LAB 0

Objective

The objectives of this lab are:

1. To familiarize you with BlueJ (the Java development environment we will be using)
2. To teach you how to write a simple Java program that uses just the `main` method of a class;
3. To review programming concepts such as simple data types conditionals and loops and to introduce the syntax and semantics of these constructs in Java.
4. To familiarize you with procedures for submitting work done in the lab.

What You Should Do

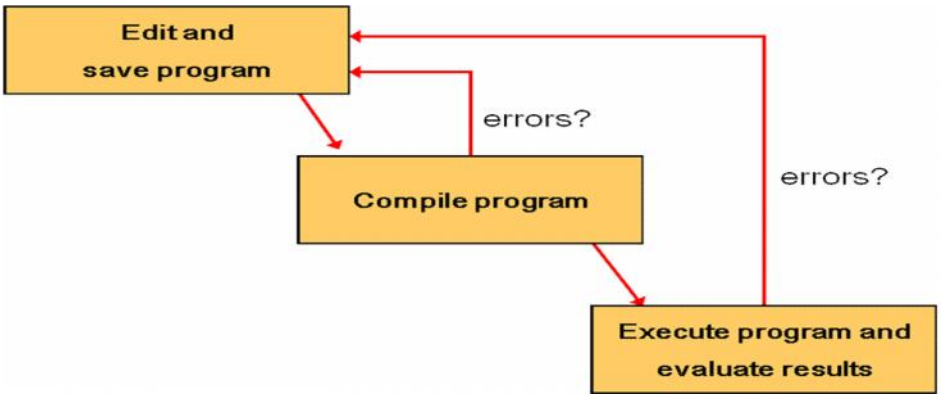
Read these instructions carefully. This document is designed to help you get the most from your lab.

Before You Start

Ensure that the machine you are sitting in front of is running Windows. These instructions are only for Windows users using the computers in the UWI CS lab.

What is BlueJ?

Writing Java programs requires that you use several tools. First you use an editor to create a file with the program code. After you have created your program you use a compiler to convert the program code to byte code. The byte code is then run on the Java Virtual Machine (JVM).



If you work at the command line you carry out these tasks by running each of these tools as is needed. An Integrated Development Environment (IDE) is a single program that provides all these facilities for you in an "integrated environment". The IDE usually has a Graphical User Interface (GUI) so that you simply point and click on icons instead of typing commands. It is an easier, more convenient way to work.

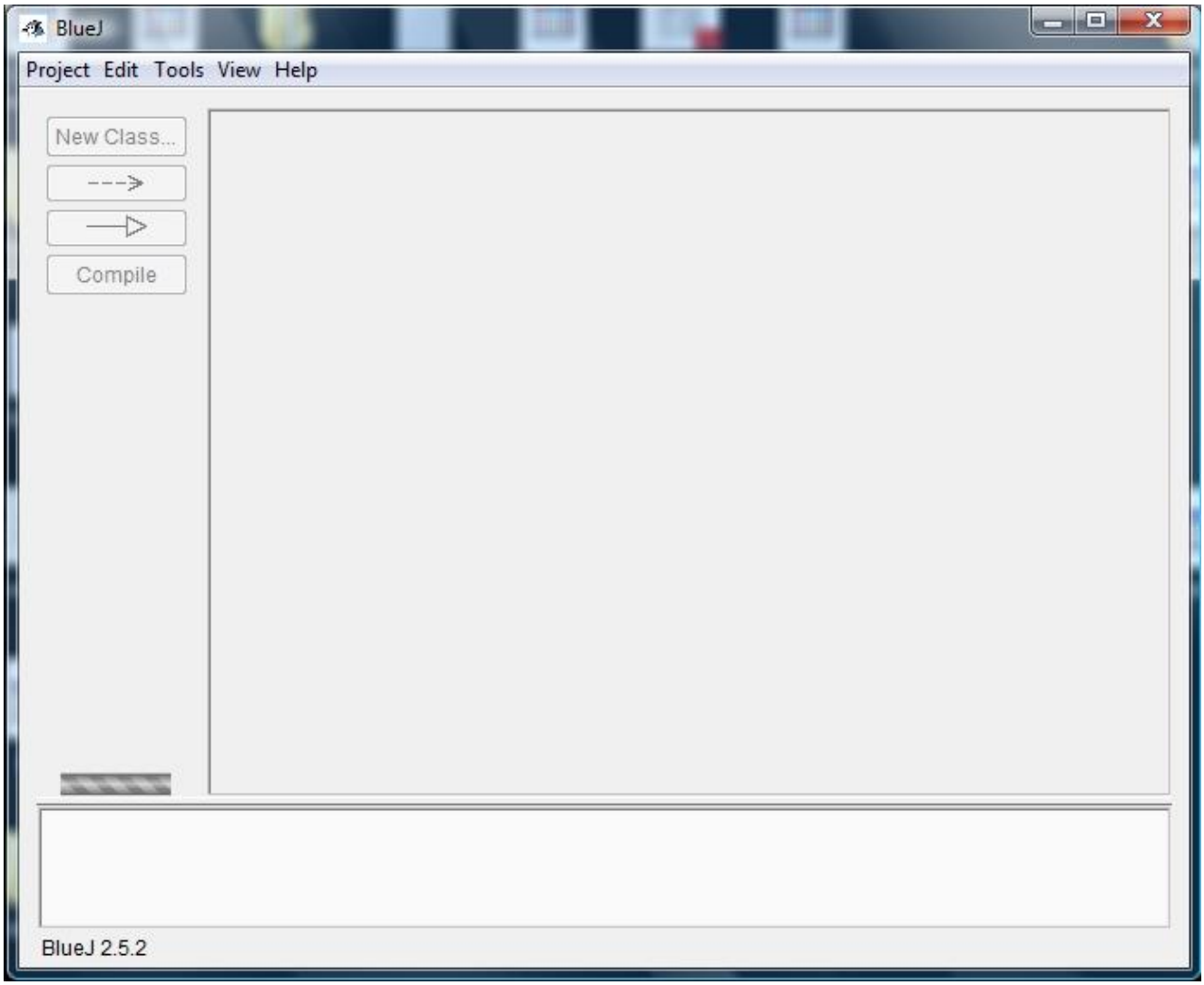
BlueJ is an IDE that can be used to create and manage Java programs. BlueJ is especially useful for the student environment as it is much “lighter” than IDEs such as NetBeans and Eclipse which are used by professional programmers. At the same time, BlueJ gives you everything you will need to be an effective Java programmer.

Running the BlueJ Java development environment.

Click Start, select All Programs, select Programming, select Object Oriented, select BlueJ and then click on the BlueJ option at the top of this menu.

See the next page.

When you have successfully started BlueJ you should see this on the screen:



This is the first screen you will see when developing your Java programs. The next time you run it you will probably see more items in the main window. That's OK, that's correct.

Setting Up BlueJ

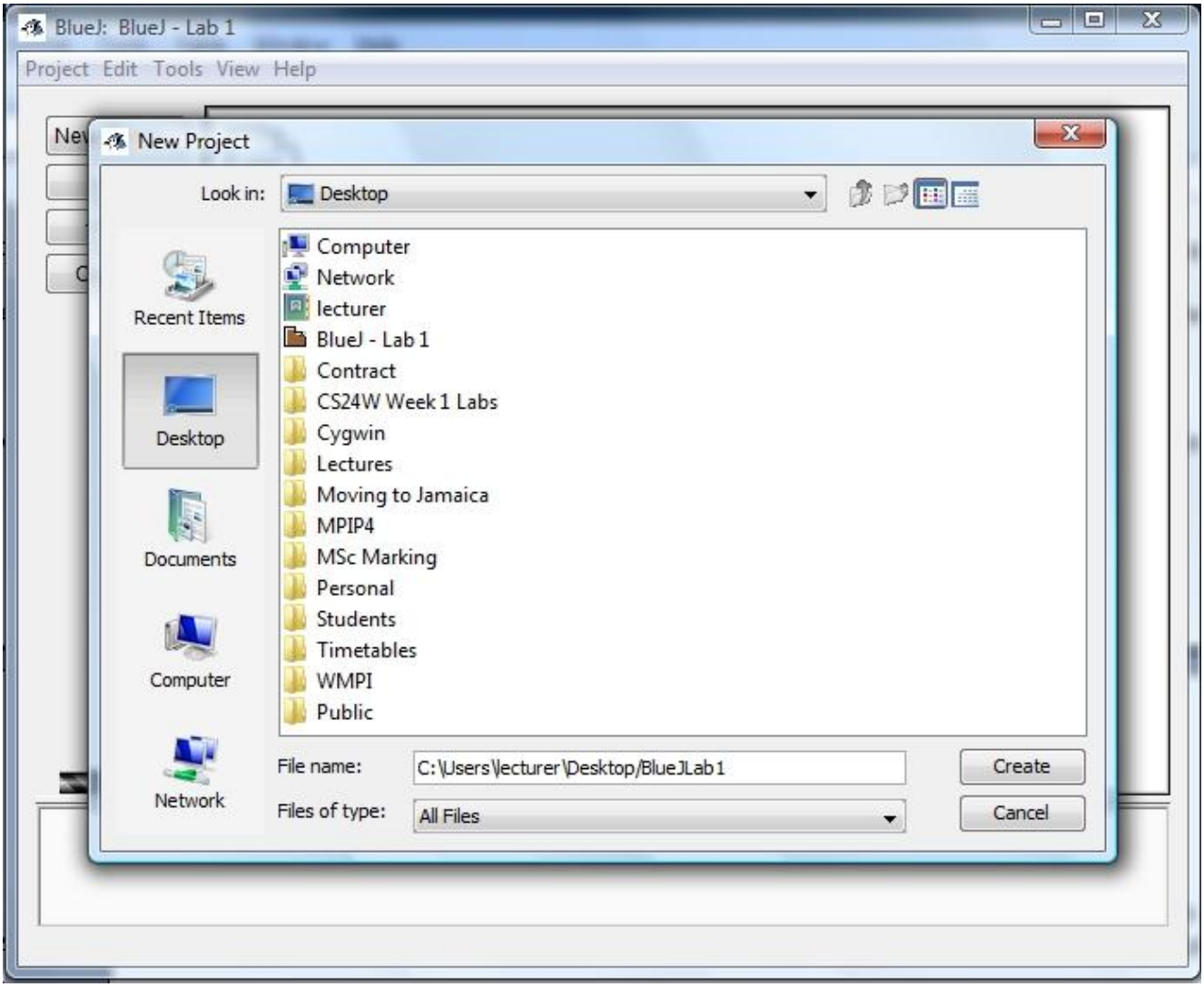
Before we write our first program we have to set things up first. From now on, all the instructions are to be performed within BlueJ unless this document says otherwise.

In BlueJ, do the following:

1. Click the Project drop-down menu (it's in the bar at the very top)
2. Select New Project…
3. On the window that p to create a new folder on your computer's desk n another location (e.g. on a thumb drive) you sh e)
4. You will be prompted project in BlueJ (and other code developm ogram source files and other data a develop m.

(See the next page for v ike)

[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Text Box Tools tab to change the formatting of the pull quote text box.]

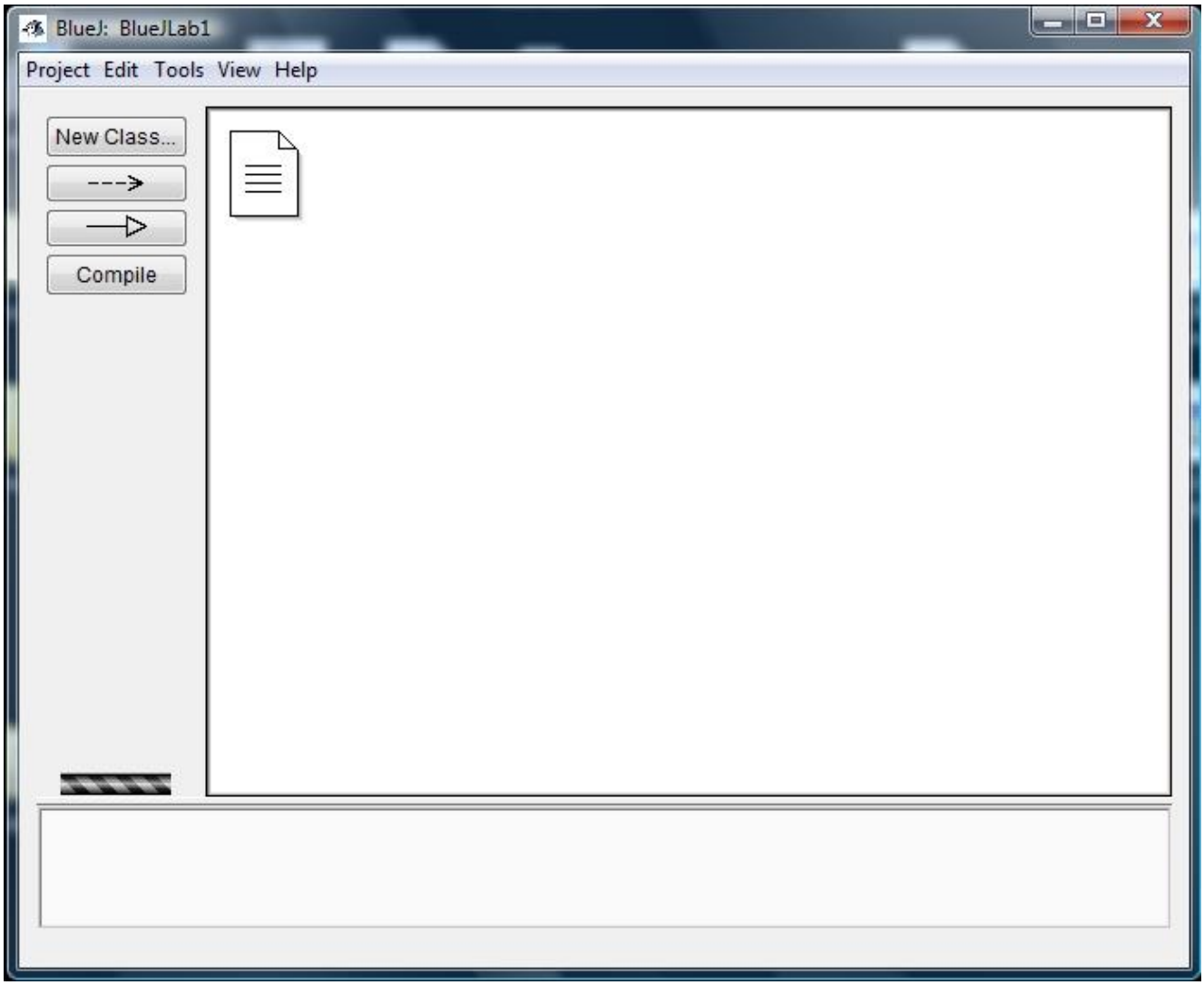


5. Click create

BlueJ has created a folder on your Desktop (or other location) called BlueJLab1. BlueJ will place all of your Java files and everything else it needs to work in this folder.

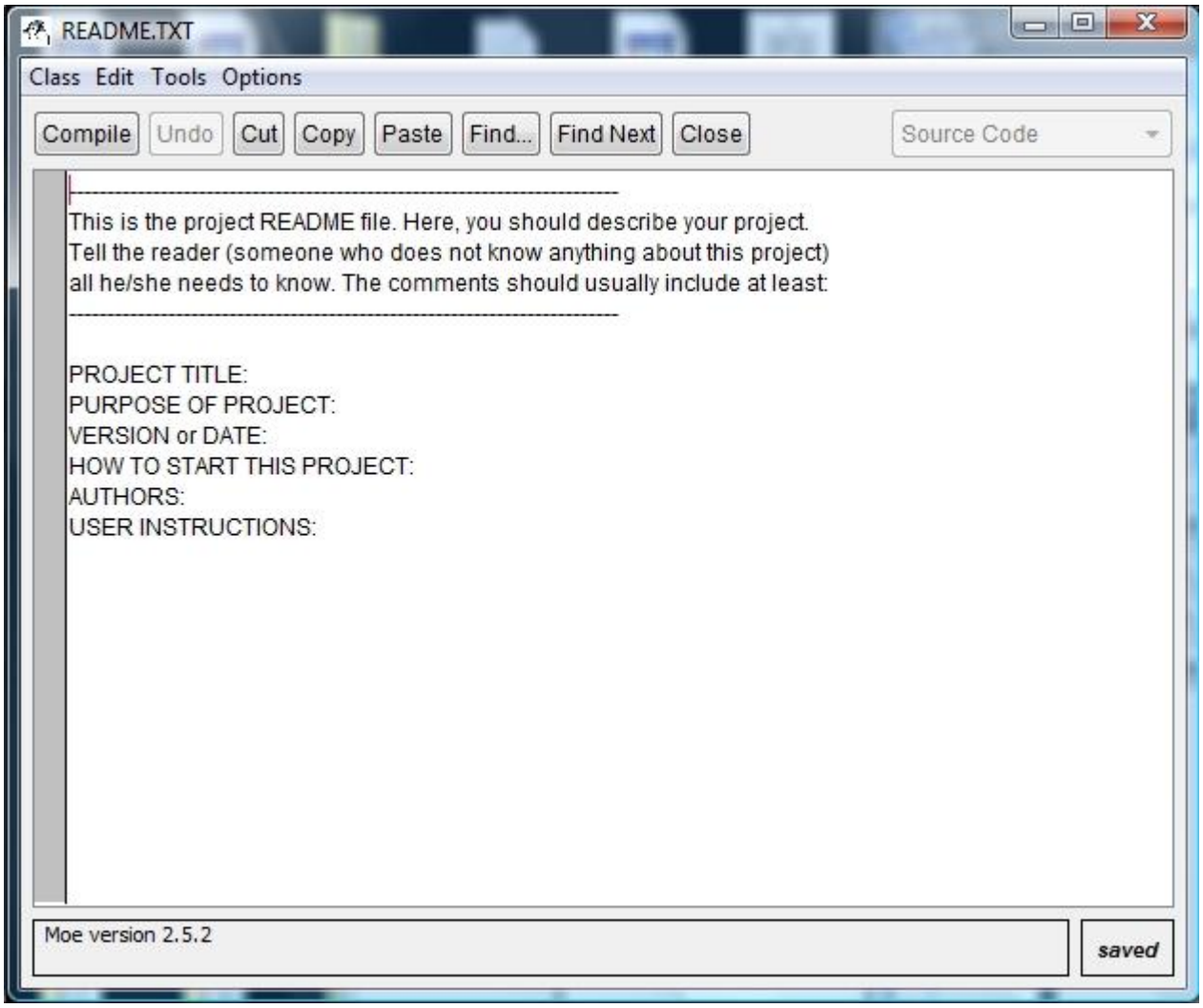
See the next page.

The BlueJ interface should now look like this:



BlueJ has created a single file (shown by the file icon) and placed it into your project. Right-click on the file and select open. See the next page.

You will see this:



This is a README.txt file that has been created by BlueJ. This file is BlueJ specific and has nothing to do with Java.

The file contains information that will be useful to you when you look back at this code in the future. Fill in the above information: the project title can be Lab1. The purpose of the project is to learn some Java. Add today's date. In How to Start this Project say "right-click on the Main class and select ??". For Authors add your full name and your student id. For User Instructions say "Just run the program with no parameters".

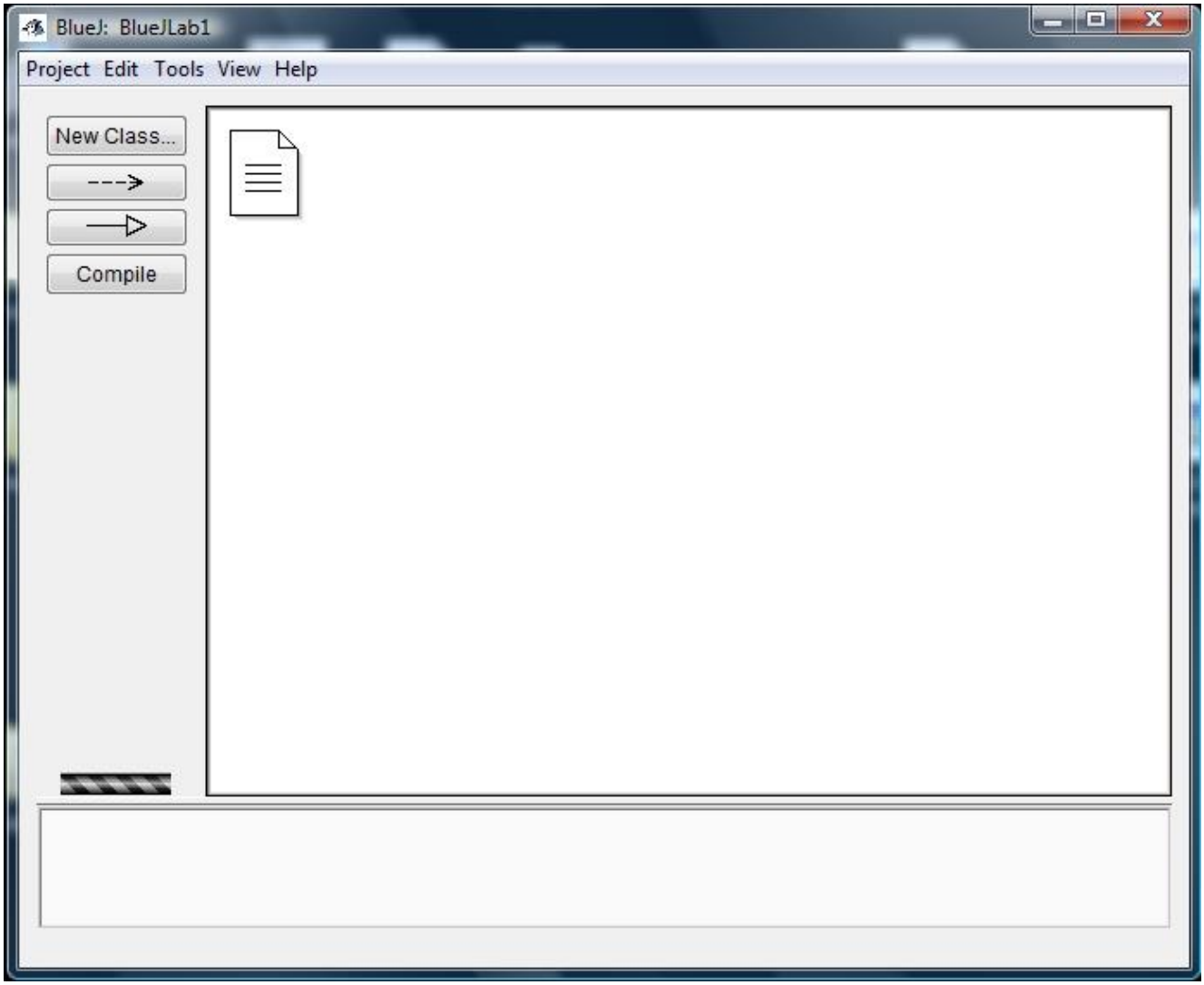
Once you have filled in all the information, select the "Class" drop down menu (in the bar at the very top) and select save. What you have typed has then been saved.

Now click the "Close" button which is on the far-right of the row of buttons.

You are now ready to write your first Java program. See the next page.

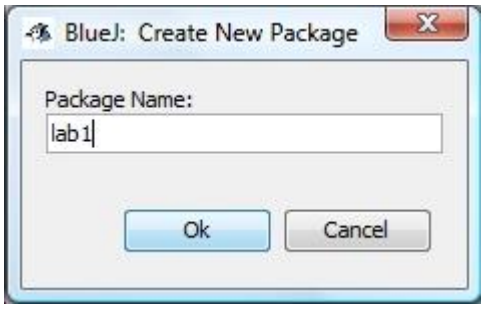
1. Writing Our First Java Program

You have BlueJ in front of you on the screen and it should look like this.



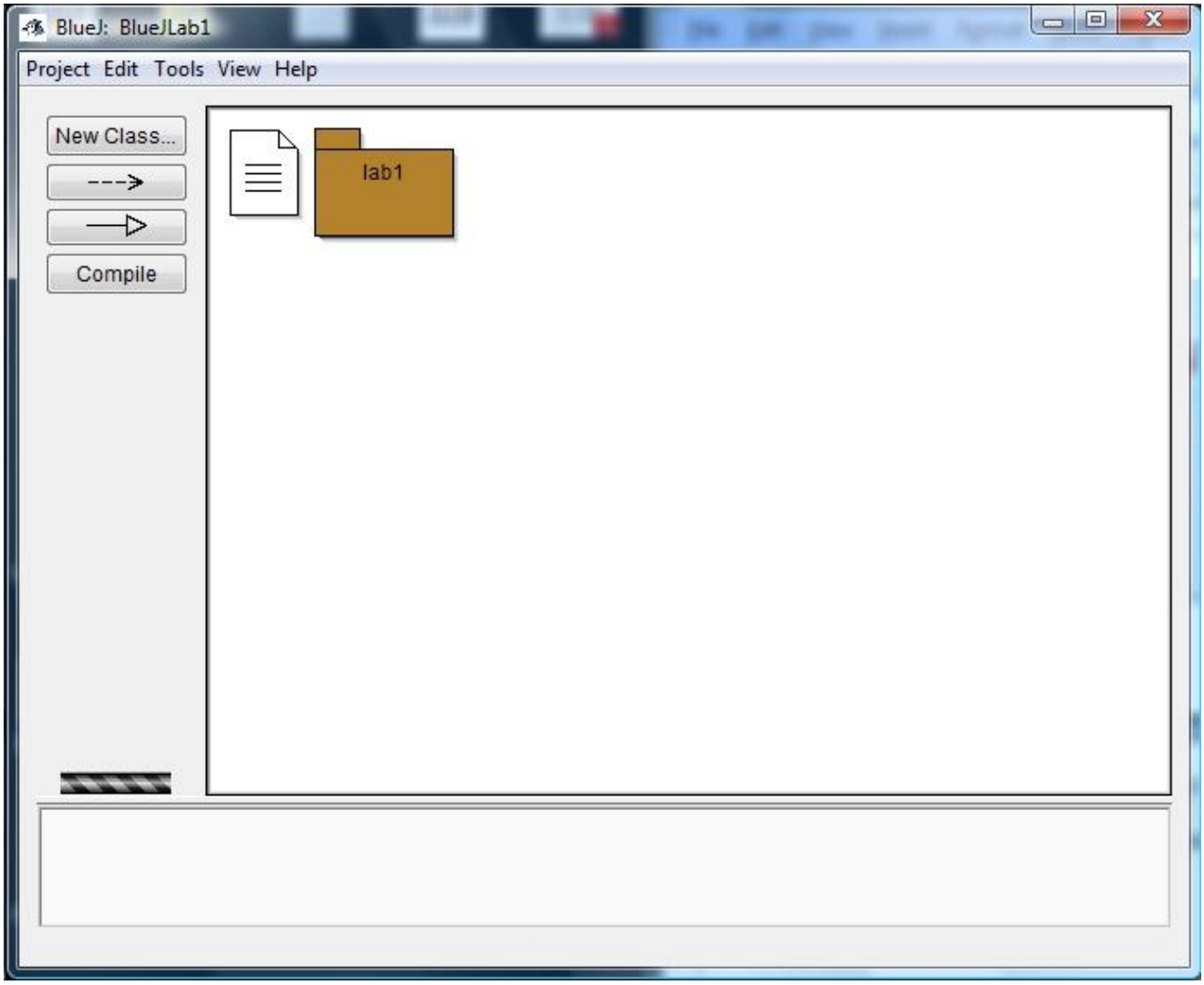
All Java code has to exist within a class and each class should exist within a Java package. We will see how packages work in later lectures. For now think of a package as a container into which we are going to place a number of Java classes.

To create a new Java package with BlueJ, click on Edit in the top level menu bar. Then select "New Package…". A new window pops up asking for the package name. Type lab1 as below.



Click Ok.

Your main BlueJ window will change and it will now look like this:



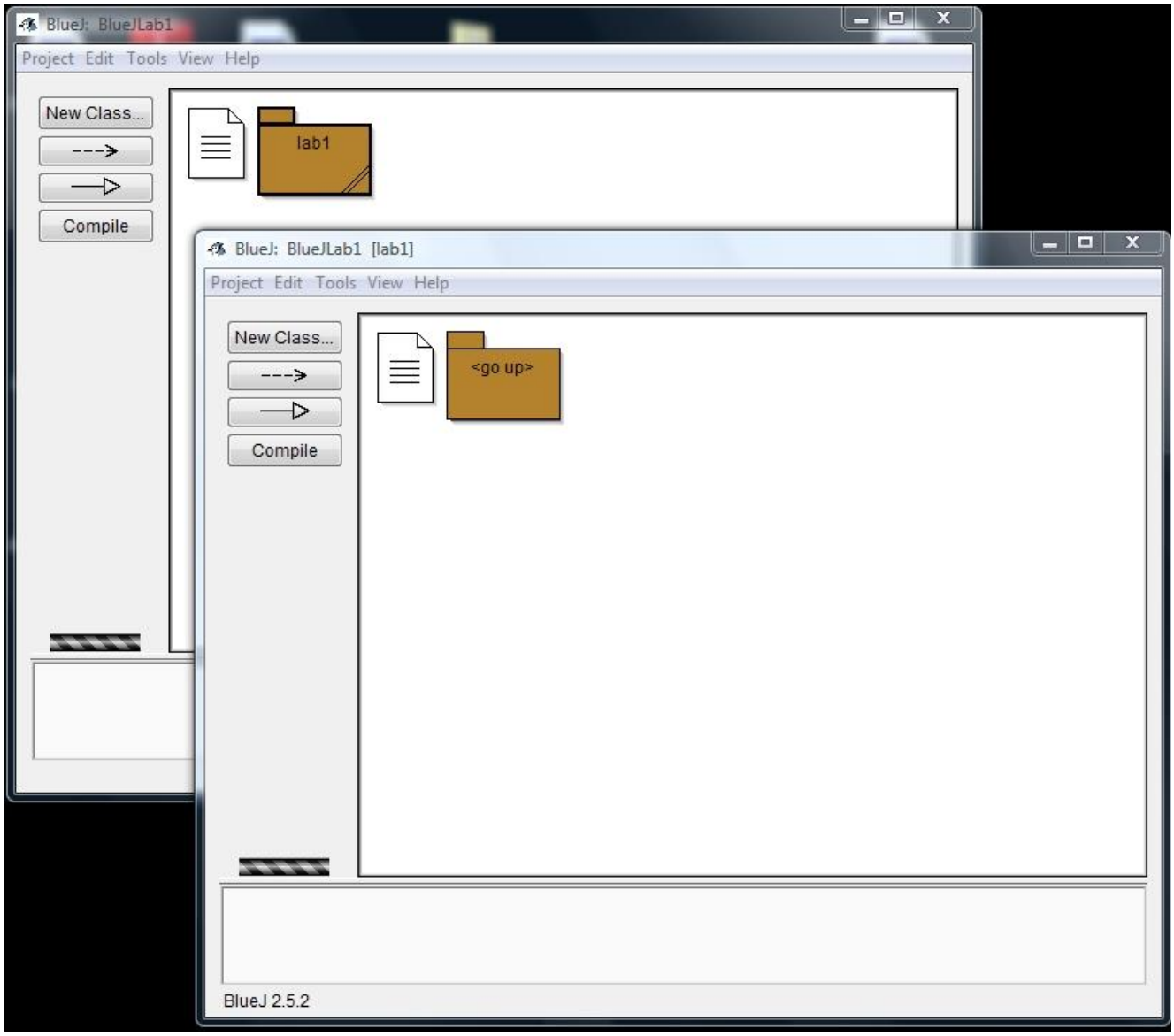
This brownish shape with "lab1" written in it is BlueJ's way of telling you that you have defined a package within your project.

Remember that a Java package is a container for any number of Java classes.

Right-click on the package icon and you will see a drop down menu appear with two items in it. Select open. BlueJ will then create another main window for you which will look like that below. See the next page.

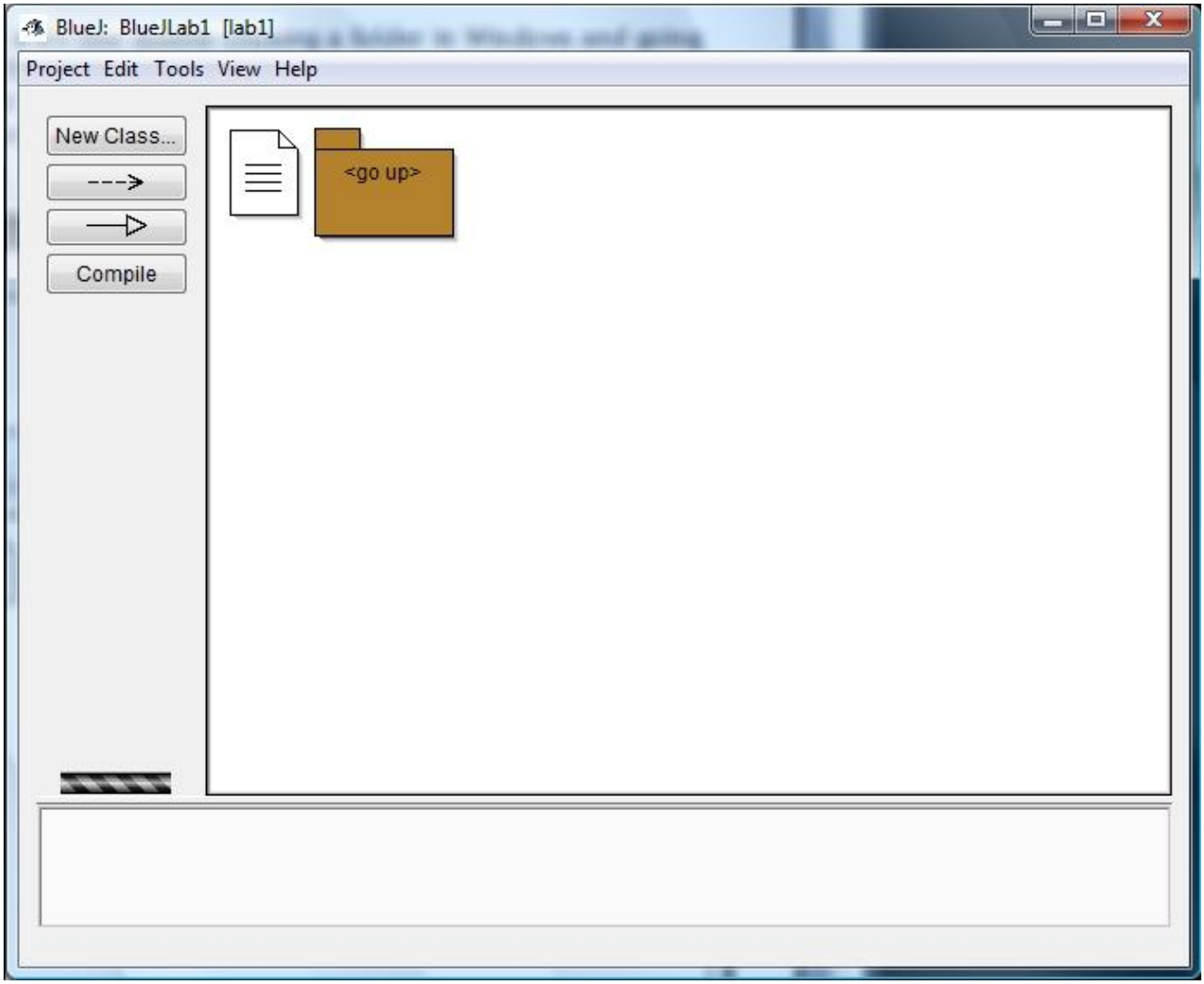
By opening the "lab1" package icon you have gone into the package, you have made BlueJ go into the container for the Java classes. Java lets packages contain other packages, as well as Java classes.

Moving into a package like this is a bit like double-clicking a folder in Windows and going into the folder. In the Windows file hierarchy you have a folder above the folder you are currently in. The same is true for Java. One package will have a parent package which is the package above it in the hierarchy.



You can tell the two windows are different because in the new one it says "go up" in its package definition. For the rest of this lab we are going to create code in the new package called "lab1" so you **MUST** use the window as shown above that currently says "go up" in it. To keep things easy make sure you minimize the other window, i.e., the window at the top of the above diagram, the one that has "lab1" in it. Minimize that window by clicking on the "_" sign at the very top right of the window.

You should only have the following window on your screen.

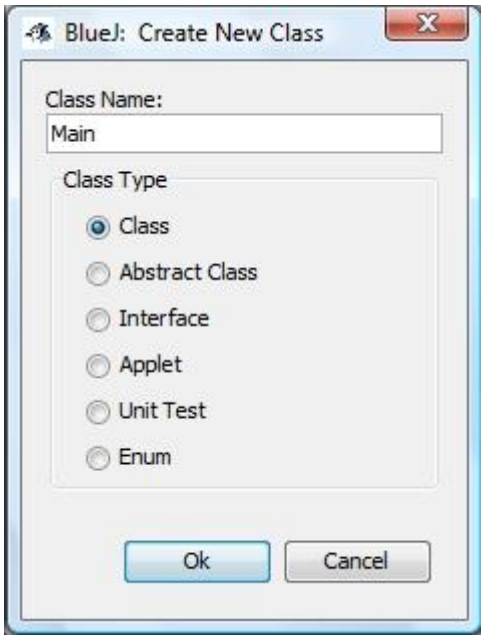


You will notice BlueJ creates a README file (that you completed above) for each Java package. You only have to complete the first one that BlueJ creates as you did above.

Now we have created a package (class container) called lab1 we have to put some classes into it.

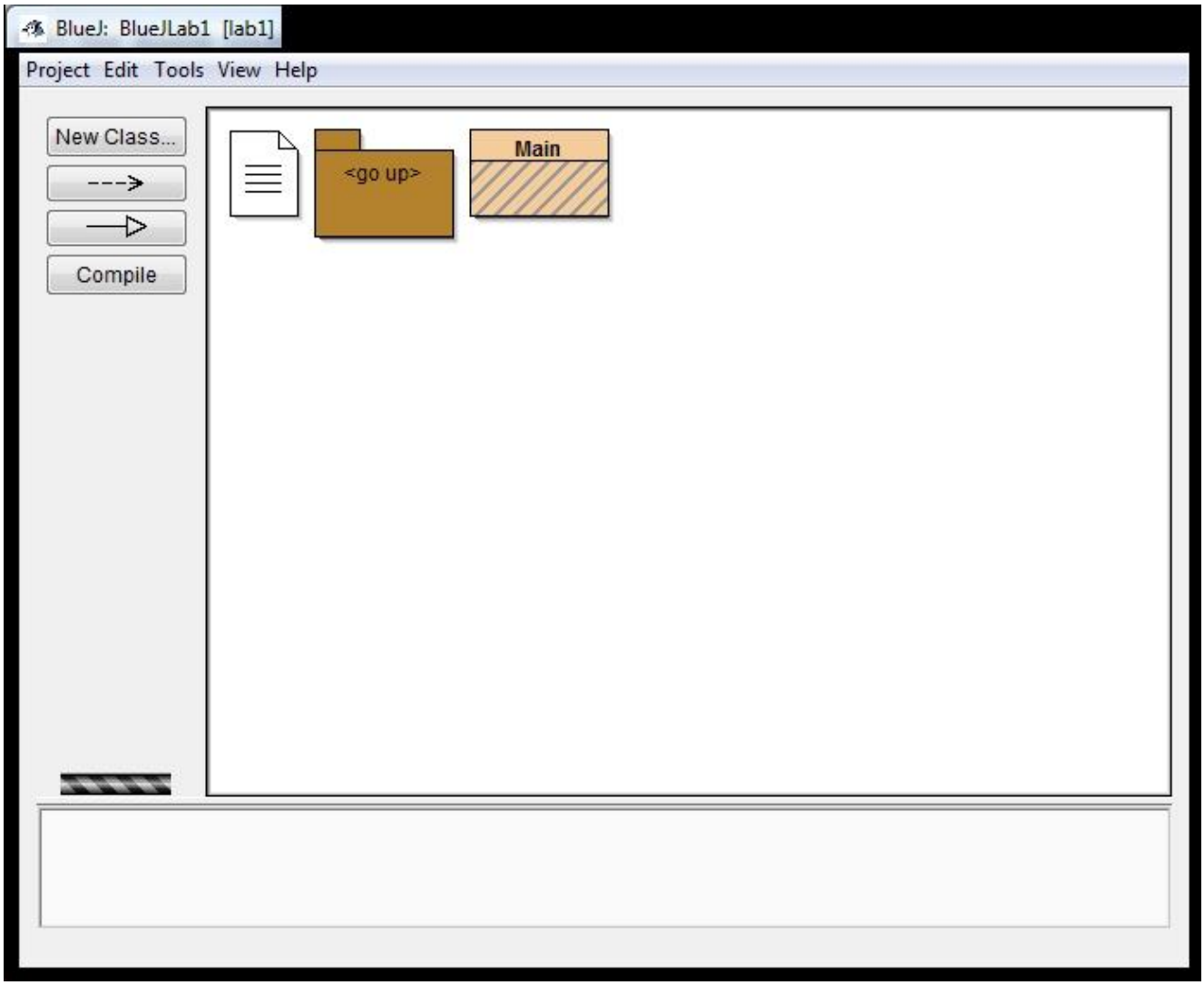
In Java all code has to exist within a class, therefore, we need to first of all create a new class.

Select the "New Class…" button in the top left of the above window. A small window appears. Enter Main where it ask for the class name and ensure that "Class" is selected within the Class Type list, like this:



Click Ok. See the next page.

You will now see this in the main BlueJ window.



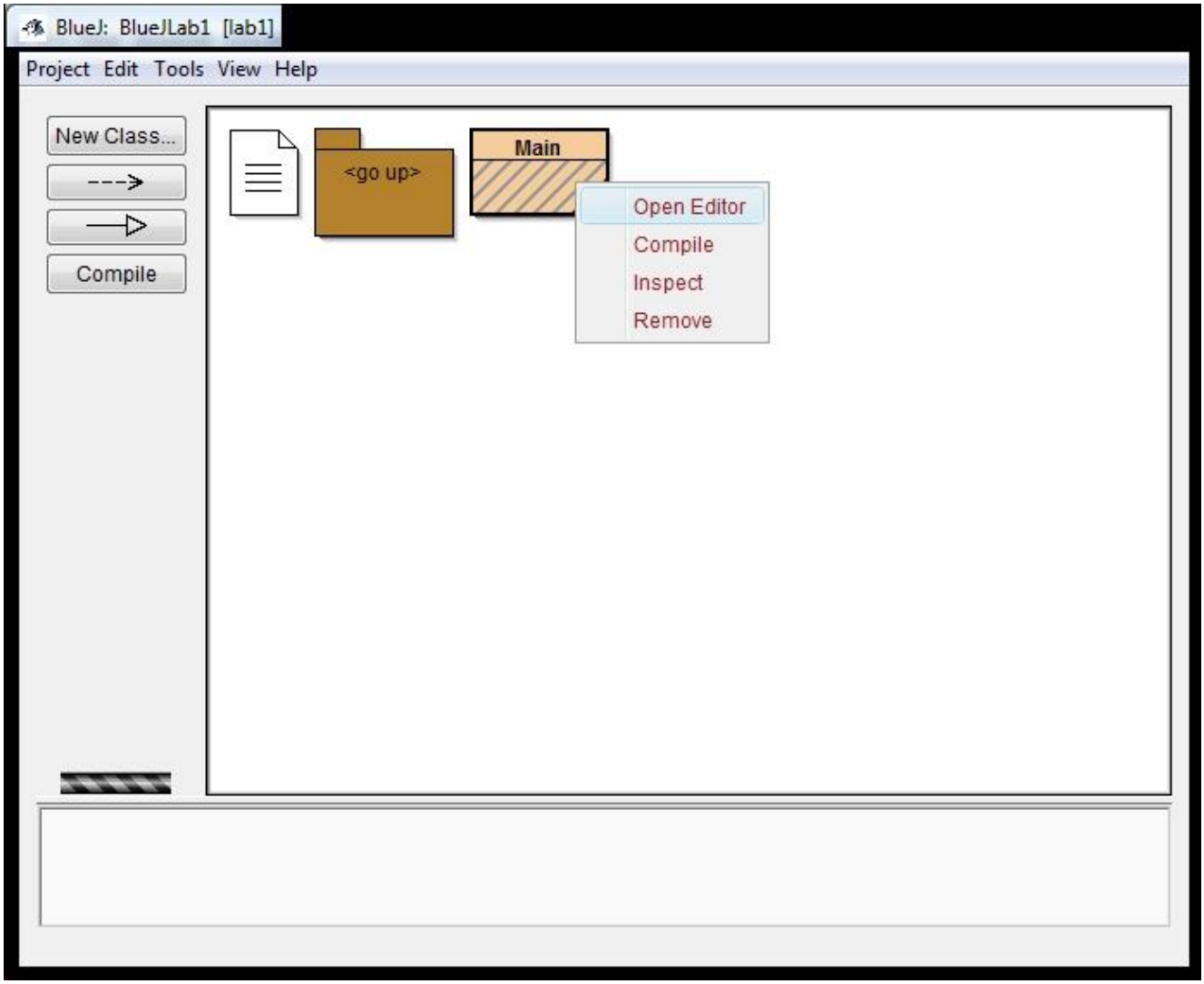
The orange box with the word Main in it is a pictorial representation of your Java class called Main. BlueJ has created a file called Main.java for you and it has placed it into your project (which you called BlueJLab1). BlueJ represents the file in the project as above.

As you create classes (and other items), BlueJ will display them as a diagram. You can link items together to tell BlueJ more information about your program, but for the moment, we are not going to do this.

We are going to edit the Main.java file.

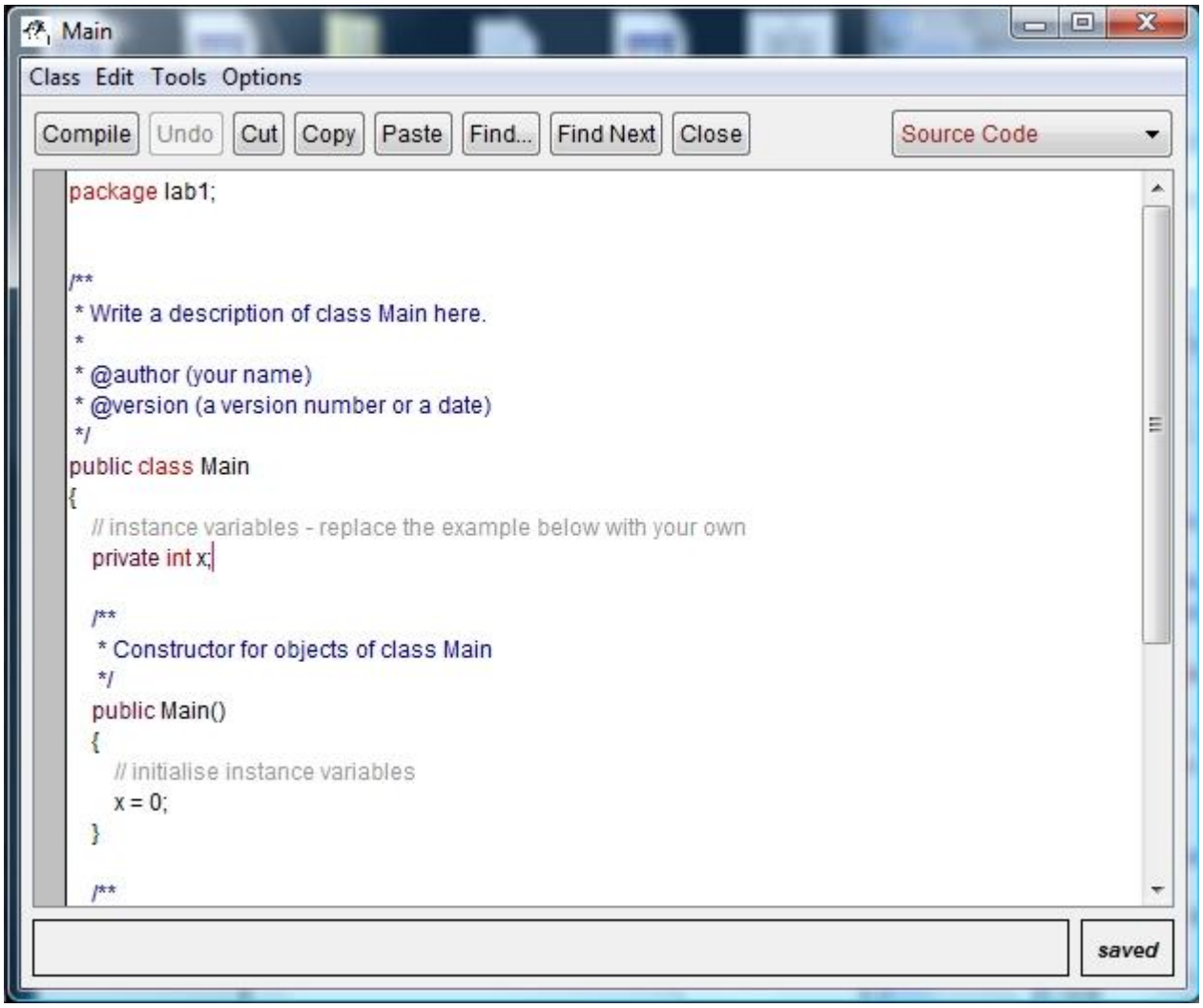
Editing Main.java

Put the mouse over the orange Main box and click the right mouse button (also known as right-click). A drop-down menu with four items in it should appear. See the next page.



Select Open Editor. See the next page.

A new window will appear which will contain Java code.



BlueJ has provided a partially complete class, but we are not going to use it.

In this window, delete all of the program text. You can do this by pressing control on the keyboard and keeping control pressed, then press a (also known as control-a). This should highlight of all the text with a yellow background. When you see this, then press the delete key on your keyboard. You should now have an empty editor window in front of you. See the next page.

You should see this in front of you. It **MUST** be completely blank. If it is not, get some help.

This is the editor into which we are going to type our first Java program.

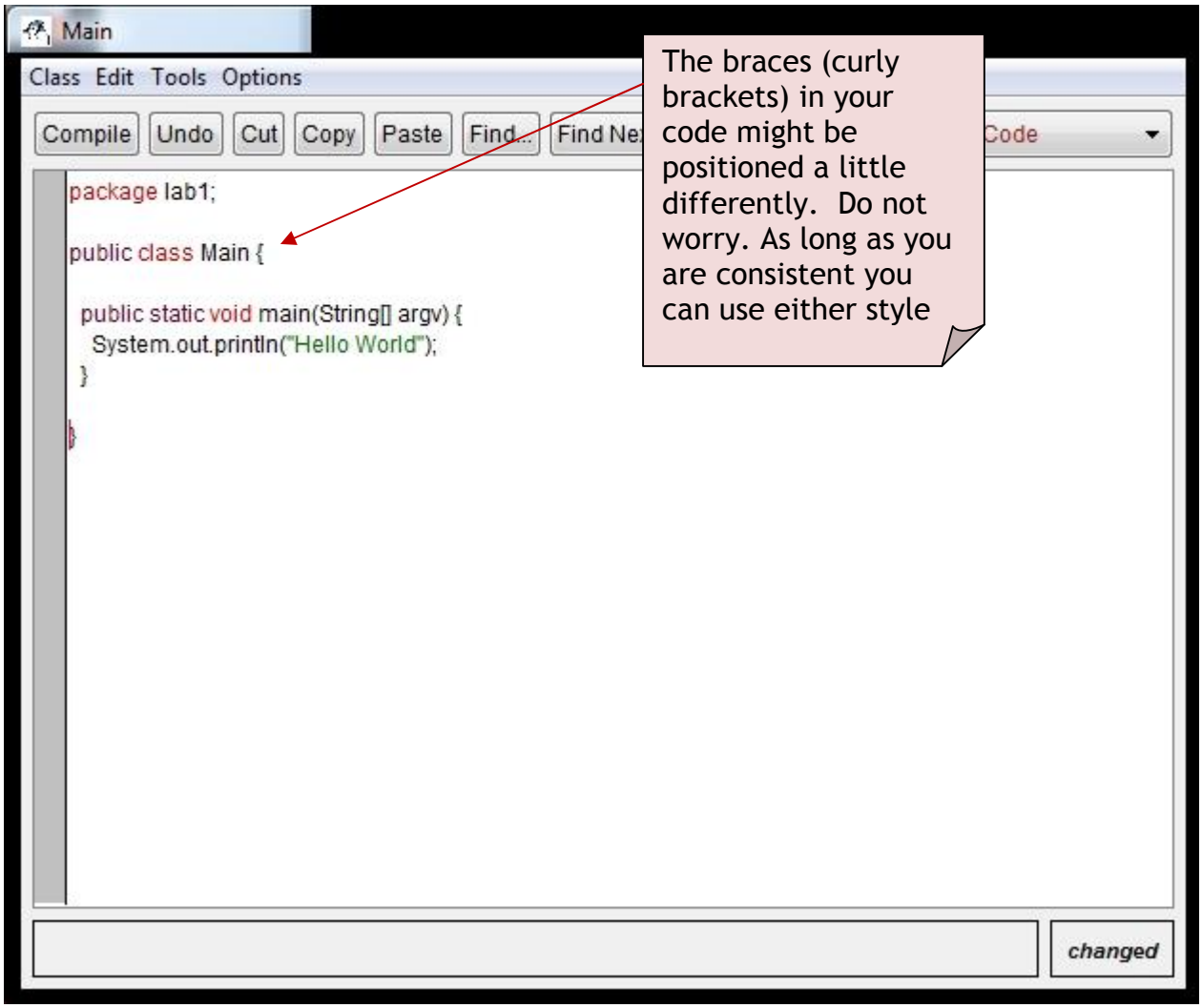
Completing Main.java

Enter the following Java code into the window above. **DO NOT COPY AND PASTE THE TEXT. TYPE IT SO THAT YOU BECOME FAMILIAR WITH JAVA SYNTAX.**

```
package lab1;

public class Main
{
  public static void main(String[] argv)
  {
    System.out.println("Hello World");
  }
}
```

When you have done this, the above window should look like it does on the next page.

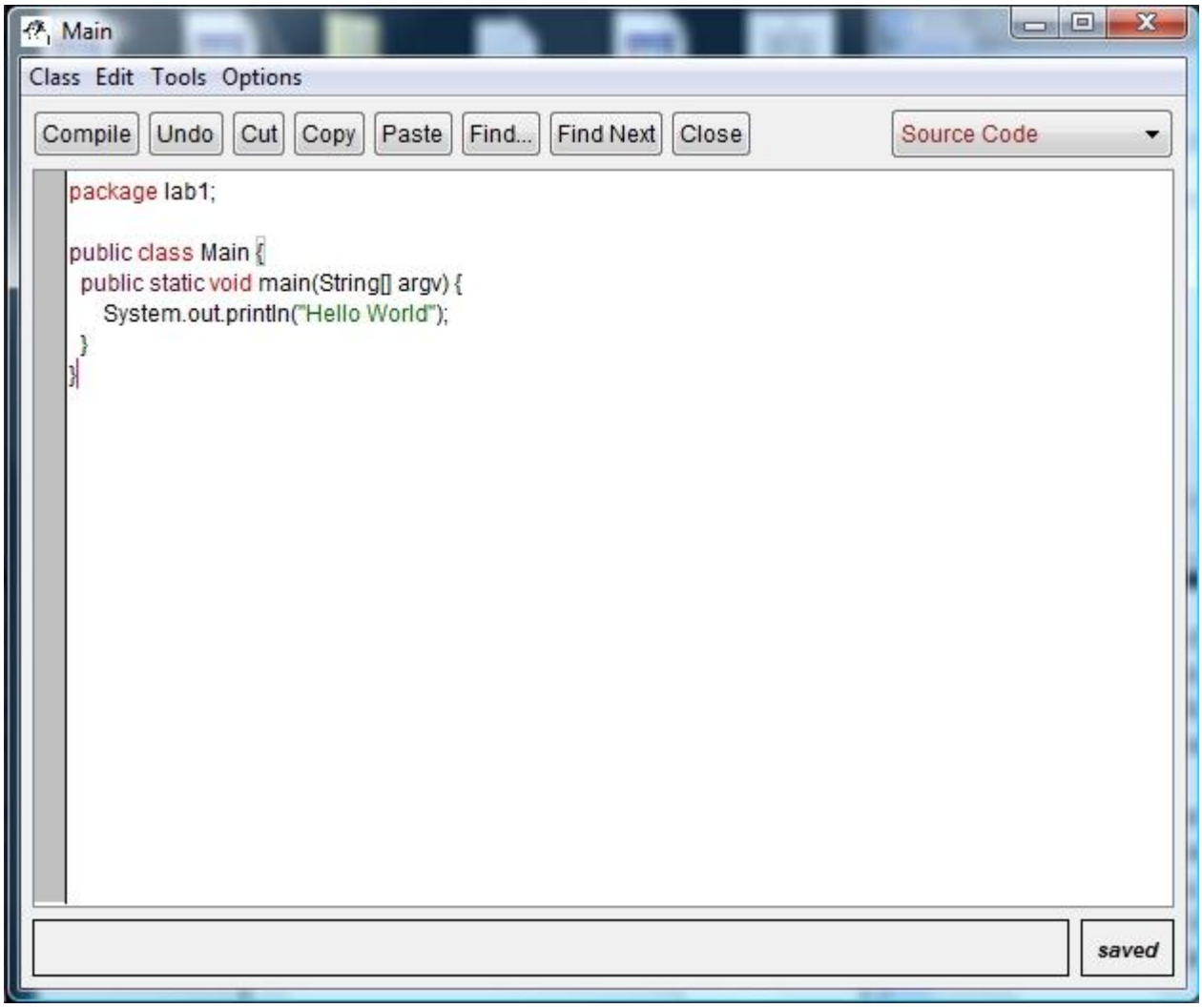


You will notice that BlueJ is highlighting different Java keywords in different colours. This is to help you develop the code. You will get used to seeing words in certain colours, for example, the "Hello World" string in green. If you don't see the string in green you know you have made a mistake, e.g., missed off one of the quote (") characters.

When you enter text into a BlueJ window, information at the bottom right changes to tell you that you have changed the text. This means that you need to save the file to disk first. If you do not save it to disk, when you try to compile it, the Java compiler will see the old version of the file which is not likely to give you a successful compilation. Always save the file after you have completed typing some new code.

You can save the file by clicking on "Class" in the menu bar at the very top and selecting the "Save" option. You will then see your window change to look like that on the next page.

The information in the bottom right of the screen has changed to "saved". This means that your Java code has been saved into a file on disk.



We have a complete Java class. We are now going to compile the class.

Compiling Main.java

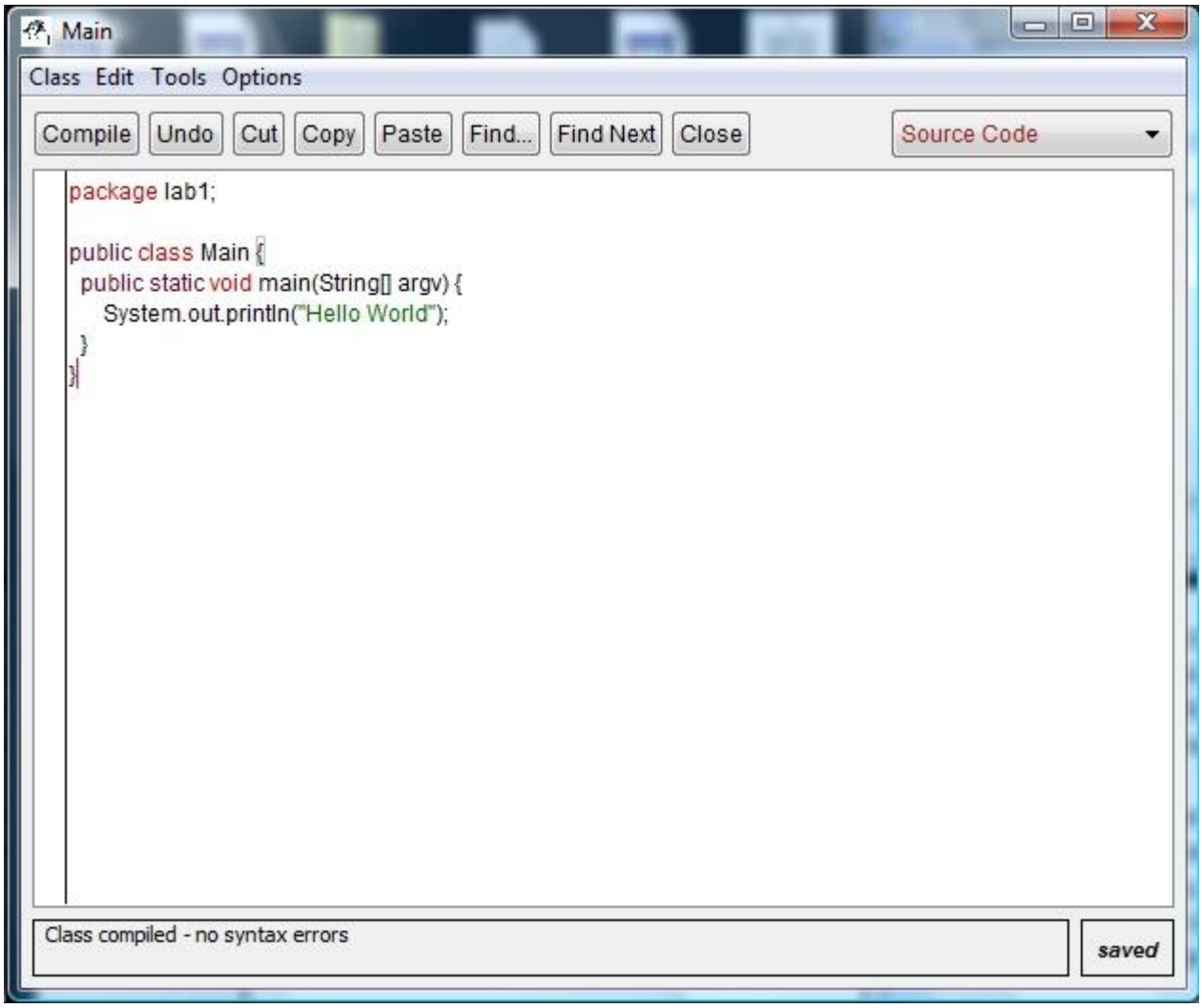
Every time you want to test your code you must first of all compile it.

When you compile code in BlueJ, information on the compilation appears in the box at the bottom of the window shown above.

To compile the above code, click on the "Compile" button in the top left corner of the above window and one of two things will happen.

Possibility 1
In the box at the bottom of the window you will see a message say "Compiling" and if the compilation has been successful you will see "Class compiled – no syntax errors" be displayed as in on the window on the next page.

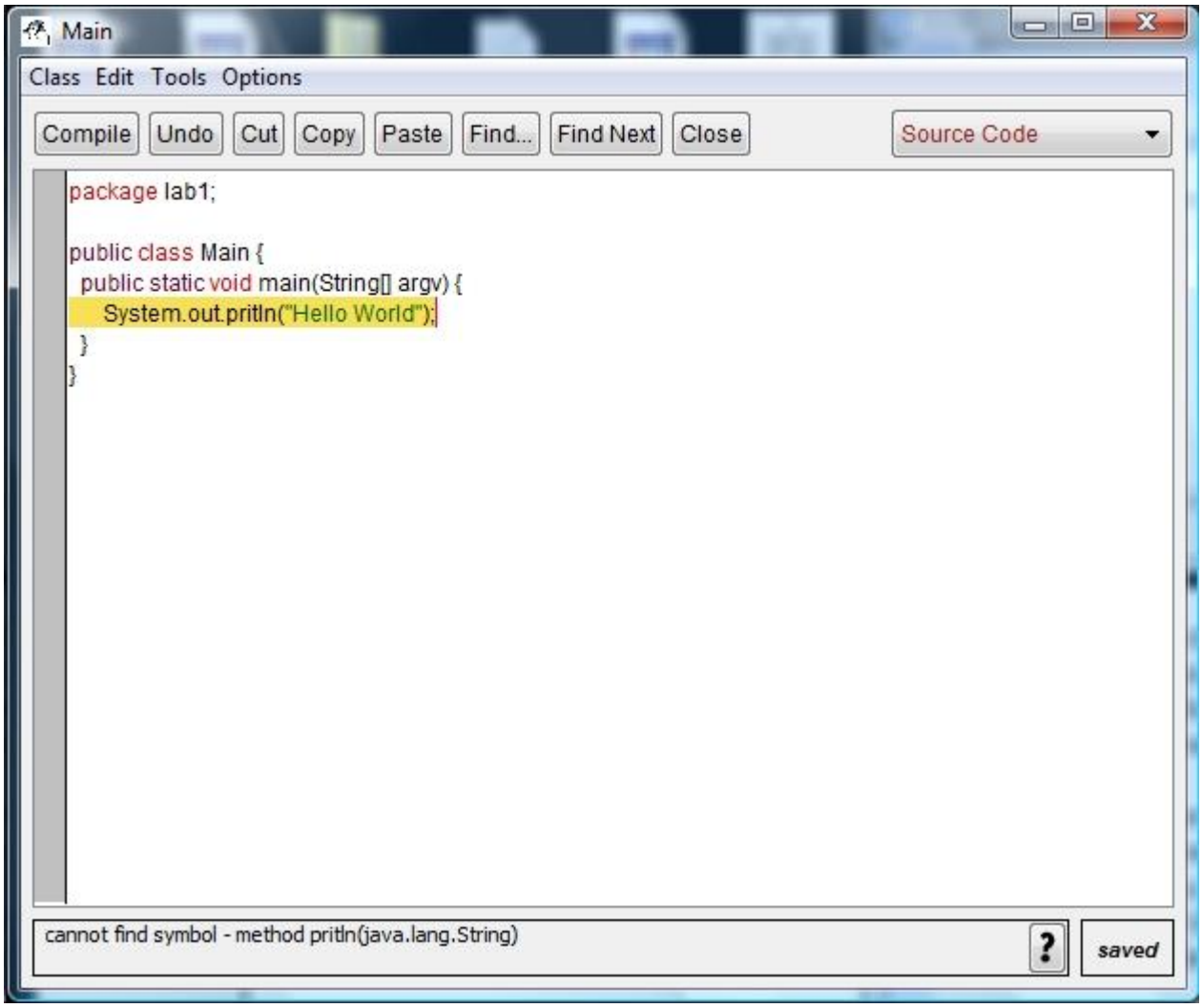
The window below shows at the bottom in the box a message that says the compilation of your code was a success.



Possibility 2

If you have made a mistake in what you have typed, then you will see an error message in the box at the bottom of the above window. See the next page.

There has been an error in the compilation in the window below and the Java compiler has not been able to successfully compile the code you have typed.

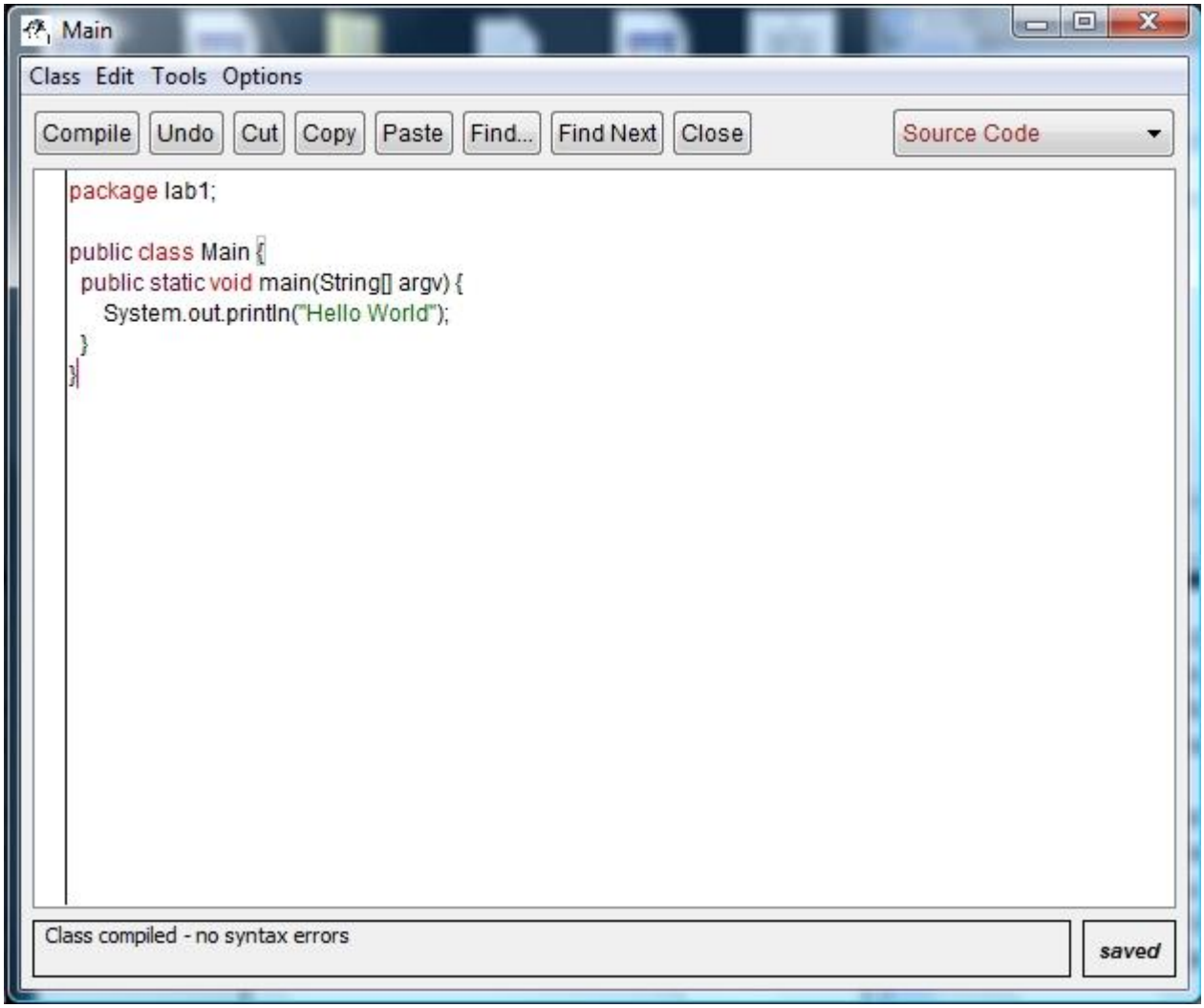


However, BlueJ highlights the line that is in error. Also, if you read the error that has been output in the box below you can see what the error is. I didn't type the name of the "println" method correctly, I made a typographic error (also known as a typo) because I typed "pritln" by mistake. This is an error because there is no "pritln" method defined on the System.out object. Therefore, the above code cannot be compiled.

You have to correct the above typo and recompile the program.

When you compile a piece of Java code you **ALWAYS MUST** get a successful compilation. If you do not, you have errors which must be corrected before you can test your program. See the next page.

As you can see I have corrected the error by typing the correct method name (println).

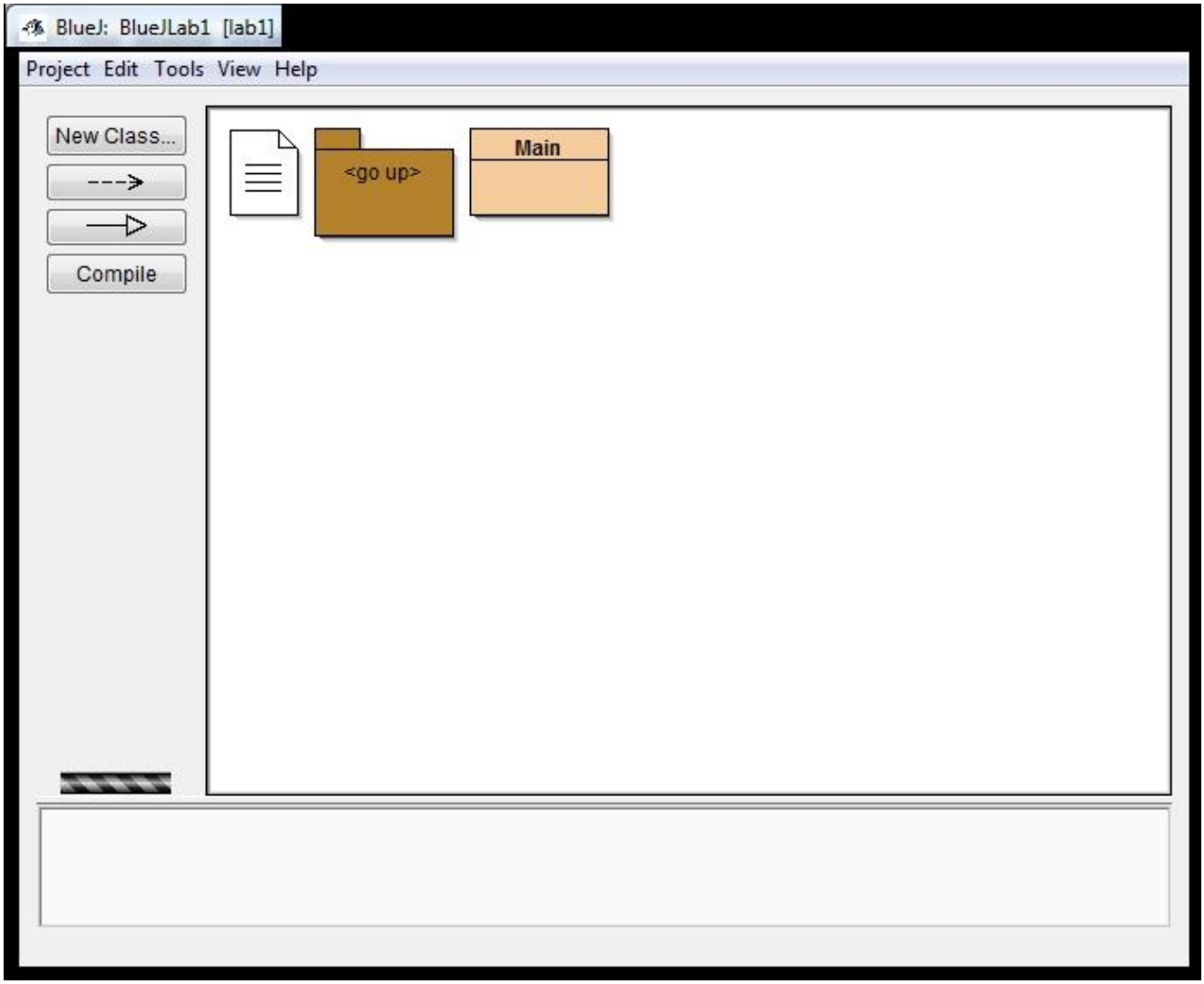


I have saved the file and successfully recompiled the code.

Now we have to test the code by running it. See the next page.

Testing the Code

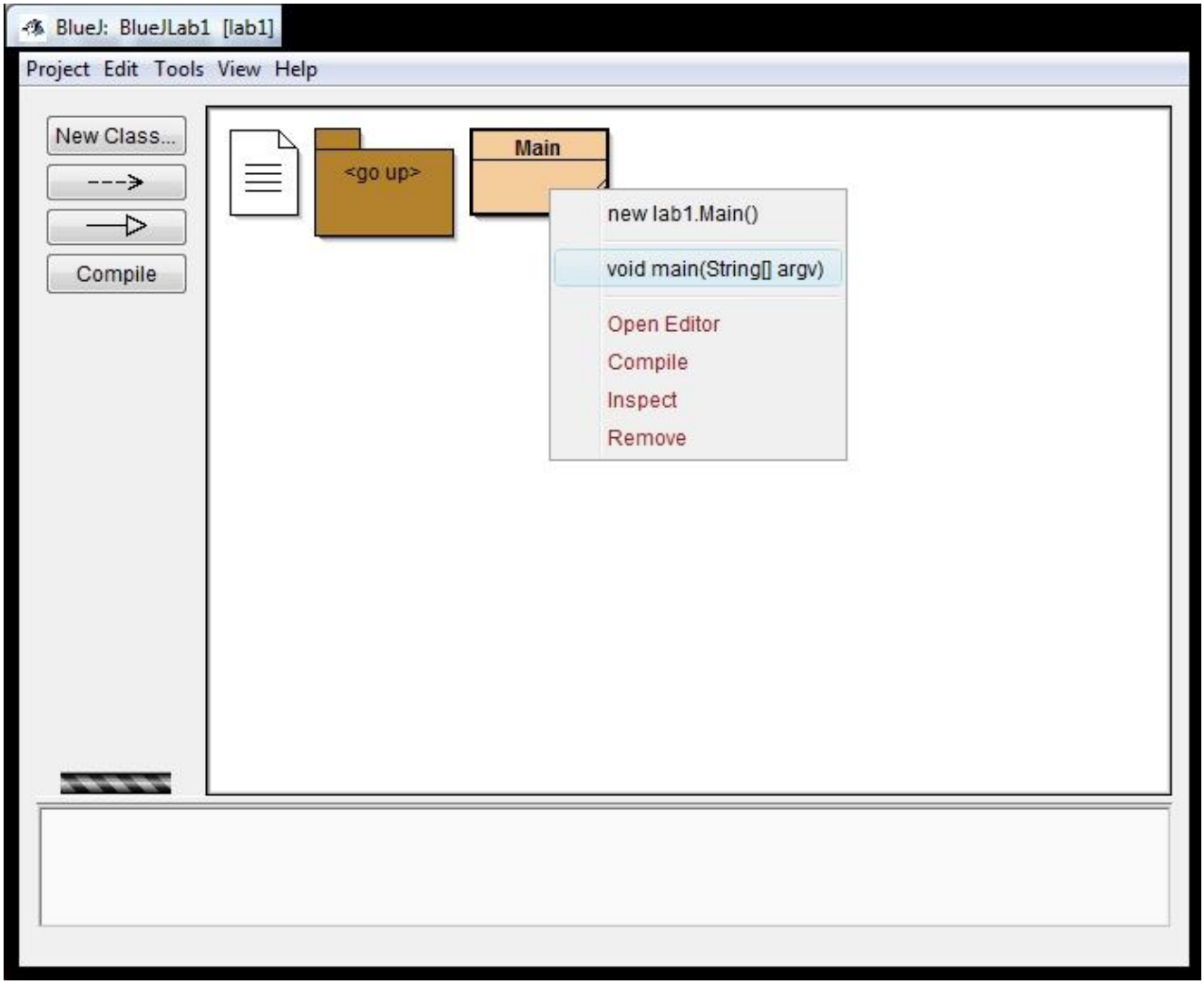
To test the code by running it, go to this window:



Right-click on the orange rectangle that has Main written in it to display a menu as you can see on the next page.

Select the second item down in the list, it will say:

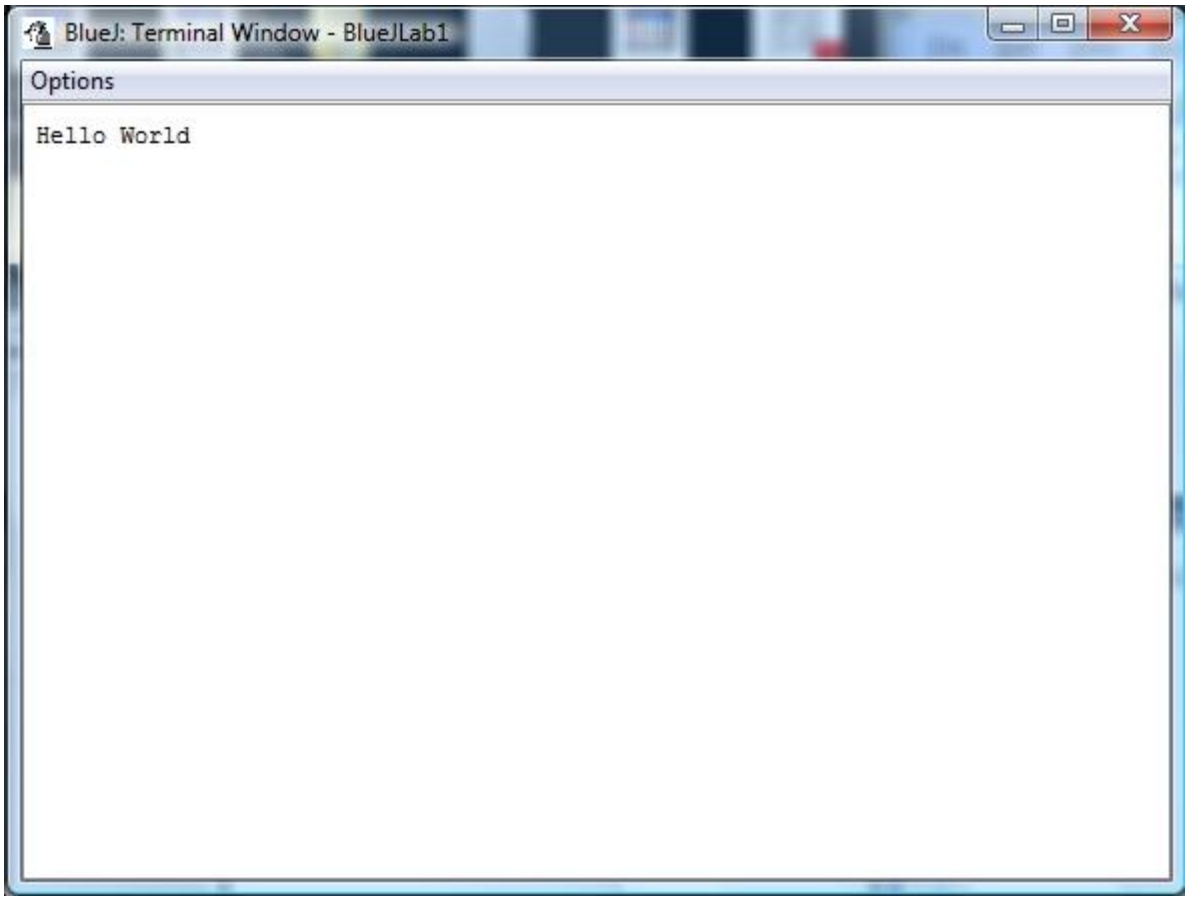
void main(String[] argv)



When you select the above menu option you will see a new window pop up.

Just click Ok. This lets you pass information to the Java program. We are not going to use this at the moment. See the next page.

When you click Ok in the above small window, BlueJ will execute your code in a new window. It should look like this:



Java has executed and the System.out.println method has been called. This method has had "Hello World" passed to it and you have seen this Java String be displayed on the BlueJ Terminal Window (as above). This is because BlueJ takes all the output generated by System.out and puts it into the above window.

As you have seen "Hello World" displayed in the above window you code has been tested and it is running correctly.

Congratulations, you have successfully written, compiled and tested your first Java program.

SOME IMPORTANT THINGS ABOUT JAVA CODE

- Java source code is written in a class
- A class contains methods
- A method contains Java statements
- The semicolon is used to mark the end of a statement
- Statements are separated by semicolons
- Comments are used to annotate you program
- The left brace "{" is used to mark the start of a block of code
- The right brace "}" is used to mark the end of a block of code

You should also notice that unlike some program languages (such as Python) Java code variables and constants must be declared explicitly. Please review the slides on Java primitive data types (Lecture 1).

There is also one aspect of the program that you might not understand now. The construct **`System.out.println(….)`** is a method (again this is like a function) that is used to display information. It is similar to `printf` (in Python) or `printf` (in C) except that is provided by an object. The object that provides this method is named `out`. It is a part of a class named `System`. There is a whole lot more to know about `System.out.println`. For now, please just use it as shown in the example to display a string. The string that is displayed can be built from an expression using the "+" as a concatenating operator. All the following are valid:

```
//Join two strings
System.out.println("Hello" + "World");

//Join a string to a numeric value
System.out.println("Mileage: " + mpg);
```

There is a whole lot more to the `System.out.println` method. Just use it as shown in the example for now.

A Few Words About Errors

Errors that occur during programming can be classified as follows:

Syntax Errors indicate where you have type code that violates the rules of the programming language. For example, forgetting to put a semicolon where one should be.

Runtime Errors occur when your program tries to carry out illegal operations. For example, an attempt to divide a number or zero will cause runtime errors.

Logical Errors are perhaps the most frustrating type. These occur where a program compiles and runs without crashing but does not produce the correct result. Logical errors are caused by programmer mistakes.

Locating Your Files

BlueJ has actually created folders on your computer's desktop (or other location that you specified). It is important that you know how to locate this folder so that you can upload the correct files for submission.

Using the **MyComputer** icon from the Windows Start Menu (or other means) open an explorer window.

Go to the location in which you created the BlueJ project. You should see a folder named **BlueJLab1**. This folder contains the packages that are in your project. Each package is a folder within the project folder. The class you created will be in the package folder. You should see several files with the name **Main**.

- The file with the name **Main.java** is the file you created (the Java source code file).
- The one with the name **Main.class** contains the bytecode that was generated when you compiled the **Main** class.

A Few Words About bytecode

When traditional programming languages (such as C) are compiled, the code that is produced is machine language. Machine language is the code that a computer's processor understands. Unfortunately, different processors might not use the same language. Consequently, the compiled program might not be portable (i.e. the code might only run on a limited set of computers.

The Java compiler uses a different. The compiler produces processor-independent code (called **bytecode**) that is highly portable. Bytecode does run directly on a processor. Instead, bytecode is run on what is called the Java Virtual Machine (JVM). The JVM runs works by interpreting bytecode and issuing machine language commands to the processor. The JVM can be downloaded from the Internet for a wide range of processors. In tis regard the Java programming language is **Architecture Independent**.

Changing Main.java

You now have a running BlueJ environment and a Main.java class that exists in the package called lab1.

From now on, please remember the following:

1. Always save the file after you have made changes.
2. Always compile your class after you have made changes. If you forget to do this the .class file (the bytecode) will be out of date. BlueJ helps you by showing hashed lines in the symbol for a class if it has not been compiled since last changed,
3. You must get rid of all errors (highlighted in yellow) before you can test your code

Lab Exercise (for submission)

You will now write a program that prompts you to enter some values, carries out some calculations, and outputs a result. The problem being solved is calculation of mileage for a vehicle.

1. Create a new class named Mileage. As before, BlueJ will automatically generate sample code. Delete all the code except for the class header and the braces that enclose the class body.
2. Using the Main class as your reference, create the main method for your new class. Leave the method empty (i.e. do not type any code between the braces}
3. Compile your program now to ensure that you have not made any mistake in typing the code.

4. Add the following code to your main method.

```
int miles;
double gallons, mpg;

Scanner scan = new Scanner (System.in);

System.out.print ("Enter the number of miles: ");
miles = scan.nextInt();


System.out.print ("Enter the gallons of fuel used: ");


gallons = scan.nextDouble();


mpg = miles / gallons;
System.out.println ("Miles Per Gallon: " + mpg);
```

5. Compile and run the program. This time the program will request input from you. It will ask you to enter some miles and then enter the number of gallons used to travel that many miles. Enter 100 for the miles and 4 for the number of gallons. Has your code given you the correct result of 25?

6. Run the program again and this time, enter 100 for the number of miles and 5 for the number of gallons. What result do you get? Is it what you expected?

7. Run the program again. This time enter 100 for the number of miles and 0 for the number of gallons. What happens? What type of error (syntax, logical, or runtime) did you get?

8. Modify the program so that it also outputs kilometers per liter. A kilometer is 0.62137 miles, while a liter is 1.76 pints (Remember 1 gallon = 8 pints). Your program must still accept its inputs as miles and gallons.

9. Add comments to your program. Most importantly, make sure that your student id is included in the class that contains the main method.

Submitting Your Lab

It is very important that you follow submissions strictly. Otherwise the job of marking your lab becomes extremely difficult. Suffice it to say that you will be given a grade of **0** for any submission that is not made in accordance with instructions. For example, do not submit a .zip, .rar archive unless that is what is asked for. In most cases you will be asked to submit a Java Archive (.jar file) which will contain the **compiled classes along with your Java source code**. A .jar file can be created as follows.

1. From the Project menu, select the command "Create jar file.."
2. Select the following options from the dialog box that is displayed:
   - The name of the file that has the main method for your application;
   - Include Source files.

   Do not include BueJ project files.

3. Save the **.jar** file to a location of your choice. Better yet, save it in the folder that BlueJ created for the project. Name the file **comp1161_lab1.**

4. From the course web site, select the upload link for your lab and follow the instructions to upload your submission.

THANK YOU!