

# Chapter 1

## Introduction



## Java Software Solutions

### Foundations of Program Design

John Lewis  
William Loftus

Addison-Wesley  
is an imprint of



Copyright © 2014 Pearson Education, Inc.

## Outline



**Computer Processing**

**~~Hardware Components~~**

**Networks**

**The Java Programming Language**

**Program Development**

**Object-Oriented Programming**

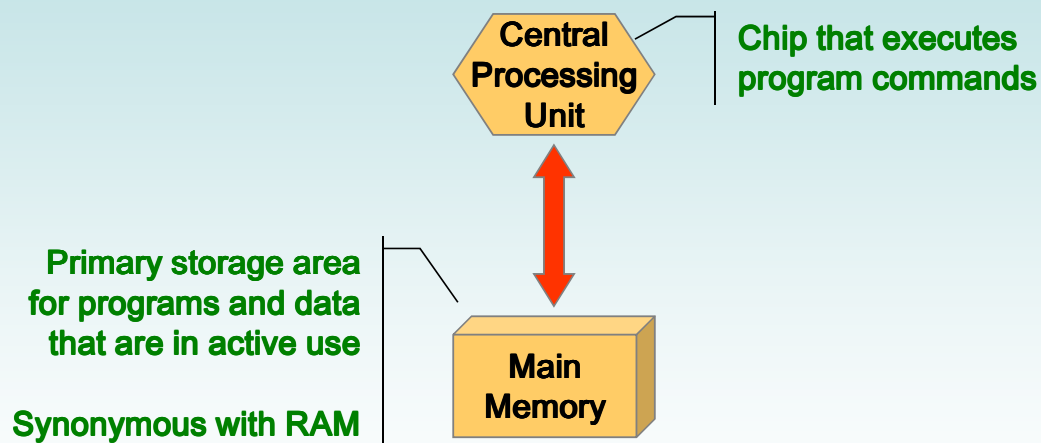
Copyright © 2014 Pearson Education, Inc.

# Hardware and Software

- Hardware
  - the physical, tangible parts of a computer
  - keyboard, monitor, disks, wires, chips, etc.
- Software
  - programs and data
  - a *program* is a series of instructions
- A computer requires both hardware and software
- Each is essentially useless without the other

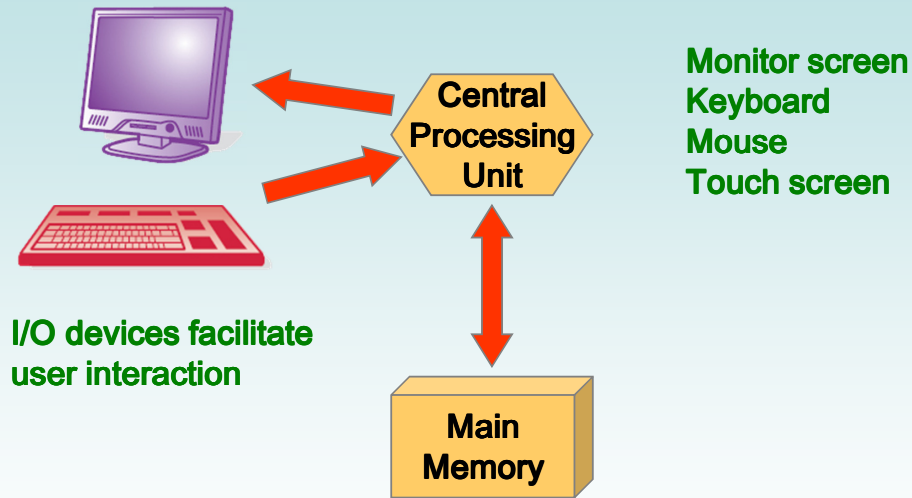
Copyright © 2014 Pearson Education, Inc.

## CPU and Main Memory



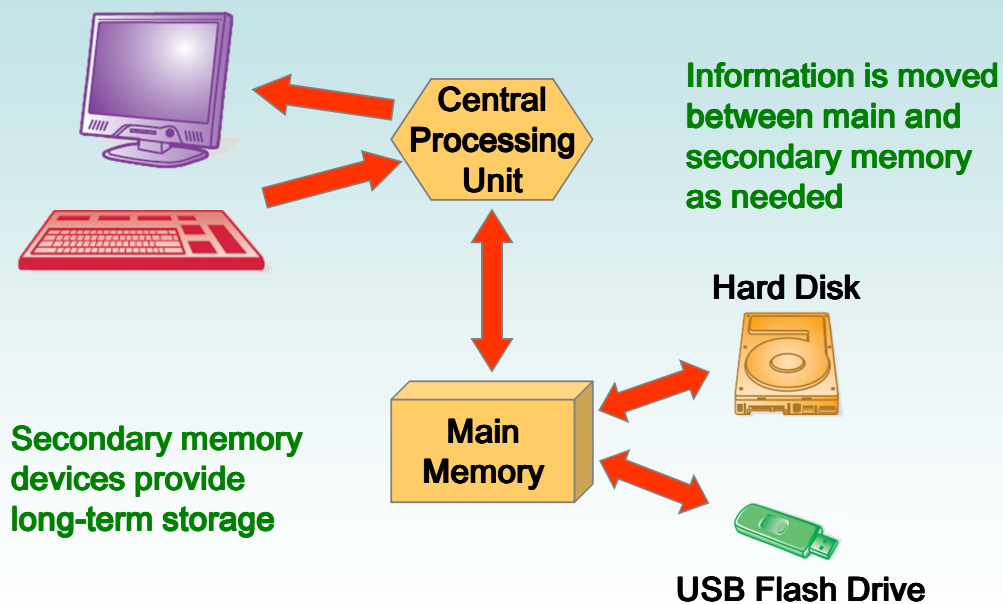
Copyright © 2014 Pearson Education, Inc.

# Input / Output Devices



Copyright © 2014 Pearson Education, Inc.

# Secondary Memory Devices



Copyright © 2014 Pearson Education, Inc.

# Software Categories

- Operating System
  - controls all machine activities
  - provides the user interface to the computer
  - manages resources such as the CPU and memory
  - Windows, Mac OS, Unix, Linux,
- Application program
  - generic term for any other kind of software
  - word processors, games
- Most operating systems and application programs have a *graphical user interface* (GUI)

Copyright © 2014 Pearson Education, Inc.

## Outline

**Computer Processing**

**Hardware Components**

**Networks**

 **The Java Programming Language**

**Program Development**

**Object-Oriented Programming**

Copyright © 2014 Pearson Education, Inc.

# Java

- The Java programming language was created by Sun Microsystems, Inc.
- It was introduced in 1995 and its 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*

Copyright © 2014 Pearson Education, Inc.

## 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`

Copyright © 2014 Pearson Education, Inc.

```

//*****
// 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.");
    }
}

```

Copyright © 2014 Pearson Education, Inc.

## Output

```

//*****
// 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.");
    }
}

```

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

Copyright © 2014 Pearson Education, Inc.

# Java Program Structure

```
// comments about the class
public class MyProgram
{
}

```

**class header**

**class body**

**Comments can be placed almost anywhere**

Copyright © 2014 Pearson Education, Inc.

# Java Program Structure

```
// comments about the class
public class MyProgram
{
    // comments about the method
    public static void main (String[] args)
    {
    }
}

```

**method header**

**method body**

Copyright © 2014 Pearson Education, Inc.

# 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 */
```

Copyright © 2014 Pearson Education, Inc.

# 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`

Copyright © 2014 Pearson Education, Inc.



# Identifiers

- Sometimes the programmer chooses the identifier (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

Copyright © 2014 Pearson Education, Inc.

# Reserved Words

- The Java reserved words:

<code>abstract</code>	<code>else</code>	<code>interface</code>	<code>switch</code>
<code>assert</code>	<code>enum</code>	<code>long</code>	<code>synchronized</code>
<code>boolean</code>	<code>extends</code>	<code>native</code>	<code>this</code>
<code>break</code>	<code>false</code>	<code>new</code>	<code>throw</code>
<code>byte</code>	<code>final</code>	<code>null</code>	<code>throws</code>
<code>case</code>	<code>finally</code>	<code>package</code>	<code>transient</code>
<code>catch</code>	<code>float</code>	<code>private</code>	<code>true</code>
<code>char</code>	<code>for</code>	<code>protected</code>	<code>try</code>
<code>class</code>	<code>goto</code>	<code>public</code>	<code>void</code>
<code>const</code>	<code>if</code>	<code>return</code>	<code>volatile</code>
<code>continue</code>	<code>implements</code>	<code>short</code>	<code>while</code>
<code>default</code>	<code>import</code>	<code>static</code>	
<code>do</code>	<code>instanceof</code>	<code>strictfp</code>	
<code>double</code>	<code>int</code>	<code>super</code>	

Copyright © 2014 Pearson Education, Inc.

# Quick Check

Which of the following are valid Java identifiers?

grade  
quizGrade  
NetworkConnection  
frame2  
3rdTestScore  
MAXIMUM  
MIN\_CAPACITY  
student#  
Shelves1&2

Copyright © 2014 Pearson Education, Inc.

# 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

Copyright © 2014 Pearson Education, Inc.

# 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`

Copyright © 2014 Pearson Education, Inc.


# Outline

**Computer Processing**

**~~Hardware Components~~**

**Networks**

**The Java Programming Language**

 **Program Development**

**Object-Oriented Programming**

Copyright © 2014 Pearson Education, Inc.

# 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

Copyright © 2014 Pearson Education, Inc.

# 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

Copyright © 2014 Pearson Education, Inc.

# 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

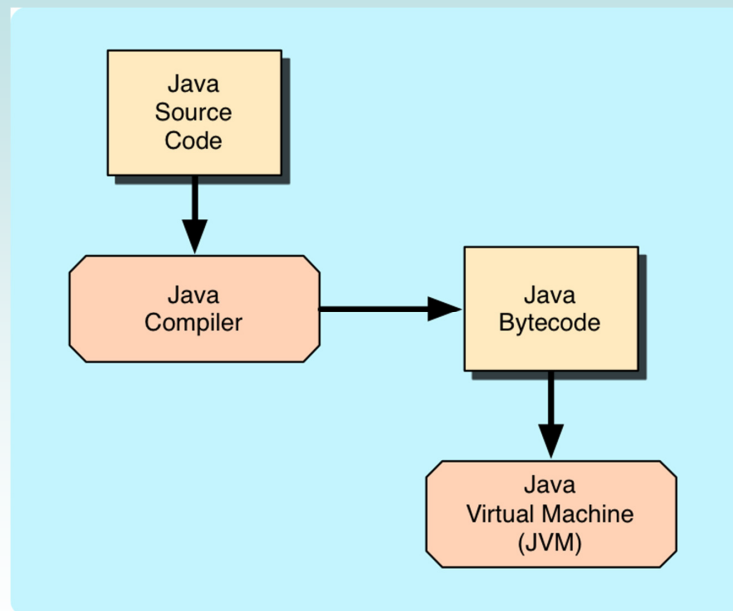
Copyright © 2014 Pearson Education, Inc.

## 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*

Copyright © 2014 Pearson Education, Inc.

# Java Translation



Copyright © 2014 Pearson Education, Inc.

## Development Environments

- There are many programs that support the development of Java software, including:
  - Java Development Kit (JDK)
  - Eclipse
  - NetBeans
  - BlueJ
  - jGRASP
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

Copyright © 2014 Pearson Education, Inc.

# 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

Copyright © 2014 Pearson Education, Inc.

# 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*)

Copyright © 2014 Pearson Education, Inc.

# Basic Program Development

