

Java Basics Iteration 1

Previous Iteration	Wiki Home	Forums	Next Iteration
Iteration 0	Wiki Home Page	Forums	Iteration 2

Previous Iteration	tWiki Home	Next Iteration
Iteration 0	tWiki Home Page	Iteration 2

Table of Contents

- [Iteration Objectives](#)
- [Your first Java Program](#)
 - [Hello World in the command line](#)
 - [Exploring an integrated development environment \(IDE\)](#)
 - [Hello World in an IDE](#)
- [Demo](#)
- [Next Steps](#)

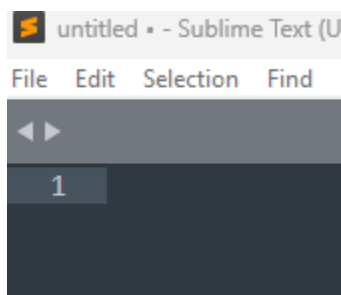
Iteration Objectives

Create a Java program and explore an integrated development environment (IDE)

Your first Java Program

In order to write our first Java program, we will be using some kind of word editor. We need a text editor to write our code. You can download Sublime Text [Sublime Text - Text Editing, Done Right](#), which is a common text editor that developers use. Sublime is a tool that we will use to write our Java programs. In the command line is a tool that will help us build and run our programs. With Sublime downloaded, we can open up a new window and write some code.

How do we write a Java program? Java is verbose programming language, so there is some boilerplate code that we add with every program. When we say verbose, we mean that there are a lot of words we have to write, compared to other programming languages, in order to get a simple program to work. It's easier to learn what you may not have known, and it makes what you are trying to do very verbosely clear. Every Java program has to have a class, and in fact, all Java code must be inside a class. We'll talk more about classes later in the course, but we'll be creating a class and calling it "HelloWorld". We'll write " HelloWorld", the name of our class, and then open and close curly brackets. The idea of Hello World is a bit of a tradition in computer science. When you are first learning a new language, or framework, or technology, all you want to do is make the program output "Hello World", and that's exactly what we'll be doing in this lesson.

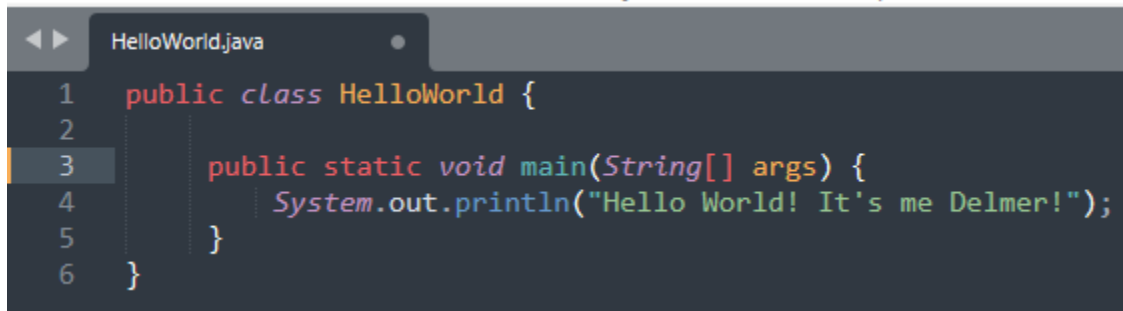


Let's put some more code in between the curly brackets. We'll write "public static void main(String[] Args)", and then open and close curly brackets.

This may look a bit overwhelming, but don't worry. All we need to know about this piece of code, is that it's called the main function, and it's the entry point to our program. This means any code I put between the inner curly brackets will be executed by the program. Everything we've written so far is boilerplate, or stuff we must have for any Java program to run. The thing that will make our program unique, is what we add in those curly brackets. As we said before, we want our program to output a "Hello World" statement. We can do that by adding the following line of code. We'll write:

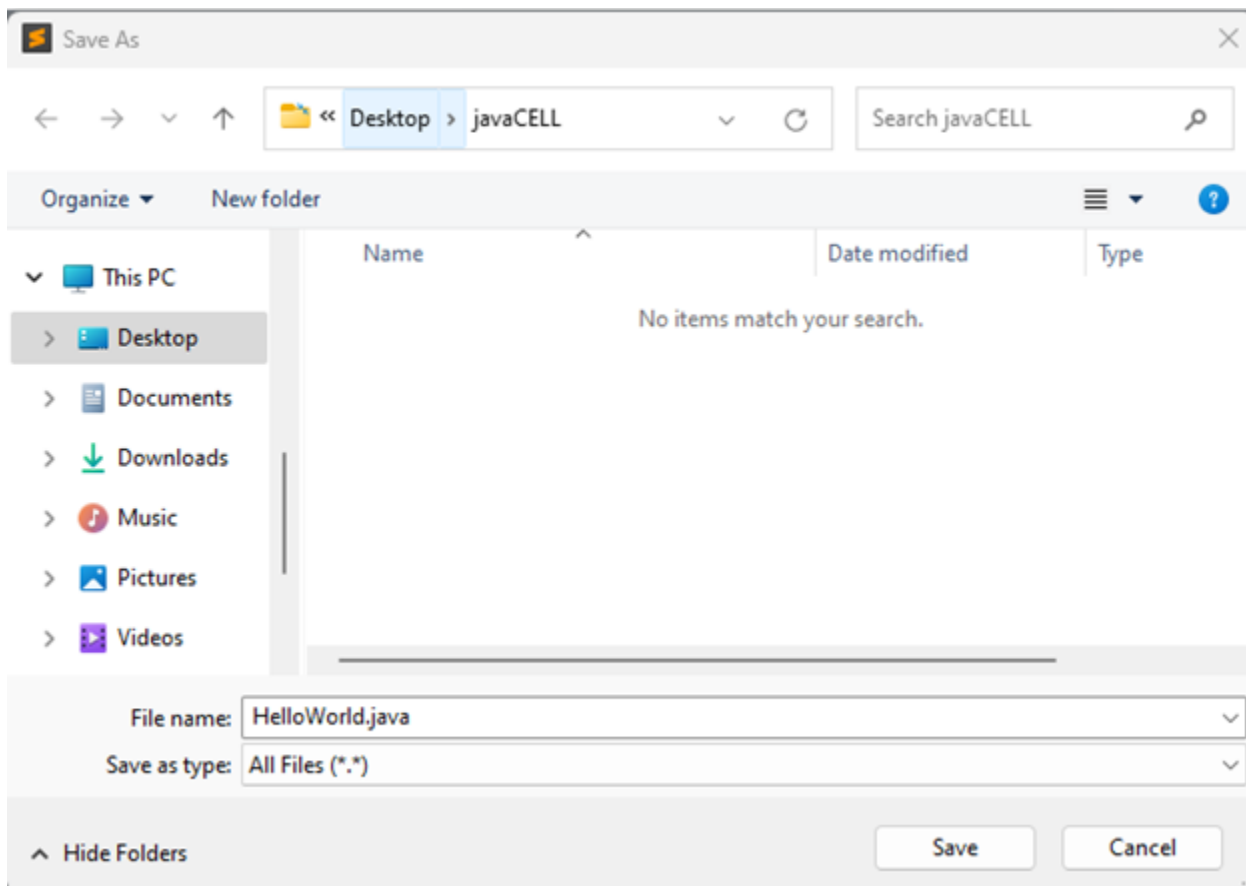
```
System.out.println ("Hello World! It's me Delmer!");
```

The "System.out.println" portion gets us access to the outputting, or printing functionality of Java. And in between the parentheses we write what we want to output. In this case, "Hello World! It's me Delmer (you can add your name)!" You might be wondering what public means, and how it relates to Java, but that is out of the scope for this introductory course.



```
1 public class HelloWorld {
2
3     public static void main(String[] args) {
4         System.out.println("Hello World! It's me Delmer!");
5     }
6 }
```

We will talk about what void and static mean in later chapters. With our code done, we'll save this file by clicking File, and then Save. We'll name this file "HelloWorld" and give it the dot Java extension because it is a Java file. Java conventions require the file name to be the same as the class name, so make sure the file name and the class name match. You can save this anywhere you'd like on your computer. We are going to be saving it to the learning Java folder for this example.



```
1 public class HelloWorld {
2
3     public static void main(String[] args) {
4         System.out.println("Hello World! It's me Delmer!");
5     }
6 }
```

Alright, let's execute and run our program. We just used Sublime to write our program; next we'll use the command line to build and run our program.

Hello World in the command line

Java code must be Compiled before it is run. This means we'll be using one command to compile our code and another command to execute or run our code. We won't be discussing the details of compilation in this course, but it is important to know that we are taking our Java Source Code and compiling it into Java bytecode. Upon compilation, Java class files with the .class extension, will be produced, and they will be used when we run the program. Before we compile our program, we need to go to the directory where the source code lives. We'll need to go where this folder lives and our HelloWorld program is. So, when we use `cd`, we'll see that learning Java folders is there, and you can actually hit tab to auto complete the rest of the folder name.

```

Microsoft Windows [Version 10.0.22621.521]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd Desktop

C:\Users\Administrator\Desktop>cd javaCELL

C:\Users\Administrator\Desktop\javaCELL>dir
Volume in drive C is OS
Volume Serial Number is 00A6-F542

Directory of C:\Users\Administrator\Desktop\javaCELL

10/07/2022  05:38 PM    <DIR>          .
10/07/2022  05:10 PM    <DIR>          ..
10/07/2022  05:38 PM                146 HelloWorld.java
               1 File(s)                146 bytes
               2 Dir(s)  20,614,840,320 bytes free

C:\Users\Administrator\Desktop\javaCELL>

```

Hit enter to go into this folder, type dir and see our Java file inside. To compile our Java source code into Java byte code, we'll write javac and then the name of our file, which is HelloWorld.java.

```

C:\Users\Administrator\Desktop\javaCELL>javac HelloWorld.java

C:\Users\Administrator\Desktop\javaCELL>

```

We have just compiled our code. Notice the Java class file that's now in our learning Java folder. We can now execute our program. To execute our program, we can write java and then the name of the Java class folder without the .class extension, HelloWorld. Hit enter and see the output right here in the command line.

```

C:\Users\Administrator\Desktop\javaCELL>dir
Volume in drive C is OS
Volume Serial Number is 00A6-F542

Directory of C:\Users\Administrator\Desktop\javaCELL

10/07/2022  06:05 PM    <DIR>          .
10/07/2022  05:10 PM    <DIR>          ..
10/07/2022  06:05 PM                442 HelloWorld.class
10/07/2022  06:05 PM                147 HelloWorld.java
               2 File(s)                589 bytes
               2 Dir(s)  20,575,748,096 bytes free

C:\Users\Administrator\Desktop\javaCELL>java HelloWorld
Hello World! It's me Delmer!

C:\Users\Administrator\Desktop\javaCELL>

```

Hello world, it's me. That's exactly what we wrote in our print statement. Let's try running the command again. We'll use the up arrow to access the previously run command. Hit enter, and we see the same output in our terminal. Now let's try changing our Java source code, so something else is printed out. In our Java code, we'll add on the phrase, "and I love tacos". We'll save this and try rewriting the program.

A screenshot of an IDE window titled 'HelloWorld.java'. The code is as follows:

```
1 class HelloWorld {
2
3     public static void main(String[] args) {
4         System.out.println("Hello World! It's me Delmer and I love tacos!");
5     }
6 }
```

A screenshot of a terminal window with the following commands and output:

```
C:\Users\Administrator\Desktop\javaCELL>java HelloWorld
Hello World! It's me Delmer!

C:\Users\Administrator\Desktop\javaCELL>javac HelloWorld.java

C:\Users\Administrator\Desktop\javaCELL>
```

Hit that up arrow, then hit enter, and we still see Hello World, It's me. Since we've changed the source code, we need to recompile and re-execute the program in order to get the latest version. If we just re-execute, which we just did, we get the old version of the program. To recompile we'll type javac HelloWorld.java. This compiles it into Java bytecode, and then to run our program, we'll go java HelloWorld, and we get the "and I love tacos" in the output.

A screenshot of a terminal window with the following commands and output:

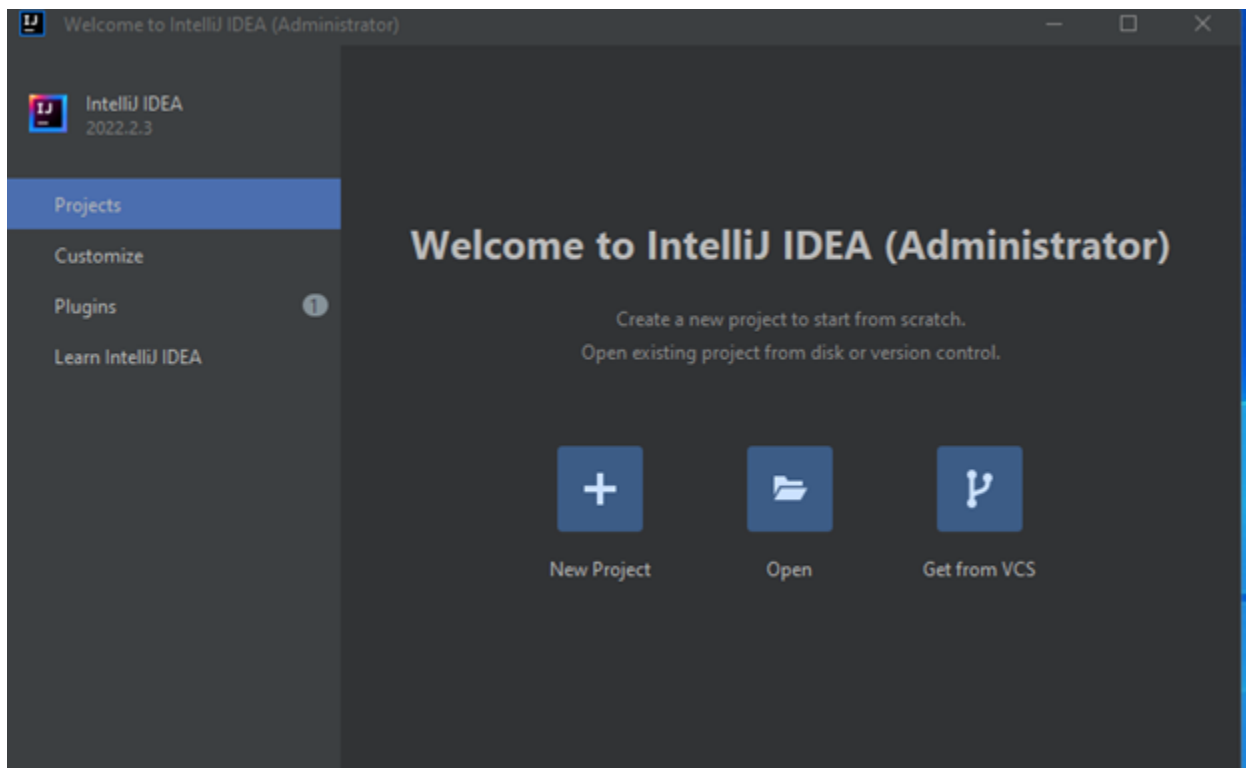
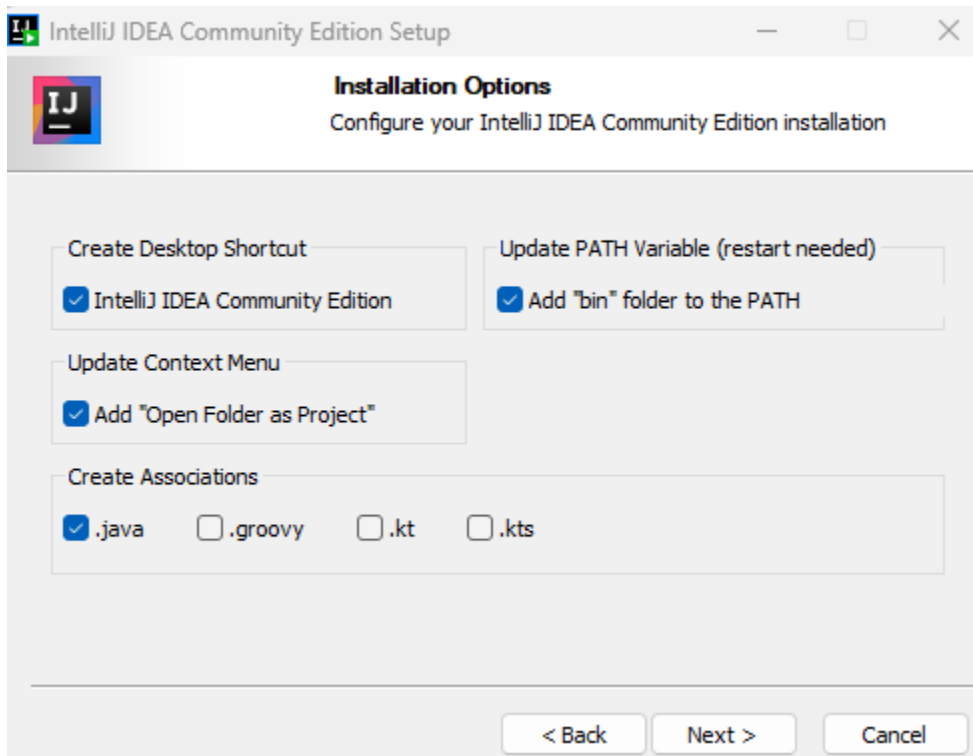
```
C:\Users\Administrator\Desktop\javaCELL>java HelloWorld
Hello World! It's me Delmer and I love tacos!

C:\Users\Administrator\Desktop\javaCELL>
```

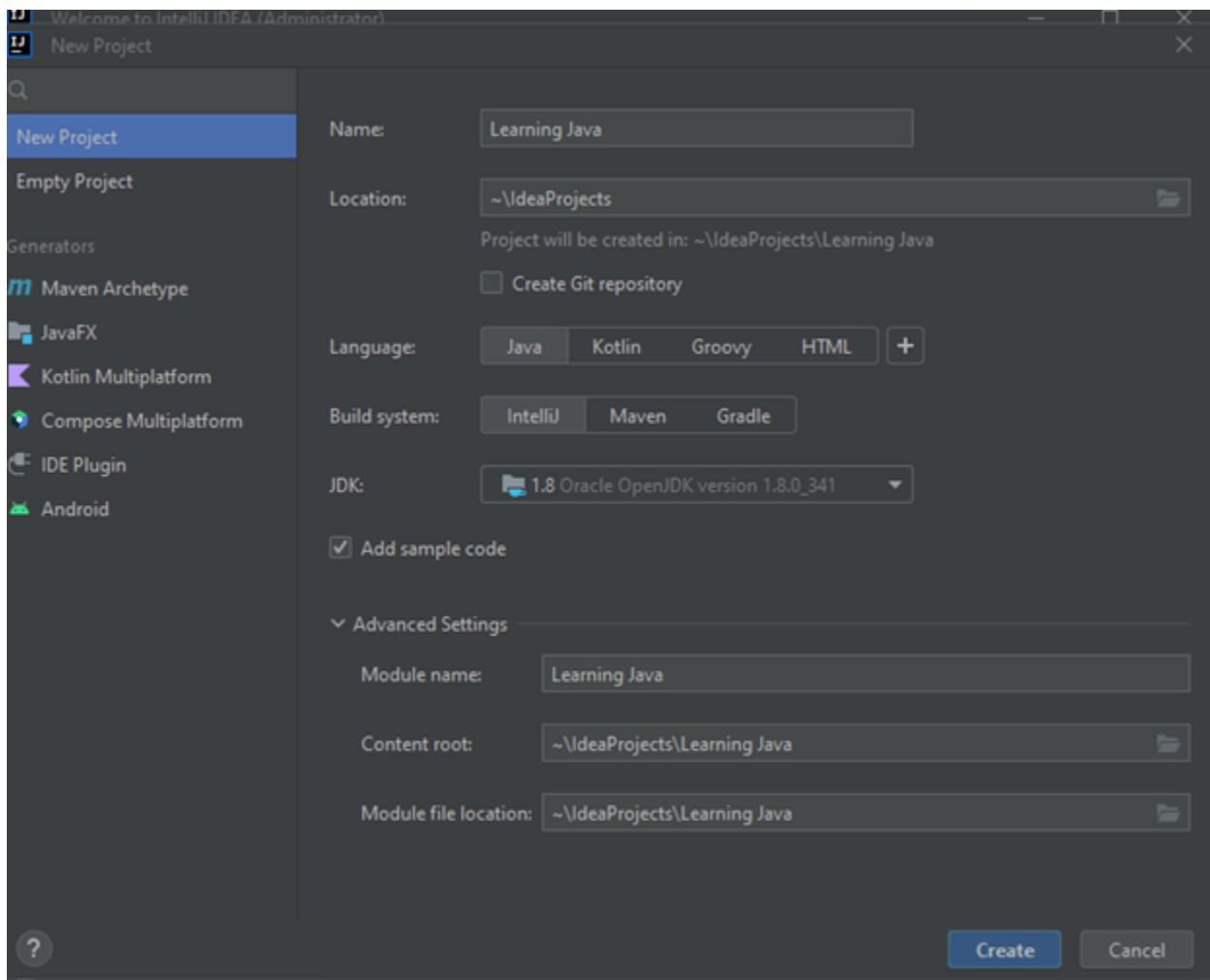
Now, running these two commands, javac and java, can be annoying. Especially if we are making lots of incremental changes to our code. There are tools that exist that can make our lives as developers a lot easier, and that actually combine the functionality of a text editor, that's our sublime, and the Java commands in the command line. We'll take a look at those next.

Exploring an integrated development environment (IDE)

Over time, programs get larger and larger. And although you can compile and run Java programs with the terminal, it's often faster to use an IDE. IDE stands for integrated development environment and an assembly of software that we can use to build and run our Java programs, just like the command line. The main difference is that this tool is built specifically for building and executing code and has a lot of built-in features specifically for that purpose. It also has a GUI, or a graphical user interface, meaning we won't have to type in commands. We can just click a button for our code to build and run. Ultimately, just like we use Photoshop to edit photos, we can use a special software, an IDE, to edit and run our code. There are many different IDEs, or integrated development environments, that you can write code with, but in this course, we'll be using IntelliJ, which is made by a company called JetBrains. The download can be found on the Java Basics CELL MS Teams channel. Now, this IDE has a paid ultimate version and a community free version. We'll use the community version for this course. Let's open our IntelliJ.

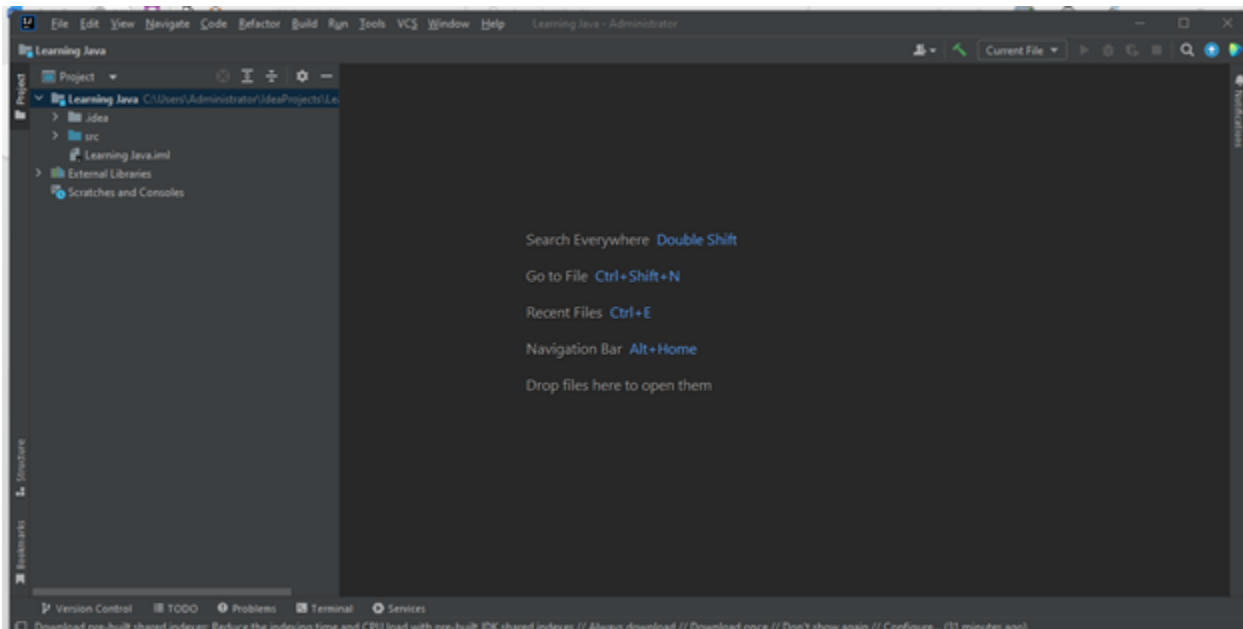


We will not import settings; we'll accept. Here, you can choose what type of UI you want for your IDE: do you want dark mode or light mode - dark mode. Every large-scale program you create should be in its own project. Since we are learning the fundamentals in this course, we will be using one project for the whole course. In the Project SDK setting, you should see your version of the JDK. Click next. For the project name, we are going to call this Learning Java.



For this example, it will be put in the learning-java folder for the course. However, you could put this wherever you'd like. Click Create. Now IntelliJ is configured.

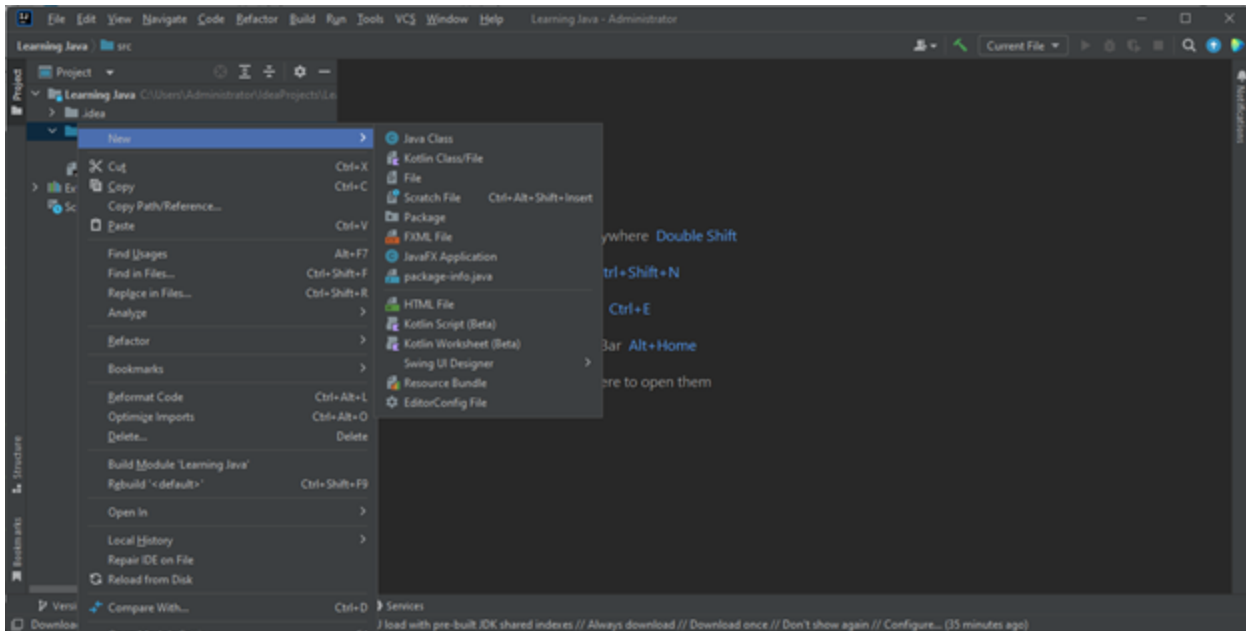
What, exactly, does this IDE have? On the left, we have the navigation pane. This allows us to navigate to different files in our project. The .idea folder holds metadata about our project, but the main code for our Java program will be in the SRC folder, or the source code folder. You'll also notice, in the external library section, our JDK is linked. We'll be working more with this navigation pane in the next lesson. At the top right-hand corner, we have some buttons that will allow us to run, debug, and step through our program.



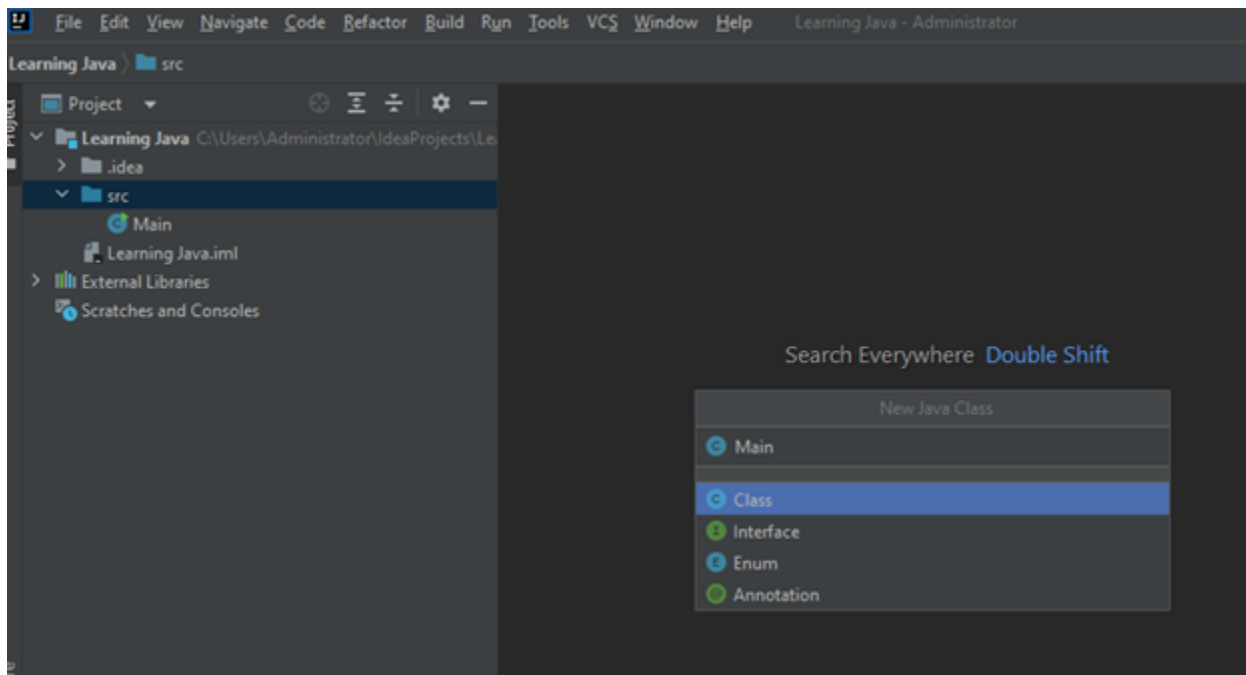
We'll learn more about what debugging and stepping through the program mean later on in the course, but this play button acts as the Java C and Java commands from the command line. Once we have a Java file created in the IDE, we'll be able to click this button and it will compile and run our program. Right now, the button is grayed out because we do not have a Java file with a main function configured in this IDE yet. The middle portion is also blank right now because we have not created and selected a Java file.

Hello World in an IDE

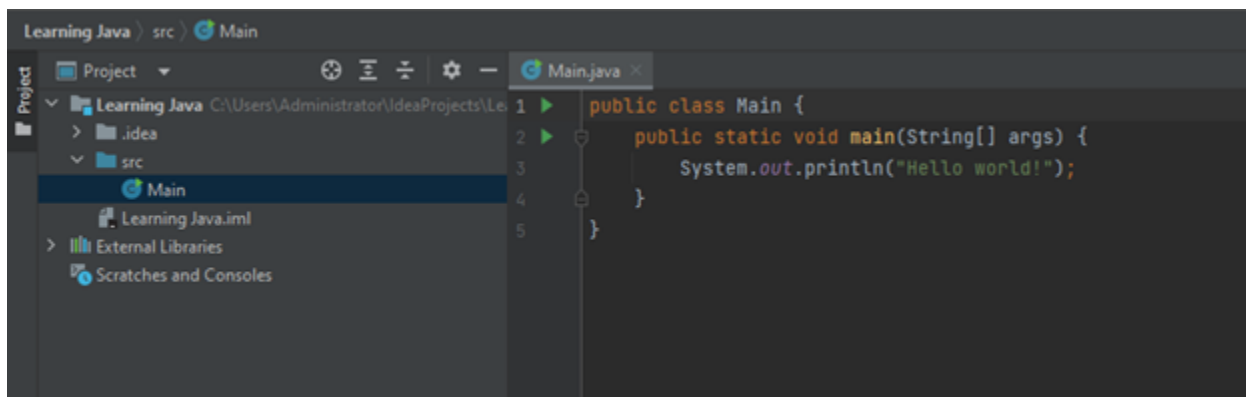
Let's create our first Java File in the IntelliJ IDE. We'll left-click the SRC, or Source Code Folder, hover over New and click Java Class.



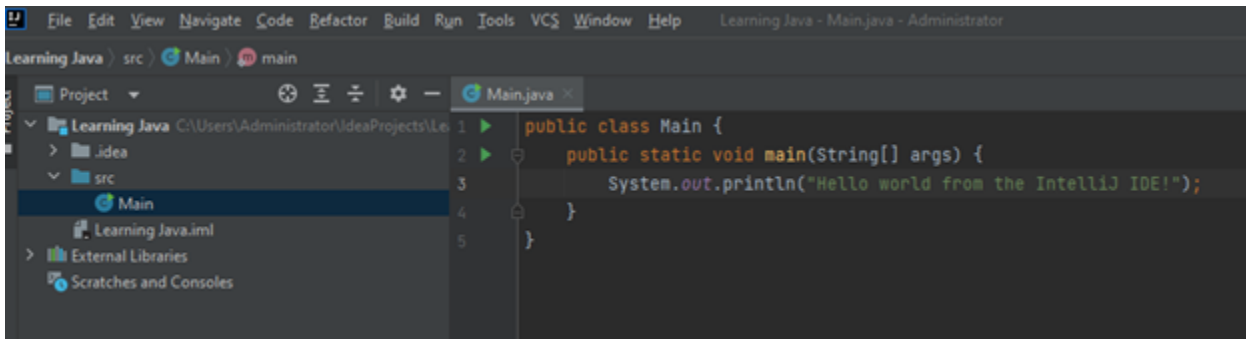
We'll call this class, Main. This is still creating a Java File, but it's pre-populating a class called Main.



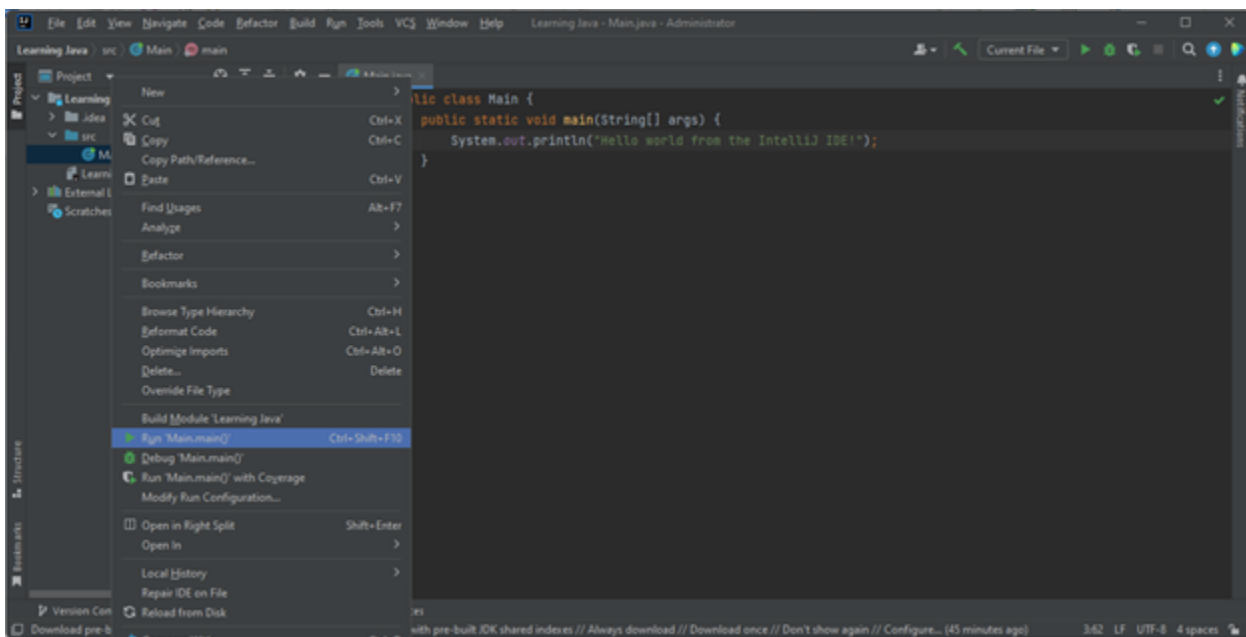
This middle area is the code editor, and it looks a lot like what we saw in Sublime. The code editor is where we can access the content of our Java Files. And in this case, the content of the Main.Java File. We can recreate some of the Java code we created before in Sublime. Instead of typing out that long name function with public static void, we can just type or double click on Main, and the IDE pre-populates it for us.



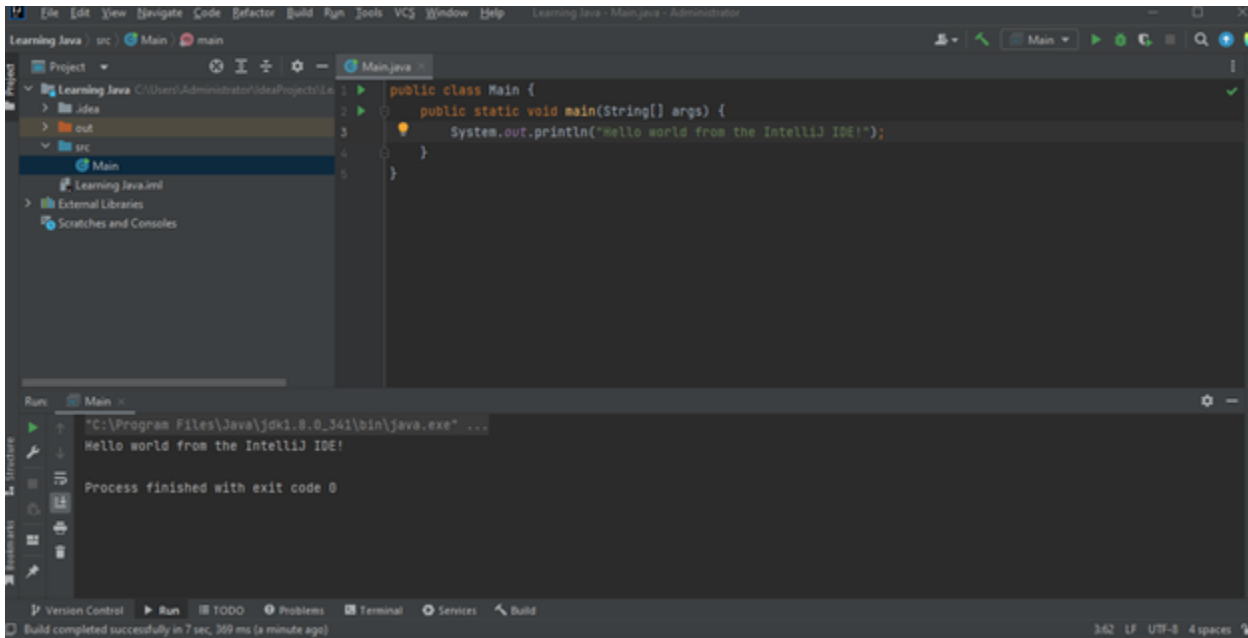
Remember when we talked about boilerplate code? Every Java program must have a Main function, so the IDE has shortcuts that make some of that boilerplate code easier and faster to write. This shortcut, where we type in a word and the IDE completes it for us, is called code completion, and this is one of the benefits to using an IDE. In this Main function, we'll output something to the user, just like we did before with Sublime and the command line. To output to the user, we can write: `System.out.println`, then whatever we want the system to output. In this case, we'll say, "Hello World from the IntelliJ IDE!" And then we'll do a parenthesis and semicolon.



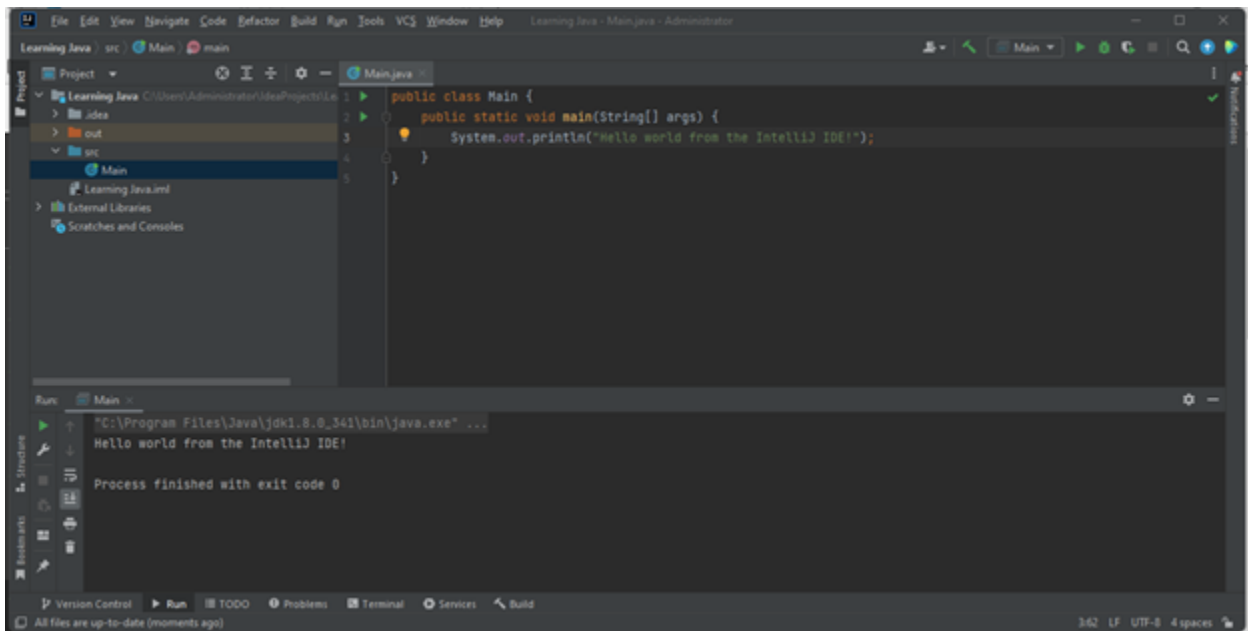
Now our program is ready to run. To run this file, we can left-click the Main class, hover over Run 'Main.main' and click it. Here, the first Main was the class Main, and the second Main was the Main function.



By selecting Run 'Main.main', the IDE compiles the program and runs it for us. There are no commands to type in, like before. We just click the button, and it runs. After running the program, you may notice a small window at the bottom of the screen containing our output. This window is called the Console and it's a text display where the executor of the program can see output messages. Here we see, Hello World from the IntelliJ IDE.



When we ran this program, we had to do two button clicks. We had to left-click the Main class and then select Run 'Main.main.' What if we could run the program with just one button click? At the top right-hand corner, click the play button. Our program was just run again.



Congratulations, you just created and ran a Java program using the IntelliJ IDE. What does this IDE do for us? It contains a code editor where we write our code. It also contains a debugger, which we will explore in later, and can use to figure out any errors in our program. You may have already noticed the bug button, next to the play button in the IDE. We will come back to that. An IDE also has other features that make writing code easier, like the code completion we saw earlier. Although an IDE is not necessary to write code, it can be very helpful because it has extra tools that make developing code a little faster and more straight forward.

Demo

Schedule a 1x1 demo with your group's TA. If you are unsure of who your TA is, post in the CELL - Java Basics MS Teams channel.

Next Steps

- Schedule a demo with a Topic Advisor (TA).
- Check your CELL group schedule in [SharePoint](#)
- Continue to collaborate with others and be active in MS Teams.
- If you noticed any errors on this page, please let the CELL Program Team know by sending an email to the Technology Learning Team mailbox: g31488@att.com
- Navigate to Iteration 2 and continue your CELL journey.

Previous Iteration	tWiki Home	Next Iteration
Iteration 0	tWiki Home Page	Iteration 2

Previous Iteration	Wiki Home	Forums	Next Iteration
Iteration 0	Wiki Home Page	Forums	Iteration 2