

Lesson 1: Introducing Java



Introduction

"Computers are good at following instructions, but not at reading your mind."

—Donald Knuth, computer scientist and professor emeritus, Stanford University

Welcome to *Introduction to Java Programming*! My name is Alan Simpson. I'll be your teacher for the next six weeks. My interest in programming started way back in the 1970s when the Apple computer was brand new. Since then I've written over 100 books on programming, databases, and web design, published in dozens of languages throughout the world. Currently, I manage the Applications and Databases team for my county's Information Technology department.

My master's degree is actually in the psychology of learning, and I've been teaching programming and databases for my entire adult life. I enjoy writing computer code and teaching others how to write code. Some of you may recognize me from my HTML, CSS, and JavaScript courses here.

Over the next few weeks, we're going to introduce you to *Java*, a powerful computer programming language that people use for lots of different tasks.

If you've ever tried to learn a new language, you're already pretty familiar with the skills you're going to need to master Java. To become fluent in any language, all you really have to do is master a bunch of new vocabulary words and then learn the rules for stringing those words together.



We use the Java language the same way we use any other language: to communicate. The big difference between Java and human languages like Spanish or French is that we use Java to communicate with computers.

By the time you finish this course, you'll have a good understanding of what Java is, what it can do, and how it works. You'll be able to create your own Java programs from scratch—even if you have no experience in computer programming right now.

Ready to start? Then please proceed to the next chapter.

What Java Is

Java is a programming language, but that's not such a great definition for someone who hasn't used one before. After all, how many people really know what that means?

Let me explain a bit better. A *programming language* is a tool we use to create computer programs.

And what on earth is a computer program, you may ask?

Well, computer programs (which are sometimes referred to as *software* or as *applications*) are just lists of instructions that tell a computer to perform a series of tasks in a certain order.

In essence, then, Java is nothing more than a tool you can use to write instructions that you want your computer to carry out.

Let me give you an analogy.

Once upon a time, back in the dark ages before microwave ovens, my uncle Bill had to fend for himself at home while the rest of the family was out. This particular incident happened while he was in high school. His mom left him a note telling him how to heat up a TV dinner. The note said:

Text equivalent start.

Bill, the TV dinner is in the freezer. To cook it, preheat the oven to 450. After the oven is hot, put in the TV dinner for 30 minutes before you eat it. When you finish eating, throw the tray in the trash, and wash any silverware you used. We should be home by 10:00

Text equivalent stop.

That little note you just read fits my definition of a program, doesn't it?

Remember, I defined a computer program as a list of instructions that tell a computer to perform a series of tasks in a certain order.

Okay, so Bill isn't a computer. But if he were a computer, that note would be a program. After all, isn't the note just a list of instructions asking him to perform a series of tasks in a certain order?

Look at it again. What's it say?

First, find the dinner. Preheat the oven, and then heat the food. Then eat it. When you're done, clean up.

If you've ever written a note similar to the one Bill's mom left for him, you have what it takes to create simple computer programs.

There's just one important thing to remember: Computers aren't anywhere near as smart as people are. You and I can find TV dinners in a freezer because we're familiar with them. Unfortunately, computers don't have any such experience. If you wanted to teach a computer how to do that, you'd have to start at the beginning. In other words, you'd need to teach the computer how to recognize a freezer and a TV dinner, how to open a freezer, and so on.

And of course, you'd also have to teach the computer how to move before you could accomplish any of the above. If your computer is anything like mine, that isn't very likely.

But in another way, my little story will show you how literal computers are. My uncle is somewhat literal too. He did exactly what the note said, just as a computer would. And his mom forgot one important step: She didn't tell him to take the dinner out of the box before heating it! She assumed he would know to do that. Instead, he just stuck the TV dinner, box and all, into the oven. In a few minutes, he had a nice fire going in the kitchen!



Computers are like that. They do exactly what you say and in the order you say it. So we can make no assumptions when dealing with computers. In our programs, we'll need to tell them every detail of what we want them to do.

What Java Isn't

You may have heard of JavaScript, a programming language that helps people design interactive sites. Java and JavaScript are different programming languages that are only distantly related—even though their names are similar.



In the next chapter, we'll download the software you need to write and run your Java programs.

The Java JDK

The Java SE Development Kit (JDK) is the platform we'll use to develop our Java programs. Java has several editions:

- **Java ME (Micro Edition)** is for developing programs for small devices like cellphones, digital video recorders, cameras, microwave ovens, and so on.
- **Java SE (Standard Edition)** is the one we'll use. It's the most commonly used edition for development and for training.
- **Java EE (Enterprise Edition)** is for large organizations that need to develop complex applications that large numbers of people can use across networks.



Java's Platform Independence

Java is one of the first programming languages that was designed to be *platform independent*, which means that a program written in Java will get along fine with almost any processor and almost any operating system.

Before Java came along, programmers had to develop one version of a program for a Windows user, a second version of the same program for a Macintosh user, a third version for a Linux user, and so on. As you might imagine, the process was complicated, time-consuming, and expensive.

But those days are history. With Java, you only need to create a program once, and it'll work on all those platforms—and others as well. This is a tremendous step forward in the world of computer programming. It's also why Java has become so popular for programs that run over the Internet. Internet programmers have no idea how many different types of computers their programs will need to run on. So Java makes their lives much easier!

What's really exciting is the fact that Java programs can run on almost any computer-like device, as long as it has a processor and an operating system. You could (at least in theory) write a single program on your home computer and then see the thing run not only on other types of computers but also on Java-enabled cellphones, handheld organizers, home theater systems, your car, or even your microwave oven! (Okay, that may be taking things a little far, but you get the idea.)

Because Java makes multi-platform software development relatively effortless and convenient, it's one of the most popular programming languages in the world.

Downloading the JDK

Okay, that's enough background. Let's download the JDK.

First of all, if you're running a Mac with Mac OS X, the JDK is already part of your operating system, so you can skip to the next chapter to get BlueJ.

If you're using Windows or Linux, please click this link now:

[Java Download Site](http://www.oracle.com/technetwork/java/javase/downloads/index.html) (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
(<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)

That should get you to this page at Oracle's website:










The Lesson 1 Discussion Area contains videos showing how to perform all of the downloads in this lesson.

Near the center of the screen, click **Download JDK Download**, and on the next page, scroll down until you see the list of available versions, which will look something like the example below. If you're using Windows 7, 8, or 10, click the Windows x64 Installer link and follow the on-screen instructions. For Mac, you don't need to download anything. For Linux, you'll need to choose the version with which you're most familiar.

Java SE Development Kit 13.0.2

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

| Product / File Description | File Size | Download |
|--------------------------------|-----------|--|
| Linux Debian Package | 155.72 MB |  jdk-13.0.2_linux-x64_bin.deb |
| Linux RPM Package | 162.66 MB |  jdk-13.0.2_linux-x64_bin.rpm |
| Linux Compressed Archive | 179.41 MB |  jdk-13.0.2_linux-x64_bin.tar.gz |
| macOS Installer | 173.3 MB |  jdk-13.0.2_osx-x64_bin.dmg |
| macOS Compressed Archive | 173.7 MB |  jdk-13.0.2_osx-x64_bin.tar.gz |
| Windows x64 Installer | 159.83 MB |  jdk-13.0.2_windows-x64_bin.exe |
| Windows x64 Compressed Archive | 178.99 MB |  jdk-13.0.2_windows-x64_bin.zip |

Getting ready to download

Remember, if you're using a Mac, you don't need to download anything. But if you want to check to make sure you have a recent version of Java installed, see the Lesson Videos post in the Lesson 1 Discussion Area titled **Mac - Check your Java Version**. For Windows 7, 8, or 10 see the **Windows - Download Java** video there. If you get stuck, you can also post a question in the Discussion Area, and I'll get back to you as soon as possible.

After you've completed the Java installation, you *won't* have a new Java icon on your menu or desktop because the Java Development Kit is an interactive app. So please don't go looking for that or be concerned about it. The Java Development Kit stays in the background until you call on it to execute some Java code that you've written.

To write Java code, you'll need to use a code editor. In this course, we'll use BlueJ because it's free and relatively easy to work with. You'll learn how to download and install the BlueJ editor in the next chapter.

Downloading BlueJ

First let's talk about what BlueJ is and why we're going to use it.

BlueJ is an *integrated development environment*. An IDE is a set of tools that helps you create software programs. It provides an editor, a compiler, a debugger, and other tools that we'll use over the next few weeks.

You can compile and run Java programs using just the JDK, without an IDE. But to do that, you have to use the old *command-line* computer interface that requires you to type commands instead of pointing and clicking. For example, here are the commands that I entered to compile and run one program:

Text equivalent start.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Merrill>cd Documents
C:\Users\Merrill\Documents>cd Java
C:\Users\Merrill\Documents\Java>cd "BlueJ Examples"
C:\Users\Merrill\Documents\Java\BlueJ Examples>cd hello
C:\Users\Merrill\Documents\Java\BlueJ Examples\hello>javac Hello.java
C:\Users\Merrill\Documents\Java\BlueJ Examples\hello>java Hello Hello, world
C:\Users\Merrill\Documents\Java\BlueJ Examples\hello>
```

Text equivalent stop.

Unless you're particularly fond of this type of interface, it's probably easier for you to click a button than enter those commands. So we'll use BlueJ as our development tool.

I chose BlueJ because it's designed for students, and it's easy to learn and use. If you already have another IDE you want to use (like NetBeans, Eclipse, or IntelliJ IDEA), feel free to use that. But my discussion in the lessons will center on BlueJ.

BlueJ provides several tools for us in a window environment that most of us are more familiar with.

- It contains a *text editor* that we'll use to create and edit our programs.
- It contains links to the Java compiler that will let us compile programs with the click of a button. This link will also capture messages from the compiler and display them for us. (We'll talk more about text editors and compilers in Chapter 5.)
- It'll execute our programs with a mouse click instead of requiring us to type commands.

A Few Words of Encouragement







BlueJ is an example of what programmers can do in Java. It's written entirely in Java using Java classes and methods. So don't be discouraged if it seems like we're moving slowly. In less time than you think, you'll be able to write programs that'll do some interesting things.

Now let's download BlueJ. To find the download file, start with the BlueJ home page link in this lesson's Supplementary Material or click this link, and you should see the page below:

[BlueJ Download Site](https://bluej.org/) (<https://bluej.org/>) (<https://bluej.org/>)

Download and Install

Version 4.2.2, released 4 October 2019 (fixes extra space in terminal, Submitter preferences, and more)

| Windows | Mac OS X | Ubuntu/Debian | Other |
|--|---|--|--|
|  |  |  |  |
| Requires 64-bit Windows, Windows 7 or later. Also available: Standalone zip suitable for USB drives. | Requires OS X 10.11 or later. | Requires 64-bit, Debian buster or Ubuntu 18.10 or later. Please read the Installation instructions . | Please read the Installation instructions . (Works on most platforms with Java/JavaFX 11 support). |

Note: BlueJ now uses Java 11+, which requires a 64-bit operating system, which 95+% of users will have. For 32-bit operating systems, download [BlueJ 4.1.4](#) instead.

[Download previous versions or source code.](#)
The copyright for BlueJ is held by M. Kölling and J. Rosenberg.
BlueJ is available under the GNU General Public License version 2 with the Classpath Exception (full license text, licenses for third party libraries).

Useful Resources

[BlueJ home page](#)

There may be more than one release available on the download page. The BlueJ developers sometimes post preview and beta releases there so people can try them out. To minimize problems, though, please stick to the official release.

Tip:



If you prefer videos, take a look at the videos titled [Mac - Download BlueJ](#) or [Windows - Download BlueJ](#) depending on whether you're using a Mac or Windows.

Several versions of the official release of BlueJ are available at the download page. There are versions for Windows, for the Mac, for Linux systems like Ubuntu, and one for all other systems. So be sure to get both the official release and the correct version for your computer. At the time of this writing, the version is BlueJ 4.2.2, but if you see a later version, go ahead and download that instead.

One more note for Mac users: There are two releases for Macs based on the version of Mac OS X you have. If you have version 10.7.3 or newer, BlueJ includes Java version 7 in its download. If you have an older version of OS X, you'll need to get the "legacy" version of BlueJ that runs with Java 6, which is already installed as part of OS X. If this is a problem for you, please let me know in the Discussion Area.

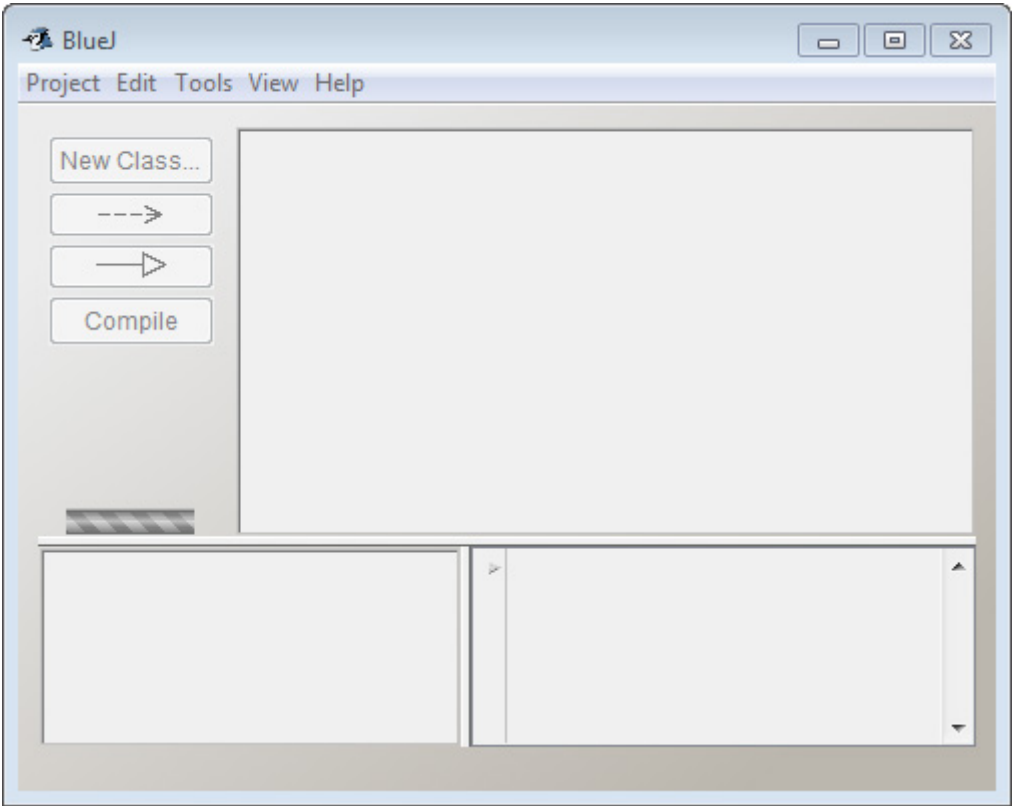
Once you've found the correct release and version, click the filename to its right, and save the file onto your computer. Again, the easiest place to find it quickly is probably your desktop.

Click the **Windows** or **Mac** link, depending on which kind of computer you're using, and follow the instructions to install BlueJ. On a Mac, I suggest you drag the BlueJ app to your Applications folder so it's easy to find. For extra help, see the Mac or Windows BlueJ video under **Lesson Videos** in the Lesson 1 Discussion Area.

Finally, you'll need a place to store the sample programs you'll be writing. I already have some sample code you can work with in a folder you can use for any Java program you write. To download that folder, click this link:

[Download Example Files](#)

Once you've downloaded the zip file, extract the Examples folder and put it on your desktop. See the Mac or Windows Download Examples video in the Lesson 1 Discussion Area, under Lesson Videos for specifics.

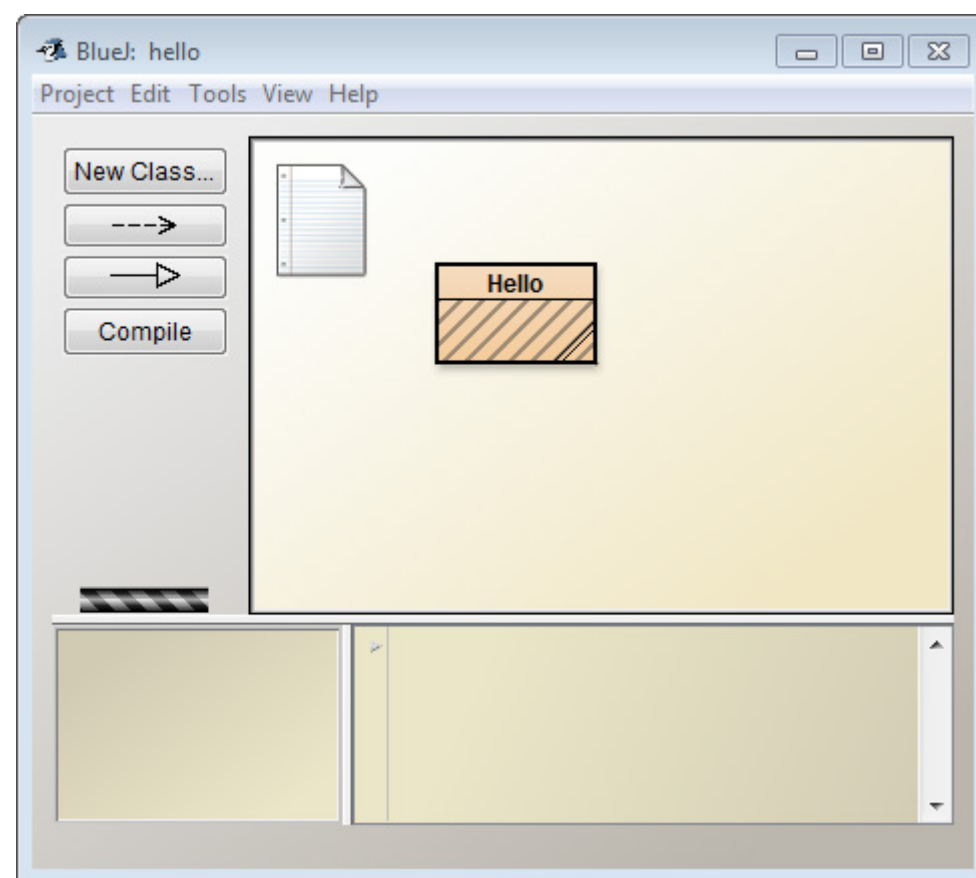


BlueJ window

Don't worry about all the different menus and buttons at this point. We'll be using them later. For now, just follow these instructions to make sure everything is working right.

In this video, I'll walk you through the process of opening and running a project in BlueJ.

Click **Project > Open Project**. The normal file open dialog box will appear. Open the examples folder you saved earlier, and within it, open the **Hello** project. You should see the project appear in the BlueJ window so that it looks something like the image below:

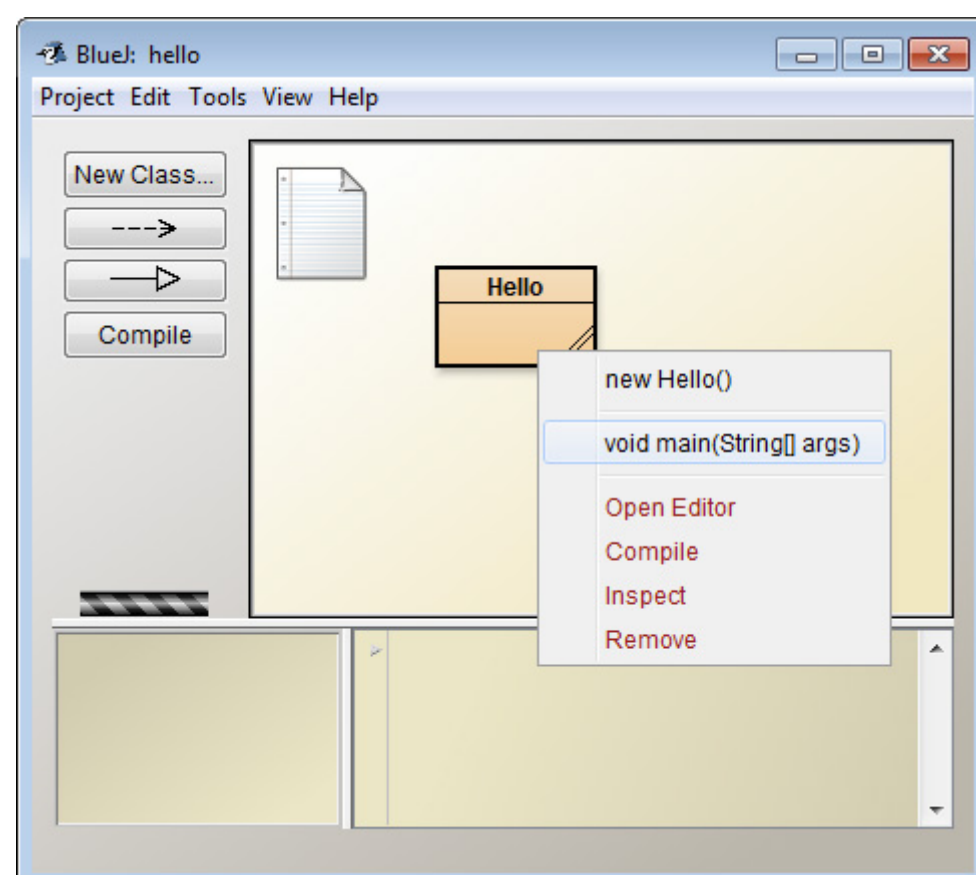


The Hello project

The icon named Hello represents a simple Java program that writes the phrase, "Hello, world" to a window on the screen. The diagonal blue lines in the box mean that the program hasn't been compiled yet. (I'll explain more about compiling in the next chapter.) So click the **Compile** button to tell BlueJ to compile the program.

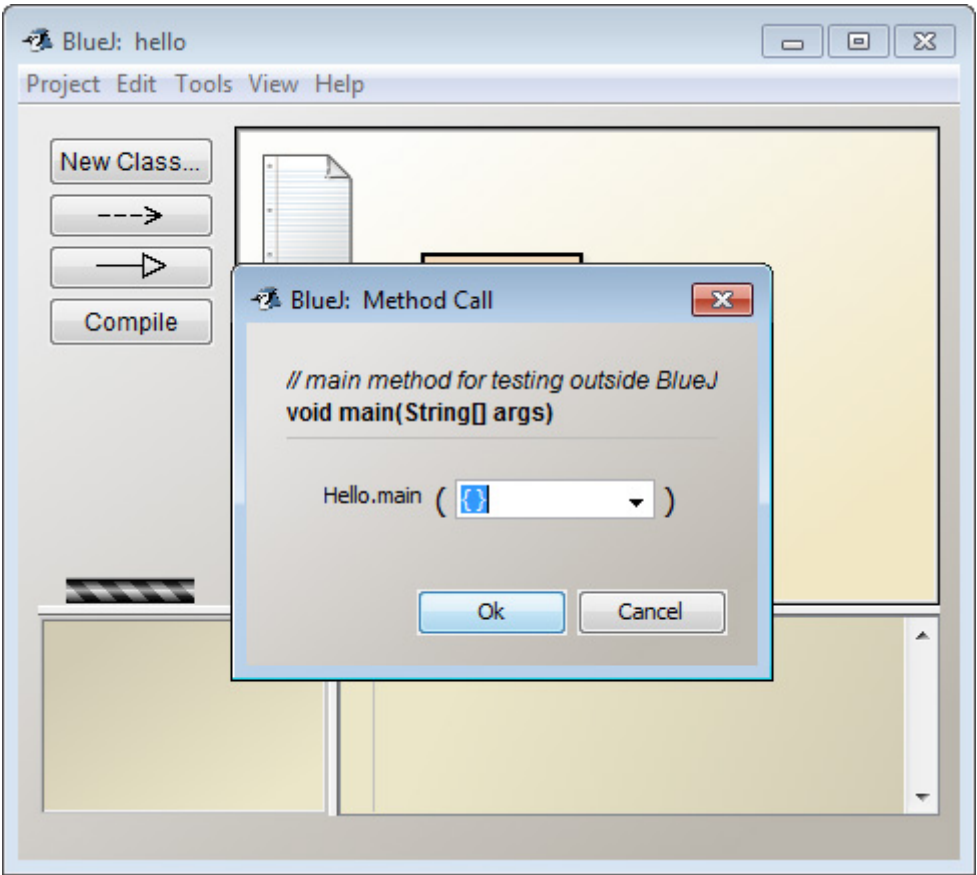
The message *Compiling* will appear in the bottom left corner of the BlueJ window for a few seconds, and when the program has compiled, it'll change to *Compiling . . . Done*. The blue lines will also disappear from the Hello box. That means the program is compiled and ready to run.

To run it, right-click the Hello box (Mac users should hold down the *CTRL* key while clicking) and choose the second option from the drop-down menu, "**void main(String [] args)**" as in the image below. This is the signature of the program's *main method*, where the highest-level logic of the program goes, and where Java always starts when it executes a program.



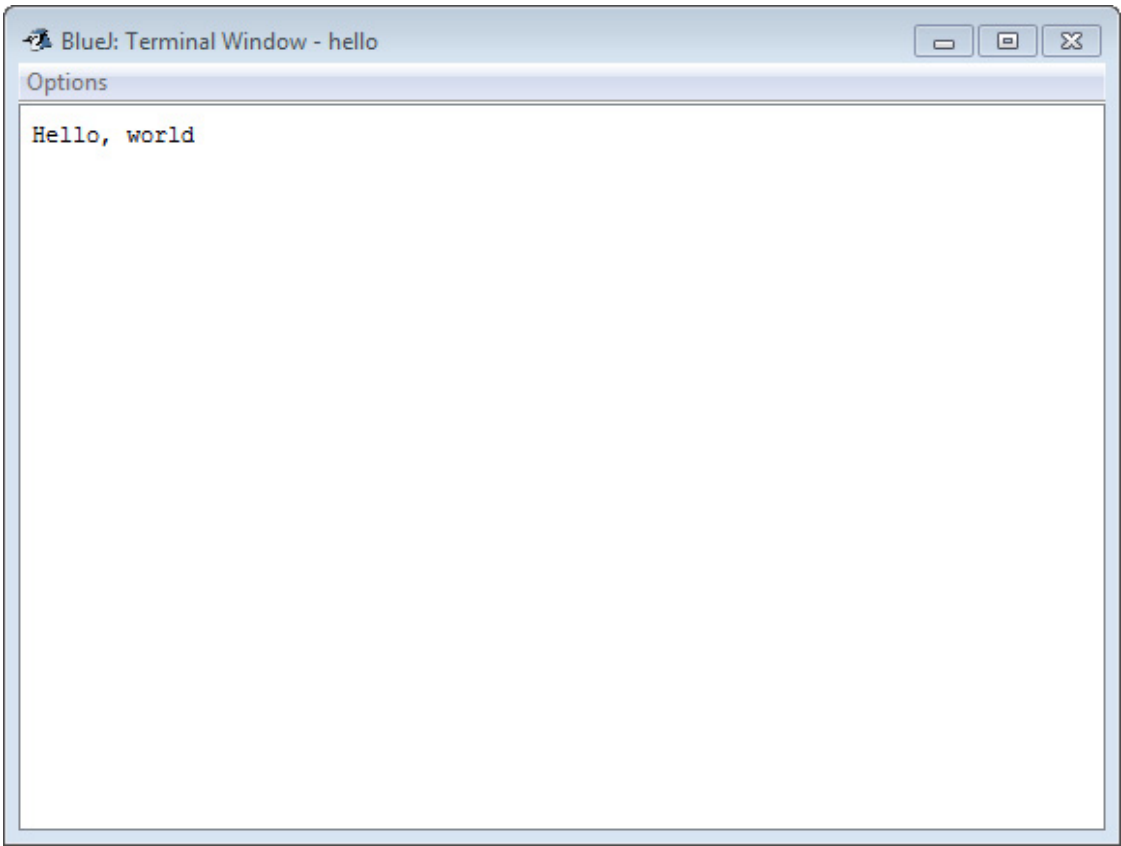
Starting a program

When you click that menu option, you'll see a *Method Call* dialog box open up like the one in the image below. This is a dialog box that BlueJ uses to run a Java method. Click **OK** to run the program.



Method Call dialog box

When the program runs, all it does is display some text onscreen. BlueJ opens a new window, called the *terminal window*, to show you the text. We'll use this window for a lot of our programs' input and output. It should look like this:



BlueJ's terminal window

Congratulations! You've just compiled and run your first Java program!

Creating and Running a Java Program

Now that you have Java and BlueJ installed and running, let's take a step back and look at the process of creating and running a Java program. Writing and running a Java program takes several steps. I've numbered them from one to four in the diagram:

Text equivalent start.

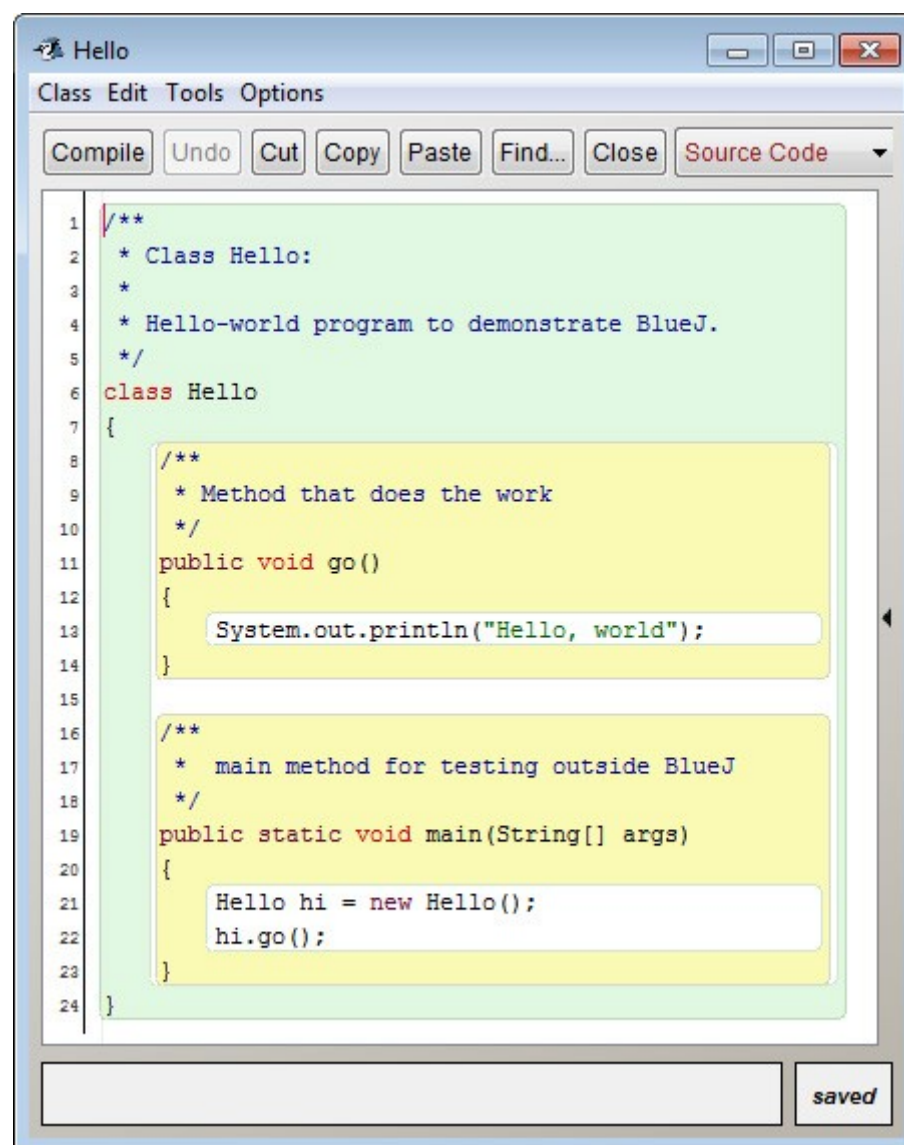
The edit, compile, and run cycle begins with the text editor. It continues on to the source code and the compiler. If there are syntax errors, the process goes back to the text editor. If not, it moves to Java Bytecode. From there, it generates either applets or applications. Applets go to a Web browser and an applet viewer. If there are runtime errors, the process goes back to the text editor. Applications go to a Java interpreter. As with the applets, if there are runtime errors, the process goes back to the text editor.

Text equivalent stop.

I know this looks complicated, but I'll walk you through it step by step. The squares in the diagram are files, the circles are tools, and the arrows show the flow of the process.

The first step in creating a Java program is to write it using a text editor. You could use any text editor, but we'll use the one that BlueJ provides.

The programs that we write in text are called *source code*. (This is true of any programming language, not just Java.) If you'd like to look at the source code for the program you just ran, you can do that by double-clicking the program icon in the BlueJ project window. But whatever text editor a programmer uses, the product is the same: Java source code.



Code editor window displaying 'Hello' class source code

As the diagram shows, the text editor saves the source code in a source code file.

Once we've written a program, we have to get it into a form that a computer can run. So Step 2 in creating a Java program is to compile it. Special programs called *compilers* do that for just about every programming language. A compiler translates the source code you've written into a form that a computer can understand. In many cases, this is a string of ones and zeroes.

Compilers also let you know about *syntax errors*, such as spelling a name wrong, leaving out some punctuation, and so on. Any program more than a few lines long will likely have some syntax errors in it the first time it goes through the compiler. When there are syntax errors, we fix them by editing the source code file, and then we save and compile it again.

After we fix syntax errors, the compiler creates a new file. Some compilers create a finished file that will run directly on the computer. Files like that are *object code*, and they're the strings of ones and zeroes I mentioned a minute ago.

Java doesn't do that, though. As the diagram shows, the Java compiler creates something called *bytecode* and puts it into a file. It's not object code, since it won't run directly on a computer. But in this case, that's a good thing, since object code can only run on the type of computer it's created for.

If you remember, one of Java's main benefits is that you can compile a Java program without knowing where it will run. That's often the case with Internet programs. It works because the bytecode gets translated into object code on the computer where it's executed. I can take the bytecode from a PC, and it should run just as well on a large computer like a mainframe as long as the mainframe has a *Java Runtime Engine* (JRE), which is sometimes called a *Java Virtual Machine* (JVM).

But anyway, back to the main topic. Once I have a bytecode file, I can run it (Step 3 in the diagram). If my program is an *application*, I can run it as a separate stand-alone program, like BlueJ.

If my program is an applet, I'll need a Web browser or Java's AppletViewer to run it. *Applets* are small applications (which is why they're called applets) that are specially designed to run in a browser so they'll work on the Internet. We'll see how to create both types of programs in this course.

Step 4 in the diagram happens only if there are *runtime errors*. If the program crashes or otherwise doesn't do what it should, then the programmer has to figure out why and fix it in the text editor. That starts the whole process again.

The final result, which usually doesn't happen until we've been through this cycle a few times, is a working program that'll run on many different computer platforms and always do what it's designed to do. By the time we finish this course you'll have quite a bit of experience with this process and will be able to go through it with your eyes closed, so to speak.

I know I've thrown a lot of terms at you in this chapter, so try this matching game to help you remember them. I won't keep track of your score—this is just for practice.

Text equivalent start.

Instructions: Read the clue in the first column, and guess the term. Then read the second column for the answer.

| Clue | Answer |
|---|---------------|
| A crash or other glitch that prevents your program from running properly. | Runtime Error |
| A versatile program that the Java compiler creates. | Bytecode |
| A problem such as a misspelled word or incorrect punctuation within code. | Syntax Error |
| A program written within a text editor. | Source Code |
| A program that a particular computer can understand. | Object Code |
| A program that translates source code into object code or bytecode. | Compiler |
| A mini-application that runs within a Web browser. | Applet |

Text equivalent stop.

How did you do? Were some of these terms familiar to you before you started this course?

Summary

We covered quite a bit of ground today, from introducing you to Java to downloading the software we need (both the Java JDK and BlueJ) to compiling and running a Java program using BlueJ. That's quite an accomplishment.

We also discussed the complete process for writing, compiling, and running Java programs. We'll be going through this cycle of steps quite a few times over the next few weeks, so it's a good idea to be sure you understand how it works.

In the next lesson, we'll start writing, compiling, and running our own Java programs. That's when things will really get interesting. I'll see you then!

Next Steps

To finish the lesson, you'll need to complete the steps outlined below. Simply click "Next Up" at the bottom of the page to access the next activity. Or, if you wish to skip around, click the Book Icon in the top-right corner. There you'll find links to all the activities in this lesson. Here are your remaining activities:

- **Check out the FAQs.** Since learning something new usually raises questions, every lesson in this course comes with an FAQs section.

- **Browse the Supplementary Material section.** Here you'll find links to helpful online resources relating to the lesson.
- **Do the assignment.** Get some hands-on practice applying what you've just learned.
- **Take the quiz.** Reinforce what you learned with a short five-question quiz.
- **Participate in the Discussion Area.** Ask questions about anything that came up in the lesson, and share your insights. This is where we'll create a learning community.

Additional Resources

In addition, there are some additional resources you may find helpful throughout the course. Access these resources by clicking the link for Resources listed after Lesson 12. There you'll find:

- **Recommended books and resources.** This is a list of books and other resources that you can consult to extend your learning.