

# Lab: Introduction To Java Programming

## Student Exercises

Clifton Nock, IBM Rochester  
**Session 15LF (403692)**

# Lab: Introduction To Java Programming

Goal .....	4
Tools .....	4
Setup .....	4
Hints .....	4
Exercise 1: Create and run a simple Java application on Windows .....	5
<b>What This Exercise Is About</b> .....	5
<b>What You Should Be Able To Do</b> .....	5
<b>Introduction</b> .....	5
<b>Exercise Instructions</b> .....	5
<b>Open An Editor</b> .....	5
<b>Enter And Compile The Source</b> .....	6
<b>Run The Application</b> .....	7
Exercise 2: Run a simple Java application on AS/400 .....	8
<b>What This Exercise Is About</b> .....	8
<b>What You Should Be Able To Do</b> .....	8
<b>Introduction</b> .....	8
<b>Some AS/400 Specifics</b> .....	8
<b>Exercise Instructions</b> .....	9
<b>Use the emulator to sign-on to the AS/400</b> .....	9
<b>Use FTP to copy the Java class file from the PC to the AS/400</b> .....	9
<b>Run the Java application on the AS/400.</b> .....	10
Exercise 3: Optimize an AS/400 Java application .....	11
<b>What This Exercise Is About</b> .....	11
<b>What You Should Be Able To Do</b> .....	11
<b>Introduction</b> .....	11
<b>Exercise Instructions</b> .....	12
<b>CRTJVAPGM, DSPJVAPGM</b> .....	12
Exercise 4: Create a simple web page .....	14
<b>What This Exercise Is About</b> .....	14
<b>What You Should Be Able To Do</b> .....	14

<b>Introduction</b>	14
<b>Exercise Instructions</b>	14
<b>Create The Web Page</b>	14
<b>View The Web Page</b>	15
<b>Exercise 5: Create and run a simple Java applet</b>	16
<b>What This Exercise Is About</b>	16
<b>What You Should Be Able To Do</b>	16
<b>Introduction</b>	16
<b>Edit and Compile The Source</b>	17
<b>Insert The Applet Into Your Web Page</b>	18
<b>Exercise 6: Use Qshell on an AS/400</b>	20
<b>What This Exercise Is About</b>	20
<b>What You Should Be Able To Do</b>	20
<b>Introduction</b>	20
<b>Using QShell</b>	20
<b>Exercise 7: Create and run a simple Java servlet</b>	21
<b>What This Exercise Is About</b>	21
<b>What You Should Be Able To Do</b>	21
<b>Introduction</b>	21
<b>Edit and Compile The Source</b>	22
<b>Run the servletrunner test server</b>	23

## Goal

In this lab, you will work through a series of exercises that provide you with an opportunity to create some simple Java programs in a variety of environments. You will explore the differences between an application, applet, and servlet, as well as learn how to use some Java development tools.

## Tools

This lab uses primitive tools. The most basic editor and command line compiler are used to create Java source code and compile your Java programs. You will use Windows Notepad to edit your Java source code and the Java Development Kit (JDK, J2SE, SDK) to compile and run. The JDK is installed on the lab PCs.

It is common for Java developers to use these tools everyday. However, there are some very good integrated development environments (IDEs) that make programming easier. They provide syntax highlighting, point-and-click compile and run, integrated help text and much more. IBM VisualAge for Java is one such example. IDEs are not used in this lab because the lab would turn into an exercise in learning the tool instead of writing some Java code.

## Setup

During this lab you will need a userid and password for the AS/400 and you will need to know the name of the AS/400 system. The userid and password will be JAVAx where **xx** is the number 01 to 99 (two digits number). This information will be given to you by your instructor. Please fill in this information below so that you have it for reference during the lab.

The **name** of my AS/400 is: \_\_\_\_\_  
My AS/400 **userid** is: \_\_\_\_\_  
My AS/400 **password** is: \_\_\_\_\_  
The AS/400 **directory** is: /LAB15LF/**Javaxx** (Javaxx is your userid)  
The PC **directory** is: C:\LAB15LF  
The PC **hostname** is: \_\_\_\_\_

## Hints

The most important tip that you should remember while performing the following exercises is that the Java language is **HIGHLY** case sensitive. Please be extra careful to double check the case of all code that you enter to ensure that you have input your code correctly.

## Exercise 1: Create and run a simple Java application on Windows

### What This Exercise Is About

In this exercise you will create and run a simple Java application on Windows.

### What You Should Be Able To Do

At the end of this exercise, you should be able to:

- Edit Java source code
- Compile the Java source code into a Java class file
- Run the Java program from the command line
- Know how to print text to the screen in a Java application

### Introduction

Lets create a simple Java application. This application will print out the current date and time.

### Exercise Instructions

#### Open An Editor

The Java Developers Kit comes with all the tools you need for Java development *except* for a source code editor. You can use any editor you prefer. We will use Windows Notepad for this lab.

1. Open an MS-DOS prompt. Select the **Start->Programs->Command Prompt** menu.
2. Create a directory for your work. Type the following command in the MS-DOS prompt:

```
mkdir c:\LAB15\F
```

3. Make this directory the current directory:

```
cd \LAB15\F
```

4. Use Notepad to begin editing the Java source file. We will call the source file CurrentInfo.java. (Remember: Java is case-sensitive!)

```
notepad CurrentInfo.java
```

When Notepad asks you if you want to create a new file, click on **Yes**.

Now you are ready to start coding...

### Enter And Compile The Source

Type in the following source code. Please note the punctuation as well as the capitalization. Both are very important elements of Java's syntax. Unlike some other languages, Java is not column specific so indentation does not matter.

```
import java.util.Date;

public class CurrentInfo
{
    public static void main(String[] args)
    {
        System.out.println("Current date and time: " + new Date());
    }
}
```

**Hint:** The class name ("CurrentInfo" here) must match the name of the source file ("CurrentInfo.java") exactly - with the same case.

1. Save the file using the **File->Save** menu.
2. Now it is time to compile your source code into Java byte codes using the javac compiler. javac comes with the Java Developers Kit (JDK) which is installed on the lab PC. Type the following command in the MS-DOS prompt:

```
javac CurrentInfo.java
```

3. If the source code compiles successfully, you will be presented again with another MS-DOS prompt. No news is good news! If there are errors in your source code, they will be displayed.
4. If you have any errors in your source code, go back to your Notepad window, fix the errors, save the file, and try to compile again. Common errors are misspellings, inconsistent capitalization, or punctuation mistakes.
5. After you compile with no errors, type `dir` in the MS-DOS prompt. You will notice that there is a new file called CurrentInfo.class. This file contains the Java byte codes generated by the compiler.

## **Run The Application**

Once you are able to successfully compile CurrentInfo.java, it is time to run it.

1. At the MS-DOS prompt, type:

```
java CurrentInfo
```

This starts a Java Virtual Machine(JVM), loads the Java byte codes contained in CurrentInfo.class, and runs the program.

2. Verify that you see the correct output from your program :

```
Current date and time: Fri Aug 25 12:38:39 CDT 2000
```

**(Leave the MS-DOS window open. You will use it in other exercises.)**

## Exercise 2: Run a simple Java application on AS/400

### What This Exercise Is About

This exercise will demonstrate the “write once, run anywhere” ability of Java. In this exercise you will run the CurrentInfo Java application on the AS/400.

### What You Should Be Able To Do

At the end of this exercise, you should be able to:

- FTP Java class files from the PC to the integrated file system of an AS/400
- Run Java programs on the AS/400

### Introduction

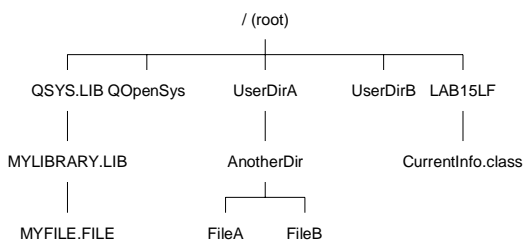
In the previous exercise you created a Java program and ran it on the PC. Now you will run the same program on the AS/400.

## Some AS/400 Specifics

### The Integrated File System on the AS/400

To understand how Java programs are stored and run on the AS/400, knowledge about the AS/400’s integrated file system is helpful. The integrated file system is a part of OS/400 that supports stream I/O and file management similar to UNIX and PC file systems. Key features of the integrated file system include a hierarchical directory structure and support for storing information in stream files. Benefits of the integrated file system include fast access to OS/400 data and efficient handling of stream data including images and audio. All Java programs that reside on the AS/400 are stored in the integrated file system. Libraries and files are accessible via the QSYS.LIB directory. An example of a directory tree is shown below:

### AS/400 Directory Tree Structure





## Exercise Instructions

### Use the emulator to sign-on to the AS/400

In order to run a Java program on the AS/400, the class file must be in the integrated file system of the AS/400. First, you need to create a directory in the integrated file system as a place to copy the class file that you compiled in the previous exercise.

1. Start the 5250 emulator using the icon located on the Windows desktop.
2. Sign-on to the AS/400 using the user id and password assigned by the instructor.
3. Create a directory called /LAB15LF/JAVAx<sub>xx</sub>, where <sub>xx</sub> is a number assigned by the instructor. This is the directory where you will copy your Java class file.

```
MKDIR ` /LAB15LF/JAVAxxx `
```

### Use FTP to copy the Java class file from the PC to the AS/400

You will now copy the Java class file from the PC to the AS/400. Java programs are stored in class files. You will FTP your class file from the PC to the AS/400. (Another option is to use a Client Access/400 network drive to copy the file. To keep it simple you will use FTP in this lab). From the MS-DOS prompt, enter the following commands where system is the name of your AS/400 and Javax<sub>xx</sub> is your userid.

1. ftp system
2. ... Enter your AS/400 userid and password when prompted
3. cd /
4. cd /LAB15LF/Javax<sub>xx</sub>
5. bin
6. put CurrentInfo.class
7. quit

These FTP commands copied CurrentInfo.class to your AS/400. If you are curious about what each of these FTP commands did, study the response text issued after each one.

**HINT:** Notice that the Windows file system uses backslashes (\) and the AS/400 integrated file system uses forward slashes (/).

### Run the Java application on the AS/400.

The Java class file is now on the AS/400. Let's run the Java application on the AS/400.

1. The first step is to tell Java the location of your program. Java uses the CLASSPATH environment variable to find Java programs. You will tell it to look in /LAB15LF/Javaxx. *Environment variable names (i.e., CLASSPATH) are case sensitive* so make sure you type CLASSPATH in uppercase. Enter the following on the AS/400 command line:

```
ADDENVVAR CLASSPATH '/LAB15LF/Javaxx'
```

2. Run the application on the AS/400:

```
java CurrentInfo
```

Notice that this is the exact same command you used when you ran the application on Windows in an MS-DOS prompt.

3. Verify that you see the correct output from your program:

```
Current date and time: Fri Aug 25 12:54:06 GMT+00:00 2000
```

4. Java applications run in a "Java shell" on the AS/400. This Java shell looks a lot like the command entry screen. Press F3 to exit the Java shell and return to the AS/400 command line.

**(Leave the emulator open. You will use it in other exercises.)**

## Exercise 3: Optimize an AS/400 Java application

### What This Exercise Is About

Java class files contain byte codes. All platforms that support Java must convert these byte codes into machine instructions. The AS/400 has two ways of doing this -- interpreted (the byte codes are converted each time the Java application is run) and static (the byte codes are converted once and saved). In the previous exercise, the Java byte codes were interpreted as you ran the application. In this exercise you will examine the **CRTJVAPGM**, **DSPJVAPGM** and **DLTJVAPGM** commands on the AS/400. These commands deal with static conversion.

### What You Should Be Able To Do

At the end of this exercise, you should be able to:

- Use **CRTJVAPGM** to make a Java program run faster
- Use **DSPJVAPGM** to analyze how a Java program was created
- Use **DLTJVAPGM** to delete a AS/400 Java program

### Introduction

All Java Virtual Machines (JVMs) convert from Java byte codes to machine instructions. Most JVMs perform this conversion at run time. The byte codes are converted to machine instructions but the machine instructions are not saved. The byte codes must be converted every time the class is run. In order to make Java run faster on the AS/400, the class file can be converted once and the machine instructions saved for future use. The converted version is called an **AS/400 Java Program**. An AS/400 Java Program is a pre-verified, compiled version of a class file. The AS/400 command **CRTJVAPGM** creates the Java Program. The original byte codes still exist in the integrated file system and the AS/400 Java Program is not visible in the integrated file system. Only commands like **CRTJVAPGM**, **DSPJVAPGM** and **DLTJVAPGM** allow you to interact with the Java Program. The AS/400 keeps track of the two versions (byte codes and AS/400 machine instructions) and discards the AS/400 Java Program if the byte codes are changed (for example if the program is recompiled or if you move a new copy from a client).

The AS/400 Java Program is created in one of two ways: by explicitly using the **CRTJVAPGM** command on a class or jar file, or when the file is first used. In this exercise you will look at some of the options of how AS/400 Java Programs are created. **DSPJVAPGM** displays the attributes of a class or jar file in integrated file system. **DLTJVAPGM** will destroy the Java program that is associated with a class or jar file in the integrated file system (leaving the class or jar file intact).

## Exercise Instructions

### CRTJVAPGM, DSPJVAPGM

We will work with the class file we created in the previous exercise.

1. On the AS/400 command line, change the current directory to be the location of the class files:

```
cd '/LAB15LF/Javaxx'
```

2. Now you can investigate the Java program associated with the CurrentInfo.class file that you created in the previous exercises. As was mentioned in the introduction, Java programs are created when a class file is used for the first time. Since you have already run this Java application, there is already a Java program for the class file. To look at it, type:

```
DSPJVAPGM CLSF(CurrentInfo.class)
```

3. Notice the various fields, most importantly the optimization field. Optimization level 10 is the default when a Java program is created. Other possible optimization levels are \*INTERPRET (meaning there is no optimization), 20, 30, and 40. In general, the higher the optimization level, the faster the program will run. On the downside, higher optimization levels will result in longer optimization times (usually a one-time cost) and larger file sizes (more storage is needed).
4. Press Enter to return to the AS/400 command line.
5. Re-optimize the associated Java program to optimization level 40:

```
CRTJVAPGM CLSF(CurrentInfo.class) OPTIMIZE(40)
```

6. Run the application again:

```
java CurrentInfo
```

7. Press F3 to exit the Java shell and return to an AS/400 command prompt. The first thing you probably will notice is that your Java application is not running noticeably faster. Here's why: CurrentInfo.class is only one of many class files that are used when your Java application is run. The other class files are part of the standard class library that are already optimized. Since CurrentInfo.class is so small, optimizing it really doesn't make much difference.

8. Let's see an example where it makes more of a difference. You will now run a program that does a little more processing. There are two class files in the /LAB15LF directory on the AS/400. (These were copied and optimized by the lab instructor prior to the lab.) The source for these two class files is exactly the same. The only difference is the optimization level. First look at the optimization level for /IntroLab/SimpleProgram\_10.class:

```
DSPJVAPGM CLSF('/LAB15LF/SimpleProgram_10.class')
```

Verify that this class is optimized at level 10.

9. Now look at the optimization level for /IntroLab/SimpleProgram\_40.class:

```
DSPJVAPGM CLSF('/LAB15LF/SimpleProgram_40.class')
```

Verify that this class is optimized at level 40.

10. Now run the program optimized at level 10:

```
JAVA CLASS(SimpleProgram_10) CLASSPATH('/LAB15LF')
```

Notice how long this takes.

11. Finally, run the program optimized at level 40.

```
JAVA CLASS(SimpleProgram_40) CLASSPATH('/LAB15LF')
```

Verify that this takes less time to complete.

## **DLTJVAPGM**

If you delete a class file from integrated file system, any associated Java program will be deleted as well. In comparison, DLTJVAPGM will remove the associated Java program, leaving the class file intact. You may want to do this to make reduce the storage used by the Java program. Let's try it out now on your CurrentInfo.class file.

1. Delete the associate Java program for CurrentInfo.class:

```
DLTJVAPGM CLSF('CurrentInfo.class')
```

Your Java program should now be deleted.

2. Let's use **DSPJVAPGM** to double check:

```
DSPJVAPGM CLSF('CurrentInfo.class')
```

You should receive a message saying no Java program is associated with the class file. The next time you run CurrentInfo, a new Java program will be created and saved at optimization level 10.

## Exercise 4: Create a simple web page

### What This Exercise Is About

Before we are able to create a Java applet, you will need a web page in which to embed it. In this lab you will create a web page using HyperText Markup Language (HTML).

### What You Should Be Able To Do

At the end of this exercise, you should be able to create a web page, and then view it using a browser.

### Introduction

Web pages are written using a scripting language called HyperText Markup Language (HTML). Learning HTML is comparable to learning another programming language. Since our page only has a little text and formatting, we will enter the HTML tags by hand. HTML editing tools are available which allow WYSIWYG (What-You-See-Is-What-You-Get) editing of web pages. (If you want, you can use the Composer feature of Netscape Communicator.)

Why are you creating a web page in a Java programming lab? In the next exercise you will create a Java applet. Java applets *always* run embedded in a web page in the context of a web browser. In order to run your Java applet, you need to have a web page for it.

### Exercise Instructions

Our page will contain only a header and a hypertext link. The link may not work if the PCs in the lab are not connected to the web.

### Create The Web Page

1. In the MS-DOS prompt, edit a new file called MyPage.html. This will contain the HTML which describes the contents of your web page:

```
notepad MyPage.html
```

2. When Notepad asks you if you want to create a new file, click on **Yes**.
3. Type in the following HTML:

```
<HTML>
<BODY>
<H1>My Web Page</H1>
<P>Click <A HREF="http://ibm.com/as400/java">here</A> for
information about AS/400 Java.
</BODY>
</HTML>
```

4. Save the file using the **File->Save** menu.

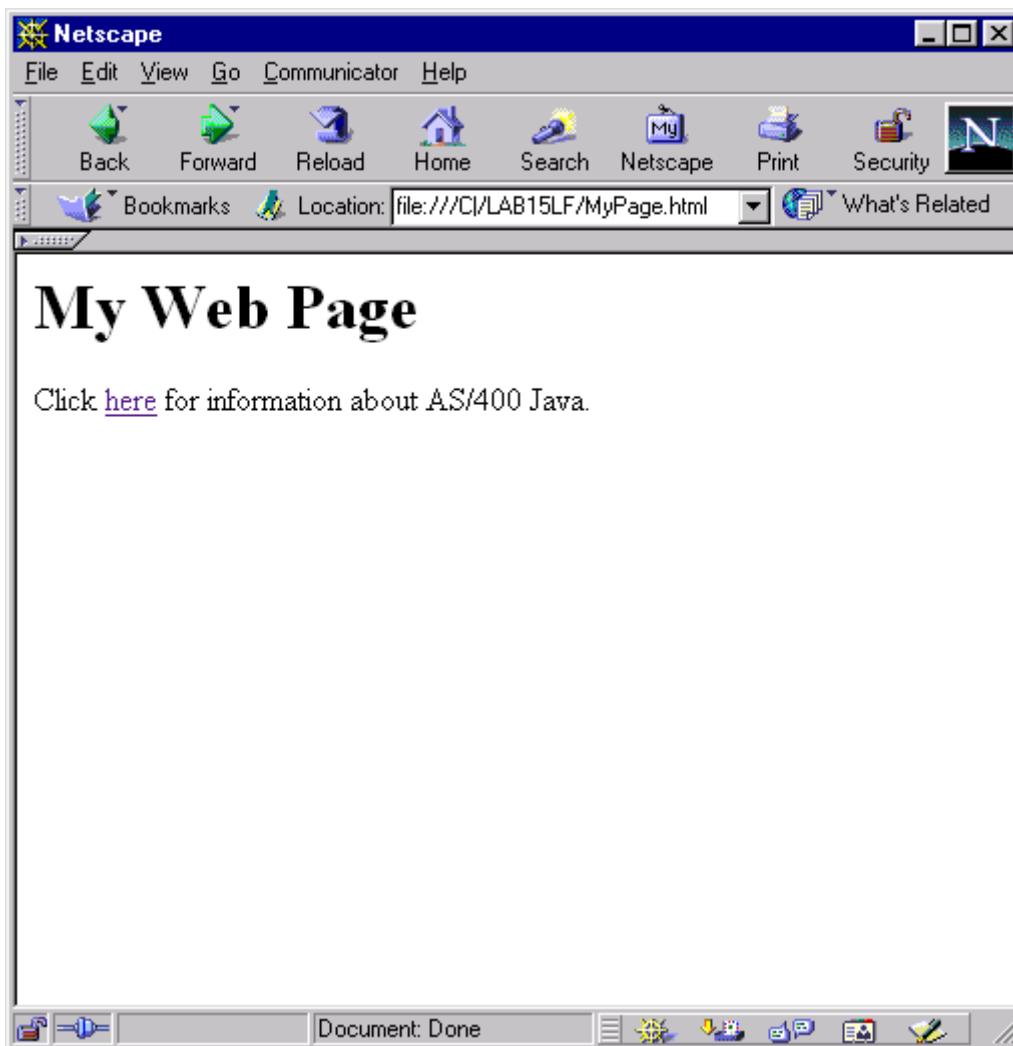
## View The Web Page

Let's see what your web page looks like in the browser.

1. Start Netscape Communicator using the icon on the desktop.
2. Open your web page using the **File->Open Page** menu, then enter

`C:\LAB15LF\MyPage.html`

Your page should appear in the main browser window:



## Exercise 5: Create and run a simple Java applet

### What This Exercise Is About

Applets are Java programs that are dynamically downloaded from a web server and run inside a browser. In this exercise you will update your Java program so it runs as an applet as well as an application.

### What You Should Be Able To Do

At the end of this exercise, you should be able to:

- Create an applet
- Compile an applet with `javac`
- Add an applet inside of an existing HTML page
- Test a web page with an applet

### Introduction

One of the powerful features of Java is the variety of environments available in which to run a program. In exercise 1 you created a Java application. In this exercise you will update this application so it can also run as an applet. In practice, Java code is often separated between the business logic and the presentation. You can write the business logic once, then easily display data as an application, applet, or servlet.

When you create a Java applet, you must extend the *java.applet.Applet* class, which is part of the Java class library available in every Java environment. This means you are going to use the *java.applet.Applet* class as a reusable part. *java.applet.Applet* will do most of the work. Your class will modify its behavior only where appropriate. In this case your class will print out the current date and time in a browser.



### Edit and Compile The Source

1. Use Notepad to edit CurrentInfo.java. (You may still have the window up from a previous exercise.)

```
notepad CurrentInfo.java
```

2. Update the source code as follows. **Changes to the original file are bold.** Please note the punctuation as well as the capitalization.

```
import java.applet.Applet;  
import java.awt.Label;  
import java.util.Date;  
  
public class CurrentInfo extends Applet  
{  
  
    public static void main(String[] args)  
    {  
        System.out.println("Current date and time: " + new Date());  
    }  
  
    public void init()  
    {  
        add(new Label("Current date and time: " + new Date()));  
    }  
  
}
```

3. Save the file using the **File->Save** menu.
4. It is time again to compile your source code into Java byte codes using the javac compiler. Type the following command in the MS-DOS prompt that you used earlier:

```
javac CurrentInfo.java
```

5. If the source code compiles successfully, you will be presented again with the MS-DOS prompt. If there are errors in your source code, they will be displayed.
6. If you have any errors in your source code, go back to your Notepad window, fix the errors, save the file, and try to compile again.

### Insert The Applet Into Your Web Page

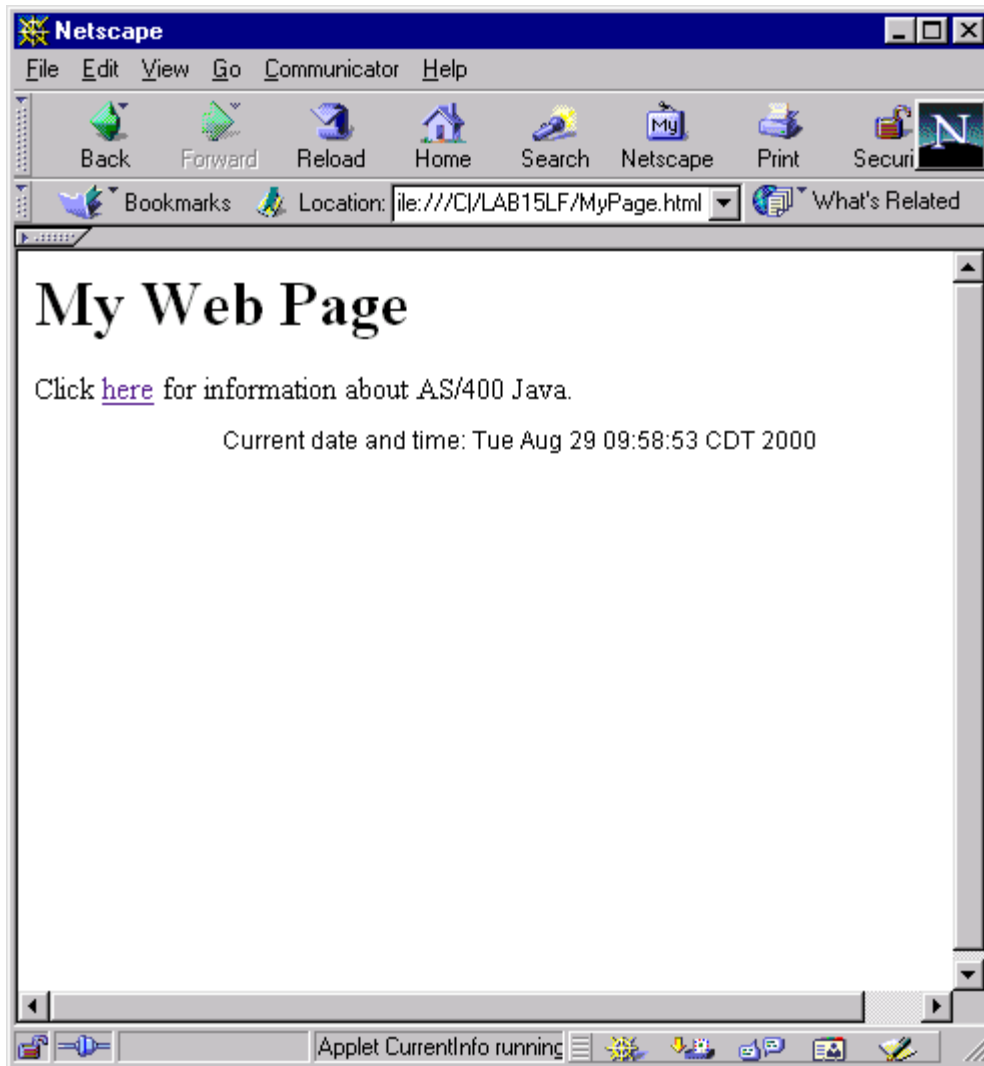
1. Use Notepad to edit MyPage.html. (You may still have the window up from a previous exercise.)

```
notepad MyPage.html
```

2. Update the HTML as follows. **Changes to the original file are bold.** These lines will result in your applet being embedded as part of the web page.

```
<HTML>
<BODY>
<H1>My Web Page</H1>
<P>Click <A HREF="http://ibm.com/as400/java">here</A> for
information about AS/400 Java.
<BR><APPLET CODE="CurrentInfo.class" WIDTH=500 HEIGHT=300>
</APPLET>
</BODY>
</HTML>
```

3. Save the file using the **File->Save** menu.
4. If the browser window is still up from the previous exercise, click the reload button in the toolbar. Otherwise, use the instructions from the previous exercise to start and load the page. The page should look like the following:



When the browser processes the APPLET tag it starts its internal Java Virtual Machine and automatically runs the CurrentInfo class. During applet processing, the `init()` method is called which is when our class defines its user interface.

## Exercise 6: Use Qshell on an AS/400

### What This Exercise Is About

Introduce you to QShell on the AS/400.

### What You Should Be Able To Do

At the end of this exercise, you should be able to use QShell to compile and run Java programs.

### Introduction

Many developers use the command line to develop, debug, and run Java programs in Windows and UNIX. The home for Java programs on the AS/400 is the Java shell. In one of the previous exercises, you used the AS/400 **java** command to run a Java application. You may have noticed that while running the program you were switched to the Java shell. You pressed F3 to exit the shell and return to the AS/400 command prompt. Another way to get into this Java shell is by running Qshell. Qshell is a full function shell environment, but for the purpose of this lab it is simply a convenient place to run Java programs. You can navigate the file system and compile and run Java programs in Qshell.

### Using QShell

From the AS/400 command prompt, enter **QSH**. This starts the shell interpreter. If you are used to Windows or UNIX, the shell has a familiar look and feel. Do the following from the shell:

1. Set up the CLASSPATH environment variable. Enter

```
export CLASSPATH=/LAB15LF/Javaxx
```

2. Change directory to the directory containing your lab exercises. Enter

```
cd /LAB15LF/Javaxx
```

3. Display the contents of the directory. Qshell is like a UNIX shell. Enter:

```
ls
```

4. Run your Java program. Enter:

```
java CurrentInfo
```

5. Press F3 when you are ready to exit Qshell.

## Exercise 7: Create and run a simple Java servlet

### What This Exercise Is About

Servlets are Java programs that run in response to a request for a web page. When you type a URL in your browser window, the browser requests a web page from the web server. In many cases, that web page is simply a file sent to your browser. In the case of a servlet, the web server runs the servlet. The output of the servlet is a web page. Servlets allow you to easily generate web pages as they are requested - you can include up-to-the-minute information like inventory, stock price, or status information. In this exercise you will create a servlet which outputs the current date and time.

### What You Should Be Able To Do

At the end of this exercise, you should be able to:

- Create a servlet
- Compile a servlet with `javac`
- Run the `servletrunner` test server
- Test a servlet with `servletrunner`

### Introduction

When you create a Java servlet, you usually extend the `javax.servlet.http.HttpServlet` class, which is part of the Java Servlet Developer Kit (JSDK). This is not part of the JDK, but rather it is a Java standard extension. This means that it is standardized, but is a separate download from `java.sun.com`. Many Java development tools include it. The `javax.servlet.http.HttpServlet` class does most of the work. Your class will modify its behavior only when needing to define the output of the servlet. In this case your class will output the current date and time in a browser.

## Edit and Compile The Source

1. Use Notepad to edit CurrentInfoServlet.java.

```
notepad CurrentInfoServlet.java
```

When Notepad asks you if you want to create a new file, click on **Yes**.

2. Enter the code as follows:

```
import java.io.*;
import java.util.Date;
import javax.servlet.http.*;

public class CurrentInfoServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws IOException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("Current date and time: " + new Date());
    }
}
```

Notice that there is a bit more to this program than the others that you wrote. The web server calls the servlet's `doGet()` method everytime a client requests it. You overrode the `doGet()` method to define what your servlet does every time it is called. First you declare that the response will be HTML text. The `PrintWriter` is used for any of your servlet's output. The output of the servlet can then be any HTML text (including tags). In this case we, just printed some plain text with the current date and time.

3. Save the file using the **File->Save** menu.
4. It is time again to compile your source code into Java byte codes using the `javac` compiler. Type the following command in the MS-DOS prompt:

```
javac CurrentInfoServlet.java
```

5. If the source code compiles successfully, you will be presented again with the MS-DOS prompt. If there are errors in your source code, they will be displayed.
6. If you have any errors in your source code, go back to your Notepad window, fix the errors, save the file, and try to compile again.

### Run the servletrunner test server

servletrunner is a test server which comes with the JavaServlet Developers Kit (JSDK). It is not really a web server, but it is helpful when testingservlets. We will use it to test our servlet in action.

1. In your MS-DOS prompt, start servletrunner, telling it to use your lab directory as the location for servlets:

```
servletrunner -d C:\LAB15LF
```

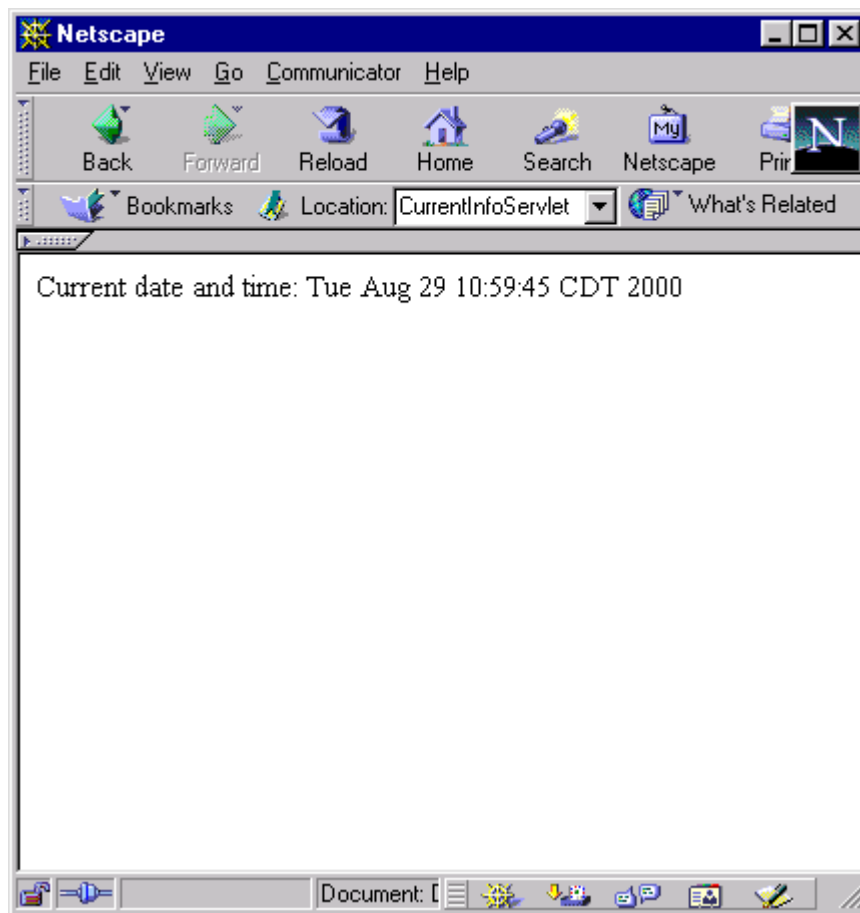
The tool will print information about its running state. After you see this information, servletrunner is ready to use.

```
servletrunner starting with settings:
port = 8080
backlog = 50
max handlers = 100
timeout = 5000
servlet dir = C:\LAB15LF
document dir = .\examples
servlet propfile = C:\LAB15LF\servlet.properties
```

2. Bring up the browser window. Enter the following URL. The lab instructor will tell you what your hostname is:

```
http://hostname:8080/servlet/CurrentInfoServlet
```

The page should look like the following.



When the browser requests the page, the servlet runs on the web server and its output is sent to the browser window. Press the **Reload** button a few times and notice that the servlet runs each time you load the page.

Note: Normally the web server will be running on a central location and clients all over the network will be accessing it. In our small example, the client and server are running on the same machine.



## Conclusion

In this lab, you had the opportunity to code simple Java programs in many environments. You saw a Java application, a Java applet, and a Java servlet. In addition, you ran a Java application on an AS/400 and learn about AS/400 Java program optimization.

Your next step is to delve deeper into the Java language itself. There are many books, classes, and online tutorials to help you with this. As with any programming language, the key to learning Java is to use it. Choose a simple project that will help you (a checkbook register, to-do list, fantasy football, etc) or your team (vacation scheduler, problem tracker, etc) and work on it a little at a time. As you add features, you can learn new Java skills, such as File I/O, network access, or graphical user interfaces.