

# ***How to Run Java Programs using J2SDK***

---

One can master programming only by writing and running programs, not just by reading the sample programs in the text. All sample programs (plus some more) are provided in a source file format so you can actually compile and run them and see how they work. This appendix is intended for those who need to install necessary tools on their computer. For those who have an access to a computer lab with necessary tools already installed may still want to read this appendix for general information.

In this appendix, we will explain a number of different ways in running Java programs. They can be divided broadly into three categories: minimalist, enhanced editor, and full integrated development environment (IDE). For the beginning programmers, we recommend the enhanced editor approach. We will describe each of the three categories here. You can find additional information, such as detailed step-by-step instructions, on using some of the tools mentioned in this appendix from our website at [www.drcaffeine.com](http://www.drcaffeine.com).



***Please read Chapter 2 before this appendix.***

At the end of this appendix, we will describe how to use classes from programmer-defined packages, such as the author-provide javabook package, in running your programs.

## Using J2SDK

---

First, we need to download a necessary compiler and other tools for compiling and running Java programs from Sun Microsystems website at <http://java.sun.com>. A collection of tools for compiling and running Java programs is called a *Java 2 SDK (Software Development Kit)*. Sun Microsystems provides three versions for SDK: enterprise edition (J2EE SDK), standard edition (J2SE SDK), and micro edition (J2ME SDK). The one you need to download is the standard edition. We describe the steps for the MS Windows platform here.

### 1. Installation

Download the most recent version J2SE SDK ver 1.4 for MS Windows. At the time of this writing (January, 2003), the address of the website for downloading J2SE SDK 1.4 is <http://java.sun.com/j2se/downloads.html>. Please be aware that website addresses change frequently. If the given address does not work, you can search for the correct page starting from the homepage at <http://java.sun.com>. You will be downloading one executable file named `j2sdk01_4_1.exe` (it may not be exactly the same, but it should be something very similar). Once the file is downloaded, double-click the file to begin the installation. You may choose any directory for installation. In this example, we assume the tools are installed in the directory

```
C:\j2sdk1.4.0_01
```

This is the default directory chosen by the installer. You may change it to any name and location you like. When the installation completes successfully, you will see a number of subdirectories, such as `bin` and `lib`, under the installation directory.

### 2. Create a Program

We are now ready to create a program. Using Notepad (or any other text editor, but don't use a word processor that saves special markers with a document), create the following program:

```
import javax.swing.*;
class MyFirstProgram {
    public static void main(String[] arg) {
        JOptionPane.showMessageDialog(null, "It works!");
        System.exit(0);
    }
}
```

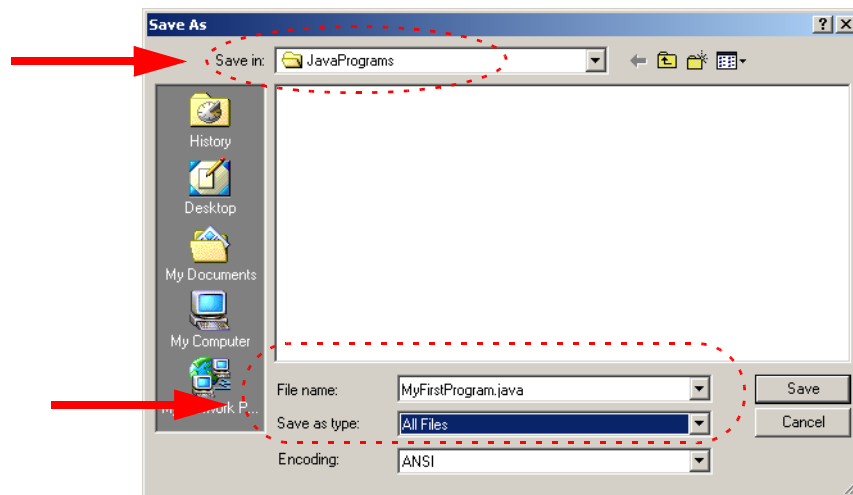
Type in the program exactly as shown, making sure the upper and lower-case letters are entered correctly.

### 3. Save the Program

Before we compile and run the program, let's save the program. First create a folder (Note: We use the word "folder" and "directory" synonymously). For this example, we will create a folder named JavaPrograms under the C: drive

```
C:\JavaPrograms
```

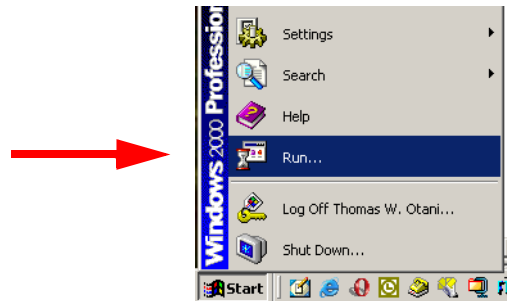
Save the program by selecting the menu choice File/Save of Notepad and giving the name MyFirstProgram.java. Put this file in the C:\JavaPrograms folder. The name of the class is MyFirstProgram, so we save it as MyFirstProgram.java. If your name the program XYZ, then save it as XYZ.java. Note that it is case-sensitive. When using Notepad, be careful that the file is not saved as MyFirstProgram.java.txt. Make sure there's no txt suffix appended to the filename. To avoid the automatic appending the txt suffix, don't forget to set the value for Save as type to All Files.



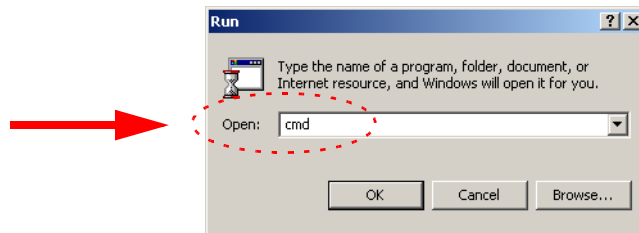
### 4. Open a Command Prompt Window

After the source file is created and saved properly, we are ready to compile and run it. We use a Command Prompt window to enter the commands for

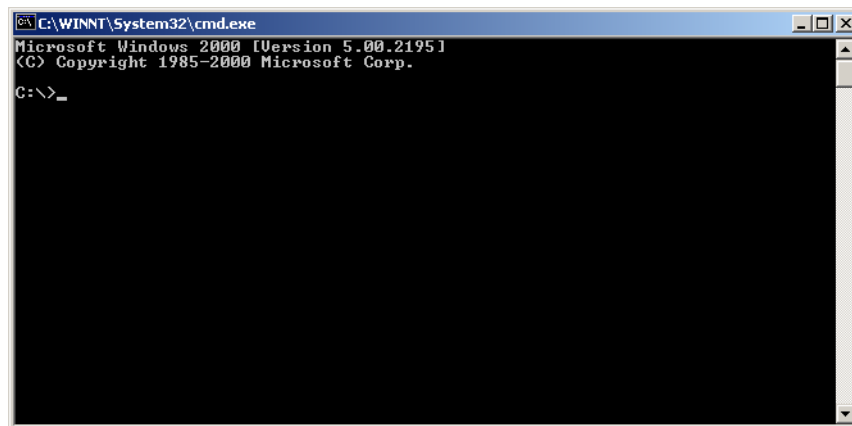
compiling and running Java programs. Open a Command Prompt window by selecting the Start/Run... option



and entering the text cmd in the text field of the Run dialog box (if cmd does not work, try command):



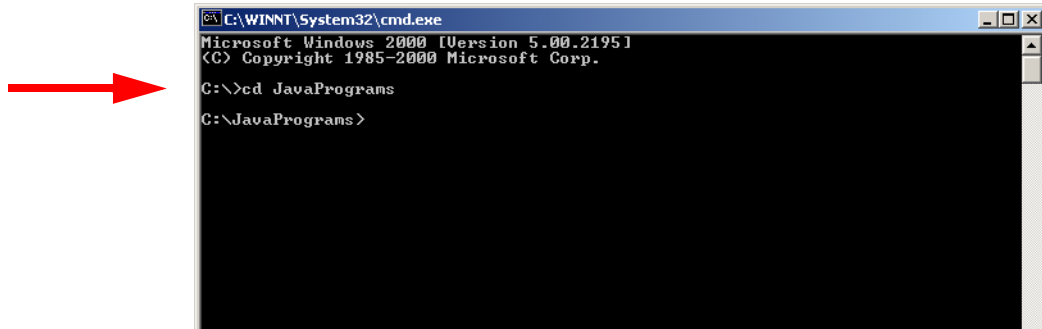
Click the OK button. A Command Prompt window appears on the screen:



From this point on, all commands are entered in this window.

## 5. Set the Environment

Before we can actually compile and run the program, we must set the environment. First change to the JavaPrograms directory where the source file is stored by entering the command `cd JavaPrograms` (and pressing the Enter key):

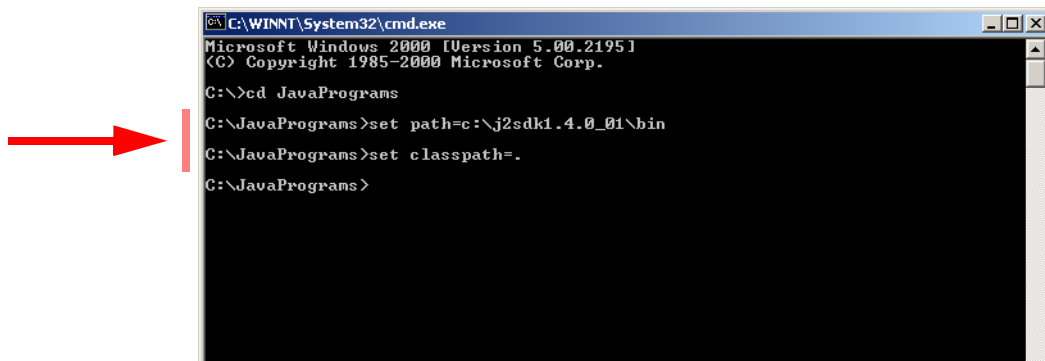


```
C:\WINNT\System32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\>cd JavaPrograms
C:\JavaPrograms>
```

*NOTE: It is beyond the scope of this appendix to explain DOS commands. Please consult other sources if you need to learn DOS commands.*

Enter the following two commands in sequence to set the environment:

```
set path=c:\j2sdk1.4.0_01\bin
set classpath=.
```



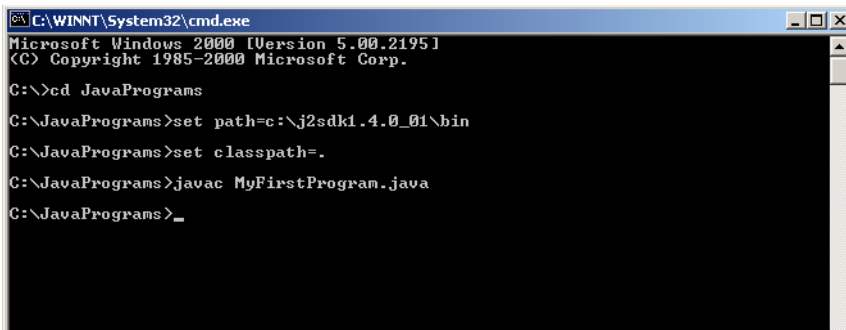
```
C:\WINNT\System32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\>cd JavaPrograms
C:\JavaPrograms>set path=c:\j2sdk1.4.0_01\bin
C:\JavaPrograms>set classpath=.
C:\JavaPrograms>
```

Enter the commands exactly as shown. Do not introduce any spaces between the equal symbols (=), for example. The first command sets the PATH environment variable so we can refer to the executable files in the bin subdirectory of C:\j2sdk1.4.0\_01. The second command tells the Java compiler and interpreter where to find the source files. The period (.) indicates the current directory. You need to enter the two commands only once.

## 6. Compile the Program

Finally we are ready to compile the program. To compile a Java source file, use the `javac` command followed by the filename of the source file. Enter the following command exactly, i.e., in a case-sensitive manner:

```
javac MyFirstProgram.java
```



A screenshot of a Windows command prompt window titled "C:\WINNT\System32\cmd.exe". The window shows the following commands and their outputs:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd JavaPrograms
C:\JavaPrograms>set path=c:\jdk1.4.0_01\bin
C:\JavaPrograms>set classpath=.
C:\JavaPrograms>javac MyFirstProgram.java
C:\JavaPrograms>_
```

A red arrow points to the `javac MyFirstProgram.java` command line.

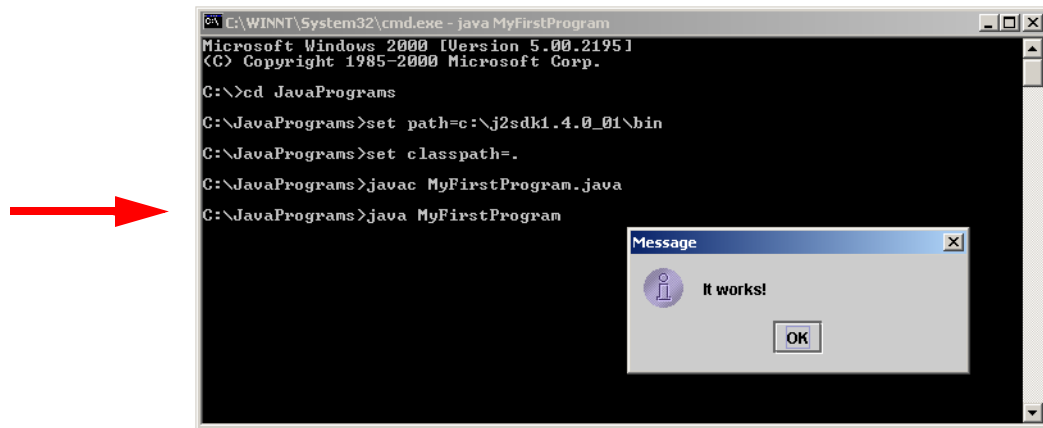
After a moment of pause, when there's no error in the program, the prompt to enter the next command appears. An error message will appear if there's an error. If that happens, go back to Notepad and check the program. Make any necessary changes and save it again. Then enter the `javac` command again. As explained in Chapter 2, successful compilation will result in a creation of a bytecode file.

## 7. Run the Program

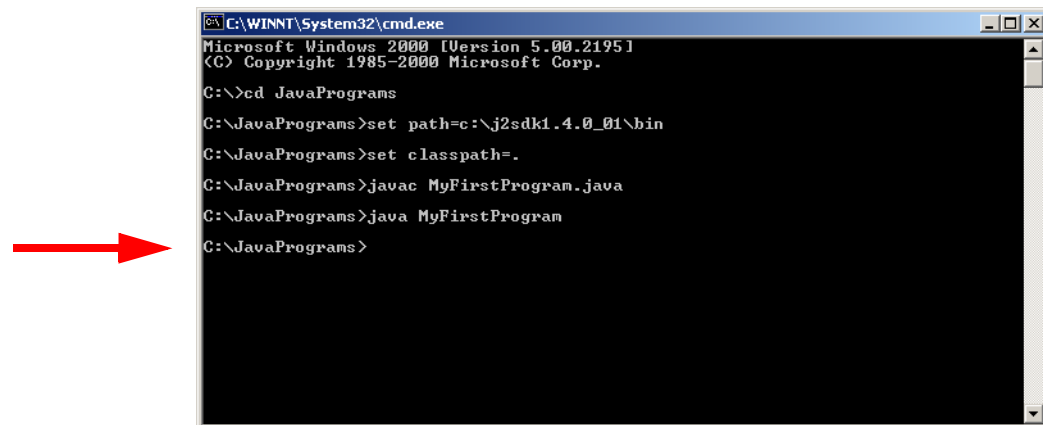
After the successful compilation of the program, we are finally ready to run the program by executing its bytecode file. To run the program, we use the `java` command followed by the name of the bytecode file (with no suffix). Enter the command

```
java MyFirstProgram
```

and press the Enter key. The program starts and you will see a message dialog appears on the screen:



Close this message dialog by clicking its OK button. The program terminates, and another prompt appears on the Command Prompt window:



Congratulations! You have successfully executed your first Java program.

### The Significance of the System.exit Statement

The last statement of the sample program was

```
System.exit(0);
```

that caused the program to terminate. If you adopt the minimalist approach of using Notepad (or another simple text editor) and a Command Prompt window,

you must include the `exit` statement to terminate the sample `MyFirstProgram` program. If you don't, then the program will not terminate. The message dialog disappears from the screen when you click its OK button, but the program is still active. When this happens, you will not get another prompt in the Command Prompt window. And, of course, without getting a command prompt, you can't enter another command anymore. Not all programs behave this way. Specifically, a program that uses console-based standard input and output does not require the `exit` statement, while a program that uses GUI-based input and output (such as `MyFirstProgram` that uses the GUI-based `JOptionPane` for output) requires the `exit` statement. When using the minimalist approach, the easiest thing to do is to include the `exit` statement for all programst.

### Using Programmer-Defined Packages

When using classes from the system packages such as `javax.swing`, `java.util`, and others, all we do is to include appropriate `import` statements in the program. this is not enough when using classes from programmer-defined packages. We must also set the environment correctly. How we set the environment is dependent on the development tools we use. We will describe how to set the environment to use programmer-defined packages with the minimalist approach and BlueJ.

We will use the author-provided `javabook` package as an example to illustrate the procedure. You can download this package and its documentation from the textbook website. We assume you have downloaded and install the `javabook` classes under the `C:\JavaPrograms\javabook` directory. Notice that the name of the package is `javabook`, and the directory that contains the classes in this package is also named `javabook`. This is the requirement. The classes in the package `xyz` must be placed in the directory named `xyz`. The directory `xyz`, however, can be placed anywhere you want. In this example, we put the `javabook` directory under `C:\JavaPrograms`. When the installation is done correctly, you should see the source and bytecode files of the `javabook` classes such as `MainWindow.java`, `MainWindow.class`, `OutputBox.java`, `OutputBox.class`, and others in the `C:\JavaPrograms\javabook` directory.



***For this example, we assume the classes from author-provided `javabook` package are placed in the directory***

```
C:\JavaPrograms\javabook
```

All you have to do is to change the setting for the classpath as follows:

```
set classpath=.;c:\JavaPrograms
```

Notice that we specify the directory that contains the javabook directory. We do specify the full pathname to the javabook directory itself.



*To use the javabook classes stored in the directory C:\Java-  
aprograms\javabook, set the classpath by entering the com-  
mand*

```
set classpath=.;c:\JavaPrograms
```

Here's how we set both path and classpath environment variables at the beginning of a session:

```
C:\WINNT\System32\cmd.exe
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\>set path=c:\j2sdk1.4.0_01
C:\>set classpath=.;c:\JavaPrograms
C:\>
```