**Getting Started:   
"Hello World!" for the NetBeans IDE**Based on: <http://docs.oracle.com/javase/tutorial/getStarted/cupojava/netbeans.html>

**Checklist  a checkmark**

To write your first program, you'll need:

1. **The Java SE Development Kit (JDK 7 has been selected in this example)**
   * For Microsoft Windows, Solaris OS, and Linux: [Java SE Downloads Index](http://www.oracle.com/technetwork/java/javase/downloads/index.html) page
   * For Mac OS X: [developer.apple.com](https://developer.apple.com/)
2. **The NetBeans IDE**
   * For all platforms: [NetBeans IDE Downloads Index](http://netbeans.org/downloads/index.html) page

**Creating Your First Application**

Your first project, HelloWorldProject, will simply display the greeting "Hello World!"

To create this program, you will:

* [**Create an IDE project**](#_Create_an_IDE)

When you create an IDE project, you create an environment in which to build and run your applications. Using IDE projects eliminates configuration issues normally associated with developing on the command line. You can build or run your application by choosing a single menu item within the IDE.

* [**Add code to the generated source file**](#_Add_Code_to)

A source file contains code, written in the Java programming language, that you and other programmers can understand. As part of creating an IDE project, a skeleton source file will be automatically generated. You will then modify the source file to add the "Hello World!" message.

* [**Compile the source file into a .class file**](#_Compile_the_Source)

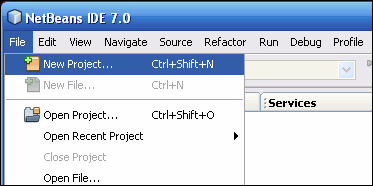
The IDE invokes the Java programming language *compiler* (javac), which takes your source file and translates its text into instructions that the Java virtual machine can understand. The instructions contained within this file are known as *bytecodes*.

* [**Run the program**](#_Run_the_Program)

The IDE invokes the Java application *launcher tool* (java), which uses the Java virtual machine to run your application.

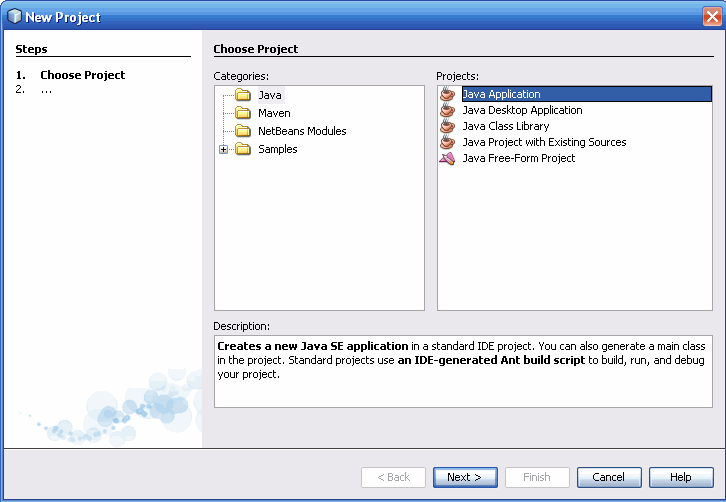
# Create an IDE Project

1. Launch the NetBeans IDE.
   * On Microsoft Windows systems, you can use the NetBeans IDE item in the Start menu.
   * On Solaris OS and Linux systems, you execute the IDE launcher script by navigating to the IDE's bin directory and typing ./netbeans.
   * On Mac OS X systems, click the NetBeans IDE application icon.
2. In the NetBeans IDE, choose File | New Project:



NetBeans IDE with the File | New Project menu item selected.

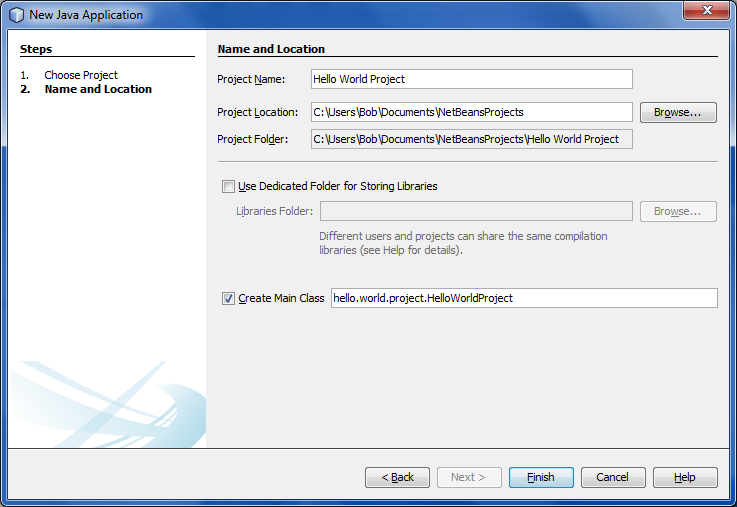
1. In the New Project wizard, expand the Java category and select Java Application as shown:



.

I suggest that you include the word **Project** in the Project name so you can easily identify project folders.  
 In the Name and Location page of the wizard, do the following (as shown in the figure below).:

* + In the Project Name field, type Hello World Project.

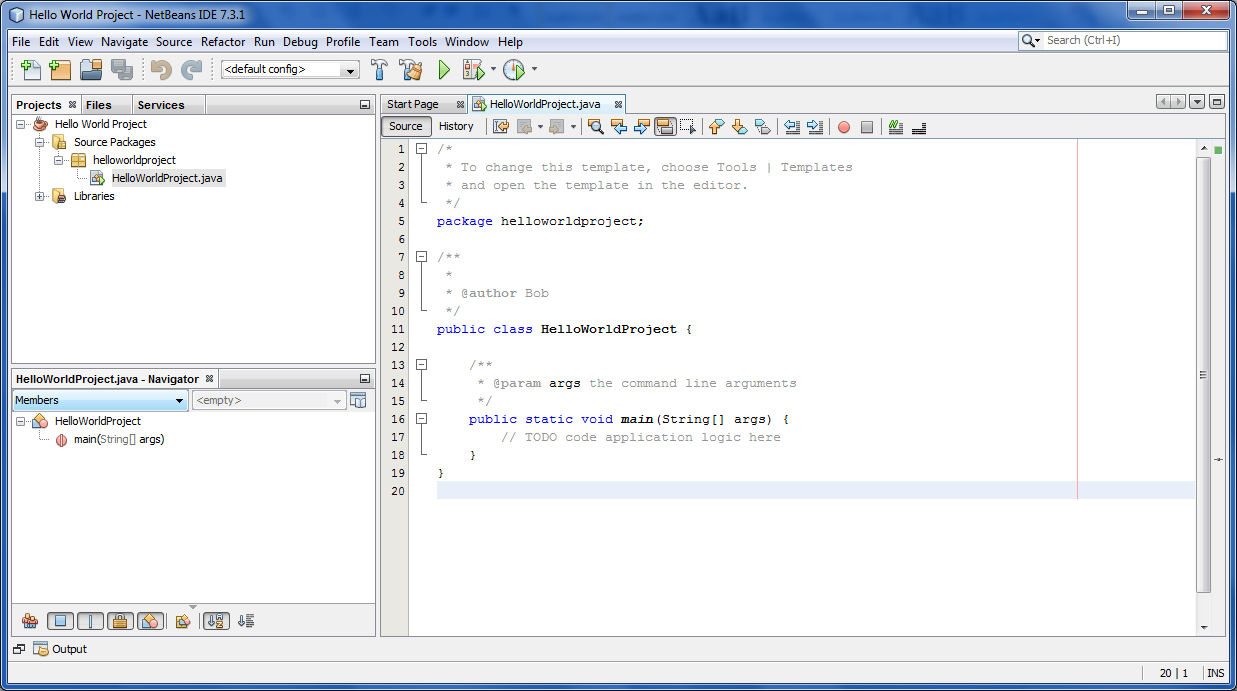


* + Leave the Create Main Class checkbox selected.
  + Change the name to **helloworldproject.HelloWorldProject** (take out the “.” between words)

1. Click Finish.

The project is created and opened in the IDE. You should see the following components:

* The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
* The Source Editor window with a file called HelloWorldProject open.
* The Navigator window, which you can use to quickly navigate between elements within the selected class.



# Add Code to the Generated Source File

When you created this project, you left the Create Main Class checkbox selected in the New Project wizard. The IDE has therefore created a skeleton class for you. You can add the "Hello World!" message to the skeleton code by replacing the line:

// TODO code application logic here

with the line:

System.out.println("Hello World!"); // Display the string.

Remember to replace “student” with your name in the @Author line:

/\*\*

\*

\* @author <your> <name>

\*/

Optionally, you can insert these lines:

\* The HelloWorldProject class implements an application that

\* simply prints "Hello World!" to standard output.

Just above the \* @author line.

These four lines are a comment for the programmer and do not affect how the program runs. Later sections of this tutorial explain the use and format of code comments.

**Be Careful When You Type** uppercase letter A   lowercase letter A

**Note:** Type all code, commands, and file names exactly as shown. Both the compiler (javac) and launcher (java) are *case-sensitive*, so you must capitalize consistently.  
  
**HelloWorldProject** is *not* the same as **helloworldproject**.

**Save your changes by choosing File | Save.**

The file should look something like the following:

**/\***

**\* To change this template, choose Tools | Templates**

**\* and open the template in the editor.**

**\*/**

**package helloworldproject;**

**/\*\***

**\* The HelloWorldProject class implements an application that**

**\* simply prints "Hello World!" to standard output.**

**\***

**\* @author <your> <name>**

**\*/**

**public class HelloWorldProject {**

**/\*\***

**\* @param args the command line arguments**

**\*/**

**public static void main(String[] args) {**

**System.out.println("Hello World!"); // Display the string.**

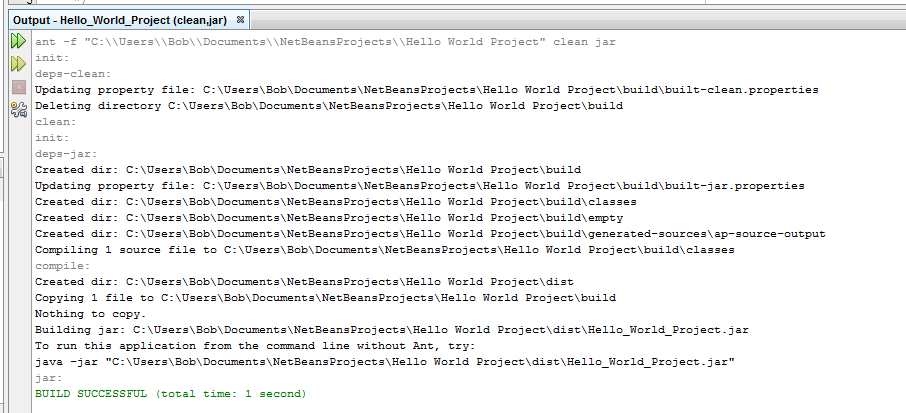
**}**

**}**

# Compile the Source File into a .class File

To compile your source file, choose Run | Build Main Project from the IDE's main menu.

The Output window displays output similar to what you see in the following figure:

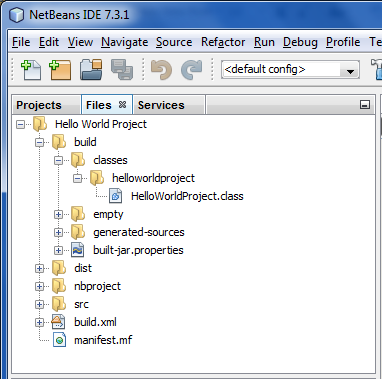


Output window showing results of building the HelloWorld project.

If the build output concludes with the statement BUILD SUCCESSFUL, congratulations! You have successfully compiled your program!

If the build output concludes with the statement BUILD FAILED, you probably have a syntax error in your code. Errors are reported in the Output window as hyper-linked text. You double-click such a hyper-link to navigate to the source of an error. You can then fix the error and once again choose Run | Build Main Project.

When you build the project, the bytecode file HelloWorldProject.class is generated. You can see where the new file is generated by opening the Files tab and expanding the Hello World App/build/classes/helloworldproject node as shown in the following figure:

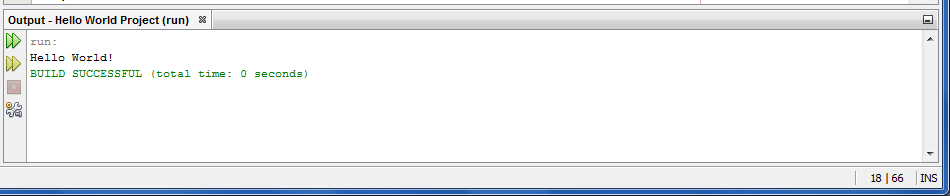


Now that you have built the project, you can run your program.

# Run the Program

From the IDE's menu bar, choose Run | Run Main Project.

The next figure shows what you should now see:



The program prints "Hello World!" to the Output window (along with other output from the build script).

Congratulations! Your program works!

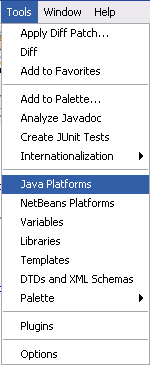
# Tips on using Java Beans IDE

The following are some tips on using the IDE and explanations of some IDE behavior that you are likely to see:

* Once you have created a project in the IDE, you can add files to the project using the New File wizard. Choose File | New File, and then select a template in the wizard, such as the Empty Java File template.
* You can compile and run an individual file (as opposed to a whole project) using the IDE's Compile File (F9) and Run File (Shift-F6) commands. If you use the Run Main Project command, the IDE will run the file that the IDE associates as the main class of the main project. Therefore, if you create an additional class in your HelloWorldProject project and then try to run that file with the Run Main Project command, the IDE will run the HelloWorldProject file instead.
* You might want to create separate IDE projects for sample applications that include more than one source file.
* As you are typing in the IDE, a code completion box might periodically appear. You can either ignore the code completion box and keep typing, or you can select one of the suggested expressions. If you would prefer not to have the code completion box automatically appear, you can turn off the feature. Choose Tools | Options | Editor, click the Code Completion tab and clear the Auto Popup Completion Window checkbox.
* If you want to rename the node for a source file in the Projects window, choose Refactor from IDE's main menu. The IDE prompts you with the Rename dialog box to lead you through the options of renaming the class and the updating of code that refers to that class. Make the changes and click Refactor to apply the changes. This sequence of clicks might seem unnecessary if you have just a single class in your project, but it is very useful when your changes affect other parts of your code in larger projects.
* For a more thorough guide to the features of the NetBeans IDE, see the [NetBeans Documentation](http://www.netbeans.org/kb/index.html) page.

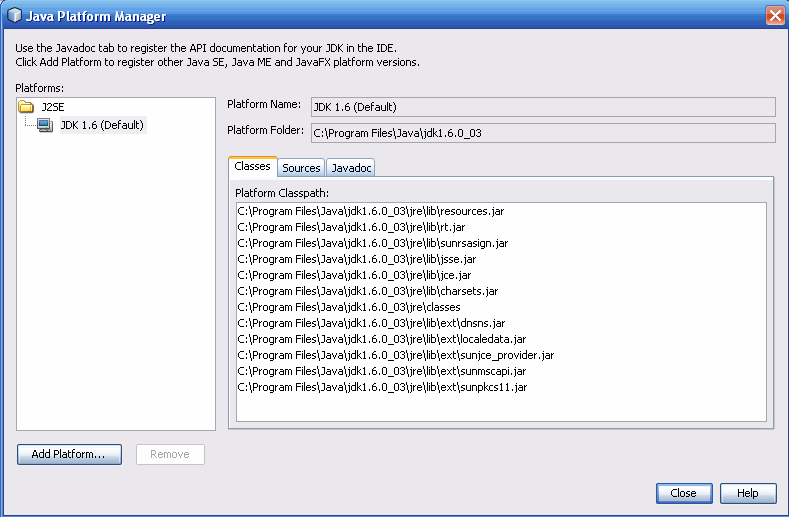
## Appendix – Add JDK 7 to the Platform List (if necessary)

It may be necessary to add JDK 7 to the IDE's list of available platforms. To do this, choose Tools | Java Platforms as shown in the following figure:



Selecting the Java Platform Manager from the Tools Menu

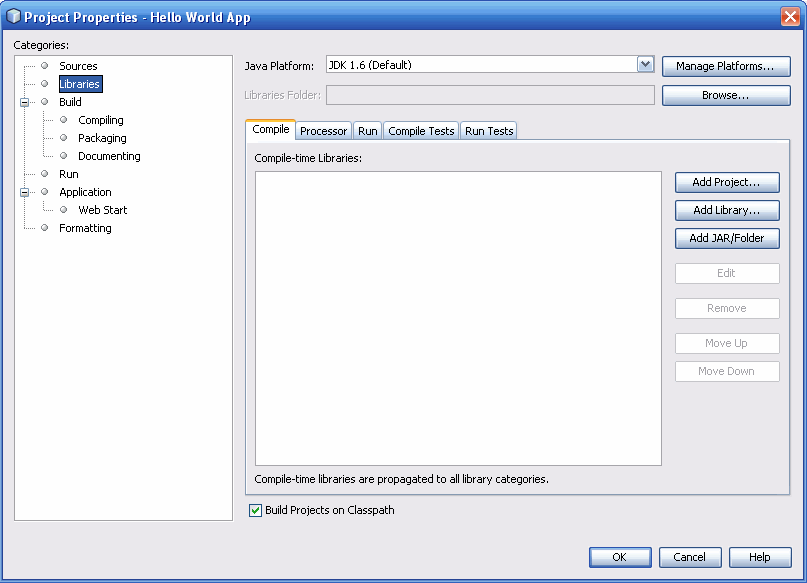
If you don't see JDK 7 (which might appear as 1.7 or 1.7.0) in the list of installed platforms, click "Add Platform", navigate to your JDK 7 install directory, and click "Finish". You should now see this newly added platform:



The Java Platform Manager

To set this JDK as the default for all projects, you can run the IDE with the --jdkhome switch on the command line, or by entering the path to the JDK in the netbeans\_j2sdkhome property of your INSTALLATION\_DIRECTORY/etc/netbeans.conf file.

To specify this JDK for the current project only, select Hello World App in the Projects pane, choose File | Project Properties (Hello World App), click on Libraries, then select JDK 7 under the Java Platform pulldown menu. You should see a screen similar to the following:



The IDE is now configured for JDK 7.