**AP Computer Science - Chapter 1 Summary**

**1.3 Programming Languages**

1. Machine Language
   1. Written in binary, the computer’s native language
   2. All processes performed by the computer are done in machine language.
   3. Punch cards were used to write programs
2. Assembly Language
   1. Early programming language that used mnemonics (add, sub) to perform simple math operations.
   2. Required a second program, an assembler, to translate the assembly language into machine language.
   3. Considered a low level programming language because it is close to machine language.
   4. Assembly was difficult to work with because at the time, different computers processed data differently, so the assembly language would have to be reworked. (machine dependent)
3. High Level Language
   1. Platform Independent – Run on different types of machines.
   2. Written in a language similar to English, than machine language.
   3. High Level languages create a *source program* (or source code), which the computer cannot understand.
   4. The source code is translated by a compiler (or interpreter) that converts source code to machine language for execution by the CPU.
   5. Common high level languages include C, C++, C#, COBOL, FORTRAN, Java, Pascal, Python, Visual Basic
   6. **Operating Systems (OS)**
4. An operating system manages and controls a computer’s activities.
5. Common operating systems: Microsoft Windows, Mac-OS and Linux
6. Major tasks of an operating system
   1. Controlling and Monitoring System Activities – Files, input/output
   2. Allocating and assigning system resources – CPU %, memory space, drive usage
   3. Scheduling Operations
      1. Multiprogramming – Multiple programs can run simultaneously
      2. Multithreading – A single program can execute several tasks simultaneously
      3. Multiprocessing – Workload can be split amongst several processors.
   4. **Java**
7. High level programming language
8. Highly integrated into web and browsers
9. Used for Stand-alone, networked and mobile applications.
10. Platform independent – Same java runs on Windows, Linux and Mac-OS.
11. Applet – Java programs designed to run in browsers.
    1. **Java Language Specification, API, JDK and IDE**
12. Java Language Specification
    1. Technical definition of the Java programming language’s syntax and semantics.
    2. Available at <http://java.sun.com/docs/books/jls>
13. API (Application Program Interface)
    1. Ever-expanding library of pre-defined classes and interfaces for developing Java programs.
    2. Available at <http://www.oracle.com/technetwork/java/index.html>
14. Three Editions of Java
    1. Java Standard Edition (Java SE) – Develop standalone applications or applets. (We use this)
    2. Java Enterprise Edition( Java EE) – Develop server-side applications
    3. Java Micro Edition(Java ME) – Use to develop applications for mobile devices
15. JDK – Java Development Kit – Software for developing Java applications. Current release is JDK 1.7
16. JRE – Java Runtime Environment – Software to run Java programs in your browser. Typically auto-updates.
17. IDE – Integrated Development Environment
    1. Software to simply the creation of Java programs
    2. The most popular IDEs for Java are Eclipse and NetBeans.
    3. The make compiling, troubleshooting and running Java applications much easier.
    4. **A Simple Java Program**
18. Classes – Every Java program has at least one class. For now, think of classes as a file that will hold your program’s code. Later we will write programs that utilize several classes.
19. Main Method
    1. public static void main(String[] args) { your code goes here }
    2. Main Method starts our program.
    3. Our program code will appear between the open and closed brackets that follow the main method.
20. Console output
    1. Means our program will be output to the monitor.
    2. Eclipse has a dedicated window to show us our console output.
21. Java Rules
    1. Every statement in Java ends with a semi-colon (;)
    2. Strings (words) are displayed in between double quotes.
    3. Java has a few reserved words, such as class, public, static, and void. There are more.
    4. Java uses braces { } to organize the parts of the program. Like parentheses, you will always have the same number of open and closed braces in your programs.
22. Comments
    1. We can use // to place a comment on a line. Any code that appears after // will not be processed.
    2. We can use /\* and \*/ to comment out a multi-line block of code.
23. System.out.println()
    1. This command is used to produce console output.
    2. Outputting Strings
       1. Use double quotes
       2. System.out.println(“Welcome to Java”);
    3. Outputting Mathematical Computations
       1. No quotes are used.
       2. System.out.println(1 + 2+ 3); Output would be the result, 6
       3. Know your order of operations

**1.8 Creating, Compiling and Executing Java Programs**

1. .java files
   1. This is the source code that you write.
   2. This code must be compiled for your program to run.
   3. Errors in the code will prevent your program from compiling.
2. .class files
   1. If a .java file successfully compiles, a .class file is produced.
   2. Class files are written in bytecode, which is essentially platform-independent machine language.
   3. JVM (Java Virtual Machine) is the software that reads the bytecode. JVM runs on all platforms.
3. Command line Commands
   1. If you are not using an IDE, you must manually compile your .java file using the javac <X.java> command.
   2. For example, if you had a Welcome.java file you wanted to compile: javac Welcome.java compiles it and creates a Welcome.class file. (bytecode)
   3. To run the class file, the command syntax is: java <classname>
   4. For our example, you would type: java Welcome

**1.10 Programming Style and Documentation**

1. Programming Style – Making your program organized and readable.
2. Documentation – Using comments to clarify and explain the parts of your logic and program.
3. Javadoc – An automated way of exporting your comments. Useful if sharing your classes with others.
4. Proper Indentation and Spacing
   1. Vital to matching braces
   2. Eclipse will help
5. Block Styles
   1. Next-line method – braces appear on lines by themselves
   2. End-Of-Line method – Open braces appear at the end of the line. Closed braces are on their own line.

**1.11 – Programming Errors**

1. Syntax Errors – Typos, missing semi-colon, missing brace
2. Runtime Errors – Cause program to terminate prematurely. Divide by zero is an example.
3. Logic Errors – Program compiles and executes, but doesn’t perform correctly. Incorrect formulas, for example.

**Chapter 1 Exercise List**

* Exercises 1.1 through 1.10 (all)
* Done and submitted in LiveLab
* Built, compiled and run in Eclipse