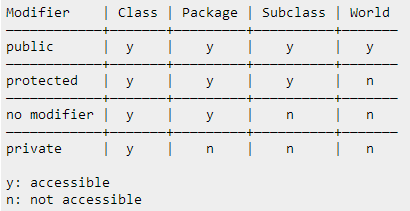
**B”H**

**Notes while programming**

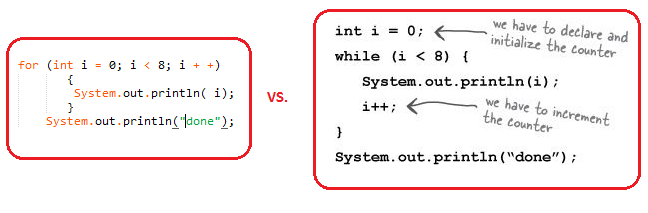
* Use good commenting such as saying what the end curly brace belongs to
* Objects live in one place and one place only — the garbage collectible heap.
* Java is pass-by-value. That means pass-by-copy.
* Encapsulation puts a force-field around the instance variables, so nobody can set them to something inappropriate.
* Access modifiers; in general only **public** or **private** will be used.



* General rule of thumb: mark all instance variables private; and make getters and setters and mark them public
* Instance variables always get a default value
* Local variables do NOT get a default value! The compiler complains if you try to use a local variable before the variable is initialized.
* Use **==** to compare two primitives, or to see if two object references refer to the same object.
* Use the **.equals()** method to see if two different objects are equal. (Such as two different String objects that both represent the characters in “Fred”)
* FIRST thing to do, is at the app level (in contrast to the class level), to create a high-level design using a flow-chart. Every app should first have a flowchart before going down to the “class” level:
* **Developing a Class:**

1. Figure out what the class is supposed to do. List the instance variables and methods. I.e. make the UML diagrams.
2. Write the pseudo-code for the methods.
   * This helps you focus on the logic without stressing about syntax
3. Write test-code for the methods.
   * A class or methods that will test the real code and validate that it’s doing the right thing.
4. Implement the class in real-code.
   * The actual implementation of the class. This is where we write real Java code
5. Test the methods.
6. Debug and re-implement as needed.

* Use the enhanced **for** loop when your loop needs to iterate over the elements in an array (or another kind of collection).
* A **while** loop has only the boolean test; it doesn’t have a built-in initialization or iteration expression. A **while** loop is good when you don’t know how many times to loop and just want to keep going while some condition is true.
* But if you know how many times to loop (e.g. the length of an array, 7 times, etc.), use a **for** loop because its cleaner.



* In general use the ArrayList over the standard array
* **&&** and **||**, are known as short circuit operators. Say you’re not sure whether a reference variable has been assigned to an object. If you try to call a method using this null reference variable you’ll get a NullPointerException. So, try this:

**if (refVar != null && refVar.isValidType() ) {**

**// do 'got a valid type' stuff**

**}**

* **EXTREME PROGRAMMING (XP):** These practices include things like:
* Make small, but frequent, releases.
* Develop in iteration cycles.
* Don’t put in anything that’s not in the spec (no matter how tempted you are to put in functionality “for the future”).
* Write the test code first.
* No killer schedules; work regular hours.
* Refactor (improve the code) whenever and wherever you notice the opportunity.
* Don’t release anything until it passes all the tests.
* Set realistic schedules, based around small releases.
* Keep it simple.
* Program in pairs, and move people around so that everybody knows pretty much everything about the code.