2. The Hello World Application

Java

Summer 2008

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- Creating Your First Application
- A Closer Look at the "Hello World!" Application
- References

Introduction

The "Hello World!" Application

- Your first application, HelloWorldApp, will simply display the greeting "Hello World!".
- We can use:
  - An Integrated Development Environments (IDE) or
  - A simple text editor.

"Hello World!" for Microsoft Windows

- To write your first program, you'll need:
  - The Java SE Development Kit 6 (JDK 6)
  - A text editor
- You can download the Windows version of JDK6 from:

  http://java.sun.com/javase/6/download.jsp
- We use a text editor to create and edit source files. We will can use ***Notepad***, a simple editor included with the Windows platforms.

Update the PATH variable

- Set the PATH variable if you want to be able to conveniently run the JDK executables (javac.exe, java.exe, javadoc.exe, etc.) from any directory without having to type the full path of the command.
- If you don't set the PATH variable, you need to specify the full path to the executable every time you run it, such as:
  - C:> **"\Program Files\Java\jdk1.6.0_<version>\bin\javac" MyProgram.java**

Update the PATH variable

- To set the PATH permanently, add the full path of the **jdk1.6.0_<version>\bin** directory to the PATH variable.
- Set the PATH as follows on Microsoft Windows:
  - Click **Start > Control Panel > System** on Windows XP or **Start > Settings > Control Panel > System** on Windows 2000.
  - Click **Advanced > Environment Variables**.
  - Add the location of bin folder of JDK installation for PATH in User Variables and System Variables.
- A typical value for PATH is: **C:\Program Files\Java\jdk1.6.0_<version>\bin**

Update the PATH variable

- PATH environment variable is a series of directories separated by semi-colons (;) and is not case sensitive.
- Microsoft Windows looks for programs in the PATH directories in order, from left to right.
- Add the path to the right end of the PATH in the User Variables.
- The new path takes effect in each new command window you open after setting the PATH variable.

Creating Your First Application

Creating Your First Application

- To create this program, you will:
  - ***Create a source file***.
    - A source file contains code, written in the Java programming language.
  - ***Compile the source file into a .class file***.
    - The Java programming language compiler (javac) takes your source file and translates its text into instructions that the Java virtual machine can understand.
    - The instructions contained within this file are known as *bytecodes*.
  - ***Run the program***.
    - The Java application launcher tool (java) uses the Java virtual machine to run your application.

Create a Source File

- Start your editor, and in a new document, type in the following code:

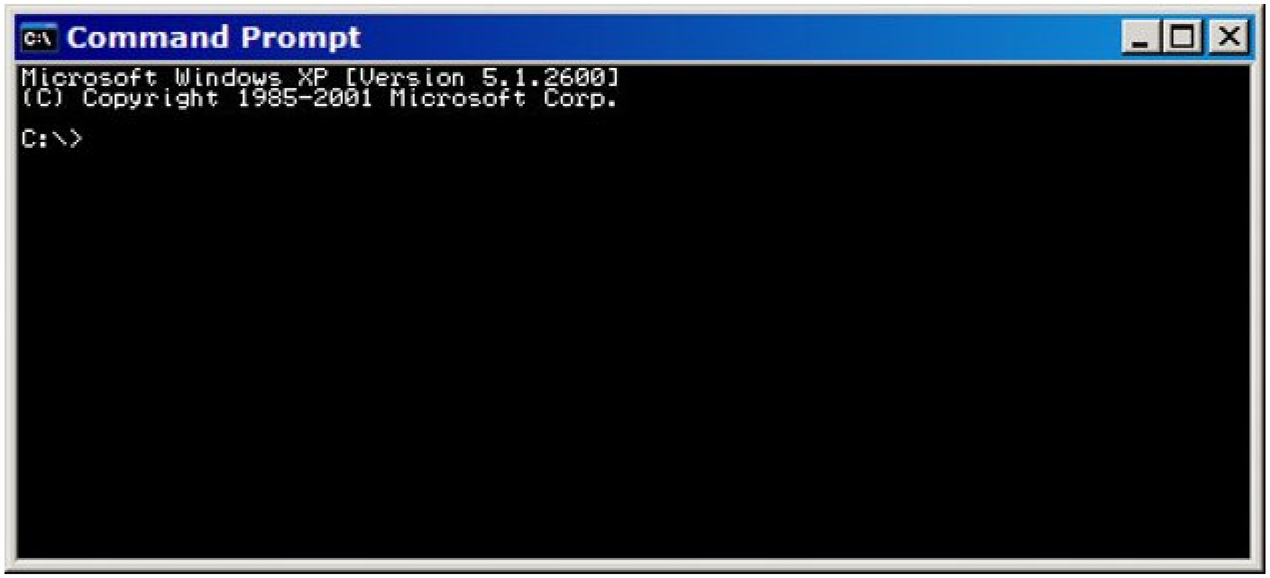
```
/**
    * The HelloWorldApp class implements an application that
    * simply prints "Hello World!" to standard output.
*/
class HelloWorldApp {
    public static void main(String[] args) {
    System.out.println("Hello World!"); // Display the string.
    }
}
```

Create a Source File

- Type all code, commands, and file names exactly as shown.
- Save the code in a file with the name ***HelloWorldApp.java***.

Compile the Source File into a .class File

- Bring up a shell, or "command," window.
- You can do this from the Start menu by choosing Command Prompt (Windows XP), or by choosing Run . . . and then entering ***cmd***.

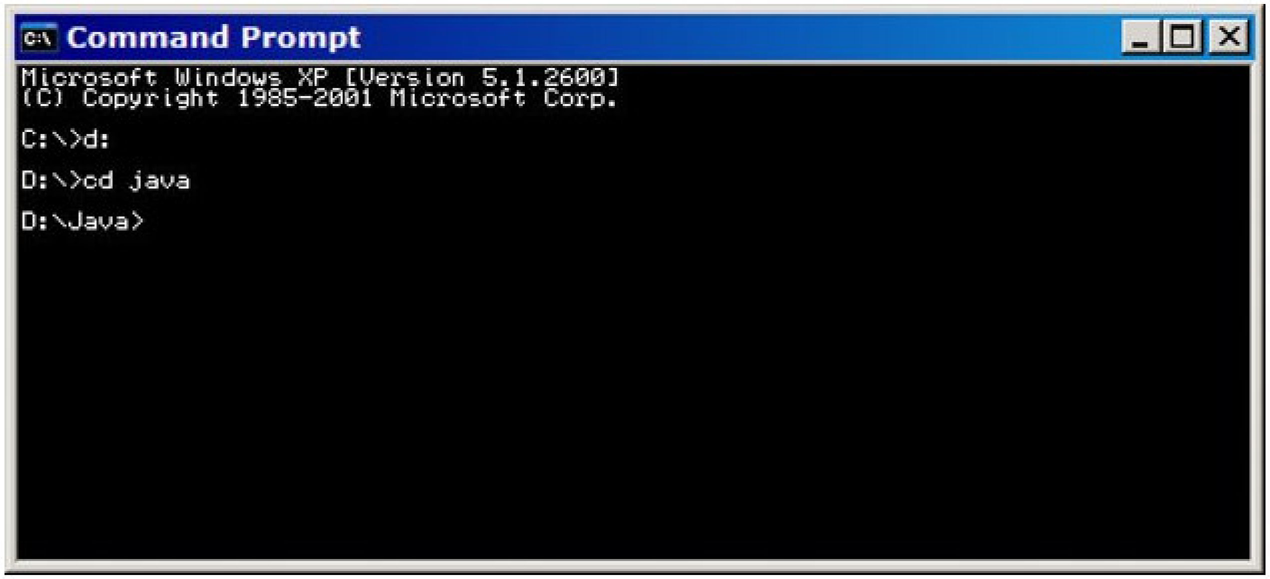


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>
```

Compile the Source File into a .class File

- To compile your source file, change your current directory to the directory where your file is located.
- If your source directory is ***java*** on the the ***D*** drive, type the following command at the prompt and press Enter:

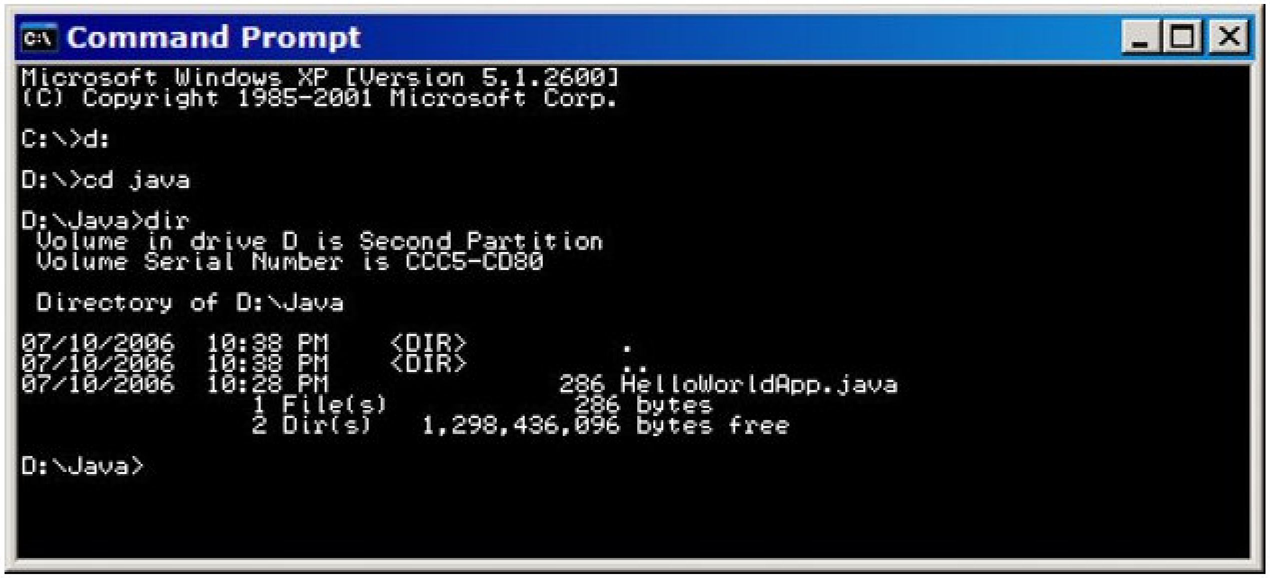
```
D:
cd C:\java
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>d:

D:\>cd java

D:\Java>
```

Compile the Source File into a .class File

- If you enter dir at the prompt, you should see your source file:

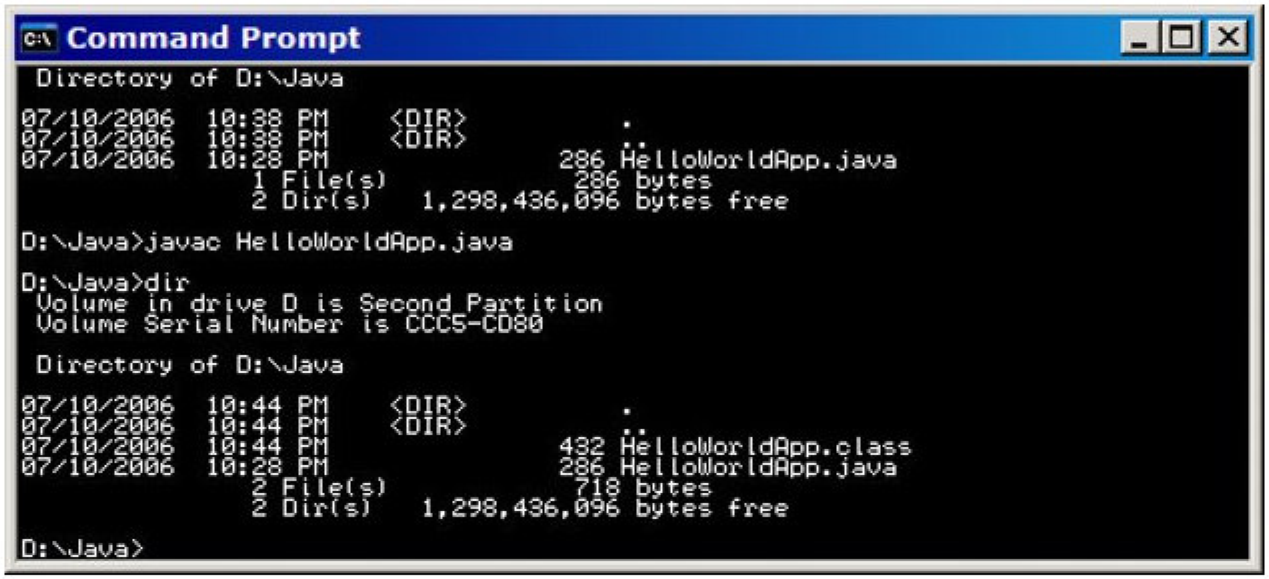


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>d:

D:\>cd java

D:\Java>dir
 Volume in drive D is Second Partition
 Volume Serial Number is CCC5-CD80

 Directory of D:\Java

07/10/2006  10:38 PM    <DIR>          .
07/10/2006  10:38 PM    <DIR>          ..
07/10/2006  10:28 PM               286 HelloWorldApp.java
               1 File(s)            286 bytes
               2 Dir(s)   1,298,436,096 bytes free

D:\Java>
```

Compile the Source File into a .class File

- At the prompt, type the following command and press Enter:

  javac HelloWorldApp.java
- The compiler has generated a bytecode file, **HelloWorldApp.class**.
- At the prompt, type dir to see the new file that was generated

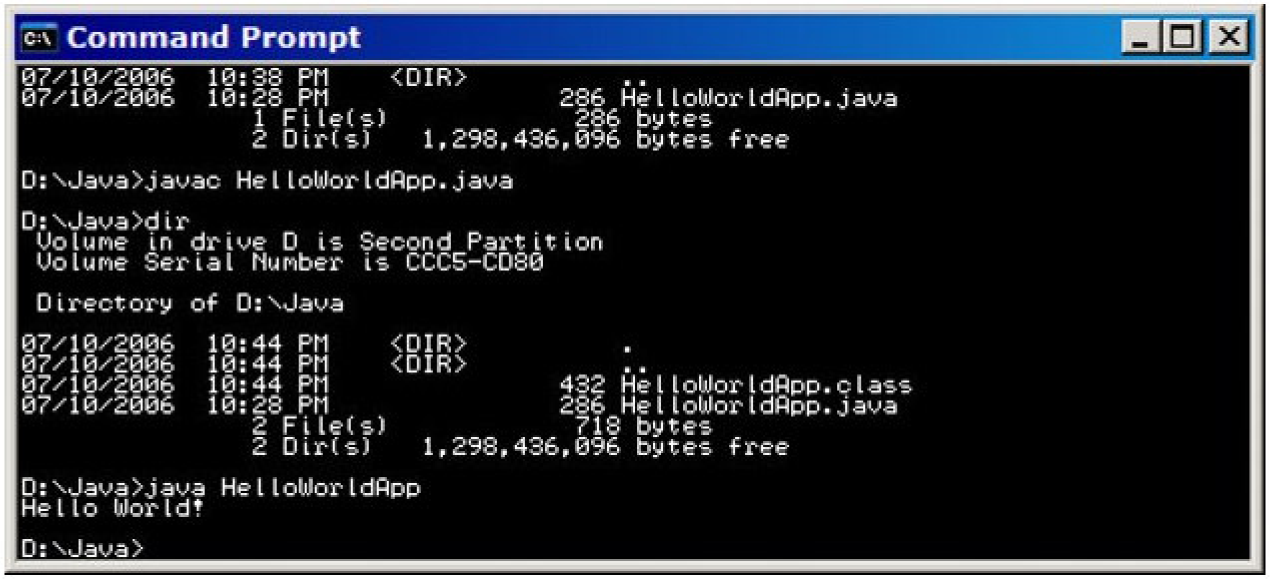


```
 Directory of D:\Java

07/10/2006  10:38 PM    <DIR>          .
07/10/2006  10:38 PM    <DIR>          ..
07/10/2006  10:28 PM               286 HelloWorldApp.java
               1 File(s)            286 bytes
               2 Dir(s)   1,298,436,096 bytes free

D:\Java>javac HelloWorldApp.java

D:\Java>dir
 Volume in drive D is Second Partition
 Volume Serial Number is CCC5-CD80

 Directory of D:\Java

07/10/2006  10:44 PM    <DIR>          .
07/10/2006  10:44 PM    <DIR>          ..
07/10/2006  10:44 PM               432 HelloWorldApp.class
07/10/2006  10:28 PM               286 HelloWorldApp.java
               2 File(s)            718 bytes
               2 Dir(s)   1,298,436,096 bytes free

D:\Java>
```

Run the Program

- In the same directory, enter the following command at the prompt:

  java HelloWorldApp



```
07/10/2006  10:38 PM    <DIR>          ..
07/10/2006  10:28 PM               286 HelloWorldApp.java
               1 File(s)            286 bytes
               2 Dir(s)   1,298,436,096 bytes free

D:\Java>javac HelloWorldApp.java

D:\Java>dir
 Volume in drive D is Second Partition
 Volume Serial Number is CCC5-CD80

 Directory of D:\Java

07/10/2006  10:44 PM    <DIR>          .
07/10/2006  10:44 PM    <DIR>          ..
07/10/2006  10:44 PM               432 HelloWorldApp.class
07/10/2006  10:28 PM               286 HelloWorldApp.java
               2 File(s)            718 bytes
               2 Dir(s)   1,298,436,096 bytes free

D:\Java>java HelloWorldApp
Hello World!

D:\Java>
```

javac & java are Case Sensitive

- Both the compiler (javac) and launcher (java) are case-sensitive.
- In other words, HelloWorldApp is not equivalent to helloworldapp.

A Closer Look at the "Hello World!" Application

A Closer Look at the "Hello World!" Application

- Here again is its code:

```
/**
    * The HelloWorldApp class implements an application that
    * simply prints "Hello World!" to standard output.
    */
class HelloWorldApp {
    public static void main(String[] args) {
            System.out.println("Hello World!"); //Display the string.
    }
}
```

A Closer Look at the "Hello World!" Application

- The "Hello World!" application consists of three primary components:
  - Source code comments,
  - The HelloWorldApp class definition, and
  - The main method.
- The following explanation will provide you with a basic understanding of the code

A Closer Look at the "Hello World!" Application

- The following bold text defines the comments:

```
/**
    * The HelloWorldApp class implements an application that
    * simply prints "Hello World!" to standard output.
    */
class HelloWorldApp {
    public static void main(String[] args) {
            System.out.println("Hello World!"); //Display the string.
    }
}
```

Source code comments

- Comments are ignored by the compiler but are useful to other programmers.
- The Java programming language supports three kinds of comments:
  - /* text */
    - The compiler ignores everything from /* to */.
  - /** documentation */
    - This indicates a documentation comment (doc comment, for short).
    - The compiler ignores this kind of comment, just like it ignores comments that use /* and */.
    - The ***javadoc tool*** uses doc comments when preparing automatically generated documentation.
  - // text
    - The compiler ignores everything from // to the end of the line.

HelloWorldApp class definition

- HelloWorldApp class definition:

```
class name {
...
}
```

- The keyword class begins the class definition for a class named name, and the code for each class appears between the opening and losing curly braces marked in bold above.
- We will classes in detail later.
- For now it is enough to know that every application begins with a class definition.

The *main* Method

- **The following bold text begins the definition of the main method:**

```
/**
    * The HelloWorldApp class implements an application that
    * simply prints "Hello World!" to standard output.
    */
class HelloWorldApp {
    public static void main(String[] args) {
            System.out.println("Hello World!"); //Display the string.
    }
}
```

The *main* Method

- In the Java programming language, every application must contain a *main* method whose signature is:

  public static void main(String[] args)
- It's the entry point for your application and will subsequently call up all the other methods required by your program.
- The modifiers public and static can be written in either order (*public static* or *static public*) but the convention is to use *public static*

The *main* Method

- The main method accepts a single argument: an array of elements of type String.

  *public static void main(**String[] args**)*

- You can name the argument anything you want, but most programmers choose "args" or "argv."
- This array is the mechanism through which the runtime system passes information to your application.

The *main* Method

- Each string in the array is called a command-line argument.
- Command-line arguments let users affect the operation of the application without recompiling it.
- The "Hello World!" application ignores its command-line arguments, but you should be aware of the fact that such arguments do exist.

The *main* Method

- Finally, the following line uses the System class from the API to print the "Hello World!" message to standard output:

  System.out.println("Hello World!");

References

References

- S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, **The Java Tutorial: A Short Course on the Basics**, 4th Edition, Prentice Hall, 2006.

The End