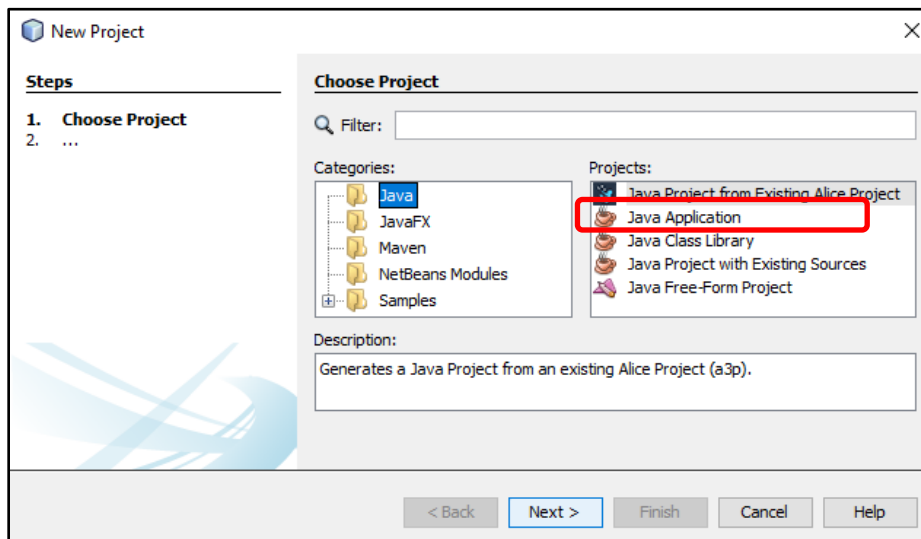


Compiling and Executing a Java Program Exercise

1. Open up the **NetBeans** environment.
2. You can close the **Start Page**. The tutorials provided in the Start Page can be confusing for a first timer.



3. Select the **File** menu and then choose **New Project**. Then choose **Java Application** as shown below.

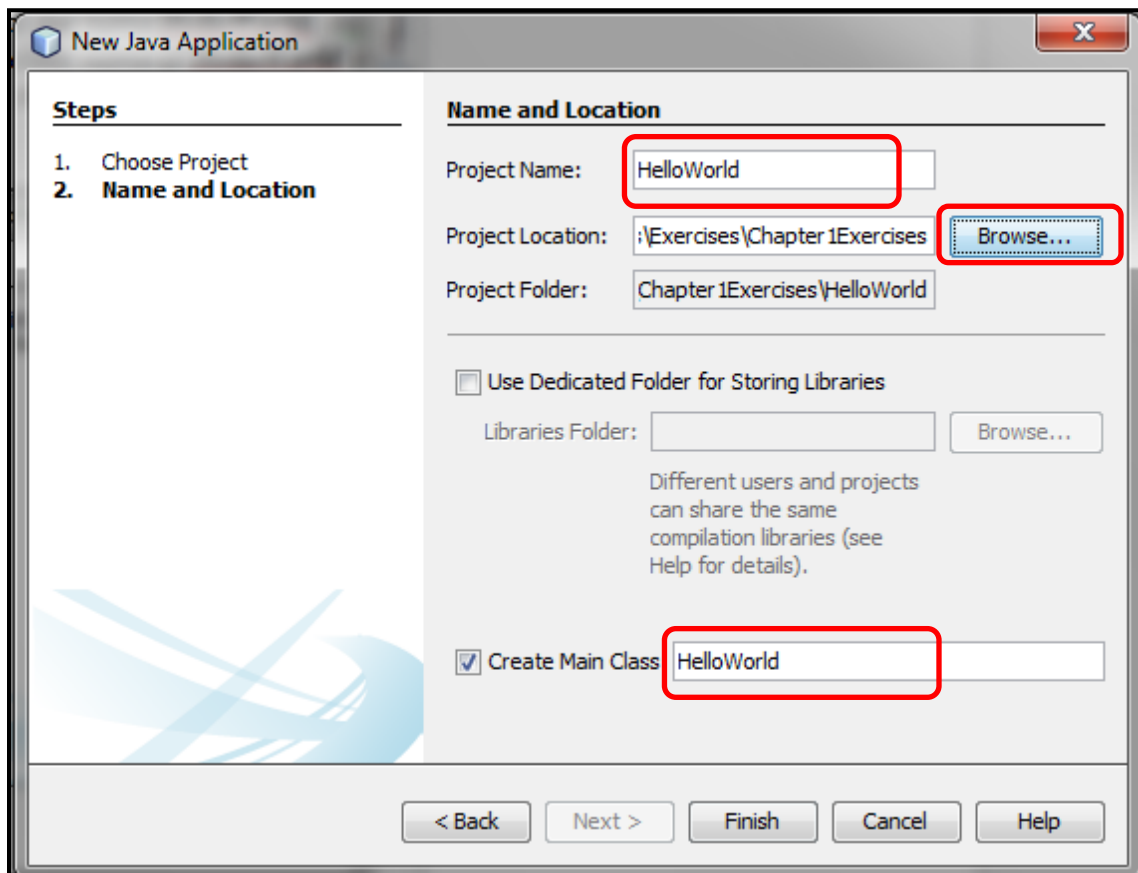


- Click **Next**. Name your NetBeans project, select the **location** of where you would like to save your file, give your file (*Main Class*) a name (*make this name the same as your project name*), and click **finish**. Although it is not necessary, we are going to name our projects and Java files (*main class*) have the same name. Therefore, make sure that the top and bottom boxes have the same name. NetBeans automatically will try to name your file (main class) helloworld.HelloWorld. Erase the **helloworld.** that NetBeans inserts before your file name. Make sure it looks like the following screenshot. Capitalization is important.

*Project the name **HelloWorld** (no spaces)*

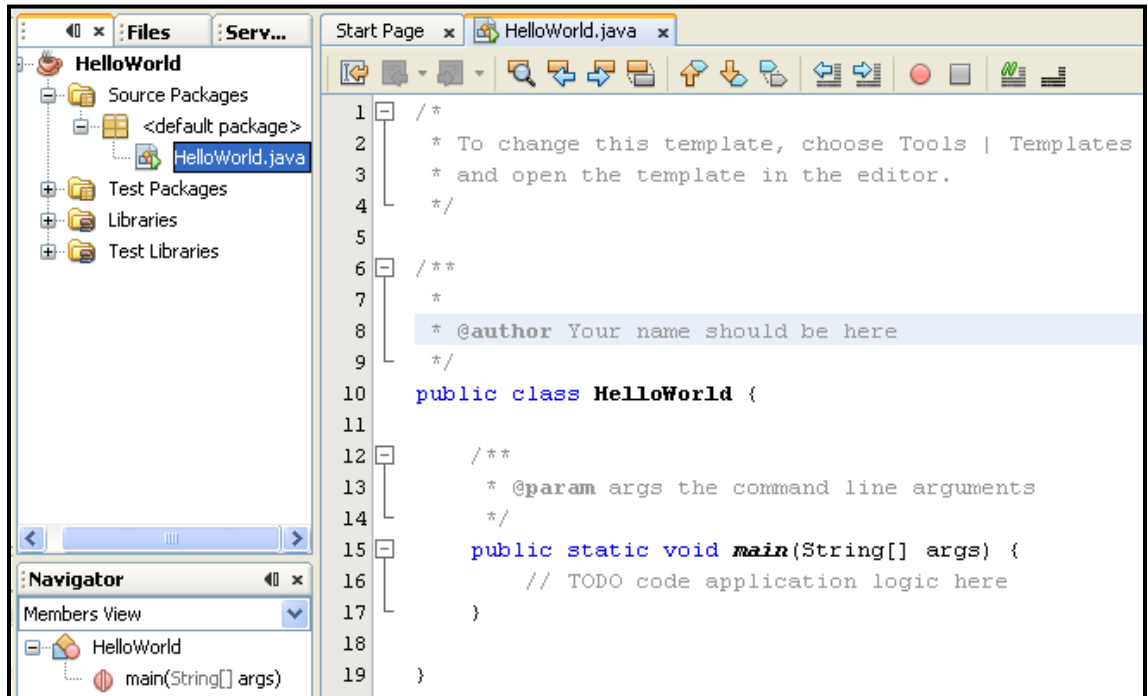
*Main class name of **HelloWorld** (no spaces)*

*Select the **Location** of where you would like to save your NetBeans project*

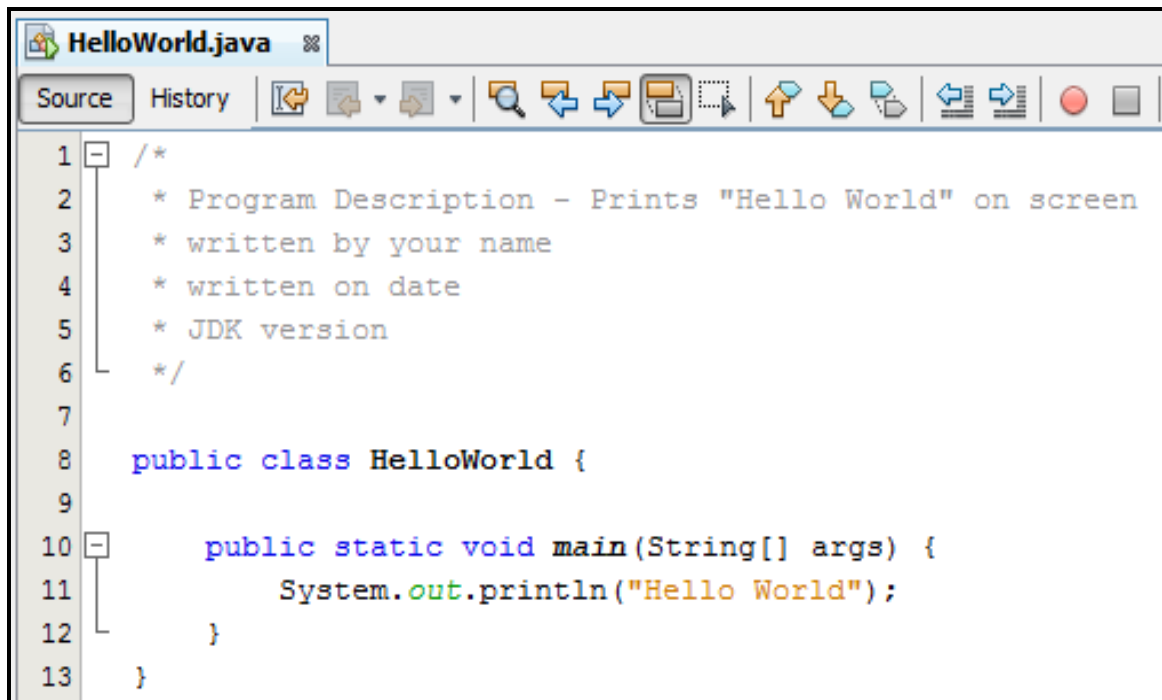


- If line numbers are not showing, click **View** from the menu, then **Show Line Numbers**.

6. Your project should look as follows. HelloWorld.java is the file that we will be working with. (Note: If your code has a package statement on line 5, then exit NetBeans and delete the project folder and do step 4 again. Make sure it looks like the screen shot provided. Look carefully at the textbox next to the Create Main Class label in the New Java Application dialog. For your information a package in Java is a folder. The package line indicates that you put your file in a folder when you created the project.)



7. Type in the following Java program. You will need to delete some comment lines and add some lines. Be careful to make your program look exactly as shown below. Try to keep your statements on the same lines as those shown below and also try to use the same approximate indentation to make your program more understandable. Change line 3 to be your name, line 4 to be today's date, and line 5 to be the JDK version that you are using (You can check to see what version of the JDK that you have by clicking **Help** from the menu and then **About** in the NetBeans environment. It will have **Java:** and then a number, this is your JDK version. You do not need the number after the underscore).



```
1  /*
2     * Program Description - Prints "Hello World" on screen
3     * written by your name
4     * written on date
5     * JDK version
6     */
7
8  public class HelloWorld {
9
10     public static void main(String[] args) {
11         System.out.println("Hello World");
12     }
13 }
```

What does the above program do? You will not understand everything about this program YET. However, here is a brief explanation line by line:

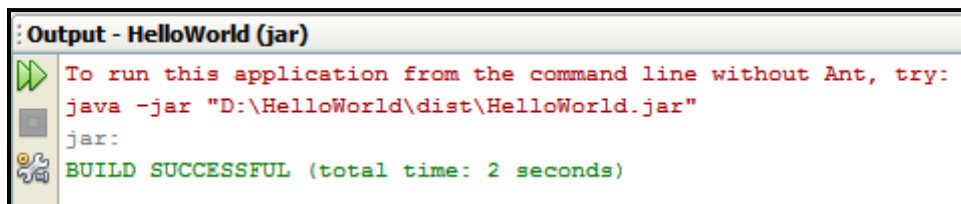
- 1-6) Lines 1-6 are known as comments. Comments are used to document code so that other people reading our code can understand our logic. Comments are useful for adding extra information to our programs that we don't necessarily want to show up in the output of our program such as: author, date, JDK used, program description, etc. Also, it is a good idea to comment your programs extensively when you are just starting out so that you have well-documented examples. This is a multi-line comment which is represented by a `/*` at beginning of comment and `*/` to end the multi-line comment.
- 7) Blank line for readability purposes. Does nothing. (not necessary)
- 8) States this will be a public program called HelloWorld. Class names should begin with a capital letter. Be careful of capitalization in Java programs.
- 9) Blank line for readability purposes. Does nothing. (not necessary)
- 10) This is the main method declaration in this Java application. Every Java application must contain a main method and it must always be public static. The arguments for a method always appear in parentheses. In this case, the argument is a String array called args. The variable name of args can be whatever the programmer wants it to be, but most programmers use the variable name of args. The square brackets appearing after the word String are found to the right of the "P" key on keyboard.
- 11) The statement of **`System.out.println("Hello World");`** prints Hello World to the screen and positions the insertion point on the next line. *System* is a Java class in the library and the *out* object is the screen. Java is case sensitive so be careful of capitalization. Java uses a punctuation method of class-dot-object-dot-method syntax. All methods have arguments in parenthesis which is a way of telling a

method from a variable. This println method has an argument of a literal string of "Hello World". All Java statement lines will end with a semicolon.

Note: The next to last character in "println" is a lowercase L, not the number 1.

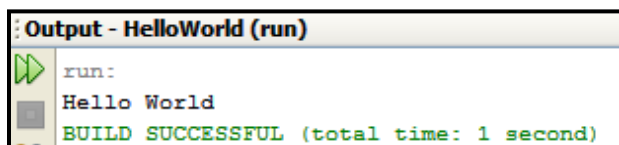
- 12) A right curly brace ends the main method. It is important to balance all your left and right curly braces and left and right parentheses in all Java programs. The right curly brace is found 2 keys to the right of the P key on keyboard.
- 13) A right curly brace to end the program.

- 8. This Java program needs compiled. Compiling a program will have the computer look at each line of your program for syntax errors such as typos, misspelling, etc. To compile your program, click on **Run** from the File menu, then **Build Project**. The compiler will check this file for syntax errors and let you know on what lines you made errors. Errors (along with line numbers of errors) will list in bottom panel of the screen. If you have errors, correct your typos on the top of the screen and compile again. Make sure you adjust the bottom output panel large enough to see your errors and your output.
- 9. If there are no compilation errors (denoted by the words BUILD SUCCESSFUL), the compiler will convert this Java program into a bytecode file called **HelloWorld.class**. This bytecode file is a generic file that may be used on any operating system. This file is located under the **project** folder, under the **build** folder, and in the **classes** folder.



```
Output - HelloWorld (jar)
To run this application from the command line without Ant, try:
java -jar "D:\HelloWorld\dist\HelloWorld.jar"
jar:
BUILD SUCCESSFUL (total time: 2 seconds)
```

- 10. Once compiled and you have a bytecode file (.class file extension), you are ready to have the Java interpreter execute your Java program. To execute your first Java program, you will click on **Run** from the NetBeans menu, then **Run Project**. "Hello World" should be displayed in the output window as shown below:

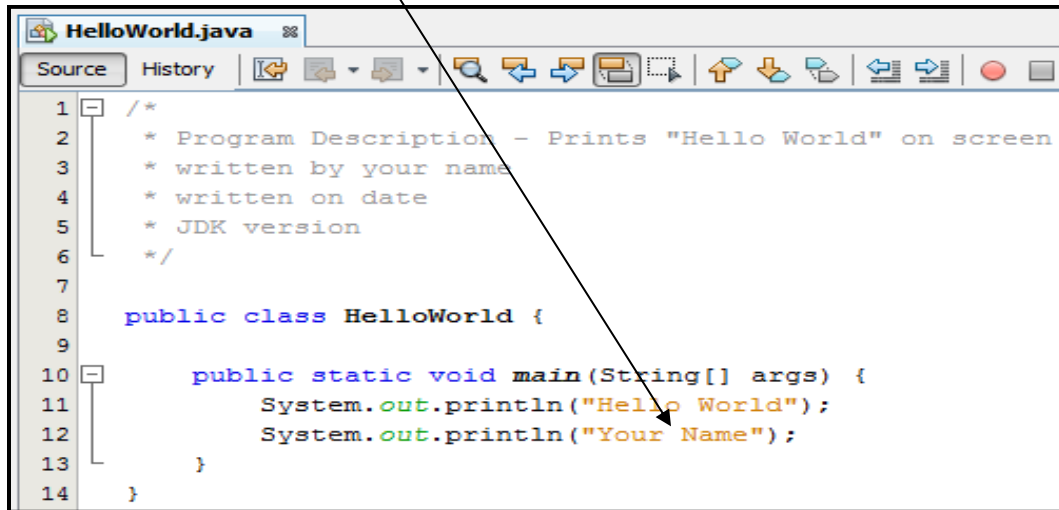


```
Output - HelloWorld (run)
run:
Hello World
BUILD SUCCESSFUL (total time: 1 second)
```

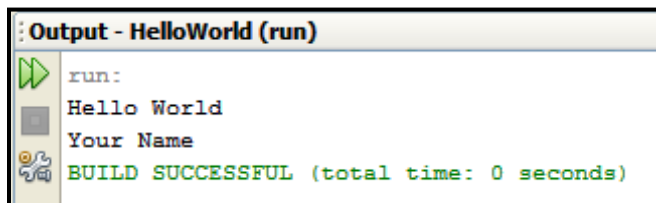
- 11. The process you have seen so far is typing a Java program into NetBeans, compiling a Java program, and executing the Java program. This is the process that you will be going through over and over as you progress through Java. The output that you have at bottom of screen is simply the computer displaying the words "Hello World".

12. Now, let's adjust the Java program. Add the following line as shown in the diagram below. (Note: if you type *sout* and hit the **tab** key, it will type the *System.out.println("")*; line for you.)

System.out.println("Your name");

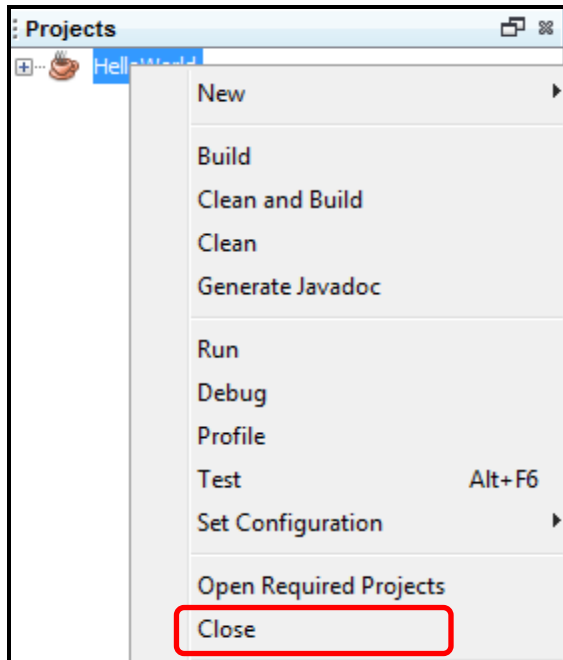


13. Now, save the new version of the program by clicking on **Save** from the File menu. (**DO NOT** click “Save As” and save this outside the project folder. NetBeans has a file structure and you cannot pull your files out of this structure or else NetBeans will not open them in the future. Compile the program (**Run** menu and then **Build Project**).
14. Execute the program (**Run** menu and then **Run Project**). Your display window should look similar to the following:

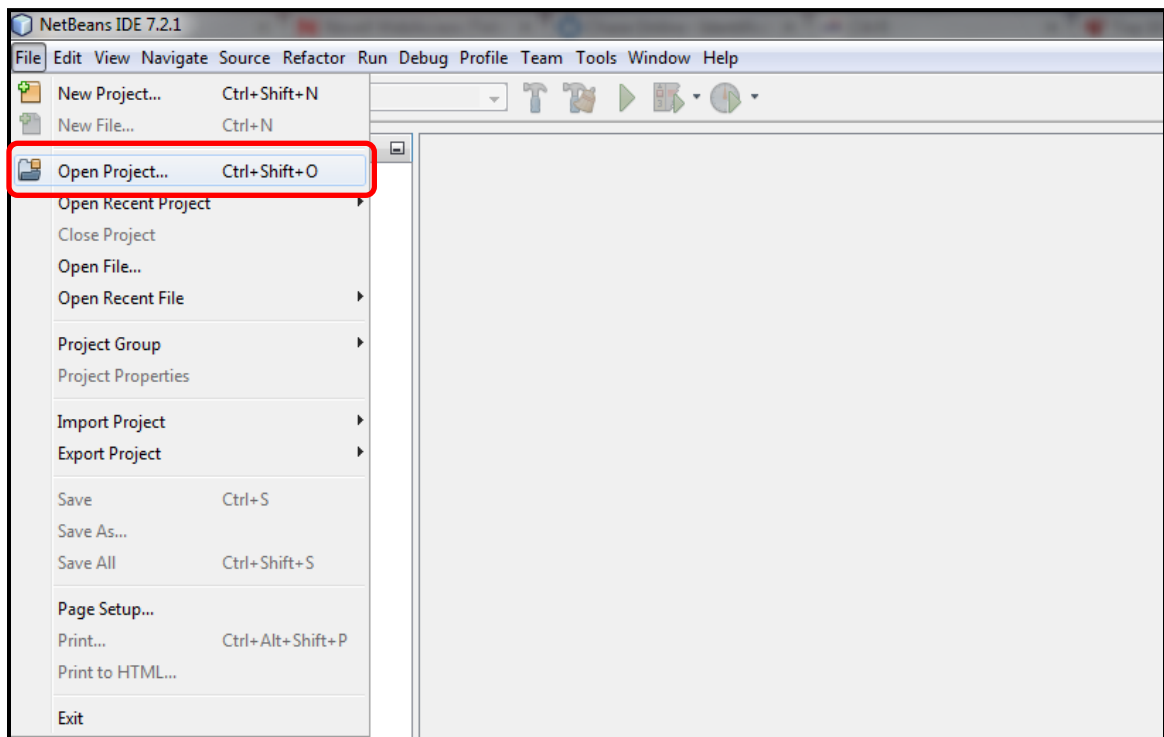


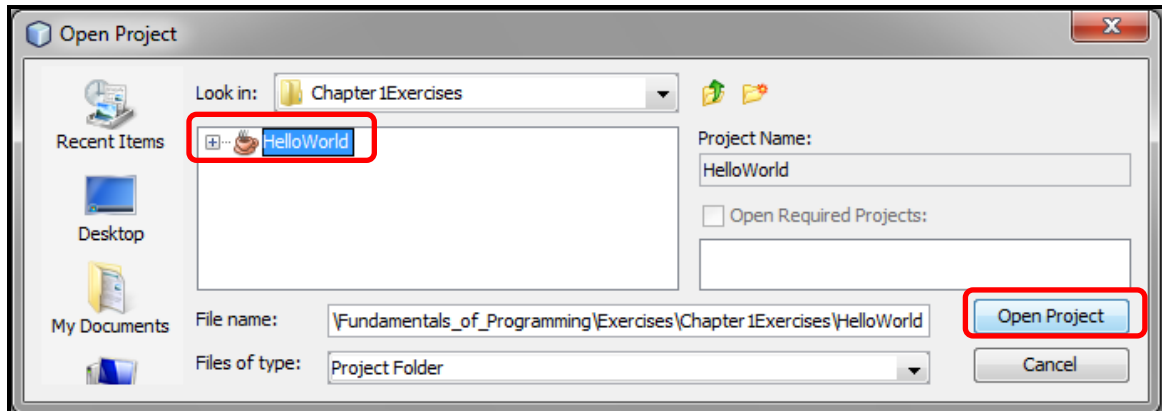
Note: If you are getting compiler errors at bottom of screen, please double check the capitalization, spelling, and punctuation.

15. To ensure that your code indentation is correct, you should always choose **Source** from the menu, then **Format**. Make sure that you compile your program (Run menu, Build Project) and run it (Run menu, Run Project).
16. Close the project by right clicking on the project on the left pane and choosing **Close**.

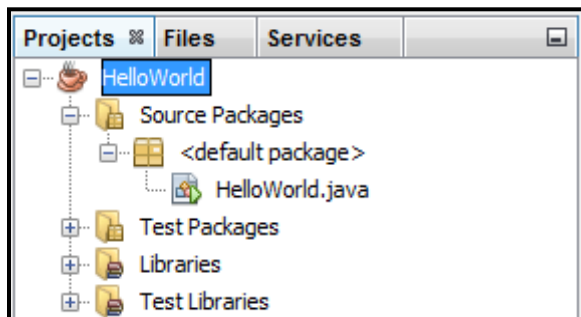


17. Choose **File, Open Project...**, select the **HelloWorld** NetBeans project (should have a coffee cup next to your project), and click **Open Project**.

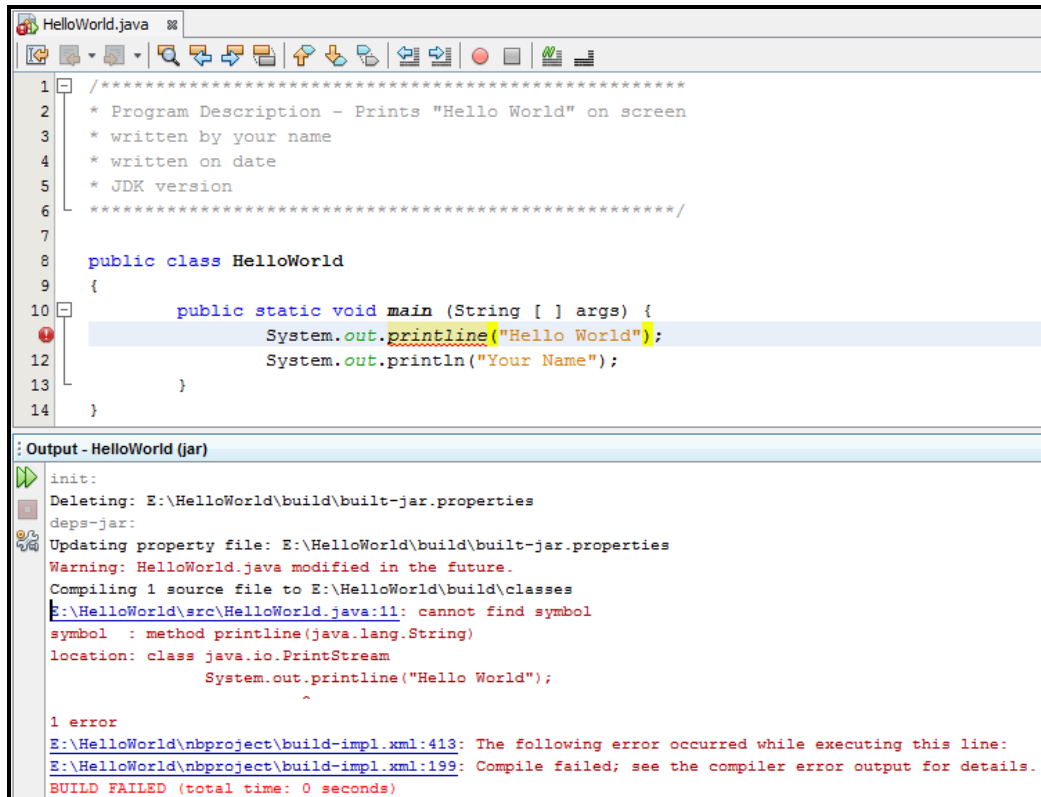




18. Open your code, by expanding the **HelloWorld** project folder, then expanding the **Source Package** folder, then expanding the **default package** folder (*this is default since we did not name this folder when we created the project*) as shown below. You will need to double click on the **HelloWorld.java** file to open the code.



19. You will now purposely make some errors in your program. Change the spelling of **println** to be **printline** as follows:
System.out.printline("Hello World");
20. Compile the program by clicking on **Run** menu and choosing **Build Project**. You should get an error at the bottom of your screen as follows:



```
1  /*****
2  * Program Description - Prints "Hello World" on screen
3  * written by your name
4  * written on date
5  * JDK version
6  *****/
7
8  public class HelloWorld
9  {
10     public static void main (String [ ] args) {
11         System.out.println("Hello World");
12         System.out.println("Your Name");
13     }
14 }
```

Output - HelloWorld (jar)

```
init:
Deleting: E:\HelloWorld\build\build-jar.properties
deps-jar:
Updating property file: E:\HelloWorld\build\build-jar.properties
Warning: HelloWorld.java modified in the future.
Compiling 1 source file to E:\HelloWorld\build\classes
E:\HelloWorld\src\HelloWorld.java:11: cannot find symbol
symbol  : method println(java.lang.String)
location: class java.io.PrintStream
        System.out.println("Hello World");
                ^
1 error
E:\HelloWorld\nbproject\build-impl.xml:413: The following error occurred while executing this line:
E:\HelloWorld\nbproject\build-impl.xml:199: Compile failed; see the compiler error output for details.
BUILD FAILED (total time: 0 seconds)
```

21. It is telling you that there is an error on line 11. It could not find a method spelled as **println**. Some errors will be obvious and those are the nice ones to solve. Change the word **println** to be **println** so that this error is corrected.
22. Compile the program. You should be rid of all errors and it should say BUILD SUCCESSFUL. Lesson learned: Names must be spelled exactly as Java expects.
23. Change line 11 by deleting the opening set of double quotes around Hello World as follows: **System.out.println(Hello World);**
24. Compile the program. You should get 3 errors at the bottom of your screen:

```
1  /*****
2  * Program Description - Prints "Hello World" on screen
3  * written by your name
4  * written on date
5  * JDK version
6  *****/
7
8  public class HelloWorld
9  {
10     public static void main (String [ ] args) {
11         System.out.println(Hello World);
12         System.out.println("Your Name");
13     }
14 }
```

Output - HelloWorld (jar)

```
init:
Deleting: E:\HelloWorld\build\build-jar.properties
deps-jar:
Updating property file: E:\HelloWorld\build\build-jar.properties
Warning: HelloWorld.java modified in the future.
Compiling 1 source file to E:\HelloWorld\build\classes
E:\HelloWorld\src\HelloWorld.java:11: ')' expected
        System.out.println(Hello World);
                           ^
E:\HelloWorld\src\HelloWorld.java:11: unclosed string literal
        System.out.println(Hello World);
                           ^
E:\HelloWorld\src\HelloWorld.java:12: ';' expected
        System.out.println("Your Name");
                           ^
3 errors
```

25. So, why would you get multiple error messages when you just made one mistake? This can happen. You may even get 15 errors for just one mistake. It really depends upon what mistake you make. In this case, it is saying it doesn't understand the *Hello World* and is saying it thinks it needs a parenthesis. This is not really the case. What it needs is a string enclosed in double quotes, but the error it shows is not real helpful. Now, insert the double quote back in front of the word *Hello*.
26. Compile the program. You should be rid of all errors. Lesson learned: The Java compiler doesn't always pinpoint the exact error. You must learn to look for errors anywhere on that line or previous line.
27. Not all errors are compilation errors. We now have a bug-free (no errors) Java program.

28. A programmer sometimes makes logic errors. Change line 11 to say:

System.out.println("Helo World");

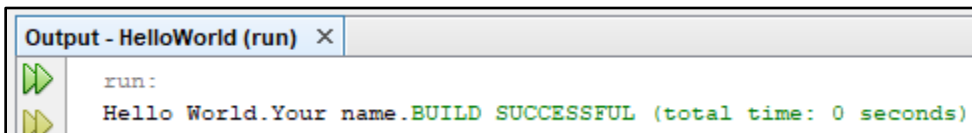
Note: Hello and World are incorrectly spelled.

29. Compile the program. The program should get a compile with BUILD SUCCESSFUL. However, when this program is executed (RUN menu, then RUN MAIN PROJECT), you will not get the words displayed that you wanted. The reason that you get no errors in the compilation is because the words inside of double quotes can be anything. The computer has no idea what you are trying to accomplish. It will display anything that is in double quotes and doesn't check that part for incorrect spellings.

30. Please fix all of your errors and recompile. NetBeans automatically saves files when your project is compiled.

31. Let's adjust your **println** statements to use the **print** method instead of **println**. You should see that our text is all on one line. This doesn't look as nice as before.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.print("Hello World.");  
        System.out.print("Your name.");  
    }  
}
```

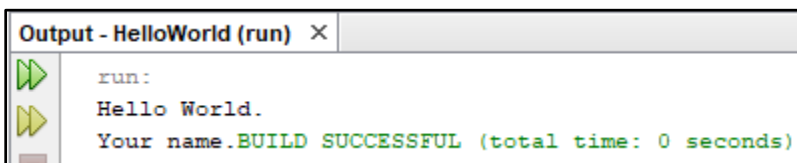


Output - HelloWorld (run) X

run:
Hello World.Your name.BUILD SUCCESSFUL (total time: 0 seconds)

32. Let's fix this by taking out the second **print** statement and adding a **\n** escape code to get your name to print on a new line.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.print("Hello World. \nYour name.");  
    }  
}
```

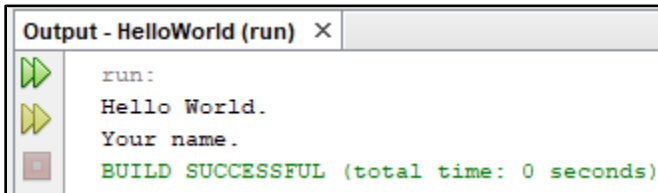


Output - HelloWorld (run) X

run:
Hello World.
Your name.BUILD SUCCESSFUL (total time: 0 seconds)

33. It might look better if we changed our **print** statement to a **println** statement so that the build information is on a new line.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World. \nYour name.");  
    }  
}
```



Output - HelloWorld (run) X

run:
Hello World.
Your name.
BUILD SUCCESSFUL (total time: 0 seconds)

You can also put the code that prints your name on a separate line of code, but it won't change your result in the output window.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World. "  
                           + "\nYour name.");  
    }  
}
```

34. Close the HelloWorld project by right clicking on the project in the left pane as shown below:

