Java Notes Chapter 01 – Creating Java Programs

**JAVA:**

* Runs on hypothetical (software) computer known as the JVM
* Architecturally neutral/WORA
* Object-oriented & case sensitive
* Literal strings cannot be broken and placed on multiple lines
* Methods that require multiple arguments are separated by commas

1. Construct source code on text editor & saved in file

2. Java compiler converts source code into bytecode (binary program)

3. Java interpreter checks bytecode and communicates w/ OS, executing instructions within the JVM

**THE MAIN METHOD:**

Public static void main(String[ ] args)

{

}

* Main is the method identifier; begins w/ lower case in Java

- Not all classes have a main( ) method BUT

- All Java applications must include a class containing a public main( ) method

- JVM always executes the main( ) method first

* Public: access specifier
* Static: works without instantiating any object of the class
* Void: no return value
* String: argument passed is an array of strings
* Args: identifier of the array

-String[ ] args represents the type of argument that can be passed to the main( ) method

-String is a built-in Java class that can be used to hold character strings; begins w/ uppercase in Java

**IMPORTANT RULES:**

* Some people consider the term object-oriented programming to be synonymous with GUI programming, but object-oriented programming means more. Many GUI programs are object oriented, but not all object-oriented programs are GUI objects.
* Class name must be the same as the file name
* If any changes to the class/source code are made:
* save file with changes
* recompile class with javac command
* interpret the class bytecode and execute the class using the java command
* If you want to retain the original version, give the new version a new class & file name

**OBJECT-ORIENTED PROGRAMMING:**

* Creating Classes 🡪 blueprints for Objects
* Creating Objects 🡪 specific Instances of those Classes
* Creating Applications 🡪 use & manipulate those Objects
* Encapsulation:
* The **enclosure** of data and methods **w/in an object**
* Allows you to treat all of an object’s methods & data as a single entity
* In OO classes, attributes & methods are encapsulated w/in an object
* Also refers to the **concealment** of an object’s data & methods **from outside sources**
* Information Hiding: concealing data
* Implementation Hiding: concealing how methods work is
* Abstraction: showing only the necessary details/info to the user
* Polymorphism:
* Feature that allows the same word/symbol to be interpreted correctly in different situations, based on the context
* Method overloading/overriding
* Inheritance:
* The ability to create classes that share the attributes & methods of the existing classes, but with more added specific features
* Extended classes