**Lecture Notes (#2b)**

**Filenames in Java:**

* If the name of your class is TestString, it must be saved in a file named TestString.java

**Comments in Java:**

*-* Single line // from double slashes to end of line is all comment

- Multiple line /\* from ‘slash star’ to ‘star slash’ is comment \*/

* javadoc /\*\* the javadoc tool generates HTML pages to document .java files. You can add your own comments to the auto-generated comments by enclosing them between the Javadoc comment delimiters \*/

**Commenting Program source code (the .java file):**

* A source file should always have a ‘header’ comment block that at the very least gives a description of what the code does.
* For the purposes of this CSC142 class, the header comment block should also contain, on separate lines: Name, Author(s), Date, Assignment number.
* Any potentially confusing code blocks (e.g. double indexed ‘for’ loops) should have comments
* Variables should often have a description of their contents, unless their purpose is obvious. (e.g. ‘tempInt’ is obviously temporary storage for an integer value.)
* Comments are not only helpful to the next programmer who reads your code, try reading **your own** uncommented code after 6 months!

**A few notes on Java comments:**

* Comment nesting inside other comments only works when single line comments(//) are nested inside multiple line comments (/\* \*/). If you try to nest a multiple-line comment inside another multiple-line comment, your code will not compile.
* When you run javac, your source code is compiled starting from the beginning of the file to the end. javac will recognize the first “/\*” it encounters, as the beginning of a comment, and it will recognize the very next “\*/” as the end of that same comment. Convince yourself why this means that nesting multi-line comments will not work in Java.
* *Why would you want to nest multi-line comments in the first place?* When debugging software, sometimes you want to comment out a block of code that might be causing a problem, then re-compile it to test. If your code is easily readable and understandable, you should probably not need more than a single line for each comment in the body of the source code. If this is the case, you can comment out a block of code that also includes comments with no difficulty)

(the short version of this: use **multiple-line** or **javadoc** comments for the header comments in your source file, and prefer **single line** comments in the main body of your code.)

**Packages in Java:**

* A **package** is a way to logically organize classes and make them available to other classes through the **import** statement. You can either explicitly refer to a class everywhere in your code

( e.g. javax.swing.JFrame) or

You can import javax.swing.JFrame and just refer to the class in your code as JFrame.

A third alternative is to “import javax.swing.\*” This statement makes all classes in the javax.swing package available to your .java file.

Any import statements need to be at the top of the .java file.

Duplicate import statements will compile, but they are totally unnecessary, and make the original programmer look uncool.

Java imports the package java.lang.\* by default. It contains useful classes used by almost every Java program. (the String and Math classes, for example)

java.util.\* is imported quite often. It provides formatted printing and scanning, among many other utility classes.

The import statement is used to make the *names* of the package contents available to the programmer without having to fully specify the path to the package.

**Declaring a class**:

class <class name> {

<class member declarations>

}

**Declaring a method:**

<modifiers> <return type> <method name> ( <parameters> ) {

<method body>

}

**main method:**

* You must include one (and only one) main method in a class for it to run as a standalone application.
* The main method is optional for all other classes you have written that are used by the application.
* However, you can include main methods in ***all*** classes you’ve written for an application! You might use these for unit testing of each class individually.
* There is only one main method allowed per class.

**Constructing Objects (continued from Notes #2)**

The **new** reserved word calls a method that allocates the memory for an object. The method always has the same name as the class, and is called a **constructor.**

(We will discuss constructors in detail later on in CSC 142)

e.g JFrame() as above, is the constructor method for the class JFrame.

A Java class may define more than one constructor (we will explore this fact in detail in the coming weeks).

If you do not write a constructor method for a class you have created, the compiler will write one for you. Usually we will want to write our own constructors for any classes we have created.

**Reserved words in Java:**

In almost all programming languages, there are **reserved words** defined that you cannot use in your source code to create an identifier. So, reserved words cannot be used to name constants, variables, classes, objects, or methods. There are 57 reserved words in the Java programming language. We have encountered several of these already: ***new***, ***import***, and ***class*** are reserved words in Java.

**Reserved words** are also called **Java keywords**.

**String class:**

* String literals are enclosed in double quotes. (e.g. “Chris”)
* For efficiency, the String class is “different”. In a sense, it is somewhere between a built-in data type and an object. Both of the following statements create a String object called ‘name’:

**String name = “Chris”;**

**String name = new String(“Chris”);**

* String is the only class where this is allowed. All other objects in Java *must* be instantiated using the **new** operator. There are subtle differences between a String created using new, and one that is created as a string literal, but those differences shouldn’t matter to us for the programs you will create.

**Some useful information on the String class:**

[**https://www.tutorialspoint.com/java/java\_strings.htm**](https://www.tutorialspoint.com/java/java_strings.htm)