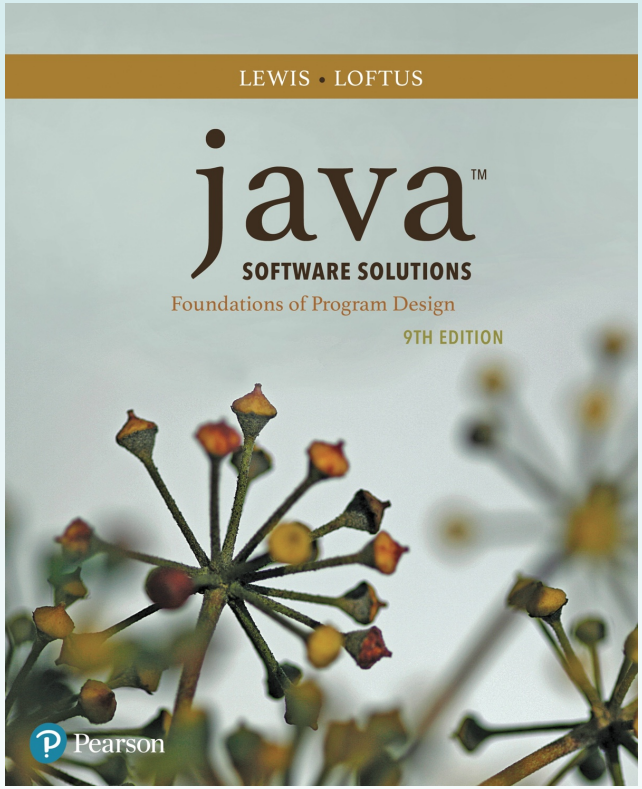
Programmieren 1



Lecture1
Introduction

Thomas Studer

Grundlagen der technischen Informatik

Neu im Raum A6 (statt B7)

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)

Neubrückstrasse 10

Institutssekretariat	D. Schroth
Prüfungssekretariat	B. Choffat

Thomas Studer	Büro 315	tstuder@inf.unibe.ch	031 631 39 84

Luca Althaus
Alessandro Esposito
Jonas Furrer
} p1-beratung@list.inf.unibe.ch

Workstations ExWi A94, A95

Systemadministration, LINUX, UNIX: *Peppo Brambilla* *root@inf.unibe.ch*

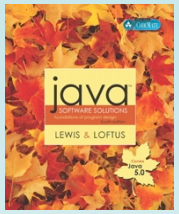
Course Home Page

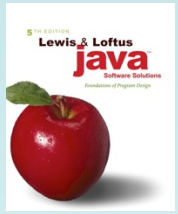
- The course home page contains lots of material and detailed information on the course
- Please visit the page regularly in order to be up-to-date on the P1 lecture
- The website is located at:

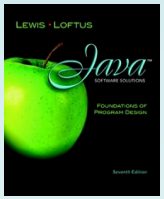
http://www.ilias.unibe.ch

- 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 or 9th edition,
  Pearson, Addison Wesley
- The book can be bought at Studentische Buchgenossenschaft, Hauptgebäude, Uni Bern

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 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	today
Exercises	Friday, 15-16	ExWi A6	today
P1 Pool	Wednesday, 12-14	ExWi Pools	Next week
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: next week
- Location: ExWi pools A94 and A95

Consulting Hour

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

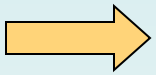
Focus of the Course

- 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

Outline



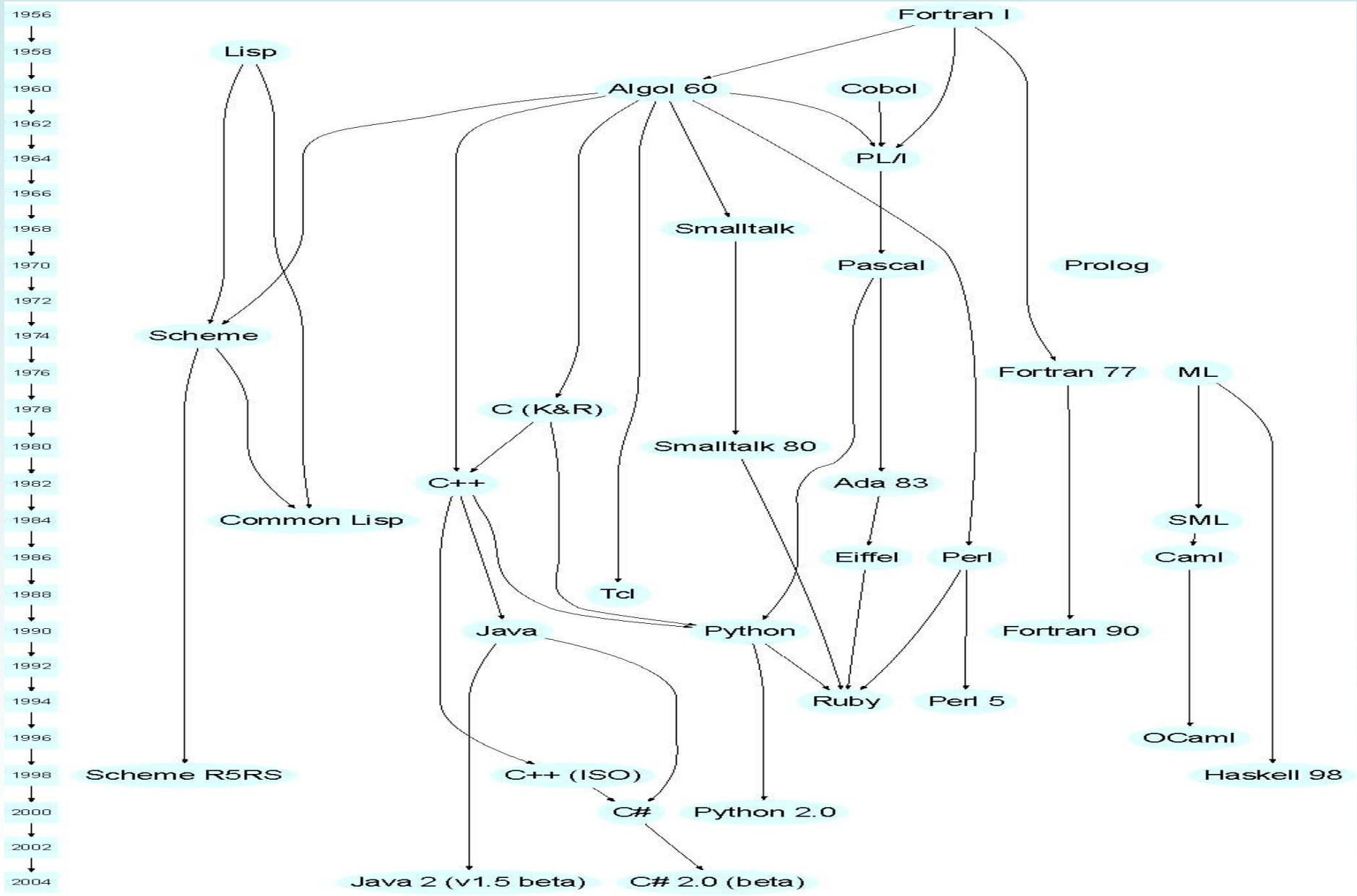
Organizational issues

The Java Programming Language

Program Development

Object-Oriented Programming

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

- The Java programming language was created by Sun Microsystems, Inc.
- It was introduced in 1995 and it's popularity has grown quickly since
- 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*

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`

```
//********************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  Demonstrates the basic structure of a Java application.
//********************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      System.out.println("A quote by Abraham Lincoln:");

      System.out.println("Whatever you are, be a good one.");
   }
}
```

Output

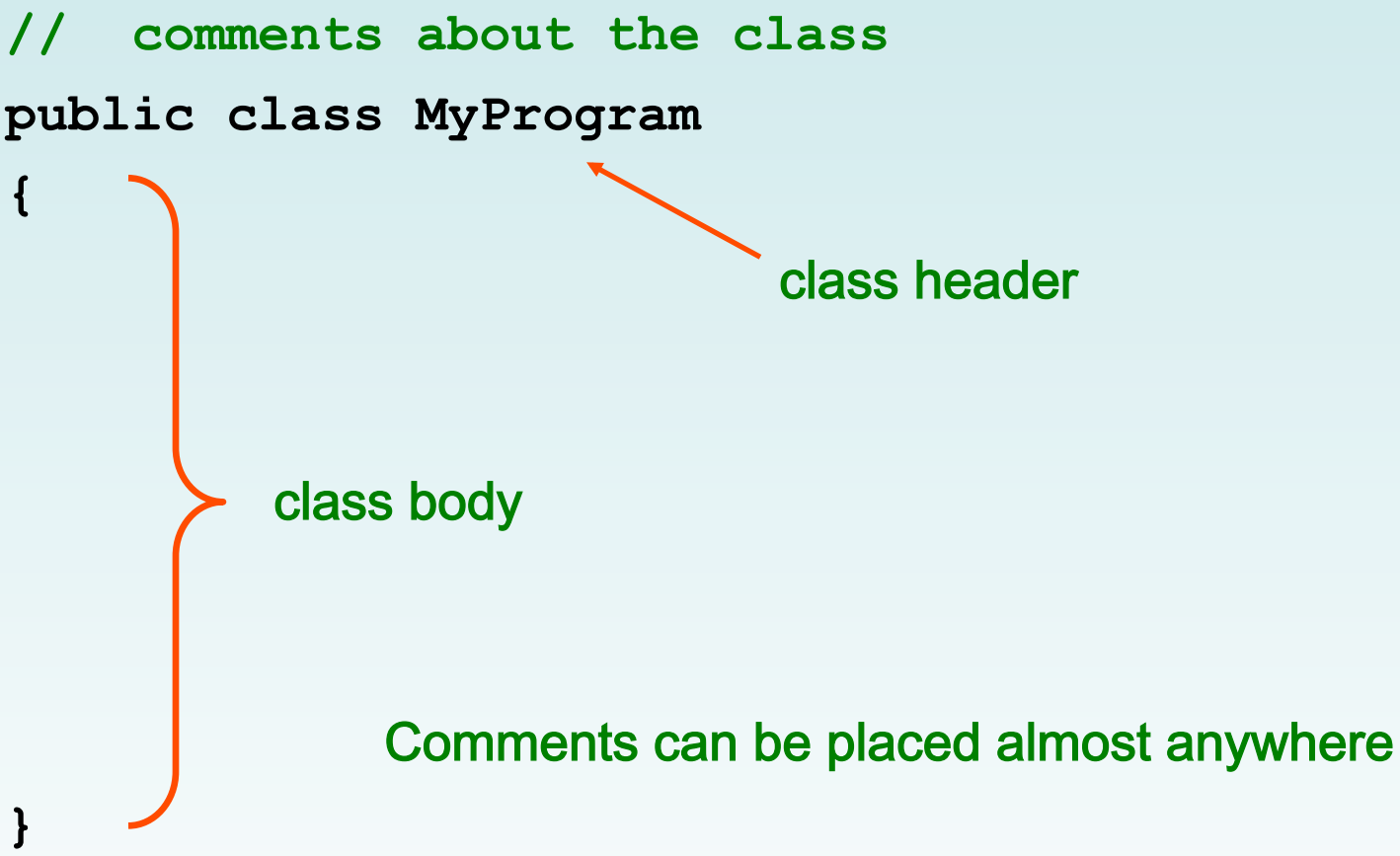
```
A quote by Abraham Lincoln:
Whatever you are, be a good one.
```

```
//********                                                    **********
//  Lincol
//
//  Demons
//*************************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      System.out.println("A quote by Abraham Lincoln:");

      System.out.println("Whatever you are, be a good one.");
   }
}
```

Java Program Structure



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 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" in a program
- A Java 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 the programmer chooses the identifer(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
```

Quick Check

Which of the following are valid Java identifiers?

```
grade
quizGrade
NetworkConnection
frame2
3rdTestScore
MAXIMUM
MIN_CAPACITY
student#
Shelves1&2
```

Quick Check

Which of the following are valid Java identifiers?

`grade`	Valid
`quizGrade`	Valid
`NetworkConnection`	Valid
`frame2`	Valid
`3rdTestScore`	Invalid – cannot begin with a digit
`MAXIMUM`	Valid
`MIN_CAPACITY`	Valid
`student#`	Invalid – cannot contain the '#' character
`Shelves1&2`	Invalid – cannot contain the '&' character

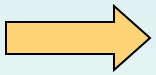
White Space

- 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` and `Lincoln3.java`

Outline

Organizational issues

The Java Programming Language



Program Development

Object-Oriented Programming

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

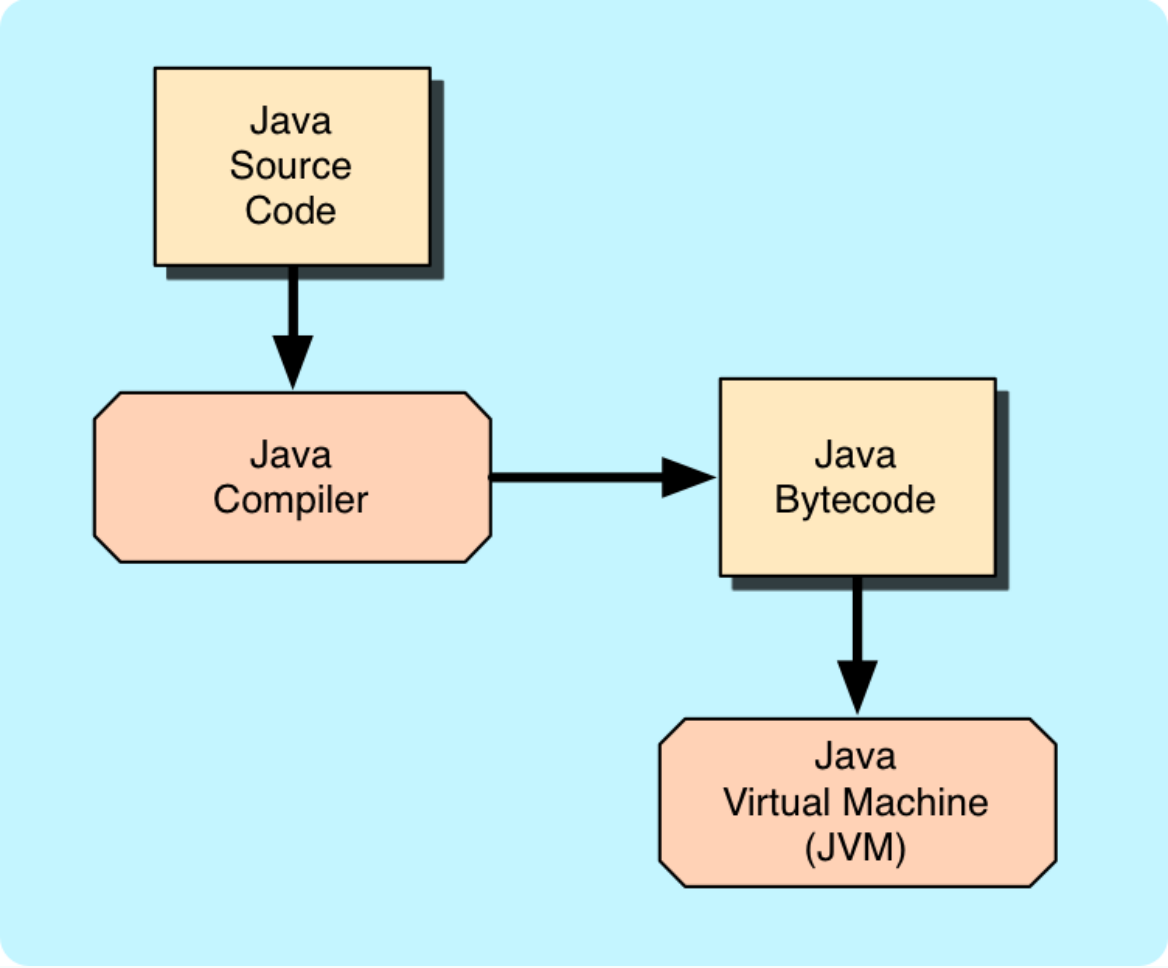
Programming Languages

- 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
- Sometimes, 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
- Bytecode is executed by the *Java Virtual Machine* (JVM)
- Therefore Java bytecode 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:
  - Java Development Kit (JDK)
  - Eclipse
  - NetBeans
  - IntelliJ
  - BlueJ
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

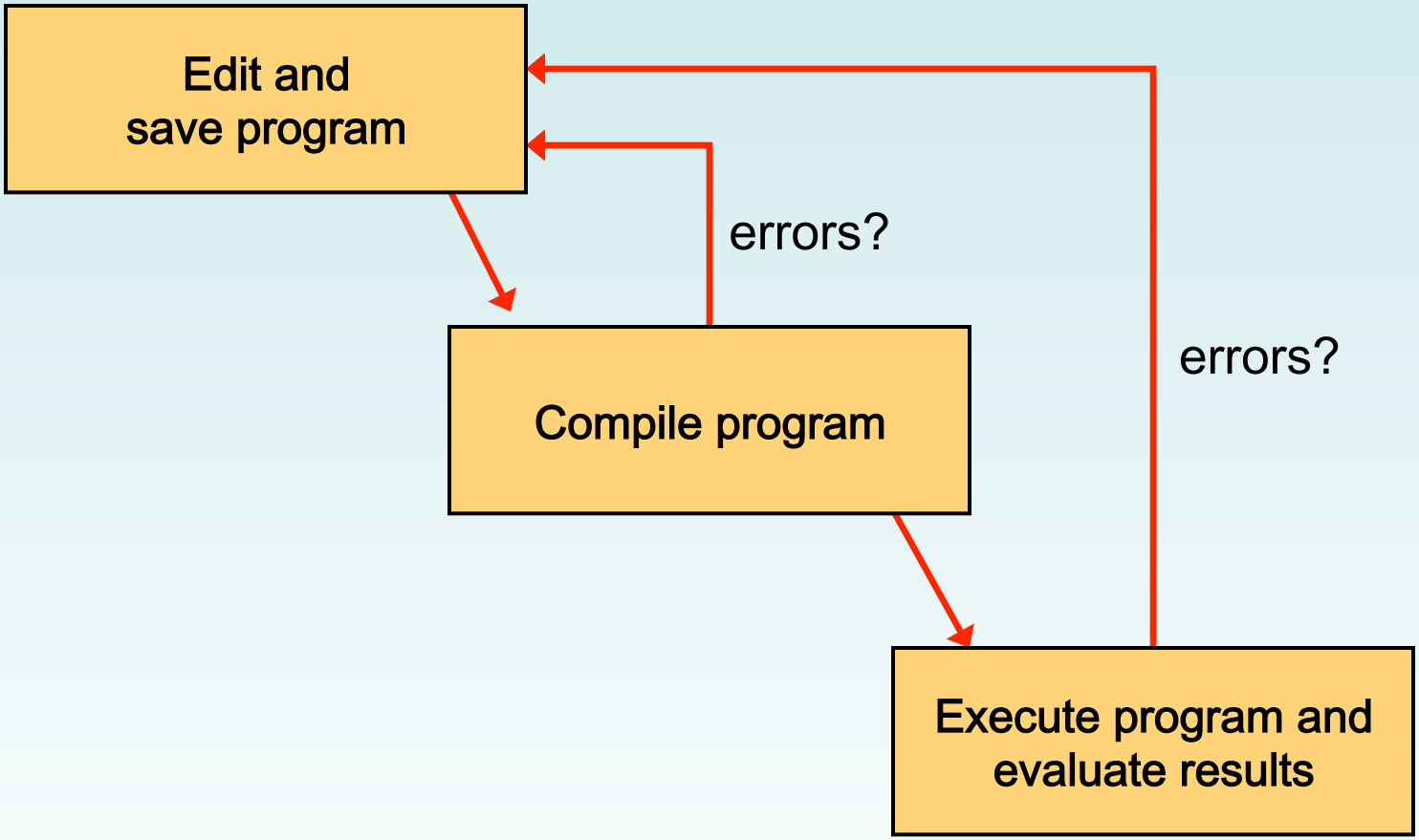
Syntax and Semantics

- 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

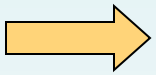


Outline

Organizational issues

The Java Programming Language

Program Development



Object-Oriented Programming

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

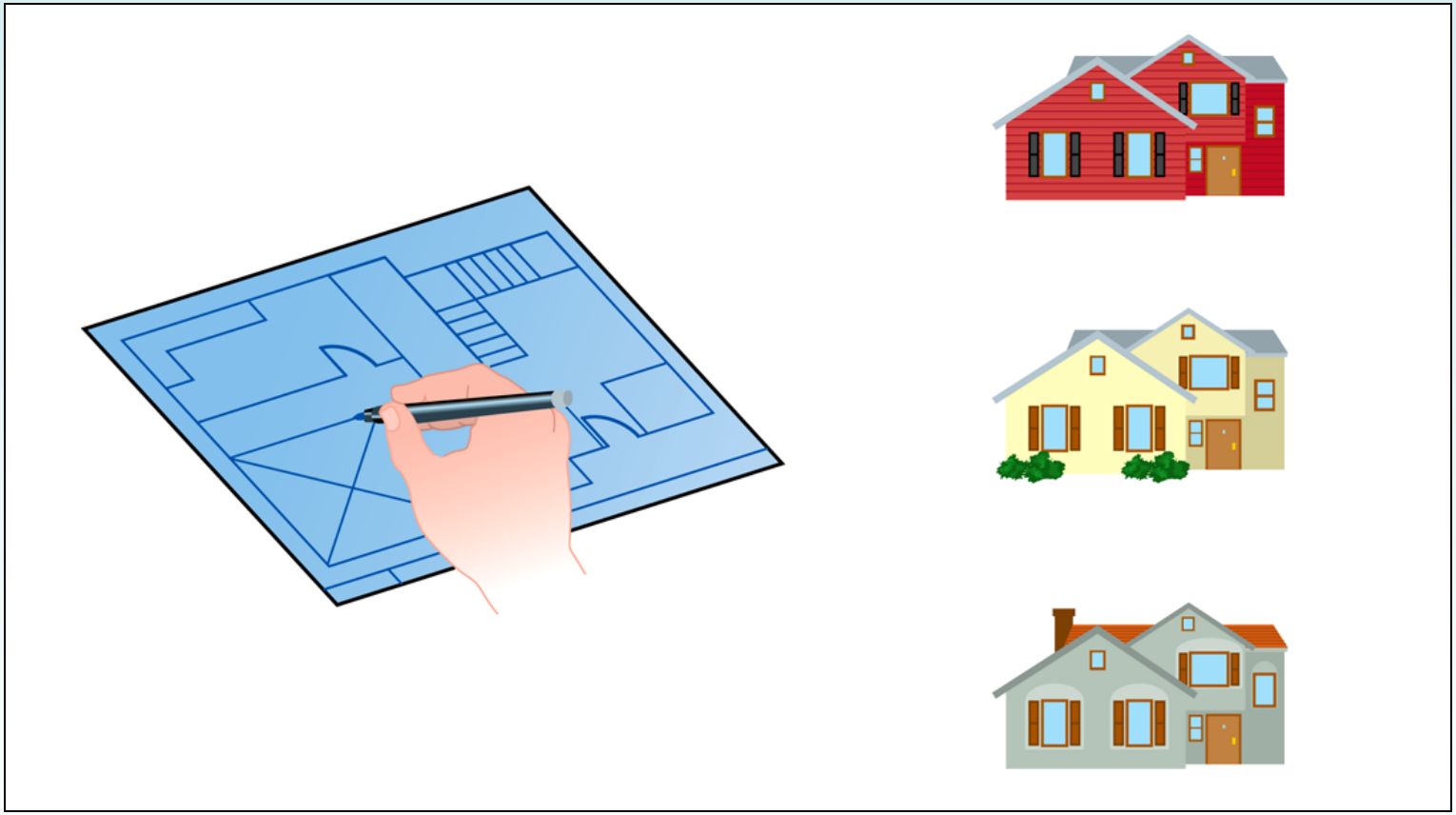
- An object has:
  - *state* - descriptive characteristics
  - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals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

Classes

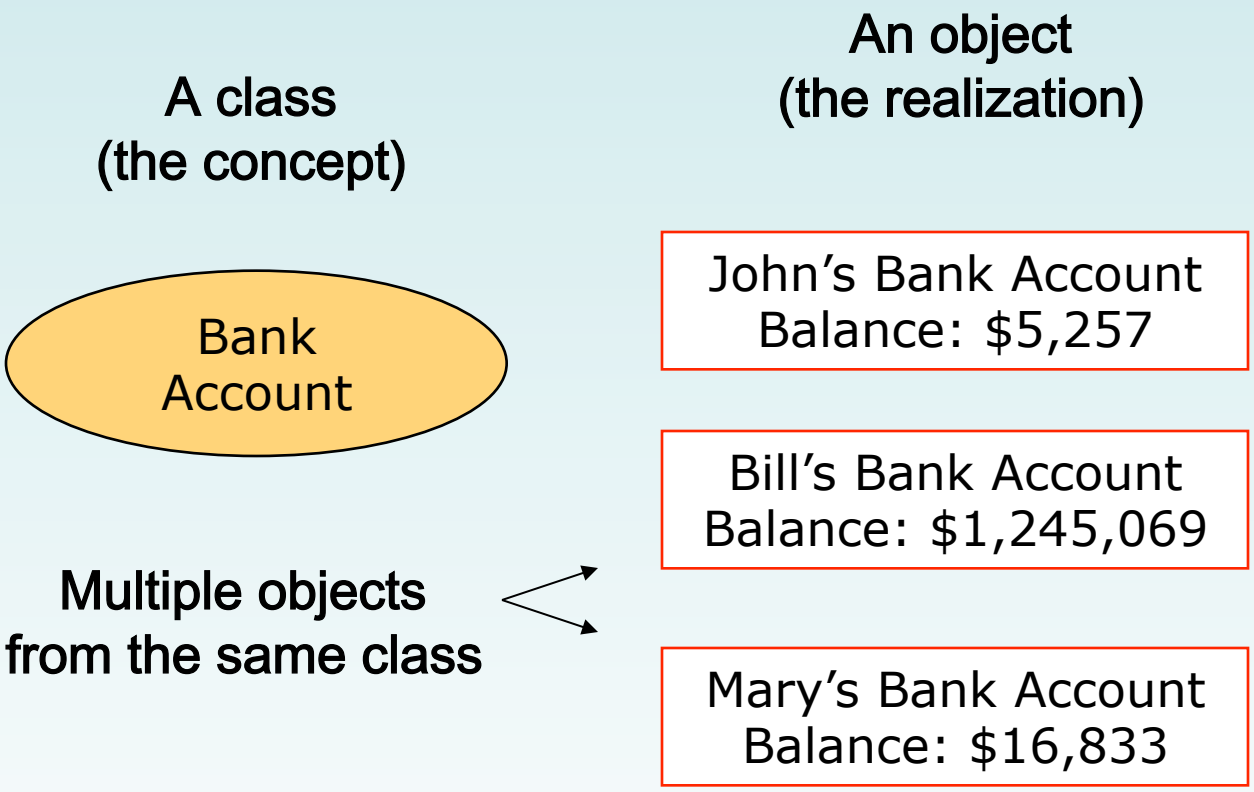
- An object is defined by a *class*
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment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

Class = Blueprint

- One blueprint to create several similar, but different, houses:

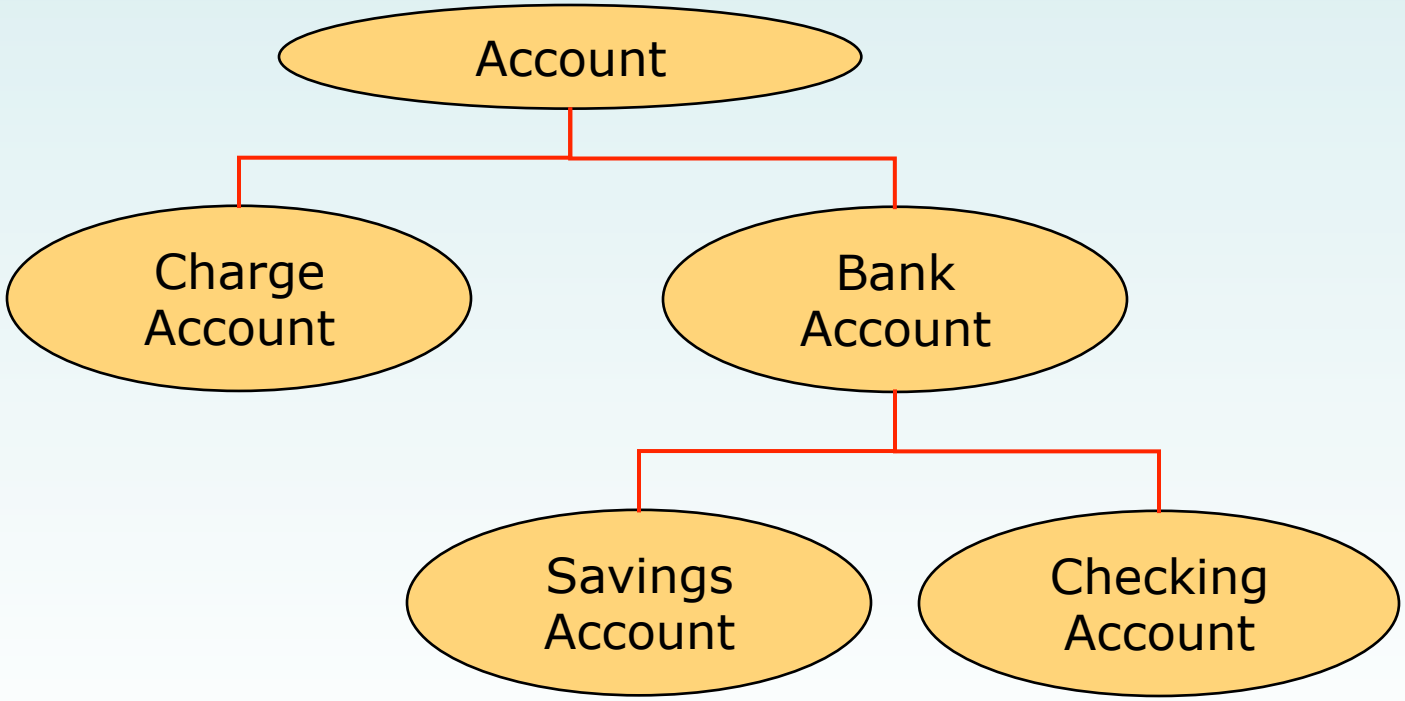


Objects and Classes



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

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