2.09 APCS

**Getting Started with Static Methods**

In Java, a [**method**](javascript:void(0);) (a segment of code that performs a specific task when invoked by a message statement) is a small segment of code that performs one specific task; when it is called by name, the method performs its task. The **main()** method can become cluttered with code that performs a wide variety of tasks, so it is time to implement a [**top-down design**](javascript:void(0);) (a style of coding in which programs are broken down into smaller subtasks) approach. This basically means that programs are going to be reorganized into concise, functional, modular units that can be called as needed and reused in programs.

This lesson is all about understanding how to call on those methods to execute their code. Pay close attention to the way the methods are called.

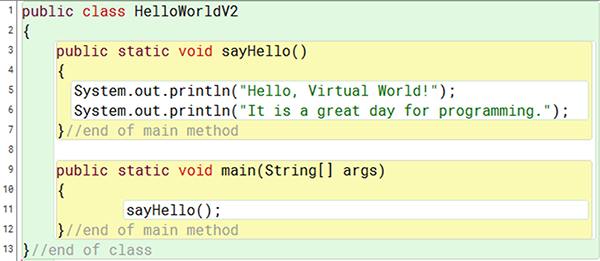
Notice that in the code snippet below we have two methods: the trusty main method that we are all familiar with, as well as a second method named **sayHello()**.

A closer look at the code in the **sayHello()** method reveals this print out:

Hello, Virtual World!  
It is a great day for programming.

In the main menu, the code **sayHello()** is what "calls" on the second method to perform its task.

There are some very important characteristics of this method and its call that you should be aware of. Take a look at the code here:



Public class HelloWorldV2

Return

Public static void sayHello open parentheses closed parentheses

Return

System.out.println open parentheses "Hello, Virtual World" closed parentheses;

System.out.printlnopen parentheses "It is a great day for programming." closed parentheses;

Return end of main method

Public static void main open parentheses String open bracket closed bracket args closed parentheses

Return

SayHello open parentheses closed parentheses

Return end of main method

Return end of class

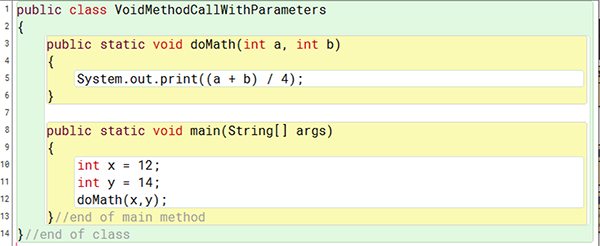
* After the name **sayHello()** there are parentheses with nothing inside, revealing that the method does not need any input from us to perform its task. We would say that this method has no arguments or parameters.
* Immediately before the name **sayHello()** is the word **void**. Anything written immediately before the name of the method identifies what type of "return" value the method has. The word **void** here means that this method will not be sending back any values to the main method either. Sometimes methods return int, double, string, or other data types.
* Note that in the main method, the line **sayHello()** is what "calls" the **sayHello()** method causing it to execute its code.

## Void Methods with Parameter

In the code below, notice two important changes:

* the once empty parentheses now contain two values: int a and int b
* the call in the main method also has two values in parentheses as well: x and y

Here, the main method is 'sending' the values of x and y to the **doMath()** method. The **doMath()** method receives these values and assigns them to the variables a and b, respectively. Finally, the **doMath()** method executes its code using the values it has for the a and b variables.



-------------------------------------------------------------------------------------------------------------------------------------

Public class VoidMethodCallWithParameters

Return

Public static void doMath open parentheses int a, int b closed parentheses

Return

System.out.print open parentheses open parenthese a + b divided by 4 closed parentheses;

Return

Public static void main open parentheses String open bracket closed bracked args closed parentheses

Return

Int x = 12;

Int y = 14;

doMath open parentheses x, y closed parentheses;

Return end of main method

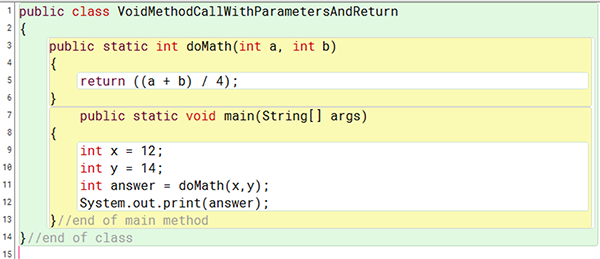
Return end of class

-------------------------------------------------------------------------------------------------------------------------------------

### Void Methods with Parameters and Return

In its most complex form, this method can both receive information from the main method and send values back to the main method. In the example below, you should notice:

* The word void in front of the method name has been replaced with int. This means that we expect this method to send (we say return) an integer value back to the main method.
* The use of the return keyword within the method, which identifies the value to be returned.



-------------------------------------------------------------------------------------------------------------------------------------

Public class VoidMethodCallWithParameters

Return

Return parentheses int a, int b closed parentheses

Return

Return open parentheses open parenthese a + b divided by 4 closed parentheses;

Return

Public static void main open parentheses String open bracket closed bracked args closed parentheses

Return

Int x = 12;

Int y = 14;

Int answer equals doMath open parentheses x, y closed parentheses;

System out print open parentheses answer closed parentheses;

Return end of main method

Return end of class

-------------------------------------------------------------------------------------------------------------------------------------

### **Assignment 02.09:**

1. Complete the reading and activities for this lesson.
2. Download and edit the [**MakingMethodCalls.java**](https://lti.flvsgl.com/flvs-cat-content/7tilc7scvh0e6rv078toqe80dr/flvs-cat-session/apcomputersciencea_v20/module02/lesson09/java/makingmethodcalls.java) file using the comments to guide your work.
3. Use the [**rubric**](https://lti.flvsgl.com/flvs-cat-content/7tilc7scvh0e6rv078toqe80dr/flvs-cat-session/apcomputersciencea_v20/module02/lesson09/rubrics/02_09d/02_09d_rubric01.htm) to guide your work.
4. Submit the updated MakingMethodCalls.java file to the **02.09 Calling Methods in Java** assignment.