nov  2  11:05 Hello.java
```

ExWi-Homepage

For more information on the ExWi-Pools:

http://www.iamexwi.unibe.ch