Vocabulary Chapter 01

Computer program: a set of instructions that you write to tell a computer what to do

Hardware: the general term for computer equipment

**Software:** the general term for computer programs

* Application software: a program that performs tasks for users (word processing, game playing)
* System software: a program that manages the computer itself (Windows, Linux)

Machine language/code: the most basic set of instructions that a computer can execute.

* A circuitry-level language that represents a series of on and off switches.
* Low-level programming language: written to correspond closely to a computer processor’s circuitry
* High-level programming language: allows you to use English-like vocabulary to write programs

Syntax: refers to the rules that define the ways language elements are used together correctly to create usable statements

* Keywords: the words that are part of the programming language
* Program Statements/Commands: orders to the computer to carry out the tasks that programs perform.

**Compiler/Interpreter**: a program that translates language statements into machine code;

* Complier: translates an entire program at once before executing any commands
* Interpreter: translates and executes one command at a time

Debugging: freeing the program of all bugs and errors

* Bug: a flaw or mistake in the program

Logic: involves executing the various statements and procedures in the correct order to produce the desired results

* **Syntax error**: results from misuse of the programming language; typing errors
* Compile-Time Error: the compiler detects a violation of language syntax and is unable to translate the source code to machine code
* **Logic Error**: a bug that allows a program to run but produces incorrect results
* Run-Time Error: when a program compiles successfully but does not execute
* Semantic Error: when you use the correct word in the wrong context

**Procedural Programming**: sets of operations are executed one after another in sequence

* Procedures: sets of operations performed by a program (method)

**Object-Oriented Programming:** involves creating classes, creating objects from those classes, and creating applications that use those objects

Variables: named memory locations that hold values that can vary

Class: collection of objects with common properties; blueprint for objects

Attributes: the characteristics that define an object as part of a class

Properties: attributes of a class

Object: an instance of a class

Instantiation: creating an object

State: the set of values an object has for its attributes

Method: a self-contained block of code (procedure)

\*\*Encapsulation: the enclosure of data and methods within an object

\*\*Inheritance: the ability to create classes that share the attributes and methods of existing classes, but with more specific features

\*\*Polymorphism: describes the features of languages that allows the same word to be interpreted correctly in different situations based on the context

**JAVA**

**JAVA:** an object-oriented language used both for general purpose business applications and for interactive, World Wide Web-based internet applications

Architecturally Neutral/“Write once, run anywhere” WORA: describes the features of java that allows you to write programs that run on any platform

Java Virtual Machine: a hypothetical (software-based) computer on which Java runs

Source Code: consists of programming statements written in high-level programming language

Development Environment: a set of tools that help you write programs by providing features as displaying a language’s keywords in color

jGRASP: a development environment and source code editor

Bytecode: consists of programming statements that have been compiled into binary format

Java Interpreter: a program that checks bytecode and communication with the operating system, executing the bytecode instructions line by line within the Java Virtual Machine

Applets: Java programs that are embedded in a web page

**Java Application**: stand-alone Java programs

* Console Applications: support character or text output to a computer screen
* Windowed Applications: creates a GUI with elements such as menus, toolbars, and dialog boxes

Literal String: a series of characters that appear exactly as entered; in Java it always appears between two double quotation marks

Standard Output Device: normally the monitor

* Standard Input Device: normally keyboard

Pascal Casing (Upper Camel Casing): a naming convention in which identifiers start with an uppercase letter and use an uppercase letter to start each new word

Identifier: the name of a program component such as a class, object, or variable

**Access Specifier:** defines the circumstances under which a class can be accessed

* **Static:** accessible and usable even though no objects the class exist

Void: the keyword that indicates the method does not return a value when it is called

**Parsing:** the process the compiler uses to divide source code into meaningful portions for analysis

Clean build: created when you delete all previously compiled versions of a class before compiling again

**Program Comments**: nonexecuting statements that you add to a Java file for documentation purposes

* Line: // - Block: /\* \*/ - Javadoc: /\*\* \*/

Dialog Box: a GUI object resembling a window in which you can place messages you want to display

Import Statement: accesses a built-in Java class that is contained in a package

Package: contains a group of built-in Java classes

Java API: the application programming interface, a collection of information about how to use every prewritten Java class

SDK: software development kit; set of tools useful to programmers

JDK: Java development kit