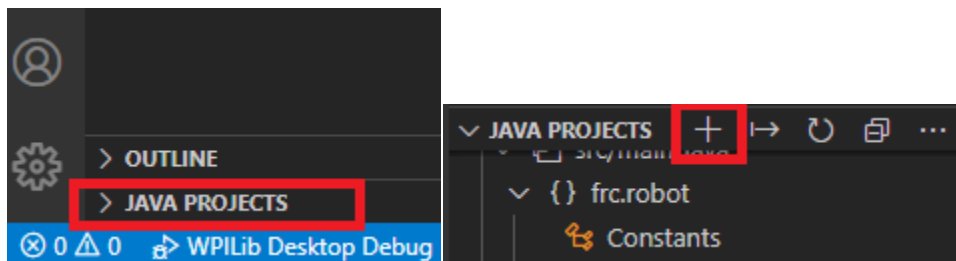


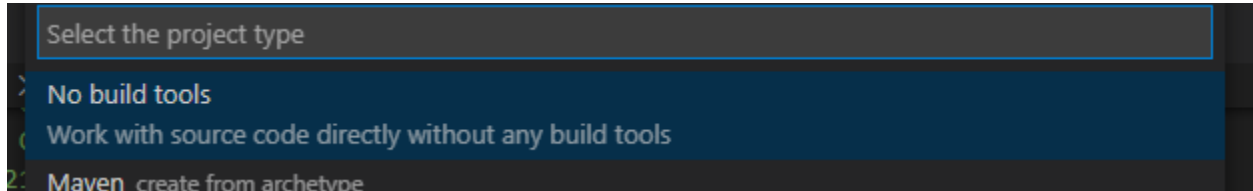
Over the past two lessons you've learned what methods are and used them, but you've probably noticed a lot of things that you don't understand about them. In this lesson we're going to start building your understanding. Remember, methods are basically self-contained chunks of code that can be called in some other location, so the code doesn't need to be repeated. There are built-in methods (for example `System.out.println`) and you can also write your own methods. To do this lesson, we're going to want a clean workspace in VSCode that doesn't interact with the Romi, so we can write some basic methods and run them without needing to connect to the robot. Then in the next lesson we'll switch back to working with the Romi and you'll write some code that gives the Romi a slightly different control scheme while you hold down a button.

Setting Up a New Workspace

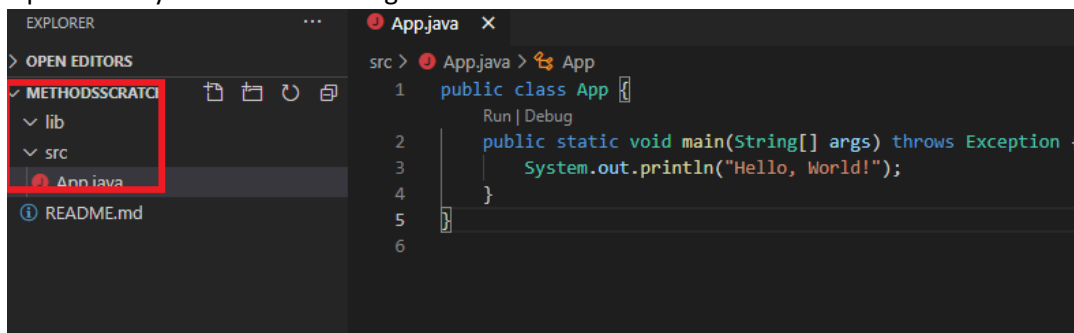
To set up a new workspace in VSCode, click on the "Java Projects" button in the lower-left corner of your screen. Then click the "+" icon. If you don't see the Java Projects button, click the Explorer icon in the upper-left corner of your screen, and it will appear.



After clicking the plus, select the "No build tools" option from the drop down:



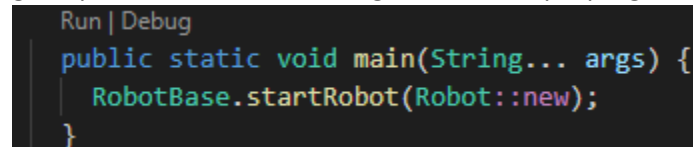
A window will pop up asking you to designate a folder for the project. You can choose whatever folder you want. You might find it helpful to create a folder somewhere on your computer to hold all of your programming projects. Regardless of where you locate it, select a folder for the project location. Type a name for your project and press enter; a good name to use is "MethodsScratch". After you press enter, you'll be greeted with a new instance of VSCode and you won't see any code, but your project will already have a file called `App.java`. You can open it by drilling into the "src" folder in explorer on the left. Open it and you'll see something like this:



Go ahead and press F5 and you'll see "Hello, World!" output in the terminal at the bottom of your screen. Outputting "hello world" is a common first program to write and is often used in examples and teaching. Try typing something else in the quotes in the SOP on line 3 and running your program again, just to prove that the code you're running is the code you're looking at. Now that you're set up, you'll be able to try out writing your own programs from scratch and running them here!

The Main Method

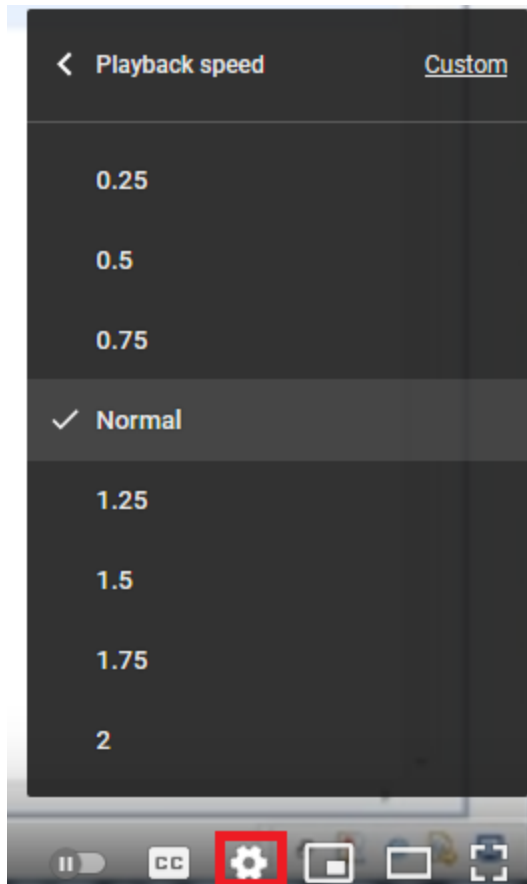
Every Java program must have exactly one "main" method, defined by the rather confusing-looking chain of word "public static void main(String[] args)". Each of those words have a meaning you will learn soon – in this lesson you'll learn what "void" means, and you can ignore the other words there for now. But the word "main" is the name of the method. Each program must have exactly one, because the main method is where your program starts, and every program needs a defined starting point. In practice there's usually not much code in main methods; it simply calls some other method with a single line. As an example here's the main method of your Romi project (you can find this in Main.java in java/frc/robot if you want to look for yourself.) Having only one line of code in your main method is a good practice to follow although for the sample programs you'll write today it's not super important.

A screenshot of a code editor with a dark background. At the top left, there is a tab labeled "Run | Debug". Below it, the following Java code is displayed with syntax highlighting:

```
public static void main(String... args) {  
    RobotBase.startRobot(Robot::new);  
}
```

YouTube Videos

Now that you have a work environment setup where you can code, it's time to get some practice. Throughout this course there will be links to various YouTube videos. There are a couple things to keep in mind with YouTube videos. First, they're often in a different application for writing code. You might see the author use keyboard shortcuts or interface buttons that don't exist in VSCode. However, the code itself runs the same way, so don't worry about matching the keyboard shortcuts. Second, you may find it beneficial to change the playback speed to go either faster or slower depending on your prior experience and understanding. You can change the playback speed by clicking the gear in the lower-right corner and selecting a speed.



Method Types

Watch this video on YouTube for a walkthrough on writing some simple methods, and follow along by writing out and running the methods yourself in VSCode: <https://www.youtube.com/watch?v=-IJ5izjbWIA> In the video, two different kinds of methods are used – some that perform some action without returning any data, and some that return data. This is where the *void* keyword comes in. A method that does not return any data is declared using the void keyword. A method that does return data is declared using the type of data it returns. Thus far we've worked with ints, doubles, booleans (true/false), and a couple of objects. Another important data type that is used in the video is String. A string is simply text. You'll notice that unlike int and double, String is capitalized like an object. This is because Strings *are* in fact objects, but they are built-in objects that are part of Java. Strings being objects will matter more later but doesn't matter too much right now for following along with the video. It's simply good to know so you understand why String is capitalized while int is not.

Scope

After finishing the methods video, watch one more video about variable *scope*, and again follow along by writing out the methods in VSCode: <https://www.youtube.com/watch?v=Y2iN3TO5qQQ> "Scope" is the concept of where a variable exists and for how long. Basically, variables declared in a method only exist in that method, and variables declared in a class exist throughout that entire class. A variable cannot be accessed or changed from a location in code where it does not exist.

Getting User Input

With the Romi, we got user input in the form of joystick and button presses. With this desktop app that we're working on, we don't have a joystick or buttons, but we do have our computer's keyboard. Let's write a quick program that gets input from the user and then prints the input back out. To do this, we'll use Java's built in *Scanner* class. The scanner accepts input in the terminal window where your output shows up. You will be able to type a number, press enter, and your program will read that number. Make sure that if your program is looking for an integer, you give it an integer! If you give it a decimal or a String (text), you will get an error. Here is a sample program you can write in VSCode:

```
App.java X
src > App.java > ...
1  import java.util.Scanner;
2
3  public class App {
4      public static void main(String[] args) throws Exception {
5          // Declare a built-in Java Scanner object
6          Scanner scanner = new Scanner(System.in);
7
8          // Get an integer from the user
9          System.out.println("Enter an integer:");
10         int userInput = scanner.nextInt();
11
12         // Output the user's input
13         System.out.println("User input: " + userInput);
14
15         // Closes the scanner to prevent memory leaks
16         scanner.close();
17     }
18 }
19
```

Run this program and look at the terminal in the bottom of your screen, and it will be asking you for input. Type an integer, press enter, and the program will echo your input:

```
Enter an integer:
12
User input: 12
```

Practice

You're now familiar with several concepts in Java: variables, methods, classes, objects, scope, and getting user input. That's a lot of knowledge, and it's enough for you to start writing some programs on your own. We'll make a small calculator app. Create a new project in VSCode (refer back to the start of this lesson if you forget how to do so) and call it Calculator. In your main method, declare a Scanner

object and use it to initialize two integer variables. For example:

```
Scanner scanner = new Scanner(System.in);  
int firstNumber = scanner.nextInt();  
int secondNumber = scanner.nextInt();
```

Now add five methods to your program as follows.

1. A "welcome" method that takes no parameters and returns nothing, that outputs a simple "Welcome to calculator" message to the user.
2. An "add" method that takes two integer parameters, adds the two integers together, and outputs the result. This method does not return anything.
3. A "multiply" method that takes two integer parameters, multiplies them together, and then returns the result as an integer.
4. A "subtract" method that takes two integer parameters, subtracts them, and then returns the result as an integer.
5. A "cube" method that takes a single integer parameter, cubes it, and returns the result as an integer.

Then, in your main method, call each of these six methods using one or both numbers that the user input. For the methods that return a value instead of outputting a result, output the value they return in your main method. You can use a variable to do this like so:

```
int product = multiply(firstNumber, secondNumber);  
System.out.println(product);
```

When you're done and you run your program, it should ask for two inputs, and then have five lines of output.