1

Introduction to Computers and Java
OBJECTIVES

In this chapter you will learn:

- Basic computer hardware and software concepts.
- Basic object technology concepts, such as classes, objects, attributes, behaviors, encapsulation, inheritance and polymorphism.
- The different types of programming languages.
- Which programming languages are most widely used.
- A typical Java development environment.
Introduction

• Java Standard Edition  Java SE
• Java Enterprise Edition  Java EE
  – geared toward large-scale distributed applications and web applications
• Java Micro Edition  Java ME
  – geared toward applications for small, memory constrained devices
• Java Development Kit  JDK
  – development and run time environment [ JDK and JRE ]
• Object-Oriented Programming
• Java is language of choice for networked applications
Computers: Hardware and Software

• Computer
  – Performs computations and makes logical decisions
  – Millions or billions of times faster than human beings

• Computer programs
  – Sets of instructions for which computer processes data

• Hardware
  – Physical devices of computer system

• Software
  – Programs that run on computers
Computer Organization

- Six logical units of computer system
  - Input unit
    - Mouse, keyboard
  - Output unit
    - Printer, monitor, audio speakers
  - Memory unit
    - Retains input and processed information
  - Arithmetic and logic unit (ALU)
    - Performs calculations
  - Central processing unit (CPU)
    - Supervises operation of other devices
  - Secondary storage unit
    - Hard drives, floppy drives
  - Assembly Language
Early Operating Systems

- **Batch processing**
  - One job (task) at a time
  - Operating systems
    - Developed to make computers more convenient to use
    - Made transitions between jobs easier
    - More throughput

- **Multiprogramming** [multitasking]
  - “Simultaneous” jobs
  - Timesharing operating systems
    - mainframe/mini-computers and terminals
Types of Computing

• **Personal** computing
  – Computers for personal use

• **Distributed** computing
  – Networked computers
  – Computing performed among several computers

• **Client/server** computing
  – Servers offer common store of programs and data
  – Clients access programs and data from server

• **Utility / Grid / Cloud** computing
Machine Languages, Assembly Languages and High-Level Languages

- **Machine language**
  - “Natural language” of computer component
  - Machine dependent

- **Assembly language**
  - English-like abbreviations represent computer operations
  - Assembler converts to machine language

- **High-level language**
  - Allows for writing more “English-like” instructions

- **Intermediate Language** [bytecode / MSIL]

- **Translator**
  - **Compiler** – translate from a high-level language to another language
  - **Interpreter** – translate and execute statements [JITer]
History of C and C++

• **C++** evolved from C, which evolved from BCPL and B

• **C** 1972
  – Developed at Bell Labs
  – Popularized as the language of the UNIX operating system

• **C++ 1980’s**
  – Developed at Bell Labs
  – Developed by Bjarne Stroustrup
  – Provides object-oriented programming capabilities
  – Hybrid language

• **Objects**
  – Reusable software components that model real-world items
  – Attributes and behaviors  [ fields / methods in Java ]
History of Java

• Java 1995
  – Original intent was for intelligent consumer-electronic devices
  – Then used for creating web pages with dynamic content
  – Now also used to:
    • Develop large-scale enterprise applications
    • Enhance web server functionality
    • Provide applications for consumer devices (cell phones, etc.)
Java Class Libraries

• Java programs consist of classes
  – Include methods that perform tasks
    • Return information after task completion

• Java provides class libraries
  – Known as Java APIs (Application Programming Interfaces)
    – Java SE 6 API

• To use Java effectively, you must:
  – Learn the Java programming language
  – Become familiar with the extensive set of class libraries [API]
Software Engineering Observation

- Use a building-block approach to create programs.

- Avoid reinventing the wheel—use existing pieces wherever possible.

- Called *software reuse*, this practice is central to object-oriented programming.
Software Engineering Observation

When programming in Java, you will typically use classes and methods:

- from class libraries
- you create yourself
- that others create and make available to you.
Performance Tip

Using **Java API classes** and methods instead of writing your own versions can improve program performance, because they are carefully written to perform efficiently.

This technique also shortens program development time.
FORTRAN, COBOL, Pascal and Ada

- **Fortran** mid 50’s
  - FORmula TRANslator
  - Developed by IBM for scientific and engineering applications

- **COBOL** late 50’s
  - COnmon Business Oriented Language
  - Used for commercial applications requiring precise/efficient manipulation of large amounts of data

- **Pascal** 1971
  - Developed by Prof. Niklaus Wirth [who also created Modula]
  - Designed to teach structured programming

- **Ada** 70’s and 80’s
  - Developed under the sponsorship of the U.S. Department of Defense
  - Needed a single language to fill most of its needs
  - Provided multiprocessing so programmers could specify parallel tasks
BASIC, Visual Basic, Visual C++, C# and .NET

• BASIC  mid 60’s
  – Beginner’s All-Purpose Symbolic Instruction Code
  – Developed at Dartmouth College to familiarize novices with programming techniques

• Visual Basic introduced by Microsoft in the early 1990’s

• .NET platform  [ similar concept to Java platform ]  2000
  – Part of Microsoft’s corporate-wide strategy to incorporate the Internet and web into computer applications

• .NET programming languages
  – Visual Basic  (based on BASIC)
  – Visual C++  (based on C++)
  – Visual C#  (based on C++ and Java, new in VS 2000)
  – Visual F#  (new in VS 2010)
Typical Java Development Environment

- Java programs go through five phases
  - **Edit**
    - Programmer writes program using an editor; stores program on disk with the `.java` file name extension
  - **Compile**
    - Use javac (the Java compiler) to create bytecodes from source code program; bytecodes stored in `.class` files
  - **Load**
    - Class loader reads bytecodes from `.class` files into memory
  - **Verify**
    - Bytecode verifier examines bytecodes to ensure that they are valid and do not violate security restrictions
  - **Execute**
    - Java Virtual Machine (JVM) uses a combination of interpretation and just-in-time compilation to translate bytecodes into machine language
Fig. 1.1 | Typical Java development environment.
Fig. 1.1 | Typical Java development environment.

Bytecode verifier confirms that all bytecodes are valid and do not violate Java’s security restrictions.

To execute the program, the JVM reads bytecodes and translates them into a language that the computer can understand. As the program executes, it may store data values in primary memory.
Notes about Java and Java How to Program Textbook

Stresses:

- clarity

- **Portability** [ Holy Grail ]
  - An elusive goal due to differences between compilers, JVMs and computers
  - Always test programs on all systems on which the programs should run
Good Programming Practice

Write your Java programs in a simple and straightforward manner.

This is sometimes referred to as KIS ("keep it simple").
Good Programming Practice

Read the documentation for the version of Java you are using.

Refer to it frequently, especially to familiarize yourself with new features and deprecated classes/methods.
Introduction to Object Technology and the UML

- **Object orientation**

- **Unified Modeling Language (UML)**
  - Graphical language that uses common notation
  - Allows developers to represent object-oriented designs
Introduction to Object Technology and the UML

• Objects
  – Reusable software components that model real-world items
  – Look all around you
    • People, animals, plants, cars, etc.
  – Attributes
    • Size, shape, color, weight, etc.
  – Behaviors
    • Babies cry, crawl, sleep, etc.
  – Messages
    • Causes the invocation of a behavior
Introduction to Object Technology and the UML

- **Object-oriented design (OOD)**
  - Models software in terms similar to those used to describe real-world objects
  - Class relationships
  - Inheritance relationships
  - Models communication among objects
  - **Encapsulates attributes and operations** (behaviors)
    - Information hiding
    - Communication through well-defined interfaces

- **Object-oriented language**
  - Programming in object-oriented languages is called **object-oriented programming (OOP)**
  - Java
Introduction to Object Technology and the UML

- **Classes are to objects as blueprints are to houses**

- **Associations**
  - Relationships between classes
    - is-a related by inheritance
    - has-a object composition
    - uses-a classes with *static* methods

- **Packaging software in classes facilitates reuse**
Introduction to Object Technology and the UML

• History of the UML
  – Need developed for process with which to approach OOA/D
  – Brainchild of Booch, Rumbaugh and Jacobson
  – Object Management Group (OMG) supervised
Introduction to Object Technology and the UML

• UML
  – Graphical representation scheme
  – Enables developers to model object-oriented systems
  – Flexible and extensible
Software Technologies

• **Open source software**
  – Individuals and companies contribute to developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge
  – Code typically scrutinized by much larger audiences, so bugs get removed faster
  – **Java is open source**
  – Some open source organizations
    • Eclipse Foundation, Mozilla Foundation, Apache Software Foundation, SourceForge
Software Technologies (cont.)

• Linux
  – Open source operating system
  – One of the greatest successes of the open source movement

• MySQL [ now part of Oracle ]
  – an open source database management system

• PHP
  – most popular open source server-side Internet “scripting” language for developing Internet-based applications

• LAMP—Linux, Apache, MySQL and PHP (or Perl or Python)
  – An acronym for the set of open source technologies that many developers used to build web applications
Software Technologies (cont.)

• Ruby on Rails
  – Combines the scripting language Ruby with the Rails web application framework developed by 37Signals
  – Many Ruby on Rails developers report significant productivity gains over using other languages when developing database-intensive web applications

  – Getting Real (gettingreal.37signals.com/toc.php)
    • a must read for today’s web application developers