Chapter 1

Getting Started

1.1 What Do Computers Do?

- A ***computer system*** is an integrated collection of hardware and software components.
- ***Hardware*** refers to the electronics inside a computer.
- ***Software*** consists of programs that tell the hardware what to do.
- [Q: Which can be seen and touched?]

Types of Computer Systems

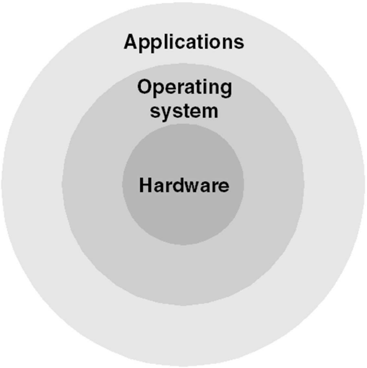
- Some computer systems are embedded within other objects. These are called ***embedded systems.*** [E.g., …]
- Other computer systems are intended for direct use by humans (***users***). [E.g., …]
  - Some systems support multiple simultaneous users, while others are limited to one user at a time. [E.g., …]
- Systems in the latter category are usually called ***personal computers.***

Hardware

- ***Processors***
  - ***Central processing unit,*** or ***CPU*** [E.g., …]
  - Specialized processors, such as a graphics processor
- ***Memory***
  - ***Main memory,*** or ***RAM*** (***random-access memory***) [Power off -> content will …; a good name?]
  - ***ROM*** (***read-only memory***)
  - [Secondary:] Hard disks, solid state drives, and other storage media
- ***Peripheral devices***
  - Provide an ***interface*** to the world outside the system
  - Include keyboards, mice, monitors, printers

Software

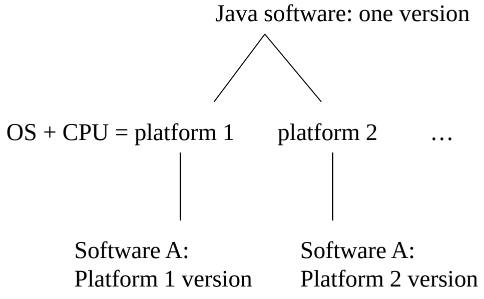
- Software consists of programs that instruct the hardware how to perform operations. [What's the hardware?]
- A ***program*** is a step-by-step set of instructions.
- Categories of software:
  - ***Operating systems.*** A collection of programs that interact directly with the computer's hardware. [E.g., …]
  - ***Applications.*** Programs designed to perform useful tasks for humans. [E.g., …]
- An operating system serves as a bridge between hardware and applications.



Platforms

- The combination of an operating system and a particular type of CPU is often called a ***platform.***
- Software usually works only on a single platform.
- Java programs, however, will run on multiple platforms without change. [why so good?]
- Most of the time, a computer system has only one operating system but many applications.
- Applications are usually designed for one particular version of an operating system.

Platforms: Web view



Java software: one version

OS + CPU = platform 1

platform 2

…

Software A:
Platform 1 version

Software A:
Platform 2 version

1.2 Ways of Interacting with Computers

- Most applications need to communicate, or “interface,” with the user by displaying information for the user to see and accepting commands from the user.
- Primary types of user interfaces:
  - Graphical user interfaces (GUI)
  - Text-based interfaces

Graphical User Interfaces

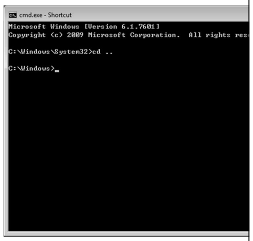
- Most applications now rely on a ***graphical user interface,*** or ***GUI*** (pronounced “gooey”) built out of visual components.
- When a GUI program is run, it displays a window on the screen.
- The window is composed of thousands of tiny ***pixels*** (picture elements), each with its own color.

Text-Based Interfaces

- Before the advent of graphical user interfaces, programs used a ***text-based interface,*** in which all input and output consisted of characters.
- In a text-based interface, no graphics are displayed, and user commands are entered from the keyboard.
- Text-based programs are normally run from a ***command line.***

Command-Line Prompts

- Typical Unix command-line prompt:
  `%`
- Typical command-line prompt:
  `C:>`
- The prompt is often configured to display the “current directory”:
  `C:\WINDOWS>`



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights res

C:\Windows\System32>cd ..

C:\Windows>_
```

1.3 What Is Programming?

- ***Programming*** means writing down a series of instructions that tell a computer what to do.
- Properties of these instructions:
  - Computation proceeds in discrete steps.
  - Each step is precisely defined.
  - The order in which steps are performed may be important.

Algorithms

- A set of instructions with these properties is said to be an ***algorithm.***
- The steps in an algorithm are not always short and simple.
  - Some steps may involve a series of smaller steps.
  - Some steps may involve making decisions.
  - Some steps may repeat.
- Algorithms are common in the real world.

A Real-World Algorithm: making a phone call

1. Key in the phone numbers and press the <call> button
2. Wait until it is connected or hear a busy signal
3. If it is a busy signal, wait for a moment and go to 1
4. If it is connected, the phone call is successful and you can start talking

Ways to Express Algorithms

- ***Natural languages.*** Allows anyone who understands that language to read the algorithm, but lacks precision.
- ***Programming languages.*** Precise, yet simple enough for computers to understand.
- ***Pseudocode.*** A mixture of natural language and a programming language. More precise than natural language but less precise than a programming language. Often easier to read (and to write) than a programming language.

1.5 Programming Languages

- Creating programs requires that algorithms be expressed in a highly precise language that’s specifically designed for computers.
- Every computer comes with such a language, known as ***machine language.***
- Each CPU has its own machine language.
- Machine language is extremely primitive, making it difficult to write even simple programs.
- Most programmers use ***high-level languages*** that aren’t tied to a particular computer.

Writing and Executing a Program

- Writing a program in a high-level language requires creating a file containing ***source code***.
- Source code is not ***executable***—there is no direct way for a computer to follow the commands that it contains.
- ***Executing*** (or ***running***) the program requires special software.
- Approaches to executing a program:
  - Compilation
  - Interpretation

Compilation

- The program’s source code is given to a program called a ***compiler.***
- The compiler checks that the source code is valid (obeys the rules of the language) and translates it to machine instructions for a particular CPU.
- The compiled program is stored in a file, and it can be run as many times as desired.

Interpretation

- The program’s source code is given to a program known as an ***interpreter.***
- The interpreter executes the program without first translating it to machine instructions.
- The interpreter itself is normally a compiled program, so it can execute machine instructions corresponding to the source code.

Java’s Approach

- Java employs a combination of compilation and interpretation.
- The Java compiler translates the original program into ***bytecode instructions*** for a computer called the ***Java Virtual Machine.***
- The resulting ***bytecode program*** is then executed by an interpreter.
- One advantage of Java’s approach is that programs don’t need a particular CPU or operating system. [What’s needed?]

1.6 Why Java?

- Simple
- Object-oriented
- Distributed
- Robust
- Architecture-neutral
- Portable
- Interpreted
- Multithreaded

1.7 The Programming Process

1. Write a specification for the program.
2. Design the program.
3. Choose algorithms and decide how data will be stored.
4. Write the program.
5. Compile the program.
6. Execute the program.
7. Debug the program.

Program Maintenance

- Most of the time, there’s an additional step in the programming process: ***maintenance.***
- Reasons for maintenance:
  - Fix bugs
  - Add enhancements
  - Adapt to changes in the program’s specification
- Maintenance is often the costliest step in the programming process.