



Introduction to Computers and Java

Chapter 1

Objectives

- Overview of
 - computer hardware and software,
 - programs and compilers
 - the Java programming language
- Example program

Hardware and Software

- Computer systems consist of *hardware* and *software*.
 - Hardware includes the *tangible* parts of computer systems.
 - Software includes *programs* - sets of instructions for the computer to follow.
- Familiarity with hardware basics helps us understand software.

Hardware and Memory

- Most modern computers have similar components including
 - Input devices (keyboard, mouse, etc.)
 - Output devices (display screen, printer, etc.)
 - A processor
 - Two kinds of memory (main memory and auxiliary memory).

The Processor

- Also called the *CPU* (central processing unit) or the *chip* (e.g. Pentium processor)
- The processor **processes** a program's instructions.
- It can process only very simple instructions.
- The power of computing comes from speed and program intricacy.

Memory

- Memory holds
 - programs
 - data for the computer to process
 - the results of intermediate processing.
- Two kinds of memory
 - main memory
 - auxiliary memory
- Measured in bits, bytes, megabytes, gigabytes...

Main memory

- Working memory used to store
 - The current program
 - The data the program is using
 - The results of intermediate calculations
- Also called RAM (Random Access Memory)
- Volatile
 - Not permanent
 - Is overwritten when a program stops running

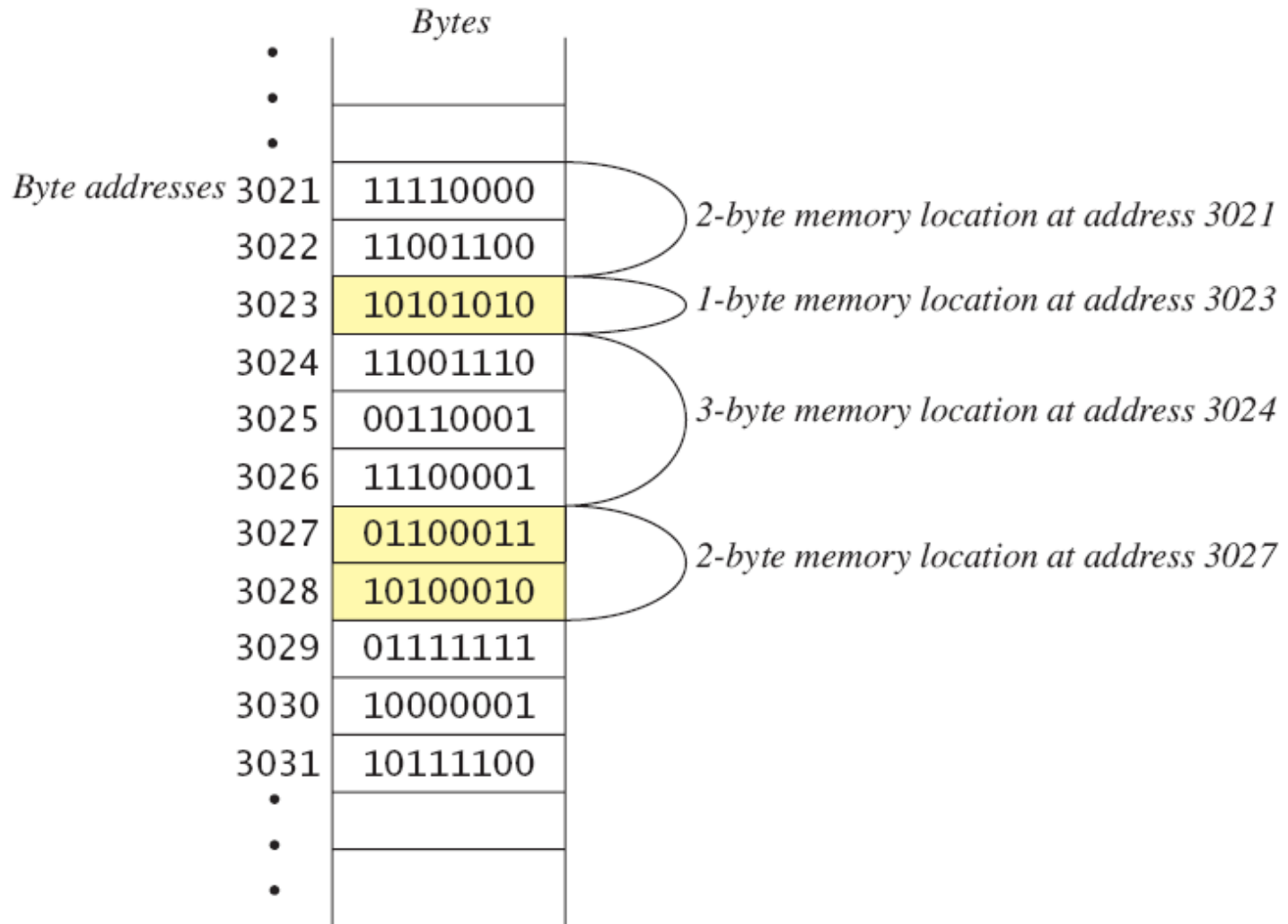
Auxiliary Memory

- Also called *secondary memory*
- Used to store:
 - Program source code
 - Data files
- Disk drives, CDs, DVDs, flash drives, etc.
- More or less permanent (nonvolatile)

Bits, Bytes, and Addresses

- Bits and bytes are quantities of memory
- A *bit* is a single digit with a value of either 0 or 1.
- A *byte* consists of 8 bits.
- Each byte in main memory resides at a numbered location called its *address*.

Main Memory



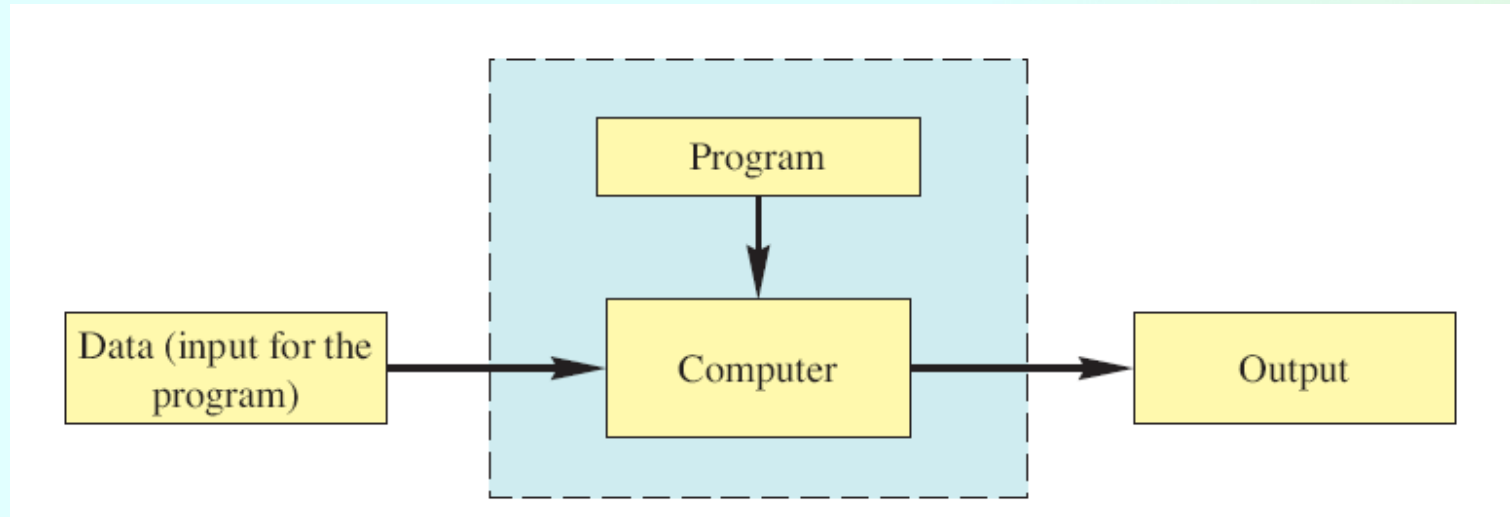
Files

- Large groups of bytes in auxiliary memory are called *files*.
- Files have names and are organized into groups called *directories* or *folders*.
- Java programs are stored in files.
- Programs files are copied from auxiliary memory to main memory in order to be run.

Programs

- A *program* is a set of instructions for a computer to follow.
- We use programs almost daily (email, word processors, video games, etc.).
- Following the instructions is called *running* or *executing* the program.

Running a Program



- Program files are copied into main memory when a program is run
- The OS (Operating System – Windows, Linux, MAC OS) loads and starts a program

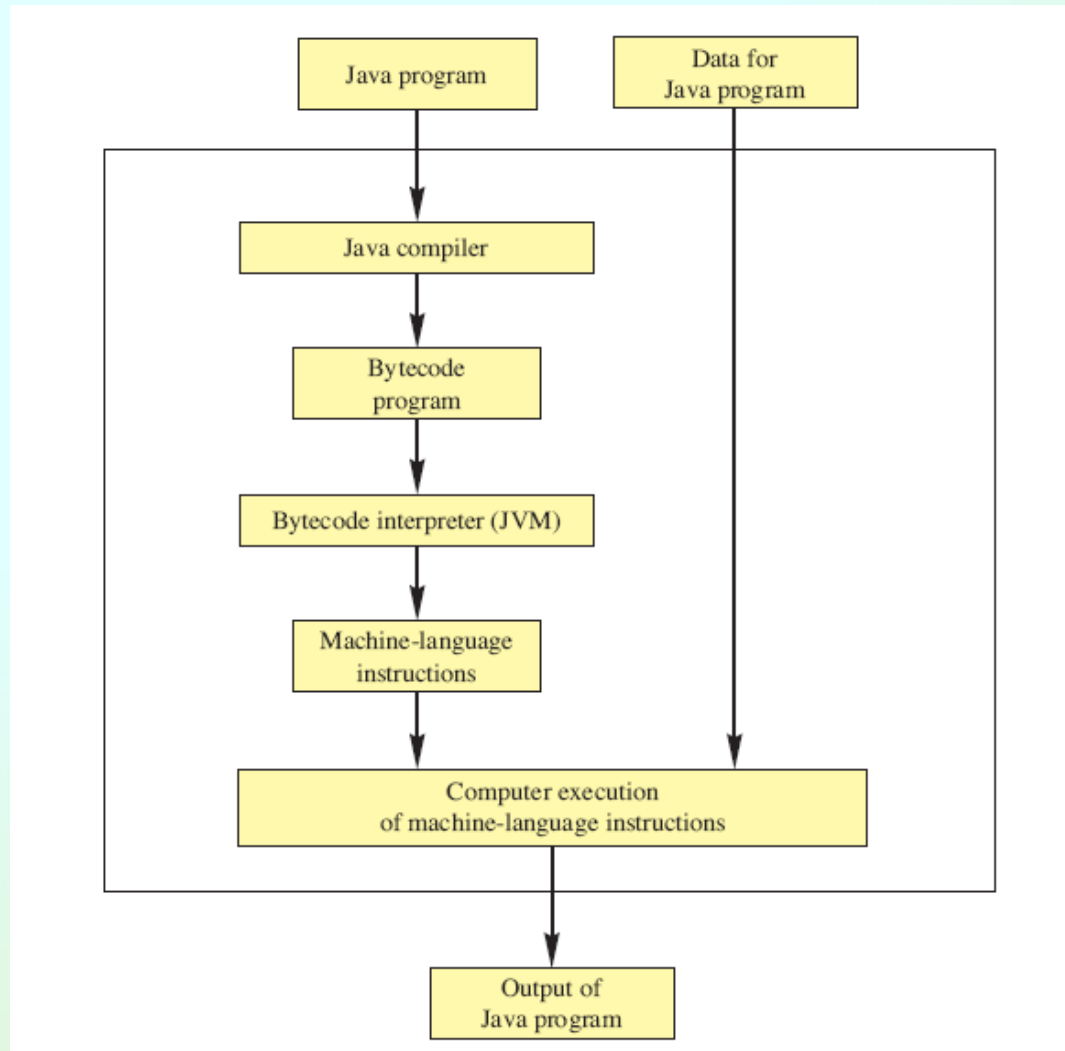
Programming Languages

- *High-level languages* are relatively easy to use
 - Java, C#, C++, Visual Basic, Python, Ruby.
- Unfortunately, computer hardware does not understand high-level languages.
 - Therefore, a high-level language program must be translated into a *low-level language*.

Compilers

- A *compiler* translates a program from a high-level language to a low-level language the computer can run.
- You *compile* a program by running the compiler on the high-level-language version of the program called the *source program*.
- Compilers produce *machine-* or *assembly-language* programs called *object programs*.
- The Java compiler produces byte-code, which is interpreted when a program is executed.

Compiling and Running a Program



A First Java Application

- Create a directory for this course
 - `mkdir java1`
 - `cd java1`
- Create a directory for examples from the book
 - `mkdir savitch`
 - `cd savitch`
- Create a subdirectory for chapter1
 - `mkdir ch01`
 - `cd ch01`

A First Java Application

- Click the “Savitch Src” link on the course webpage:

www.sfs.uni-tuebingen.de/~vhenrich/ws13-14/java/

- Download `ch01/FirstProgam.java` to `java1/savitch/ch01`

A First Java Application

- DrJava is an IDE (integrated development environment): combines text editor with commands to compile and run java programs
- Start drjava and open FirstProgram.java
- Compile and run the program

```
Hello out there.  
I will add two numbers for you.  
Enter two whole numbers on a line:  
12 30  
The sum of those two numbers is  
42
```

Sample
screen
output

Some Terminology

- *programmer*: person who writes programs
- *user*: person who interacts with the program
- *package*: library of classes that have been defined already.
 - `import java.util.Scanner;`
- *argument(s)*: item(s) inside parenthesis
- *variable*: place to store data
- *statement*: instruction – ends with ;
- *syntax*: grammar rules for a programming language

Printing to the Screen

```
System.out.println("Whatever you want to print");
```

- `System.out` is an object for sending output to the screen.
- `println` is a method to print whatever is in parentheses to the screen.
- The object performs an action when you *invoke* or *call* one of its methods

```
objectName.methodName(argumentsTheMethodNeeds);
```

Compiling Programs or Classes

- A Java program consists of one or more classes, which must be compiled before running the program.
- Each class should be in a separate file.
- The name of the file should be the same as the name of the class, with the extension **.java**
 - Class FirstProgram is stored in file FirstProgram.java
- The compiler generates a file with the extension **.class** (FirstProgram.class)

Compiling and Running

- A Java program can involve any number of classes.
- The class to run will contain the words

```
public static void main(String[] args)
```

somewhere in the file

Object-Oriented Programming

- Our world consists of *objects* (people, trees, cars, cities, airline reservations, etc.).
- Objects can perform *actions* which affect themselves and other objects in the world.
- Object-oriented programming (*OOP*) treats a program as a collection of objects that interact by means of actions.
- Java is an object-oriented programming language.

OOP Terminology

- Objects, appropriately, are called *objects*.
- Actions are called *methods*.
- Objects of the same kind have the same *type* and belong to the same *class*.
 - Objects of a class have a common set of methods and the same kinds of data
 - but each object can have it's own data values.

Algorithms

- By designing methods, programmers provide actions for objects to perform.
- An *algorithm* describes a means of performing an action.
- Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.

Algorithms

- An algorithm is a set of instructions for solving a problem.
- An algorithm must be expressed completely and precisely.
- Algorithms usually are expressed in English or in *pseudocode*.

Reusable Components

- Most programs are created by combining components that exist already.
- Reusing components saves time and money.
- Reused components are likely to be better developed, and more reliable.
- New components should be designed to be reusable by other applications.

Testing and Debugging

- Eliminate errors by avoiding them in the first place.
 - Carefully design classes, algorithms and methods.
 - Carefully code everything into Java.
- Test your program with appropriate test cases (some where the answer is known), discover and fix any errors, then retest.

Errors

- An error in a program is called a *bug*.
- Eliminating errors is called *debugging*.
- Three kinds of errors
 - Syntax errors
 - Runtime errors
 - Logic errors

Syntax Errors

- Grammatical mistakes in a program
 - The grammatical rules for writing a program are very strict
- The compiler catches syntax errors and prints an error message.
- Example: using a period where a program expects a comma

Runtime Errors

- Errors that are detected when your program is running, but not during compilation
- When the computer detects an error, it terminates the program and prints an error message.
- Example: attempting to divide by 0

Logic Errors

- Errors that are not detected during compilation or while running, but which cause the program to produce incorrect results
- Examples:
 - using subtraction where addition is required
 - an attempt to calculate a Fahrenheit temperature from a Celsius temperature by multiplying by $9/5$ and adding 23 instead of 32

Software Reuse

- Programs not usually created entirely from scratch
- Most contain components which already exist
- Reusable classes are used
 - Design class objects which are general
 - Java provides many classes
 - Note documentation on following slide

Software Reuse

Java™ Platform Standard Ed. 6

All Classes

Packages

java.applet

java.awt

SAXParseException

SAXParser

SAXParserFactory

SAXResult

SAXSource

SAXTransformerFactory

Scanner

ScatteringByteChannel

ScheduledExecutorService

ScheduledFuture

ScheduledThreadPoolExecutor

Schema

SchemaFactory

SchemaFactoryLoader

SchemaOutputResolver

SchemaViolationException

ScriptContext

ScriptEngine

ScriptEngineFactory

ScriptEngineManager

ScriptException

Scrollable

Scrollbar

ScrollBarUI

Overview Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS

FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

java.util

Class Scanner

java.lang.Object

↳ java.util.Scanner

All Implemented Interfaces:

Iterator<String>

public final class **Scanner**

extends Object

implements Iterator<String>

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from System.in:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

- Follow the link “Java 7 API” on the course website