Technical Note No. 104
Compiling and Running a Java Program in Windows

Published: August 27, 2007. Last reviewed on January 12, 2011
By Daryl Close, Professor of Computer Science and Philosophy, Heidelberg University, Tiffin, Ohio

Summary: This note provides instructions on compiling and running a Java program from the system prompt.

Downloading Java to Your Own Computer

1. Download and install the most recent Java Platform, Standard Edition (SE) Development Kit from http://www.oracle.com/technetwork/java/index.html. Under "Popular Downloads," select "Java SE." At this writing, the download link is displayed as "JDK 6 Update 23." If your instructor requires or allows your use of the NetBeans IDE (Integrated Development Environment), you can download the JDK and NetBeans together, using the download link, "JDK 6 Update 23 with NetBeans 6.9.1" You do not need to download the "jre" (Java Runtime Environment) file separately—it is included with the JDK.

2. Install the JDK on your local drive, placing it in the "Program Files" folder. The default installation subdirectory will look like this: `C:\Program Files\Java\jdk1.6.0_23`. This subdirectory contains a number of important folders, but the critical one is "bin." The "bin" folder contains the Java compiler, javac.exe, and the Java interpreter, java.exe. The Java Virtual Machine (JVM), standard libraries, etc. are in the jre subdirectory.

Configuring the Windows Command Line Environment

IMPORTANT! If your instructor requires you to use a Java IDE such as DrJava, Eclipse, NetBeans, etc., you may skip this section.

1. After writing your Java program, e.g., HelloWorldApp.java, in a Windows text editor such as Notepad, you must open a CMD command line window using Start | Programs | Accessories | Command Prompt in order to run your Java program. Everything that you do in this section will be done from the CMD command line prompt, or as it is more commonly called, the "system prompt."

2. There are two main tasks that you must perform from the system prompt every time you want to compile and execute your Java program. First, you must navigate to your source code folder using the CD command. If you do not know how to navigate your disk's directory tree from the CMD prompt, you must learn to do this first. Second, you must include the "bin" directory in the path so that you can easily invoke the Java compiler and interpreter from any subdirectory.

3. Setting the path:

   - From the prompt, go to the root directory by typing `CD \`
   - Set the path by typing in the following command:

     ```
     path=%path%;c:\;c:\Program Files\Java\jdk1.6.0_23\bin
     ```

     Mind the semicolons!
   - TIP: To "automate" this command, you may place it in a batch file. While in the root directory, type EDIT to load the CMD text editor. Type in the set path command above and save the file as "JAVAPATH.BAT" and exit. The next time that you open a CMD window, you can simply type "javapath" and the command will be executed. (This assumes that the root directory is already in the path—it normally is. If it is not, then you will have to navigate to the root directory and then type "javapath." You should store your batch file in the root of your M drive or drive C of your personal computer. Do *not* store it on drive C of a lab machine.

4. Navigating to your source code folder:

   - Assuming that your source code is in a folder on your M drive called myjava\cps201, open an MS-DOS window and type

```
cd m:\myjava\cps201
```

- As with the path command, this command may be automated with a batch file. While in the root directory, type EDIT to load the CMD text editor. Type in the "cd" command above and save the file as "JAVACODE.BAT" and exit.

5. Once you have automated the two commands above, you may then open a CMD window, navigate to the root of your M drive, type "javapath" and press the Enter key, and then type "javacode" and press the Enter key. You will find yourself in your source code directory, ready to invoke the Java compiler and virtual machine.

Compiling and Executing a Java Program from the Command Line Prompt

1. Write the class definition (your program) using EDIT from the CMD prompt, or using Notepad or another Windows text editor.

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

Save the file in m:\myjava\cps201 (or other folder), using the name "HelloWorldApp.java." In Java, the file name must be character-for-character identical to the class name. A `.java` file is called a "source code" file.

2. If you are not at the CMD prompt, open a CMD window, set the Java "bin" path as shown in Step 3 above, and navigate to m:\myjava\cps201. Here's where those batch files come in handy!

3. Compile your program with the following command:

```
javac HelloWorldApp.java
```

"**javac**" is the name of the Java *compiler* and it can be invoked from a subdirectory other than the one where your source code files are located. In that case, prepend the source file name with the complete path of its location.

4. If your program compiles without an error, nothing seems to happen! You just get another CMD prompt. However, the folder will now contain a new file, "HelloWorldApp.class." This bytecode or "class" file may now be submitted to the Java *interpreter*, java.exe, to be executed. To do this, type in

```
java –cp . HelloWorldApp
```

Your program's output will be displayed on the screen, followed by the CMD prompt.

To invoke the interpreter from another subdirectory, replace the dot in the above command with the complete path of the class file's location, e.g.,

```
java –cp m:\myjava\cps201 HelloWorldApp
```

The "-cp" switch is the class path flag that tells java where your java file is located. The dot "**.**" following the cp switch is CMD-speak meaning "current directory," just like "**..**" means "parent of current directory." The Java *compiler* is smart enough to look for the relevant files in the current directory—no sourcepath or classpath switch is necessary--but the Java *interpreter* that actually executes your program refuses to inspect the current directory for the specified `.class` file *if either* the classpath environment variable has been set on your computer (see Control Panel | System | Advanced | Environment Variables) or the

classpath switch does not specify the current directory. If the classpath variable has not be set, you may omit the "`-cp .`" switch. Note that the `.class` extension is not included when the bytecode file is sent to java.

5. Remember that while neither MS-DOS nor Windows are case-sensitive operating systems, Java is a case-sensitive programming language just like C and C++. The name of the Java compiler is thus "javac," *not* "JAVAC" or "Javac."

6. If your program does not compile, check for syntax errors. If you get a error message from the operating system stating

   ```
   'javac' is not recognized as an internal or external command, operable program or batch file.
   ```

   you have not set the jdk path correctly as shown in Step 3 above. Issue the path command again and recompile. In the worst-case scenario, you can always invoke the compiler with a full path reference:

   "c:\Program Files\Java\jdk1.6.0_23\bin\javac" HelloWorldApp.java

   Note the double quotes around the path expression are required because the CMD interpreter does not allow spaces in file or folder names. If you are in the bad habit of using spaces in file and folder names, this is a good time to break that habit.

IDEs

An IDE (integrated development environment) is basically an office suite for programmers. IDEs not only provide full-featured editors in which to write your code, but extensive file, object, and project management tools, including compiling, linking, debugging, and much more. Or, you can use Notepad and the CMD prompt in the old-fashioned way shown above. Your instructor may require you to learn the command line environment as well as an IDE. There are many Java IDEs available. Just a few are listed below.

- **Sun Microsystems** (now part of Oracle)—Sun previously provided three IDEs: NetBeans, Java Studio Creator (discontinued in 2007), and Java Studio Enterprise (now migrating to NetBeans). NetBeans is the foundational IDE from Sun and is open source software. Start with NetBeans at http://www.netbeans.org/.

- **BlueJ**—BlueJ is a Java IDE specifically designed for teaching object-oriented programming to beginners. Although not open source, BlueJ is free for noncommercial use. Because BlueJ is not designed as a commercial IDE, Sun has supported the BlueJ project to produce a version that provides students with a full-featured development environment called NetBeans IDE BlueJ Edition. Since the philosophy behind BlueJ is teaching object-oriented programming, BlueJ is a popular IDE for "objects early" CS1 courses. See http://www.bluej.org/about/why.html#environment-problems.

- **DrJava**—DrJava is a lightweight, open source Java IDE described as a "development environment. . .designed primarily for students," but with "powerful features for more advanced users" (http://www.drjava.org/). A unique feature of DrJava is the Interactions Pane, an interpreter-based pane that allows the programmer to test code snippets without writing a main method. Sun is also involved in funding the DrJava project.

- **jGRASP**—jGRASP is a free, lightweight Java IDE (not open source) that supports control structure diagrams, and UML class diagrams. Like many Java IDEs, it assumes that you have a recent Java JDK installed. See http://www.jgrasp.org.

- **Eclipse**—Eclipse is a large, not-for-profit, open source "community" that provides royalty-free software development tools, including IDEs, for a variety of languages. Eclipse was originally a propriety IBM development environment that IBM released into the public domain. See http://www.eclipse.org. Eclipse has rapidly gained a following in the professional programming community.

- **jBuilder**—jBuilder is a Java IDE from Borland, creator of Borland Developer Studio, Delphi, C++Builder, C#Builder, and the Turbo line of compilers. jBuilder is regarded by many developers as the most powerful Java development platform on the market. Borland was an original supporter of the Eclipse Foundation and is currently rewriting jBuilder on an Eclipse platform. A basic version of jBuilder is free at http://www.borland.com/us/products/ide.html.

- **Microsoft**—Microsoft Visual Studio .NET is a multi-language IDE that is widely used in commercial development. It has extensive code generation features that support rapid project development. Besides C++, C#, and Basic, Visual Studio .NET includes Microsoft's version of Java called J# ("jay sharp"). J# is a revision of Microsoft's J++ in Visual Studio 6.0. The J# language is identical in *syntax* to Sun's Java language. So, J# source code can be compiled as a Java language program by a Java compiler. However, Visual J# .NET and Java have different functionality. For example, a J# program compiles in the Visual .NET IDE to a Windows executable (.exe) file. This is a great advantage for the developer writing a Windows application. However, Visual J# .NET does not produce .class bytecode files, as javac does, so it will not directly produce applications that run on a JVM (java virtual machine). This is not a significant problem because cross-platform applications can be written in J# and then compiled outside of the Visual .NET IDE with the javac compiler. J# is available at http://msdn.microsoft.com/en-us/vjsharp/default.aspx. Students and hobbyists can download other .NET Express components for free from http://msdn2.microsoft.com/en-us/express/default.aspx. Students can download J# as a part of Visual Studio 2005 for free at https://www.dreamspark.com/default.aspx.

  N. B. In January 2007, Microsoft retired J#. It will continue to be supported for many years, but will not be included in new versions of .NET. Microsoft believes—and they're not alone—that C# is the natural successor of both Java and J#.