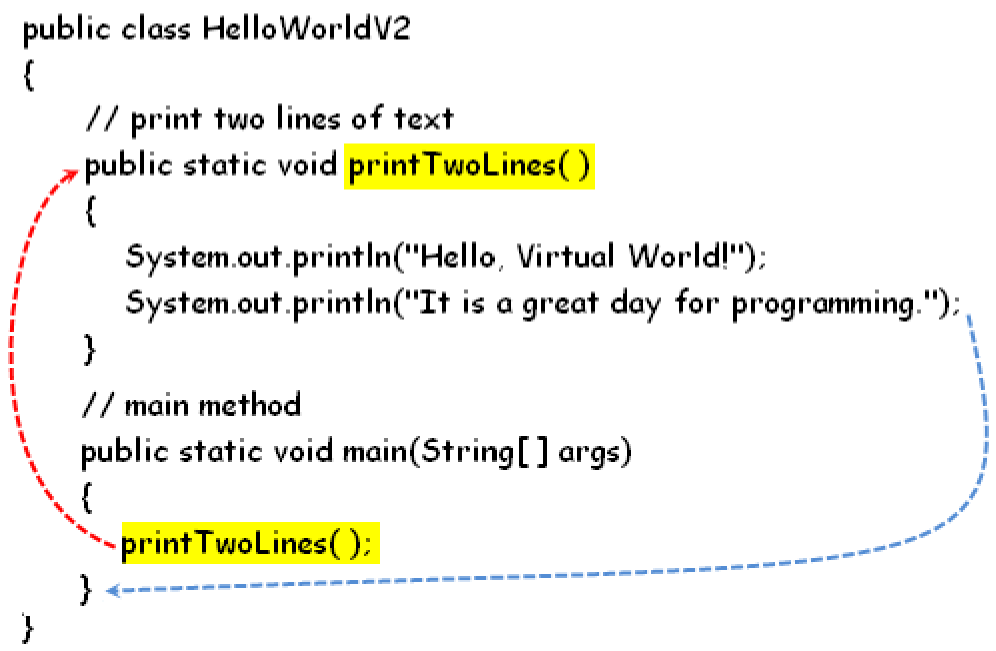
05.01 Virtual Lecture Notes (Part 1)

The original HelloWorld class can be easily re-implemented in a top-down design by placing the two print statements in a method, as illustrated in the following program.



```
public class HelloWorldV2
{
    // print two lines of text
    public static void printTwoLines( )
    {
        System.out.println("Hello, Virtual World!");
        System.out.println("It is a great day for programming.");
    }
    // main method
    public static void main(String[ ] args)
    {
        printTwoLines( );
    }
}
```

The details of method implementation are not important at this time; your focus should be on the new organizational structure and the flow of control imposed by the top-down design. Notice that the two print statements have been relocated *above* **`main()`** as a functional block of code in a separate method called **`printTwoLines()`**.

Program execution always begins from the **`main()`** method, so the highlighted line within **`main()`** calls the **`printTwoLines()`** method. As soon as the **`printTwoLines()`** method is invoked, flow of control jumps to that method (follow the red line) and the two **`String`** literals are printed. After the method is executed, flow of control returns to the line after the statement that called it (follow the dashed blue line). In this case, there are no other executable statements, so the program terminates.

Once again, this is a very simplistic example, but it clearly illustrates the new design and flow of control imposed by using methods.

Run the program and observe the output. Does the program seem to perform like the original version, or differently? It should work just like the original.

Before continuing, try adding a **`printMyLine()`** method to the class to print **`String`** literals of your choice. Just follow the pattern and add the new method above the **`main()`** method. Be sure to add an invoking statement for the new method below the highlighted line in the **`main()`** method.

Print