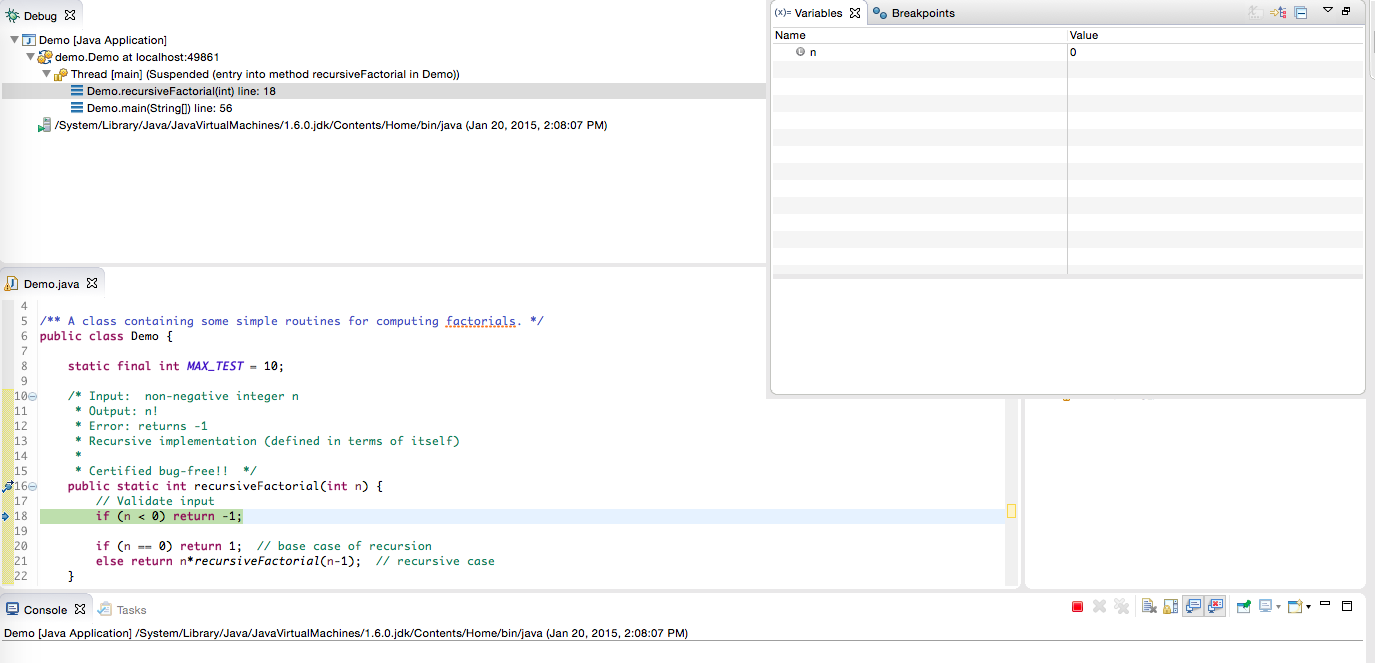
* When a method is called, a context is created and executes all code within the method
  + Local variables are created, then all the computation within a method is completed and then it returns some value back into where the method was called
  + The execution context is then deleted
* Within a debugger, you can determine the variables being created by a context
  + Every context is documented in the debugger on the top left



* Strings are objects of their own type, String
* Objects have methods, and can be called using a dot ( . )
  + Ex:

String s = "This is a string";

Int length = s.length();

* Objects also have properties or fields(descriptions), which are ALSO called using a dot ( .)
  + Ex: say you have defined a Giraffe class. You can then set the value of it's name like so:

Giraffe g = new Giraffe();

String name = g.name;

g.name = "My Giraffe";

* Methods that have 'static' vs ones that don't
  + If a method has 'static' in its signature, it is a class method
    - This means in order to invoke the method, you only need the name of the class. You don't need to create an instance of this method or create an object.
  + If a method doesn't have 'static', it is an instance method
    - This means to invoke the method, you NEED to create an object of this class type first, then call the method on that object
* Strings are enclosed in double quotes
  + To escape a string use \, \\, \t, \n, or \u####
  + You can also create strings by adding two together using the plus sign (+)

* Useful string methods:
  + Length()
  + charAt()
  + Equals()
  + Substring()
  + Trim()
  + indexOf()
  + Note: SOME OF THESE NEED PARAMETERS
* To compare two strings, you have to use the .equals() method
  + You can't use ==
* Strings in Java are immutable, meaning that the original string you create cannot be changed
  + If you do change a string, the original string still exists. You're simply creating a new string.
  + Immutability is powerful because it makes your program more efficient
    - If Java finds two of the same string, it only refers to one
  + The one exception is to create a string using the 'new' operator
    - String s1 = "A String";
    - String s2 = new String("A String");
* Arrays are a litle different because it is a type that combines other types
  + The array itself is of type array, but the individual elements must also be specified
  + In java, the elements within an array must all be the same
  + There are various ways to initialize an array:
    - Int [] array = {1,2,3};
      * Creates an array with elements of type int, with 3 elements that are 1,2, and 3
    - Int[] array = new int[3];
      * Creates an array with elements of type int, of size 3. Doesn't specify what the elements are. If you choose to do it this way, you have to then define the individual elements:
        + array[0] = 1;
        + array[1] = 2;
        + array [2] = 3;
* Length of an array is found through the length field, which means you don't need to use parenthesis
  + Size of an array can't be changed, but the individual elements can be changed
* Null is a special value that is always valid for any reference type
  + It means "no value"
  + Default value for arrays created in the second way above
* If you pass an array into a method, the method can change individual elements. Any changes that occur in this method will be permanent.
* You can also clone an array, so a new array is created with elements that point to the same reference
  + Use .clone()
  + Sometimes this is better to pass to a method, because your original array isn't changed but instead the new one is changed by the method
* Multidimensional arrays
  + Int [] [] myArray = new int[5][3]
    - Creates an array of 5 by 3
* Arrays is a library of useful functinos for manipulating arrays, such as binarySearch or sort
* Scanner objects
  + After creating a scanner object, you then can call methods such as next() or nextInt() to interpret the inputs as certain types