Introduction to Computers and Java

Chapter 1

Objectives

• Overview of computer hardware and software
• Introduce program design and object-oriented programming
• Overview of the Java programming language
• (Optional) introduce applets and graphics basics

Outline

• Computer Basics
• Designing Programs
• A Sip of Java
Computer Basics: Outline

• Hardware and Memory
• Programs
• Programming Languages and Compilers
• Java Byte-Code

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

Main memory

• Working memory used to store
  ▪ The current program
  ▪ The data the program is using
  ▪ The results of intermediate calculations
• Usually measured in megabytes (e.g. 8 gigabytes of RAM)
  ▪ RAM is short for random access memory
  ▪ A byte is a quantity of memory
Auxiliary Memory

- Also called secondary memory
- Disk drives, CDs, DVDs, flash drives, etc.
- More or less permanent (nonvolatile)
- Usually measured in gigabytes (e.g. 50 gigabyte hard drive)

Bits, Bytes, and Addresses

- A bit is a 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

- Figure 1.1
Storing Data

- Data of all kinds (numbers, letters, strings of characters, audio, video, even programs) are encoded and stored using 1s and 0s.
- When more than a single byte is needed, several adjacent bytes are used.
  - The address of the first byte is the address of the unit of bytes.

Files

- Large groups of bytes in auxiliary memory are called files.
- Files have names.
- Files 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.

0s and 1s

- Machines with only 2 stable states are easy to make, but programming using only 0s and 1s is difficult.
- Fortunately, the conversion of numbers, letters, strings of characters, audio, video, and programs is done automatically.
Programs

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

Input and Output

- Normally, a computer receives two kinds of input:
  - The program
  - The data needed by the program.
- The output is the result(s) produced by following the instructions in the program.

Running a Program

- **Figure 1.2**
- Sometimes the computer and the program are considered to be one unit.
  - Programmers typically find this view to be more convenient.
The Operating System

- The operating system is a supervisory program that oversees the operation of the computer.
- The operating system retrieves and starts program for you.
- Well-known operating systems including: Microsoft Windows, Apple’s Mac OS, Linux, and UNIX.

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.
Compilers

- Most high-level languages need a different compiler for each type of computer and for each operating system.
- Most compilers are very large programs that are expensive to produce.

Java Byte-Code

- The Java compiler does not translate a Java program into assembly language or machine language for a particular computer.
- Instead, it translates a Java program into bytecode.
  - Bytecode is the machine language for a hypothetical computer (or interpreter) called the Java Virtual Machine.

Java Byte-Code

- A bytecode program is easy to translate into machine language for any particular computer.
- A program called an interpreter translates each bytecode instruction, executing the resulting machine-language instructions on the particular computer before translating the next bytecode instruction.
- Most Java programs today are executed using a Just-In-Time or JIT compiler in which bytecode is compiled as needed and stored for later reuse without needing to be re-compiled.
Compiling, Interpreting, Running

- Use the compiler to translate the Java program into byte-code (done using the `javac` command).
- Use the Java virtual machine for your computer to translate each byte-code instruction into machine language and to run the resulting machine-language instructions (done using the `java` command).

Portability

- After compiling a Java program into byte-code, that byte-code can be used on any computer with a byte-code interpreter and without a need to recompile.
- Byte-code can be sent over the Internet and used anywhere in the world.
- This makes Java suitable for Internet applications.

Class Loader

- A Java program typically consists of several pieces called classes.
- Each class may have a separate author and each is compiled (translated into byte-code) separately.
- A class loader (called a linker in other programming languages) automatically connects the classes together.
Figure 1.3

Compiling and Running a Program

A Sip of Java: Outline

• History of the Java Language
• Applications and Applets
• A First Java Application Program
• Writing, Compiling, and Running a Java Program

History of Java

• In 1991, James Gosling and Sun Microsystems began designing a language for home appliances (toasters, TVs, etc.).
  • Challenging, because home appliances are controlled by many different chips (processors)
  • Programs were translated first into an intermediate language common to all appliance processors.
History of Java

- Then the intermediate language was translated into the machine language for a particular appliance’s processor.
- Appliance manufacturers weren’t impressed.
- In 1994, Gosling realized that his language would be ideal for a Web browser that could run programs over the Internet.
- Sun produced the browser known today as HotJava.

Applications and Applets

- Two kinds of Java programs: applications and applets
  - Applications
    - Regular programs
    - Meant to be run on your computer
  - Applets
    - Little applications
    - Meant to be sent to another location on the internet and run there

A First Java Application

- View sample program Listing 1.1
  - class FirstProgram

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

- The person who writes a program is called the **programmer**.
- The person who interacts with the program is called the **user**.
- A **package** is a library of classes that have been defined already.
  - `import java.util.Scanner;`

Some Terminology

- The item(s) inside parentheses are called **arguments** and provide the information needed by methods.
- A **variable** is something that can store data.
- An instruction to the computer is called a **statement**; it ends with a semicolon.
- The grammar rules for a programming language are called the **syntax** of the 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.
Printing to the Screen

- The object performs an action when you *invoke* or *call* one of its methods

  `objectName.methodName(argumentsTheMethodNeeds);`

Compiling a Java Program or Class

- A Java program consists of one or more classes, which must be compiled before running the program.
- You need not compile classes that accompany Java (e.g. `System` and `Scanner`).
- Each class should be in a separate file.
- The name of the file should be the same as the name of the class.

Compiling and Running

- Use an IDE (integrated development environment) which combines a text editor with commands for compiling and running Java programs.
- When a Java program is compiled, the byte-code version of the program has the same name, but the ending is changed from `.java` to `.class`. 
Compiling and Running

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

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

  somewhere in the file

Programming Basics: Outline

- Object-Oriented Programming
- Algorithms
- Testing and Debugging
- Software Reuse

Programming

- Programming is a creative process.
- Programming can be learned by discovering the techniques used by experienced programmers.
- These techniques are applicable to almost every programming language, including Java.
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.

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 within a class have a common set of methods and the same kinds of data
  • but each object can have its own data values.

OOP Design Principles

• OOP adheres to three primary design principles:
  • Encapsulation
  • Polymorphism
  • Inheritance
Introduction to Encapsulation

- The data and methods associated with any particular class are encapsulated ("put together in a capsule"), but only part of the contents is made accessible.
  - Encapsulation provides a means of using the class, but it omits the details of how the class works.
  - Encapsulation often is called information hiding.

Accessibility Example

- An automobile consists of several parts and pieces and is capable of doing many useful things.
  - Awareness of the accelerator pedal, the brake pedal, and the steering wheel is important to the driver.
  - Awareness of the fuel injectors, the automatic braking control system, and the power steering pump is not important to the driver.

Introduction to Polymorphism

- From the Greek meaning "many forms"
- The same program instruction adapts to mean different things in different contexts.
  - A method name, used as an instruction, produces results that depend on the class of the object that used the method.
  - Everyday analogy: "take time to recreate" causes different people to do different activities
- More about polymorphism in Chapter 8
Introduction to Inheritance

• Figure 1.4

Inheritance in Java

• Used to organize classes
• “Inherited” characteristics do not need to be repeated.
• New characteristics are added.
• More about inheritance in chapter 8
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.

Example: Total Cost of All Items

• Write the number 0 on the whiteboard.
• For each item on the list
  • Add the cost of the item to the number on the whiteboard
  • Replace the number on the whiteboard with the result of this addition.
• Announce that the answer is the number written on the whiteboard.
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
- Example: 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

Package names
Class names
Description of class Scanner

Summary
- You have completed an overview of computer hardware and software.
- You have been introduced to program design and object-oriented programming.
- You have completed an overview of the Java programming language.