Online Chapter B

Running Programs

This online chapter explains how you can create and run Java programs without using an integrated development environment (an environment like JCreator). The chapter also gives instructions for creating and running Java programs on non-Windows computers.

Creating a Directory

First, you need a Java program. You use this program to test your Java compiler. You can get a program from this book's CD-ROM. In this chapter, I explain how to run program on the CD-ROM named `MortgageText.java`.

NOTE: In some places in this online chapter, I refer to the program as `Mortgage.java`. It's a relic having to do with the way I put this chapter together. When I write `Mortgage.java`, I really mean `MortgageText.java`.

Where do you put this `MortgageText.java` test program? I recommend keeping all your Java programs in one tidy place. I create a directory on my hard drive named `JavaPrograms`. I put the downloaded `MortgageText.java` file in my `JavaPrograms` directory.

If you're handy with your computer's file system, then you can create a `JavaPrograms` directory by pointing and clicking. (In Windows, you can use Windows Explorer.) But you can also create a directory with the system's command prompt. Here's how:

In Windows:

Issue the following command:

```
md \JavaPrograms
```

This command creates a new `JavaPrograms` directory directly below your hard drive's top level (root) directory. Later, when you want to run Java programs, you can get to this directory by issuing the following command:

```
cd \JavaPrograms
```

This command makes `\JavaPrograms` be your working directory.

In Unix or Linux:

Issue the following command:

```
mkdir ~/JavaPrograms
```

This command creates a new `JavaPrograms` directory directly below your user home directory. Later, when you want to run Java programs, you can get to your `JavaPrograms` directory by issuing the following command:

```
cd ~/JavaPrograms
```

On a Mac with OS X:

Go to the Utilities subfolder of the Applications folder. In the Utilities subfolder, find the Terminal icon. Double click this icon to open a Terminal window. Then follow the Unix/Linux directions above.

On a Mac with the Classic OS:

As far as I know, there's no version of Java 5.0 for the Mac Classic. But if you want to compile older Java programs on a Mac, try the following:.

Open the Finder, then click anywhere on your desktop. In the Finder's File menu, select New Folder. When a new folder icon appears on your desktop, name the folder `JavaPrograms`.

Compiling and Running a Program

Are you ready to compile and run Java programs? Here's the acid test:

1. Put my `MortgageText.java` file in your `JavaPrograms` directory.

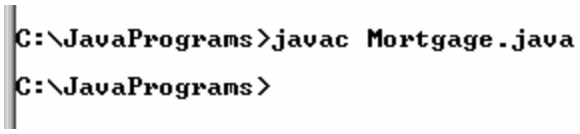
2. In a command prompt window, follow in instructions above to move to the `JavaPrograms` directory. (Type `cd \JavaPrograms` or `cd ~/JavaPrograms`, whichever is right for your system.)

3. In the same command prompt window, type the following command:

   ```
   javac MortgageText.java
   ```

   This command tells the computer to compile the `MortgageText.java` program. If all goes well, the computer responds with very little fanfare. (See Figure B-1.) You get no special messages. After a brief delay, you see yet another command prompt. (You see something like `C:\JavaPrograms>`.)

Figure B-1: Compiling a Java program.

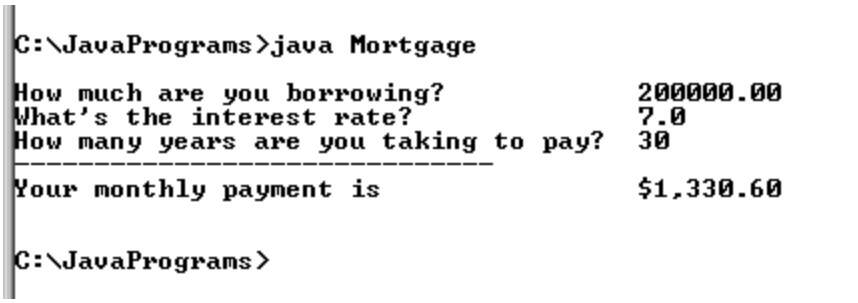


4. In the same command prompt window, type the following command:

   ```
   java MortgageText
   ```

   The command tells the computer to run the MortgageText program. If all goes well, the computer starts firing questions at you. (See Figure B-2.) You type a dollar amount (like `200000.00`), an interest rate (like `7.0`) and a number of years (like `30`). After each entry, you press Enter. After you answer the number-of-years question, the computer displays your monthly payment (a number like `$1,330.60`).

Figure B-2: Running a Java program.



<Remember>

When you type the `javac` command (in Step 3), you use a complete file name, such as `MortgageText.java`. But, when you type the `java`

command (in Step 4), you don't use a complete file name. You type `java MortgageText`. You never type `java MortgageText.java`.

What Could Possibly Go Wrong?

In Figures B-1 and B-2, I show you the picture-perfect processing of a MortgageText program. That's fine for us super heroes, but what can the average person do if things don't go smoothly? What if you get an error message or two?

This section tells you how to diagnose some of the more common difficulties. If you can't compile and run the MortgageText program, then this section's tricks will probably get you past the hurdle.

<Tip>

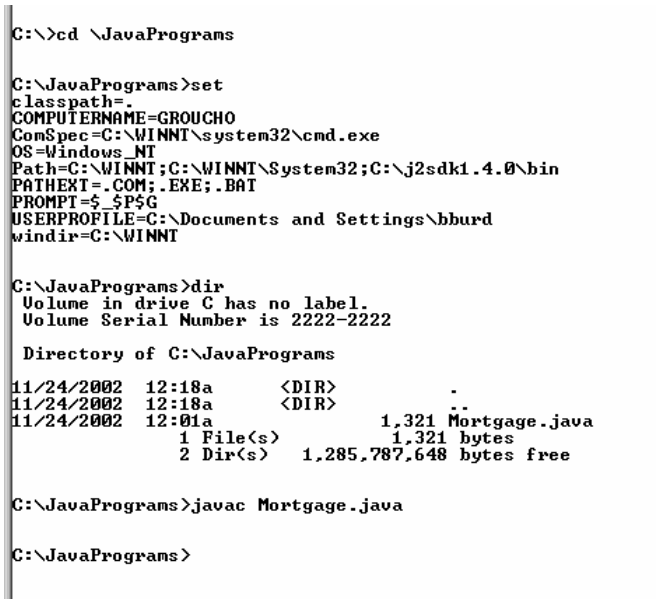
This section of the book is biased toward Microsoft Windows. If you're not a Windows user, you have to translate these tricks into your computer's native language.

The main trick is to enlarge your repertoire of commands. You retype the `javac` and `java` commands from Figure B-2, but you mix the `javac` and `java` commands with some useful diagnostic commands. Here's what you do:

1. **In a command prompt window, type the command `cd \JavaPrograms`.**

   By typing this command, you plant yourself squarely in the `C:\JavaPrograms` directory. Look at the top of Figure B-3, and notice how the prompt changes. (It changes from `C:\` to `C:\JavaPrograms`.) The directory `C:\JavaPrograms` becomes your working directory.

Figure B-3: Diagnosing your setup.



If you type this `cd \JavaPrograms` command, and your prompt doesn't become `C:\JavaPrograms`, then you should do a little checking. Here's how:

* Look again at the `cd` command that you typed. Make sure that you typed the command correctly.
* Open Windows Explorer, and make sure that you have a `C:\JavaPrograms` directory. If not, go back to the section entitled "Creating a Directory" in this chapter.

Once you've successfully issued the command `cd \JavaPrograms`, any other commands that you type refer automatically to files in that `C:\JavaPrograms` directory.

It doesn't matter where you start. As long as you're working somewhere on the `C:` drive, the command `cd \JavaPrograms` always changes your working directory to `C:\JavaPrograms`.

<Warning>

If you open several command prompt windows, then each of these windows can have a different working directory. For instance, imagine that you close one command prompt window, and open another. Then, to get to the `JavaPrograms` directory in the new window, you have to issue the `cd \JavaPrograms` command again.

2. **Type the `set` command, as in Figure B-3.**

   The computer responds by displaying a big list. The list has lines like

   ```
   classpath=.
   ```

   and

   ```
   Path=C:\WINNT;C:\WINNT\System32;C:\program
       files\java\jdk1.5.0\bin
   ```

   In fact, when you issue the `set` command, you're checking value of the `CLASSPATH` and the `PATH`.

   * Make sure that the `CLASSPATH` contains a dot. It can contain more than just a dot, but the dot must be separated from other things by semicolons. For instance, the line

   ```
   classpath=C:\Classes;.;C:\tomcat\servlet.jar
   ```

     indicates a perfectly good `CLASSPATH`.

     Notice that, on a Windows machine, the uppercase or lowercase spelling of the word `cLASspATh` doesn't matter at all. Any choice of caps and small letters works fine.

   * Make sure that the `PATH` includes your Java bin directory. For instance, in Figure B-3, my `PATH` includes the directory `C:\program files\java\jdk1.5.0\bin`.

<Tip>

Sometimes, when you type the `set` command, the resulting display is so long that the `CLASSPATH` and `PATH` lines roll quickly off of your screen. If this happens to you, try typing `set | more`. With this modified command, you see lines like `classpath=.` one screenful at a time.

3. **Type the `dir` command, as in Figure B-3.**

The computer responds by displaying a list of all files in your `JavaPrograms` directory. When you issue the `dir` command, you're looking for the file `MortgageText.java`. If this file isn't in your `JavaPrograms` directory, then you'll have trouble issuing the `javac MortgageText.java` command.

<Tip>

When you edit a file with Windows Notepad, the File->Save As... box can add the `.txt` extension to any file name that you type. For instance, if you type `MortgageText.java` in the Save As box's File name field, then you might create a file named `MortgageText.java.txt`. A file whose name has this `.txt` extension will not make the `javac` command happy. To fix this problem, do File->Save As... again, and put `"MortgageText.java"` (surrounded by double quote marks) in the Save As box's File name field.

Those Pesky File Name Extensions

The file names displayed in Windows Explorer can be misleading. You may visit the `JavaPrograms` directory and see the name `MortgageText`. Instead of just `MortgageText`, the file's full name may be `MortgageText.java` or `MortgageText.txt`. (The name `MortgageText.java` would be fine, but the name `MortgageText.txt` would be troublesome.) You may even see `MortgageText.java`, when the file's full name is something like `MortgageText.java.txt` (a bad name for a Java program file).

The ugly truth is, Windows Explorer can hide a file's `.txt` and `.java` extensions. This awful feature tends to confuse Java programmers. So, if you like using Windows Explorer, you should modify the Windows "hide extensions" feature. To do this, you have to open the Folder Options window. Here's how:

* **In Windows 95, 98 or NT:** In the Windows Explorer menu bar, select View->Folder Options (or just View->Options).
* **In Windows ME or 2000:** Go to Start->Settings->Control Panel->Folder Options.
* **In Windows XP:** Go to Start->Control Panel->Performance and Maintenance->File Types.

In the Folder Options window, select the View tab. Then look for an item labeled **Hide file extensions for known file types**. Make sure that this item's box is *not* checked.

4. **Type the `javac MortgageText.java` command, as in Figure B-3.**

   The computer should respond with very little fanfare. You should see no messages and another prompt. If you see anything else (messages like `cannot read: MortgageText.java` or `javac: invalid flag`), then something's not right. Redo steps 1 through 3, and make sure that your ducks are all in a row.

   If steps 1 through 3 look good, but step 4 still fails, then try the `type MortgageText.java` command, as in Figure B-4. The computer responds by displaying all the stuff in the `MortgageText.java` file.

If the display that you see isn't a lot like the display in Figure B-4, then there's something wrong with your `MortgageText.java` file. You should recopy the file `MortgageText.java` file from the CD-ROM.

Figure B-4: The MortgageText.java file.

```
C:\WINNT\system32\cmd.exe

C:\JavaPrograms>type Mortgage.java
import java.io.*;
import java.text.NumberFormat;

public class Mortgage
{

   public static void main(String args[]) throws IOException
   {

      BufferedReader keyboard =
         new BufferedReader(new InputStreamReader(System.in));
      double principal, rate, ratePercent;
      int years, n;
      final int paymentsPerYear = 12;
      final int timesPerYearCalculated = 12;
      double effectiveAnnualRate;
      double payment;
      NumberFormat currency = NumberFormat.getCurrencyInstance();

      System.out.print("How much are you borrowing?           ");
      principal = Double.parseDouble(keyboard.readLine());
      System.out.print("What's the interest rate?             ");
      ratePercent = Double.parseDouble(keyboard.readLine());
      rate = ratePercent/100.00;
      System.out.print("How many years are you taking to pay?  ");
      years = Integer.parseInt(keyboard.readLine());
      System.out.println("-------------------------------");

      n = paymentsPerYear * years;
      effectiveAnnualRate = rate / paymentsPerYear;
      payment = principal*(effectiveAnnualRate /
         (1 - Math.pow(1 + effectiveAnnualRate, -n)));
      System.out.print("Your monthly payment is               ");
      System.out.println(currency.format(payment));
      System.out.println();
   }
}

C:\JavaPrograms>_
```

5. **Type the ~~dir~~ command again, as in Figure B-5.**

   This time around, you're checking for the names of two files -- the old `MortgageText.java` file, and its young cousin, the `MortgageText.class` file.

Figure B-5: Continuing the diagnosis.

```
C:\JavaPrograms>dir
 Volume in drive C has no label.
 Volume Serial Number is 2222-2222

 Directory of C:\JavaPrograms

11/24/2002  12:19a      <DIR>          .
11/24/2002  12:19a      <DIR>          ..
11/24/2002  12:19a                1,442 Mortgage.class
11/24/2002  12:01a                1,321 Mortgage.java
               2 File(s)          2,763 bytes
               2 Dir(s)   1,285,713,920 bytes free


C:\JavaPrograms>java Mortgage
How much are you borrowing?           150000.00
What's the interest rate?             8.25
How many years are you taking to pay?  15
-------------------------------
Your monthly payment is               $1,455.21


C:\JavaPrograms>_
```

The new `MortgageText.class` file comes from a successful run of `javac` (in step 4). This is the bytecode file that's described in Chapter 1. If you issue the `dir` command, and you don't see a `MortgageText.class` file, then step 4 (above) wasn't as successful as you thought it was. Without a `.class` file, you can't proceed to the `java` command in the next step.

6. **Type the `java MortgageText` command, as in Figure B-5.**

   If all goes well, you should be asked `How much are you borrowing?` and two other questions. After answering these questions, you should get a monthly payment amount.

   If the program doesn't run the way it's shown in Figure B-5, then go back and try all six of these steps again. Chances are, something simple needs to be fixed.

   If you review the six steps, and you still can't figure out what's wrong, then send me a holler. My address for such questions is `Java2ForDummies@BurdBrain.com`.