

# Java Basics Unit

## Lesson 2: IDE Installation and Basic Operation

Copyright © 2016 The Learning House, Inc.

All rights reserved. No part of these materials may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of The Learning House. For permission requests, write to The Learning House, addressed "Attention: Permissions Coordinator," at the address below.

The Learning House

427 S 4<sup>th</sup> Street #300

Louisville KY 40202

## Lesson 2: IDE Installation and Basic Operation

### Overview

---

In this lesson, we will take a look at the important features of Integrated Development Environments: text editor, debugger, and build system. We will download and install the NetBeans IDE and create, compile, run, and debug 'Hello, World!' in the IDE.

### IDE Features

---

IDEs have three main features: code editor, debugger, and build/compilation automation.

#### Code Editor

The first of these, the code editor, is fairly self-explanatory — it is the part of the IDE that allows you to enter and edit Java source code. The editor in an IDE has a couple of important features that make your life, as a developer, much better:

1. Syntax highlighting
2. Code completion/Intellisense
3. Error highlighting and correction hints

**Syntax highlighting** makes code easier to read by displaying keywords and language features in different colors, as in this example:

```
package com.swcguild.hello;

public class Hello {

    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

*Listing 1: Syntax Highlighting*

**Code completion and Intellisense** are features of the IDE that give you hints and improve coding speed by autocompleting some coding constructs and giving you hints for others, as in the following example (don't worry if this doesn't make sense right now; you'll get plenty of experience with this feature throughout the course):

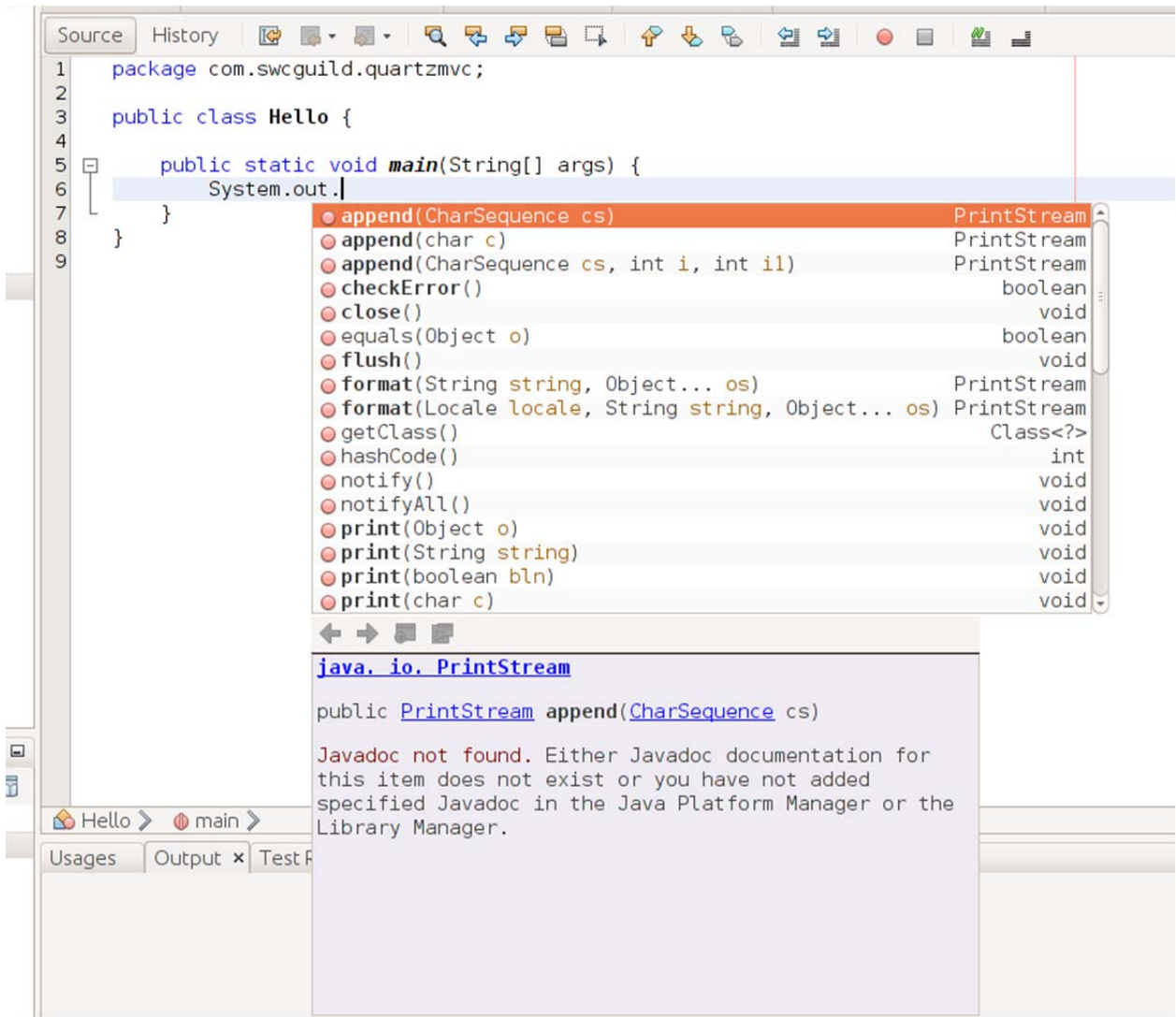


Figure 1: Code Completion and Intellisense

**Error highlighting and correction hints** are features similar to code completion and Intellisense but, rather than give you hints for coding constructs, these features highlight errors and give you hints as to how you might solve them, as in this example:

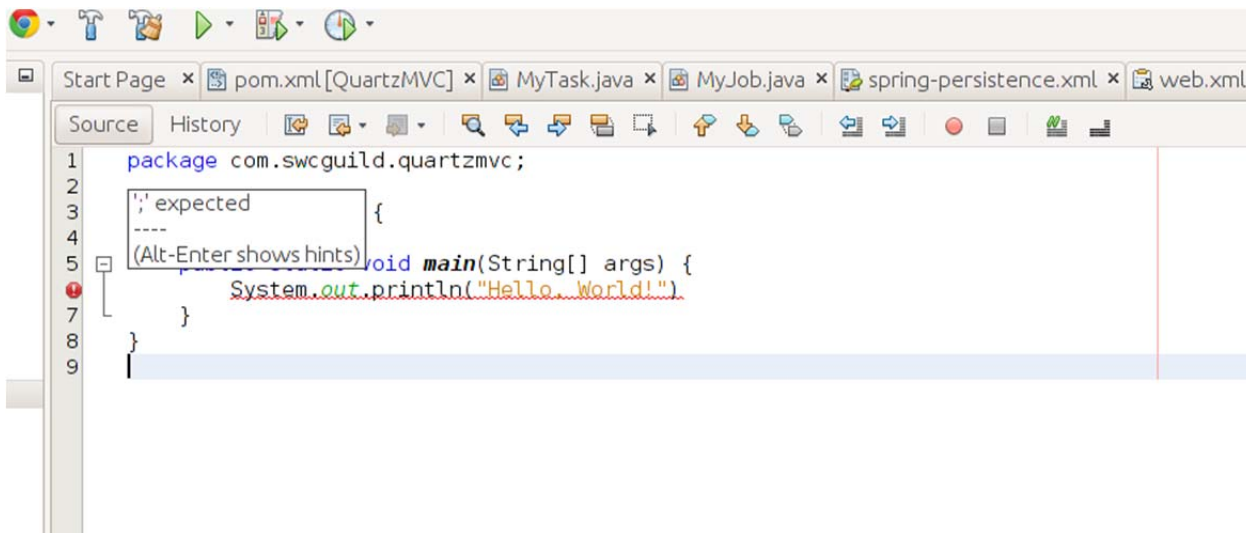


Figure 2: Error Highlighting and Correction Hints

## Debugger

The debugger is an IDE feature that allows you to step through your code line-by-line while the program is running. This allows you to see exactly which code paths your program is taking and the values of all of the variables as the program runs. As the feature's name implies, this is extremely useful when tracking down and fixing bugs.

## Build/Compilation Automation

IDEs also make it easy to build, deploy, and run your programs. When using an IDE, it is not necessary to run the Java compiler or execute your program by hand. In most cases, running the program from within the IDE will automatically recompile the program to ensure that you are running the latest version of your code.

## Download and Install NetBeans

Now that we know a little bit about IDEs, let's download the IDE that we'll be using throughout the course, NetBeans:

1. Go to [netbeans.org](https://netbeans.org) and click on the Download button for NetBeans 8.0.
2. On the "NetBeans IDE 8.0 Download" page, click on the Download button for the Java EE Download Bundle. This will download a file called `netbeans-8.0-javaee-linux.sh`.
3. Open a terminal and `cd` into the directory into which the Netbeans file was downloaded. For example, if your file was downloaded into the "Downloads" directory, you would type:  
**`cd Downloads`**.
4. In order to run the installer, we have to make this file executable. Type in the following command (this allows you — the user — to execute this shell script):  
**`chmod u+x netbeans-8.0-javaee-linux.sh`**

5. Now we will run the installer. Type in the following command:  
**`./netbeans-8.0-javaee-linux.sh`**
6. The NetBeans installer will start. On the first screen, make sure the both GlassFish and Apache Tomcat are **unchecked** (we will install Tomcat later in the course).
7. On the second screen, accept the terms of the license agreement.
8. On the third screen, accept the license terms for JUnit.
9. On the fourth screen, use the default location for the install directory and make sure that NetBeans has Java 8 selected for the JDK.
10. On the fifth screen, leave “Check for Updates” checked and click Install.
11. After installation, you will be prompted as to whether or not you would like to provide anonymous data back to NetBeans; uncheck the box if you do not wish to share your information and click Finish.

NetBeans is now on your system and is ready for use. You now have a “NetBeans IDE 8.0” shortcut on your desktop. Double-click the shortcut and verify that NetBeans starts properly. NetBeans should come up and display its Start Page. Congratulations! You have installed an IDE.

## Hello, World! NetBeans

---

Now that we have NetBeans installed, let’s see how our Hello, World program from Lesson 01 looks in NetBeans by following these steps:

1. Create a new NetBeans project
2. Type in the Java code and save it using NetBeans
3. Compile the code in NetBeans
4. Run the program
5. Step through the program with the Debugger

## Create a NetBeans Project

Select the **File** → **New Project...** menu item in NetBeans to show the New Project wizard. Select “Java” under Categories and “Java Application” under Projects, as shown in Figure 3, and click Next:

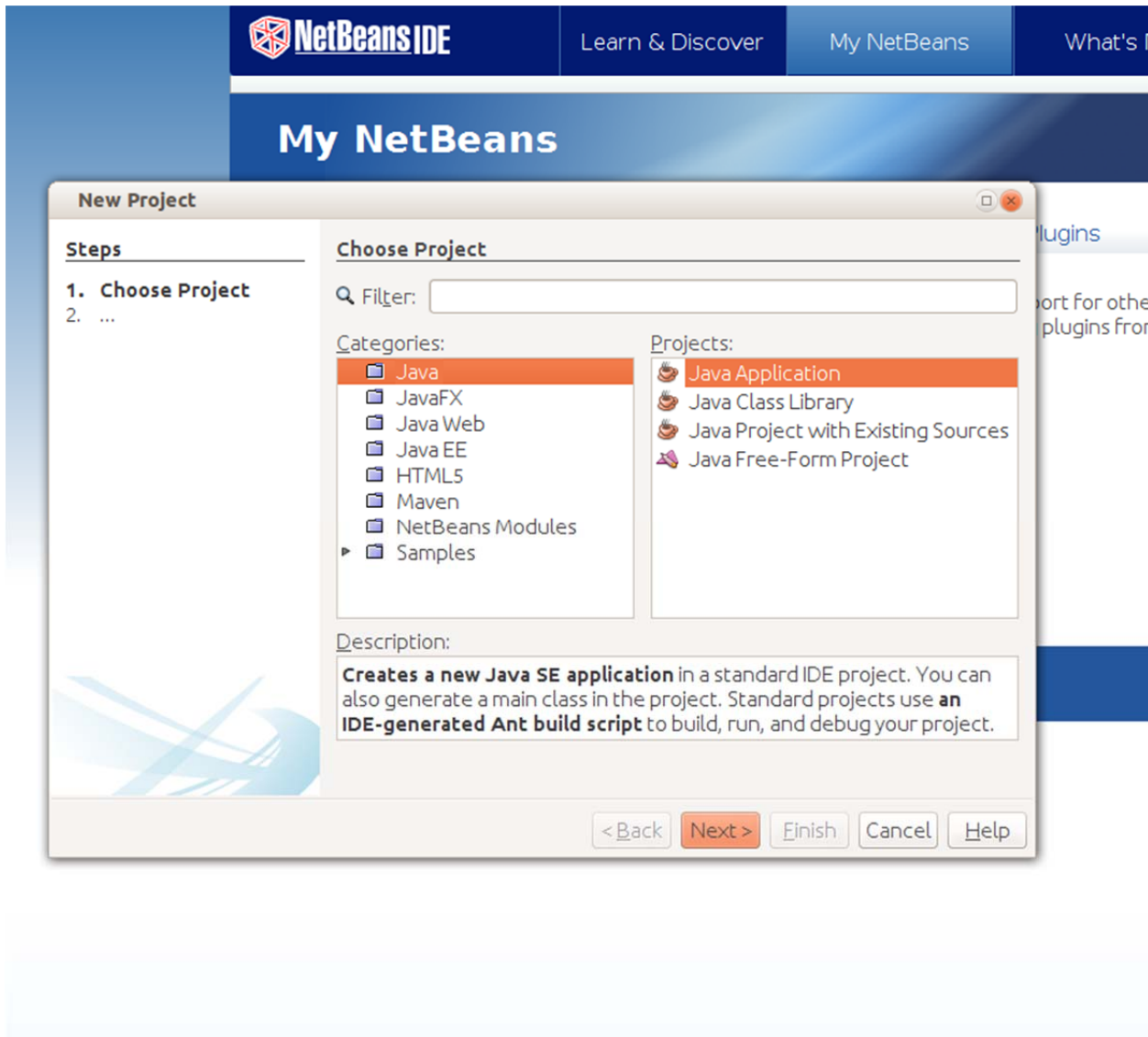
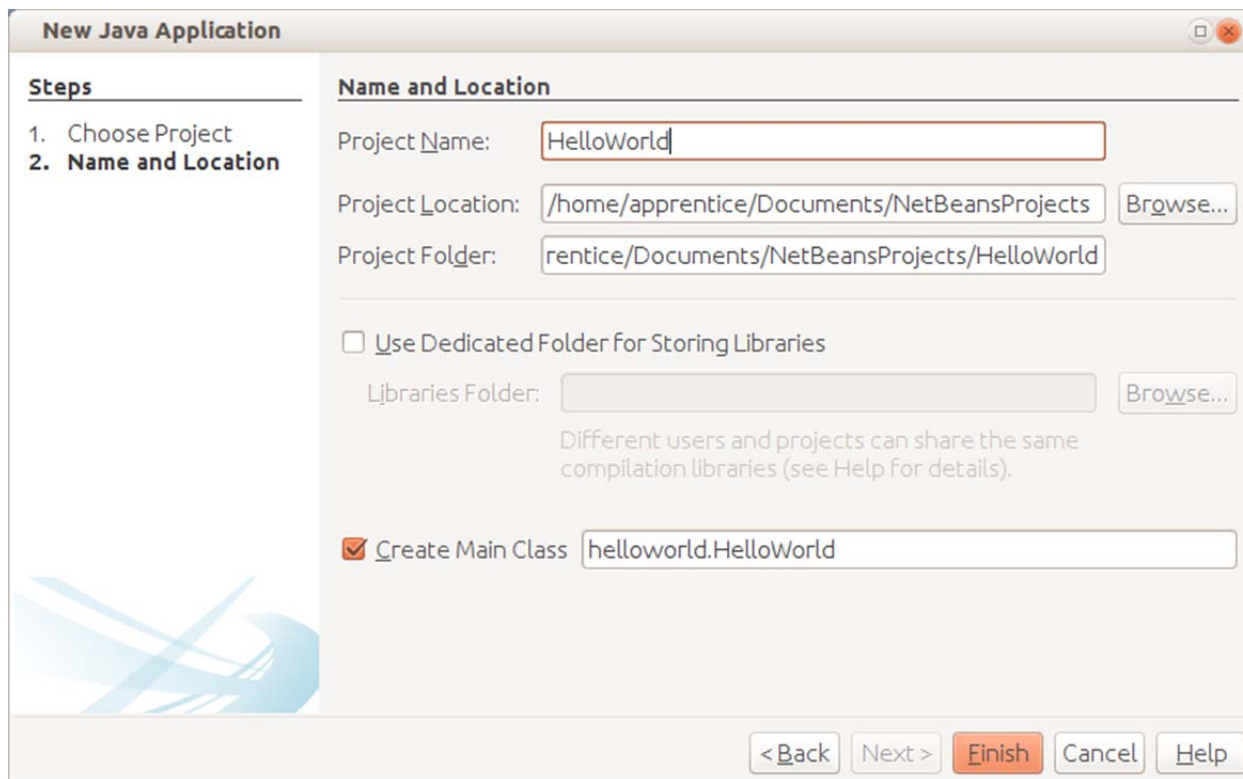


Figure 3: New Project Wizard

In Step 2 (Name and Location) of the New Project Wizard, type in “HelloWorld” for the Project Name and click Finish, as shown in Figure 4:



**New Java Application**

**Steps**

1. Choose Project
- 2. Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

*Figure 4: Name and Location*



NetBeans will now create your project and display HelloWorld.java. Your screen should look like Figure 5:

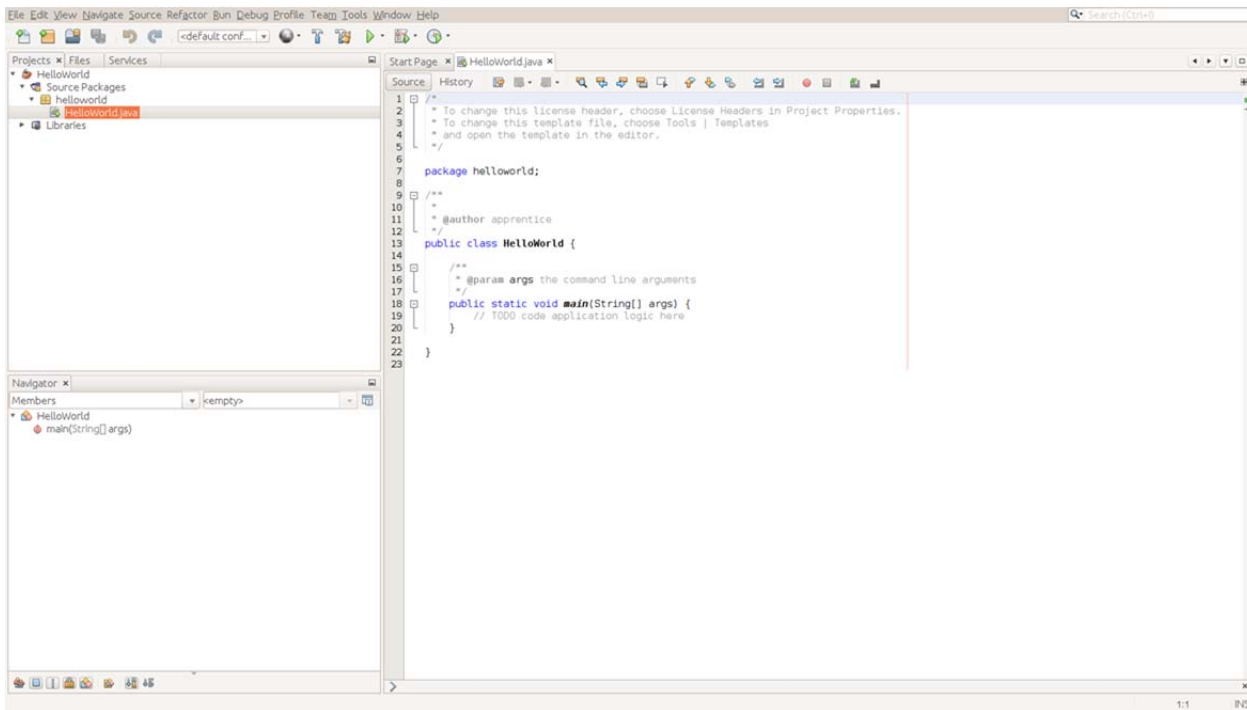


Figure 5: HelloWorld.java

## Type in Java Code

You'll notice at this point that NetBeans has already generated most of the code we need for our Hello World! application. We already have our class and our main method — in Lesson 01, we had to do that all by hand. The only thing we have to add is the magic code that sends things to the console:

`System.out.println("Hello, World!");` (review Lesson 01 if you don't remember this). Update your `HelloWorld.java` file so that it looks like Listing 2:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworld;

/**
 *
 * @author apprentice
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

*Listing 2: HelloWorld.java*

You will notice that this looks a lot like the Hello, World! program we wrote in Lesson 01 with exception of the following lines:

```
package helloworld;

...
/**
 *
 * @author apprentice
 */
...

/**
 * @param args the command line arguments
 */
```

The first difference is the **package** statement. Packages are just folders into which we organize our code. The second and third differences are **comments**, and specifically these are **javadoc** comments. The Java platform has a special tool that will generate documentation for your code based on these javadoc comments. Don't worry about these things right now; we'll learn about them later in the course. For now, we'll concentrate on the **main** method.

## Compile the Code in NetBeans

Our next step is to compile our program. In NetBeans, we can do this by clicking the “Build” button — it is the blue handled hammer icon, as seen in Figure 6. This runs the Java compiler in the background for us. The “Clean and Build” button is just to the right of the Build button (it’s the one with the broom and hammer). Clean and Build also compiles the program but it deletes all of the compiled .class files before it builds. You can see the output and status of the build in the Output window. To open the Output window, click the Window → Output menu item (or hit Ctrl-4), and the Output window will appear underneath HelloWorld.java.

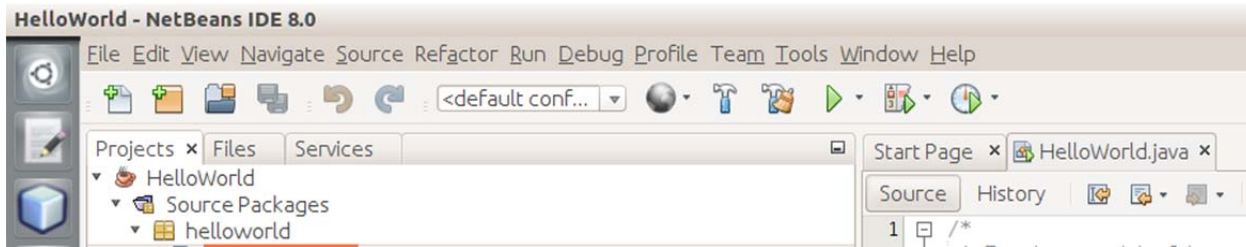


Figure 6: Build, Run, and Debugger Buttons

## Run the Program

Now that our program is compiled, we can run it. In NetBeans, we can do this by clicking the “Run Project” button — it is the green arrow icon, as seen in Figure 6. Click the Run Project button now, and your Output window should match what is seen in Figure 7:

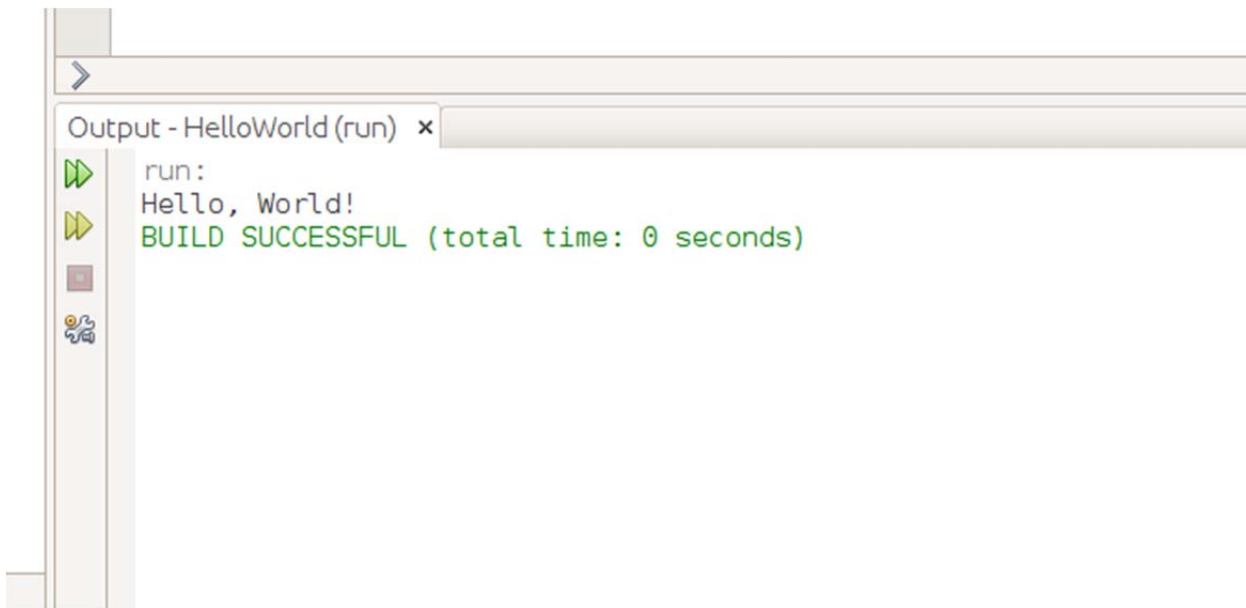


Figure 7: Output Window

## Step Through the Program with the Debugger

Finally, we’ll use the Debugger to step through our program. As mentioned above, the Debugger allows us to step through a program statement by statement and examine the program’s state while it is running. In order to do that, we have to set a **breakpoint**. A breakpoint is simply a place (it must be an executable statement) in the code where we want the Debugger to stop. When the statement associated with the breakpoint is

executed, the Debugger will pause execution of our program so we can take a look at what is going on. Don't worry too much about the intricacies of the Debugger right now — we'll take a much deeper look later on in the course.

Our program only has one executable statement (more on that later): `System.out.println("Hello, World!");` so that is our only option for setting the breakpoint. To set a breakpoint, simply click on the grey left margin of the code editing window and a pink box will appear, as in Figure 8:

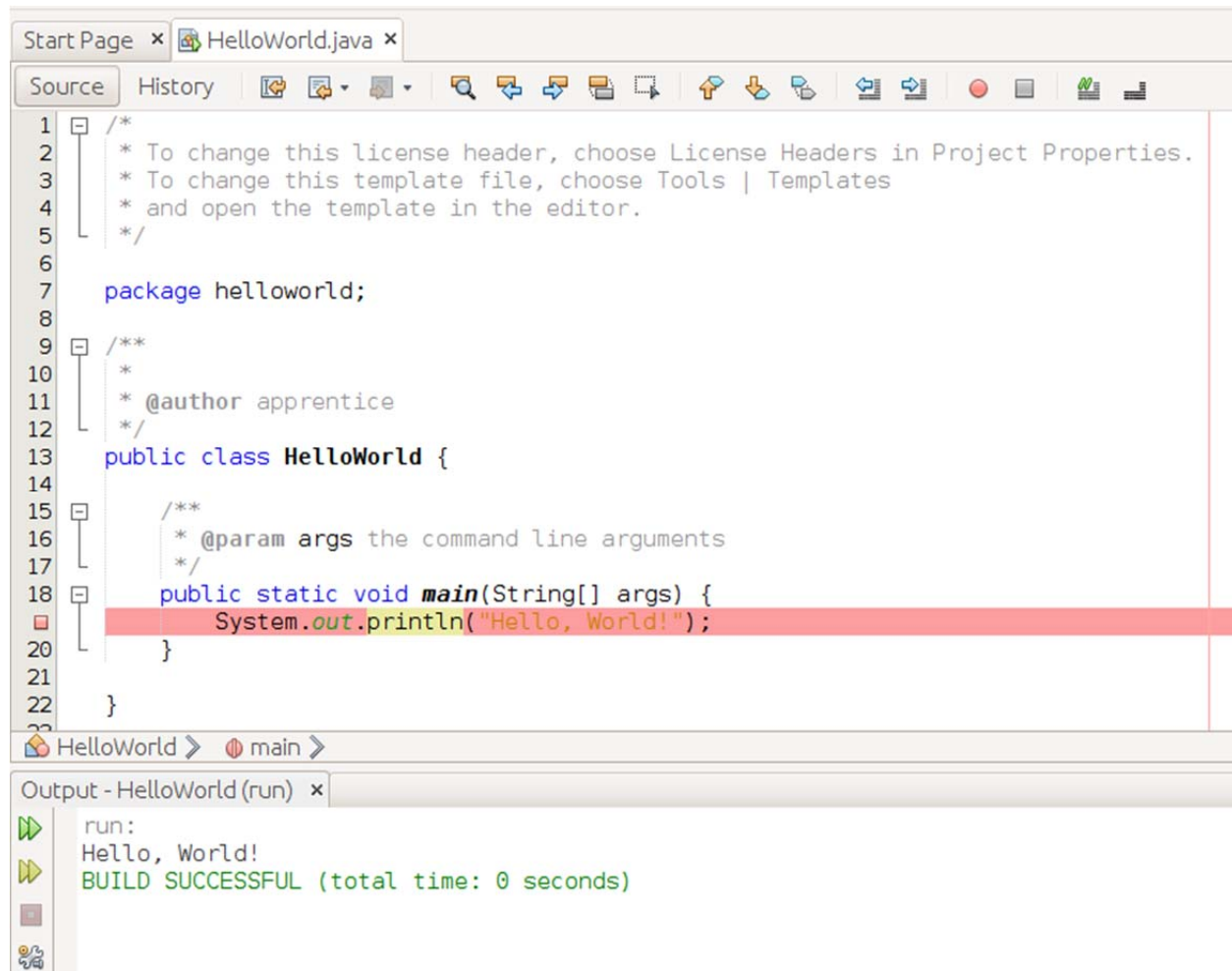
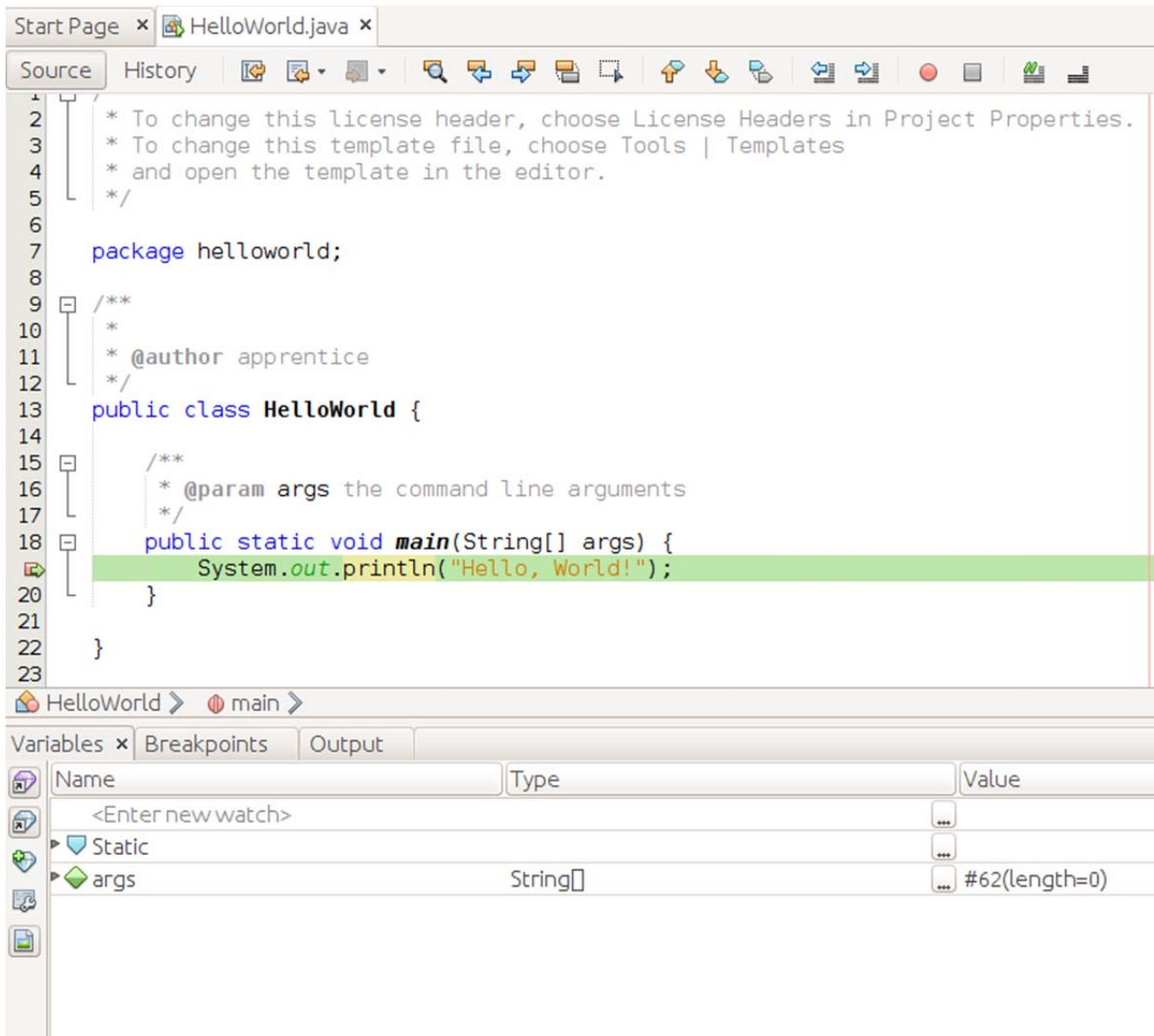


Figure 8: Breakpoint

Now that the breakpoint is set, we will run our program in **debug mode** by clicking on the “Debug Project” button which is just to the right of the “Run Project” button (green arrow) in Figure 6 above. After you click the Debug Project button, the program will begin to execute and then will pause at our breakpoint. The breakpoint line will turn from pink to green and the execution will pause as in Figure 9:



*Figure 9: Execution Paused at Breakpoint*

As you can see in Figure 9, the “Variables” window should automatically display below the code window. If this does not happen automatically, click on the Window → Debugging → Variables menu option. The variables window shows us the state of all of the variables in the program. The only variable we have is the array parameter args which, as you can see in Figure 9, has a length of zero (which means it has no values associated with it). Again, don’t worry too much at this point about what all this means — we’re just taking a tour of the IDE right now and we’ll dive deep into variables and values later on.

Now that we’ve paused execution of our program and had a look around, we want to complete execution of the program. To do this, simply press F5 and our program will run to completion.

## Wrap-up

---

Here's what we covered in this lesson:

1. We discussed and explored the main features of an IDE: code editor, debugger, and build automation
2. We downloaded and installed the NetBean IDE
3. We created, compiled, ran and debugged Hello, World! in NetBeans
4. We looked at how error messages and output are displayed in NetBeans
5. We looked at basic debugger operations in NetBeans

In the next lesson, we will dive into Java by looking at programs, statements, and variables.