**Objectives:**

Methods as sub-programs, standard Java style formatting, for-loops, arithmetic and expressions, Strings (object type - associated functionality), and Scanner (input object).

**Continue with data types. (primitive) ints, doubles, chars, (reference) Strings**

Numbers are represented differently in memory. Some of the primitive types such as counting numbers are stored as type ints which is a direct binary representation (000 ... 01 is a binary representation of quantity 1, while 000...10 is a binary representation of quantity 2 and so on). Numbers with decimals divide storage into a part for left of the decimal portion, and a part for the right of the decimal portion. Characters are encoded to/from direct binary representations. and Strings are reference types (Objects) to a collection of characters, and as objects have associated functionality such as length() which can tell you how many characters or .equals() which can tell you if one string has the same contents as another.

***Exercise 1***

Download [ForLoopDemo.java](http://webhome.csc.uvic.ca/%7Ecsc110l/2012_5/lab2/code/ForLoopDemo.java) and compile and run it. Downloading is done by right clicking on the link and selecting save as, then saving to a folder.  Fix the style according to the [Style\_Guidelines.pdf](http://webhome.csc.uvic.ca/%7Ecsc110l/2012_5/lab2/Style_Guidelines.pdf).

***Exercise 2***

From last week:

Without the aid of a program answer the following questions as a class, supporting your answers with code examples.

1. What is the difference between a literal and a variable?
2. How do you distinguish a literal double from a literal integer of the same value.
3. If an arithmetic expression involves an integer and a double the results is \_\_\_\_\_\_\_\_?
4. Create a output line with 3 string literals and 2 integer literals.
5. Why do we put temporary print lines in code as we develop it, only to remove many of them later?

***Exercise 3: Methods as subprograms***

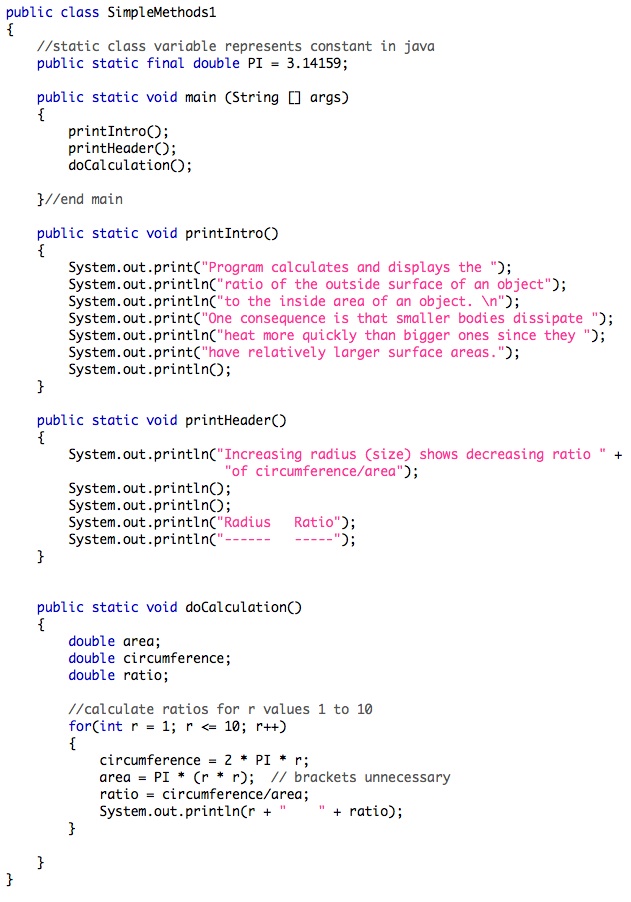
Methods as subprograms: Initially we will practice using procedural methods only. In the future we will work with functional methods too.

**Procedural methods**  are subprograms that do not return a value.  They may be passed parameters however.  The call to a procedural method does not expect a value returned.  i.e. printCoordinate(x,y)

**Functional methods** return a value that can be captured by the calling program.  A call to a functional method would expect some value to be returned. i.e. rootX = sqrt(x).  In java it is legal to ignore a returned value.  For example if I wish to ignore the rest of a line of input I could use an expression like: in.nextLine() where *in* is a variable of type Scanner and *nextLine()* is a Scanner method that returns the rest of the line as a string. The return value is not captured in a variable, and that's okay.  More about Scanner soon.

Download the program [SimpleMethods.java](http://webhome.csc.uvic.ca/%7Ecsc110l/2012_5/lab2/code/SimpleMethods.java). Compile and run it, then change it into the program pictured below. Note that the main is reduced, somewhat arbitrarily, to method calls and the work is done in three methods. Note the deference between the syntax of the **method invocation** in the main and the syntax of the **method header**.

Practice writing the method first then invoking it.  Practice testing one method at a time.  The TA will tell you how to do this and why it is most often done that way.



***Exercise 4: Working from the keyboard (reading Strings)***

Download [ScanStrings.java](http://webhome.csc.uvic.ca/%7Ecsc110l/2012_5/lab2/code/ScanStrings.java). Note the following expressions:

// reference the util library where the Scanner is defined.

import java util.Scanner;

// Define a variable of type Scanner and open it on System.in which  
// is the keyboard

Scanner keysIn = new Scanner(System.in);

// Use the Scanner method next() to get the next String.

input = keysIn.next();

// Print the strings separated by a space

System.out.print(input + " ");

**Exercise 5:**

Make ScanStrings1.java where a method is used to retrieve the Strings and print them.

After the message is printed, the main method should have a call to a method "doInput" and send the Scanner variable. The method doInput defines it's own copy of the Scanner variable and do the rest of the work. For convenience we will call the copy of the Scanner variable keysIn too. This is done in the method header, in the brackets which specify the parameter(s) type and name.

doInput(keysIn);

keysIn.close();

} //end main

public static void doInput(Scanner keysIn)

{

...

}

**Exercise 6: Written Review**

On a piece of paper (the TA may have some spare sheets), answer the questions given in the comment at the end of the program [Printing.java](http://webhome.csc.uvic.ca/%7Ecsc110l/2012_5/lab2/code/Printing.java). Hand in your answers to the TA.

The End