Programming Concepts – COP 2510

Introduction

What is computer programming?

- What is a computer program?
  - A set of instructions written in a sequence that tells the computer what to do.

- *Computer programming* is the process of writing, testing, and maintaining the instructions.

What can a computer do?

Only very simple things:

- Store numbers in memory

  in the form of 1’s and 0’s
- Retrieve numbers from memory
- Add one number to another
  - Subtract, multiply, divide
- Compare one number to another
- Read from keyboard or device
- Write to a screen or device

Computer *hardware*

What can a computer program do?

Computer *programs* can do very complex things.

- Video games
- Corporation book keeping
- Tax accounting
- Face recognition
- Communicate sounds and images
- Play chess

Computer *software*

What can a computer do?

- For every complex thing that a computer can do, someone had to describe how to do it.
  - Ultimately in terms of the very simple things that the computer can do directly.

What can a computer do?

- Fortunately programmers don’t normally have to write instructions in terms of the basic hardware operations.

- Build on existing software
  - Programming language run time system
  - Software libraries
  - Tools (existing computer programs)
    - Operating systems
    - Compilers

Programming Languages

- Virtually all software development is done in *programming languages*
  - Not basic computer hardware instructions.

- Programmer writes a text file in a programming language
  - Easier to write and more understandable that basic computer instructions.

- Programmer’s instructions are translated into machine instructions by an existing program

Programming Languages

- Assembly language
  - Close to the hardware instructions
  - Programs specific to a kind of computer
- "Higher level" languages
  - *Many* examples
  - Fortran, COBOL
  - C, C++, C#
  - Java

Focus of this Course

- Problem solving
- Program design
- Implementation
- Testing
- Debugging

Programming Experience

- Course assumes *no programming experience*
- It assumes minimal experience with computers
  - How to start up and shut down
  - How to use keyboard and mouse
  - How to browse the web.

Today’s focus

- An introduction to Java
- Program development
- Handling program errors

Introduction to Java

C-Based Languages

- C was developed at Bell Laboratories by Ken Thompson, Dennis Ritchie, and others, during the development of Unix in the late 1960s.
- C++ includes all the features of C, but adds classes and other features to support object-oriented programming.
- Java is based on C++ and therefore inherits many C features. (Created by Sun Microsystems, in 1995)
- C# is a more recent language derived from C++ and Java. (Created by Microsoft in the early 2000's.)

Java Program Structure

- In the Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements, each statement is followed by a semicolon*

  - Every method has a *name*
  - A statement in one method can invoke (or *call*) another method in the same class.
    - Or possibly in a different class.

Java Program Structure

```
// comments about the class
public class Lincoln
{




}
```

class header

class body

Comments can be placed almost anywhere

Java Program Structure

```
//  comments about the class
public class Lincoln
{

   //  comments about the method
   public static void main (String[] args)
   {
                    method body                  method header

   }

}
```

Example: Lincoln.java

```
//demonstrates the basic structure of a Java application.
    public class Lincoln
    {
          //prints a presidential quote
       public static void main (String[] args)
       {
            System.out.println ("A quote by Abraham Lincoln:");
            System.out.println ("Whatever you are, be a good one.");
       }
    }
```

statements

each statement is followed by a semicolon

```
//********************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  Demonstrates the basic structure of a Java application.
//********************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

```
//********************
//  Lincoln.java
//
//  Demonstrates the basic structure of a Java application.
//********************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

Output

```
A quote by Abraham Lincoln:
Whatever you are, be a good one.
```

Key Concepts

- A Java program consists of one or more *classes*.
  - All executable code is contained in a method within a class.
- The name of the program's source file always matches the name of the class
  - For example, the name of the file for the previous example is *Lincoln.java*
- Statements in a method can invoke other methods.

Key Concepts

- A Java program always contains a method called main.
- main is the program’s starting point
- Everything that is done by a program is done by statements in main, or by statements in methods called, directly or indirectly, from main.

Comments

- Comments should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Styles of comments:

```
 // this comment runs to the end of the line


/*  this comment runs to the terminating
    symbol, even across line breaks        */
```

Identifiers

- The words a programmer uses in a program: *Identifiers*
- An identifier can be made up of letters, digits, the underscore character (_), and the dollar sign
- Identifiers cannot begin with a digit
- Java is *case sensitive* - `Total`, `total`, and `TOTAL` are different identifiers

Quick Check

Which of the following are valid Java identifiers?

```
grade
quizGrade!
frame2
3rdTestScore
MAXIMUM
MIN_CAPACITY
student#
Shelves1&2
```

Identifiers

- By convention, programmers use different case styles for different types of identifiers, such as
  - title case for class names - Lincoln
  - upper case for constants – MAXIMUM

Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as Lincoln)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language (such as public)
  - A reserved word cannot be used in any other way
  - The compiler will tell you if you make a mistake!

Reserved Words

- The Java reserved words:

abstract	else	int	strictfp
boolean	enum	interface	super
break	extends	long	switch
byte	false	native	synchronized
case	final	new	this
catch	finally	null	throw
char	float	package	throws
class	for	private	transient
const	goto	protected	true
continue	if	public	try
default	implements	return	void
do	import	short	volatile
double	instanceof	static	while

You don't have to memorize these.

White Space

- Spaces, blank lines, and tabs are called white space
  - used to separate words and symbols in a program
  - Extra white space is ignored
- A valid Java program can be formatted many ways
- Programs should be formatted to enhance readability, using consistent indentation – to convey the structure of the program

Example of poorly formatted program

Poorly formatted program:

```
//  Demonstrates a poorly formatted, cluttered, though
// valid, program.
public class Lincoln2{public static void main(String[]args){
System.out.println("A quote by Abraham Lincoln:");
System.out.println("Whatever you are, be a good one.");}}
```

Same program nicely formatted:

```
//********************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  Demonstrates the basic structure of a Java application.
//********************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

Indentation and Blank Lines

- Goal: Format program source code in order to improve its readability.
- Indentation: To place text farther to the right to separate it from surrounding text
  - **Use indentation for methods and statements**
  - **Indentation is four spaces**
- Blank Lines
  - Around class and method declarations
  - Around a group of logically connected statements

Program Development

Program Development

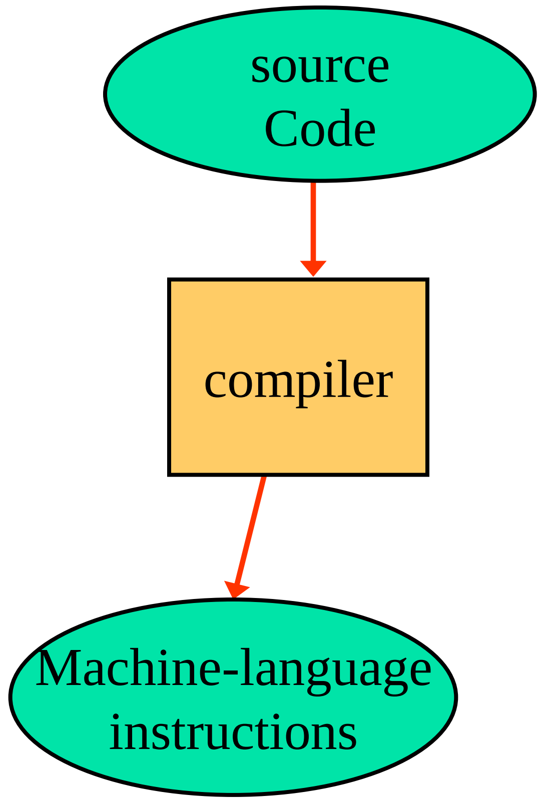
- The mechanics of developing a program include several activities
  - Writing the program in a specific programming language (such as Java) - edit
  - Translating the program into a form that the computer can execute - compile
  - Runing the program – test
  - Investigating and fixing various types of errors that can occur - debug

Program Development: Compile

- A compiler is a software tool that translates source code into a specific target language
  - first to recognize individual word and sentence units
  - then to analyze the syntax, or grammar, of the sentence
  - finally to translate the sentences into machine code.

Program Development: Compile

- Each type of CPU has its own specific machine language
- Often, that target language is the machine language for a particular CPU type

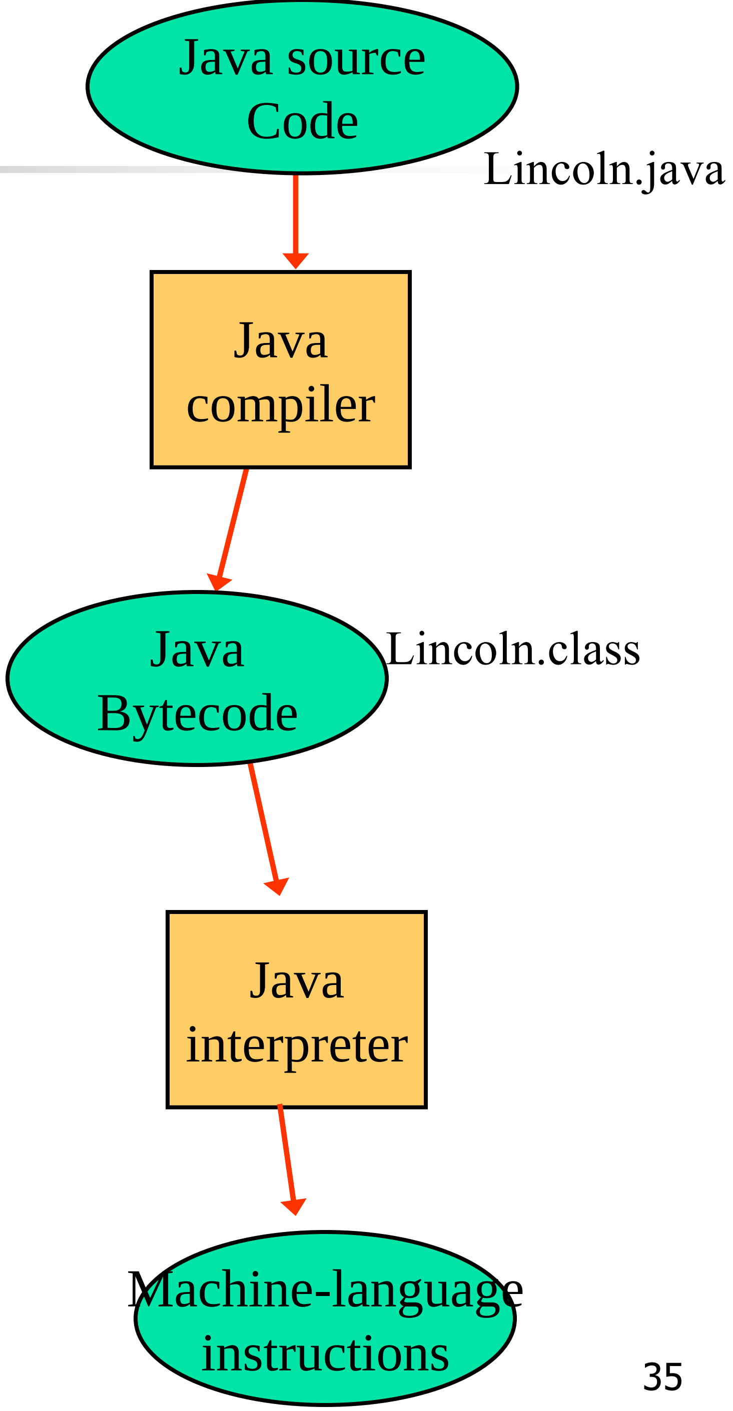


- The Java approach is somewhat different

Java Translation

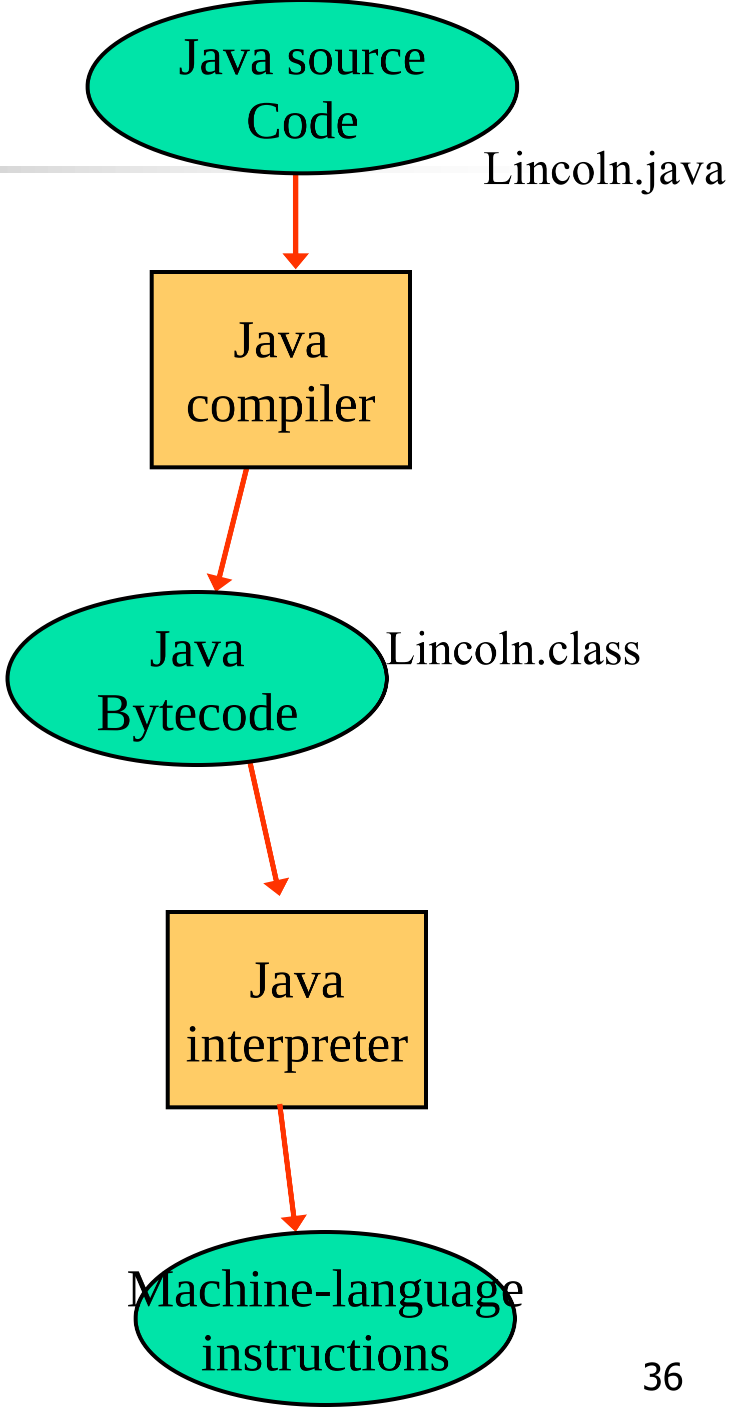
- The Java compiler translates Java source code into a special representation called bytecode

- Java bytecode is not the machine language for any traditional CPU

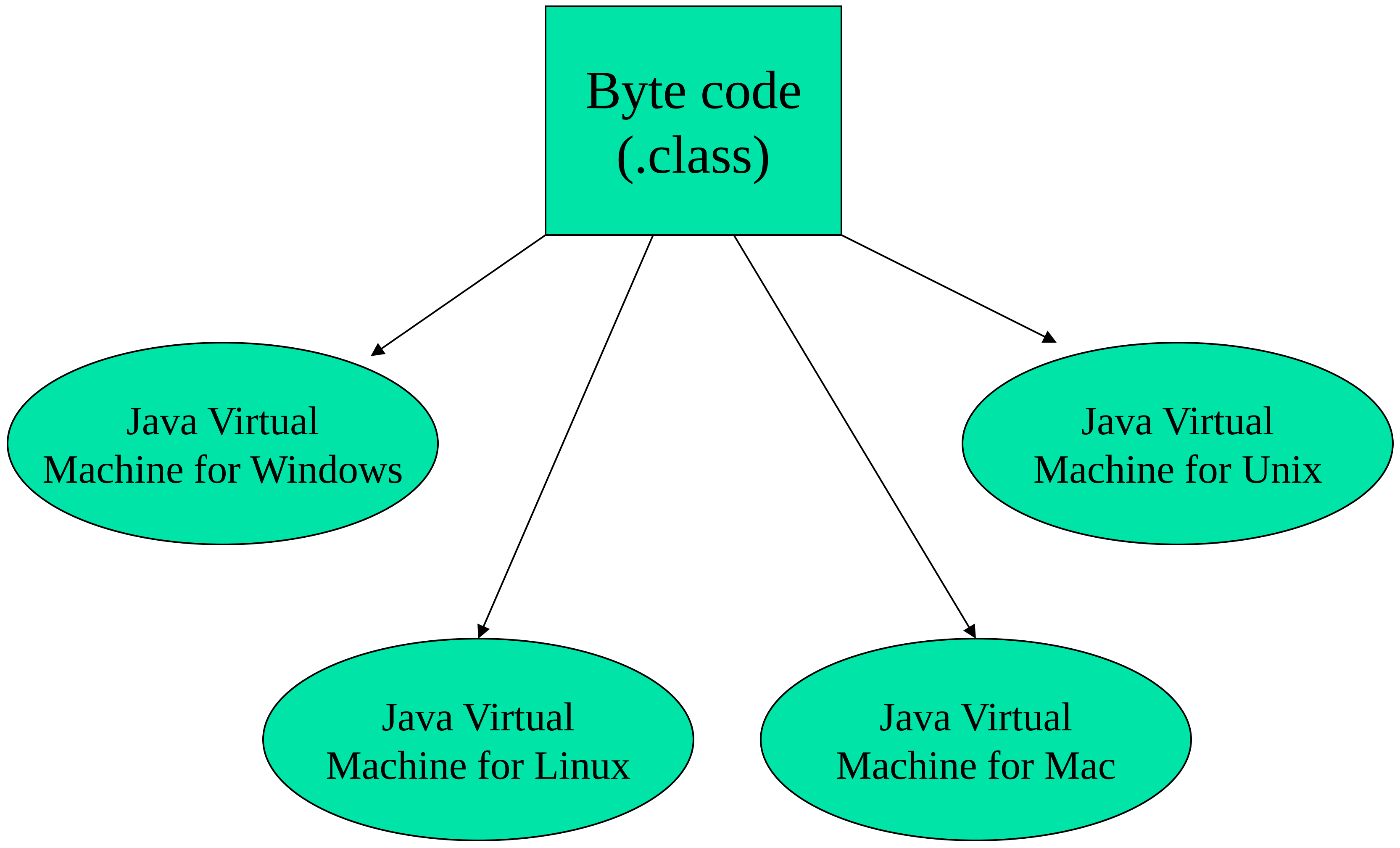


Java Translation

- Another software tool, called an interpreter, translates bytecode into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine – platform independent



Java Interpreter


Byte code
(.class)
Java Virtual Machine for Windows
Java Virtual Machine for Unix
Java Virtual Machine for Linux
Java Virtual Machine for Mac

Handling Programming Errors

Errors

- A program can have three types of errors
  - The compiler will find syntax errors
  - A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
  - A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)

Syntax Errors

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program

- If a program is not syntactically correct, the compiler will find syntax errors
  - If syntax errors exist, an executable version of the program is not created

Common Syntax Error Messages

- Syntax errors

  Error: Lincoln.java:15: ';' expected

```
System.out.println("A quote by Abraham Lincoln:")
                                                  ^
```

  Error: Lincoln.java:7: class HelloWorld should be declared in a file named HelloWorld.java

```
public class HelloWorld {
             ^
```

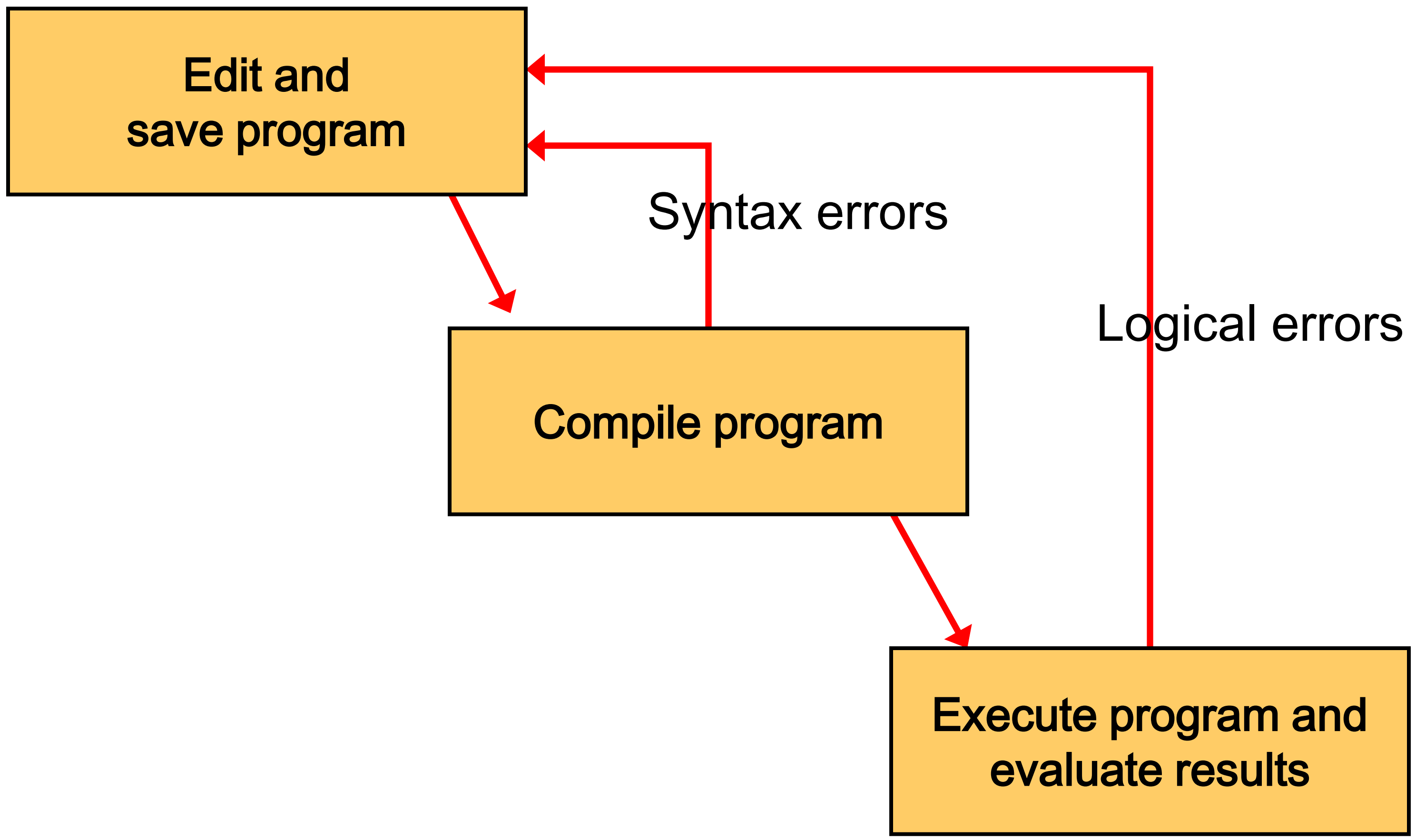
The name of the file must match the name of the class

Logical Errors

- A program may run, but produce incorrect results, perhaps using an incorrect formula
- Example: you have just created the code for a program which would display the first five powers of 2. You want to check whether it is working as intended. After compilation, your program runs and displays:

  ```
  0
  2
  4
  6
  8
  ```
- This type of error is logical error – errors (giving undesired output) due to flaws in the program design, not the syntax.

Basic Program Development



Development Environments

Development Environments

- There are many programs that support the development of Java software, including:
  - Oracle Java Development Kit (JDK)
  - Oracle NetBeans
  - IBM Eclipse
  - Borland JBuilder
  - MetroWerks CodeWarrior
  - BlueJ
  - jGRASP
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

Installing Java

To install Java on your PC see

Installing Java on Your PC

on the class web site.

http://www.cse.usf.edu/~turnerr/Programming_Concepts/

Your First Java Program

You will need a text editor program to create your source file.

A good programmer’s text editor for Windows is Notepad++

You can download it (free) from

https://notepad-plus-plus.org/download/

You can use other editor programs, even word processing programs, BUT be sure to save your program file as plain text

(NOT as a .docx or other word processing file.)

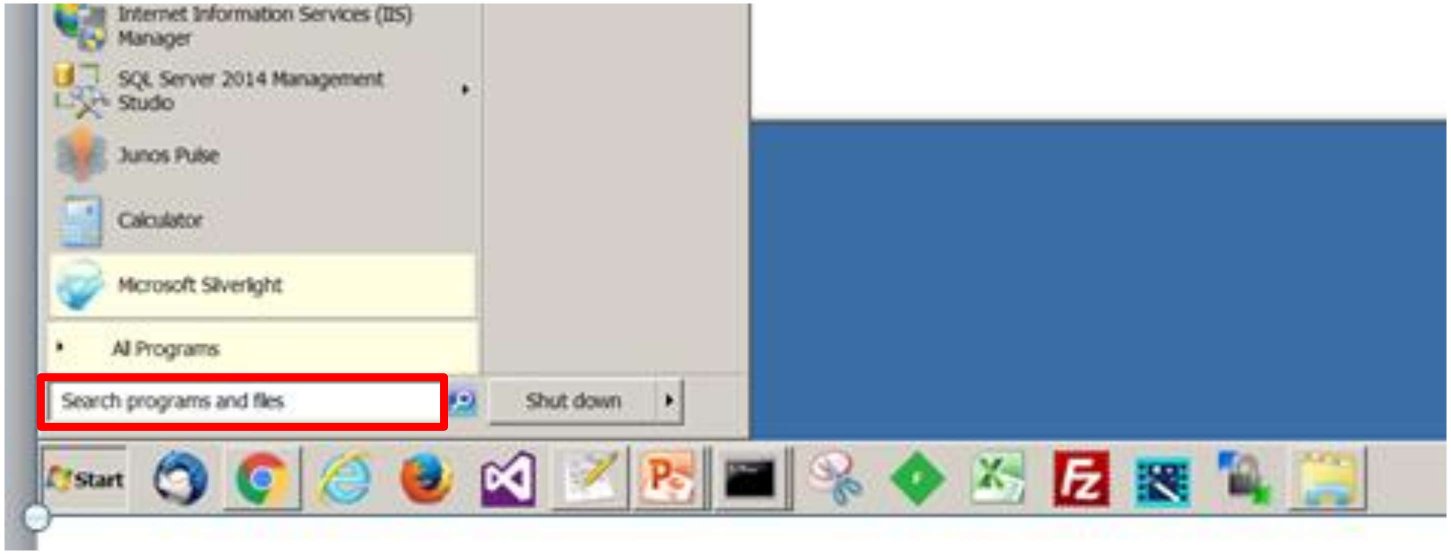
Your First Java Program

- The traditional first program is one that outputs the message “Hello, World!”

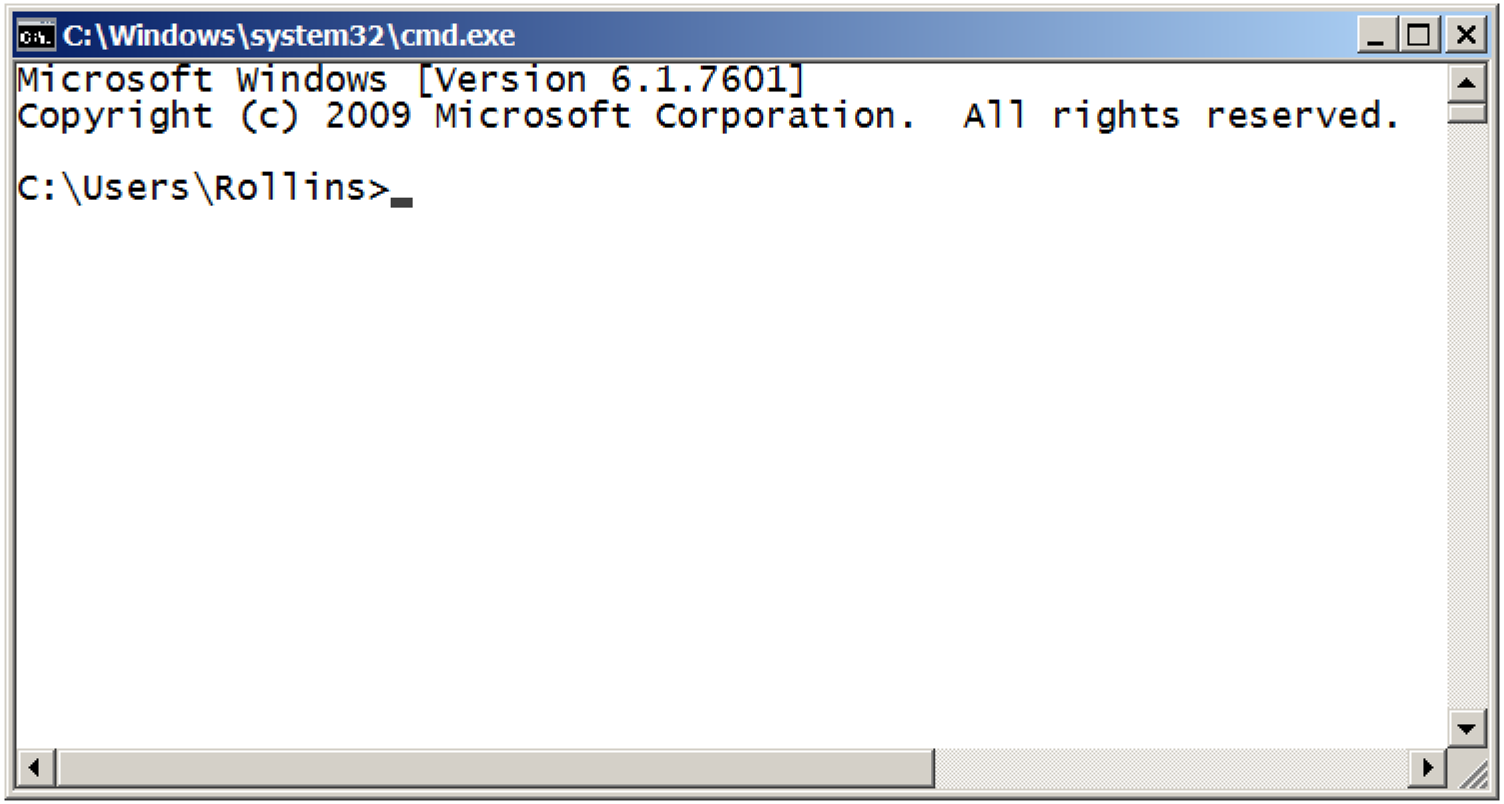
- Let’s do this in Java
  - Open a command window
  - Create a directory to hold the program.
  - Create the source file.
  - Compile the source file into a .class file.
  - Run the program.

Open a Command Window

In Windows7, click the Start button and type cmd into the search box

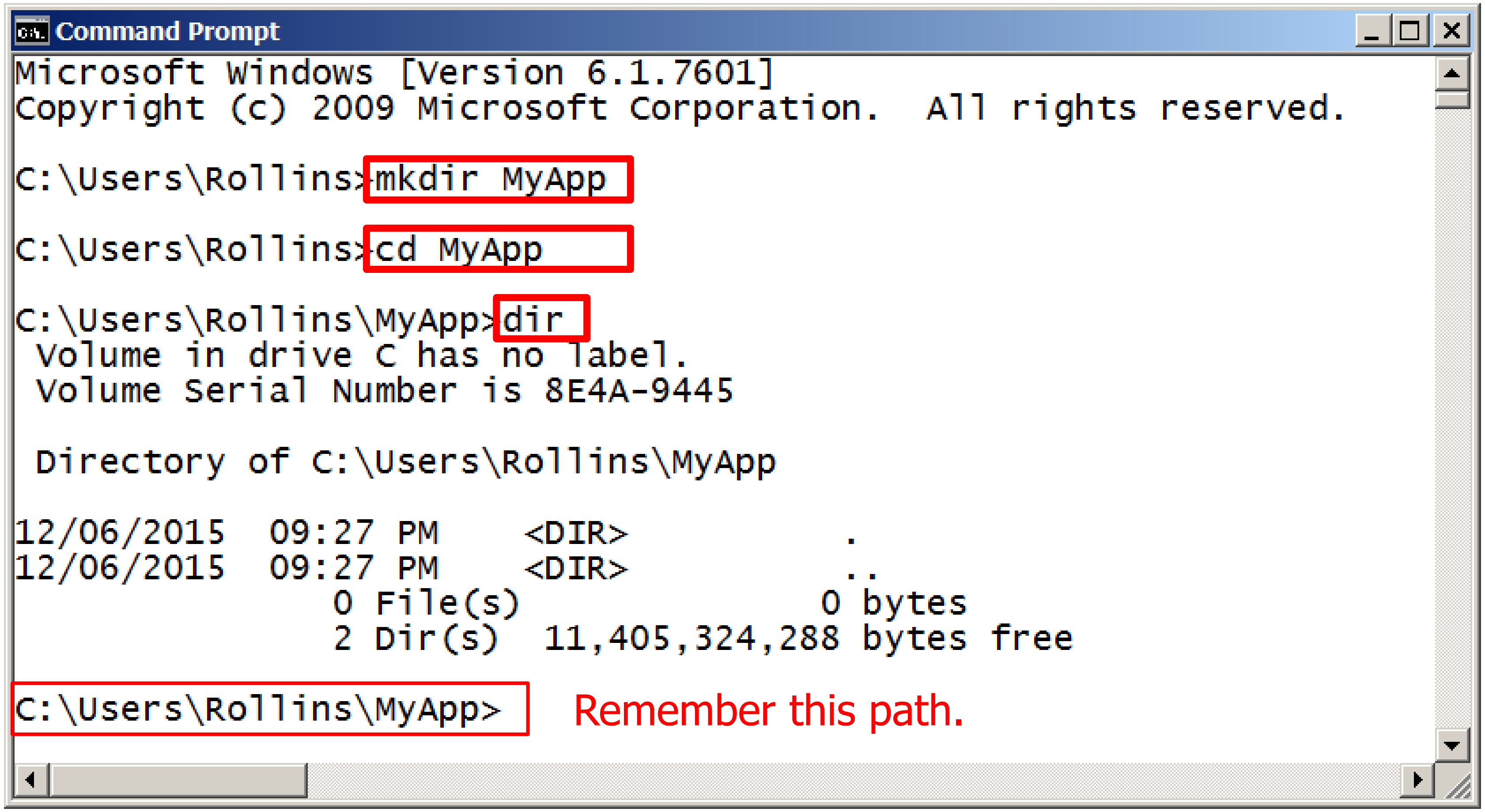

Internet Information Services (IIS) Manager
SQL Server 2014 Management Studio
Junos Pulse
Calculator
Microsoft Silverlight
All Programs
Search programs and files
Shut down
Start

The Command Window Opens



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Rollins>_
```

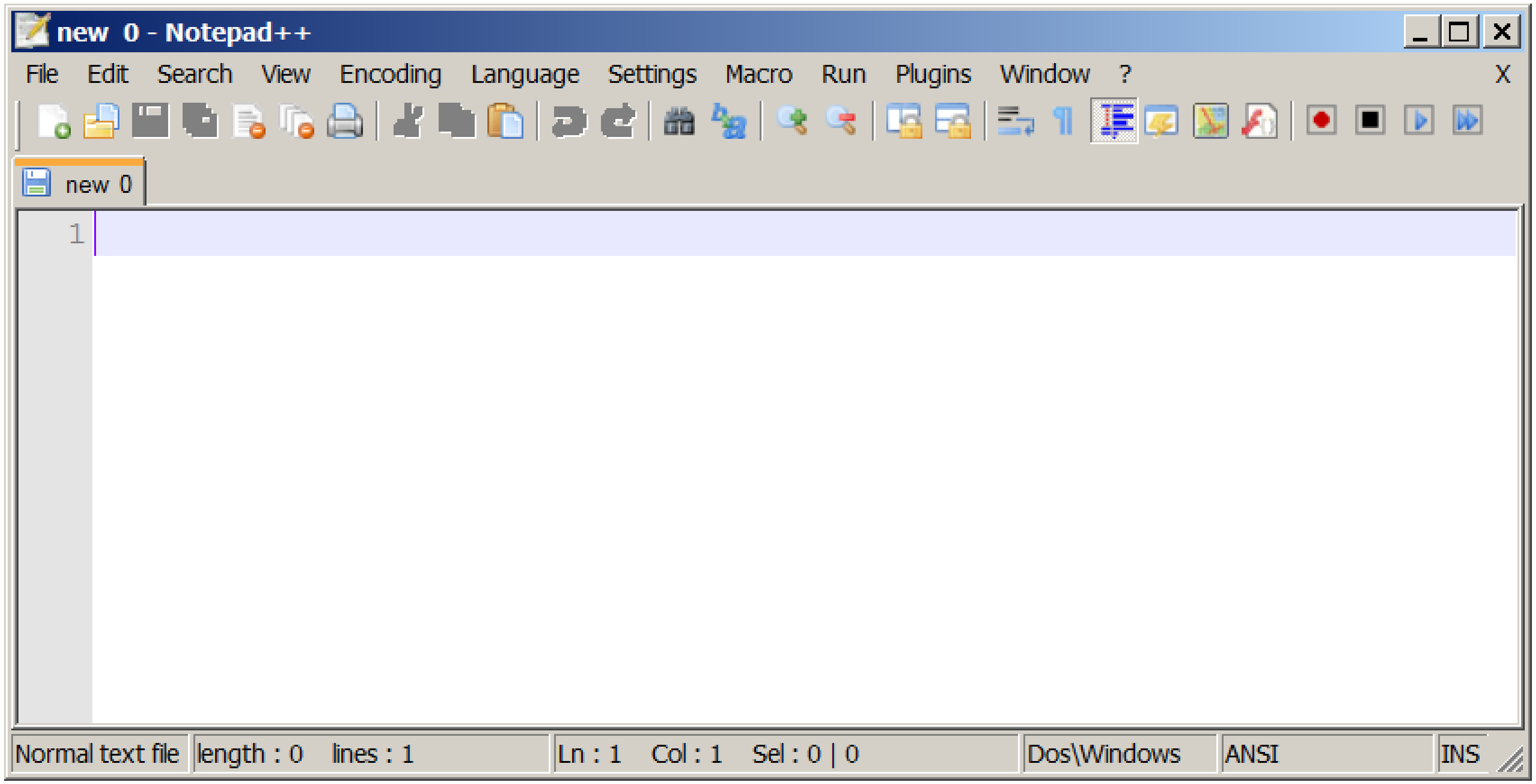
Create a directory to hold your program



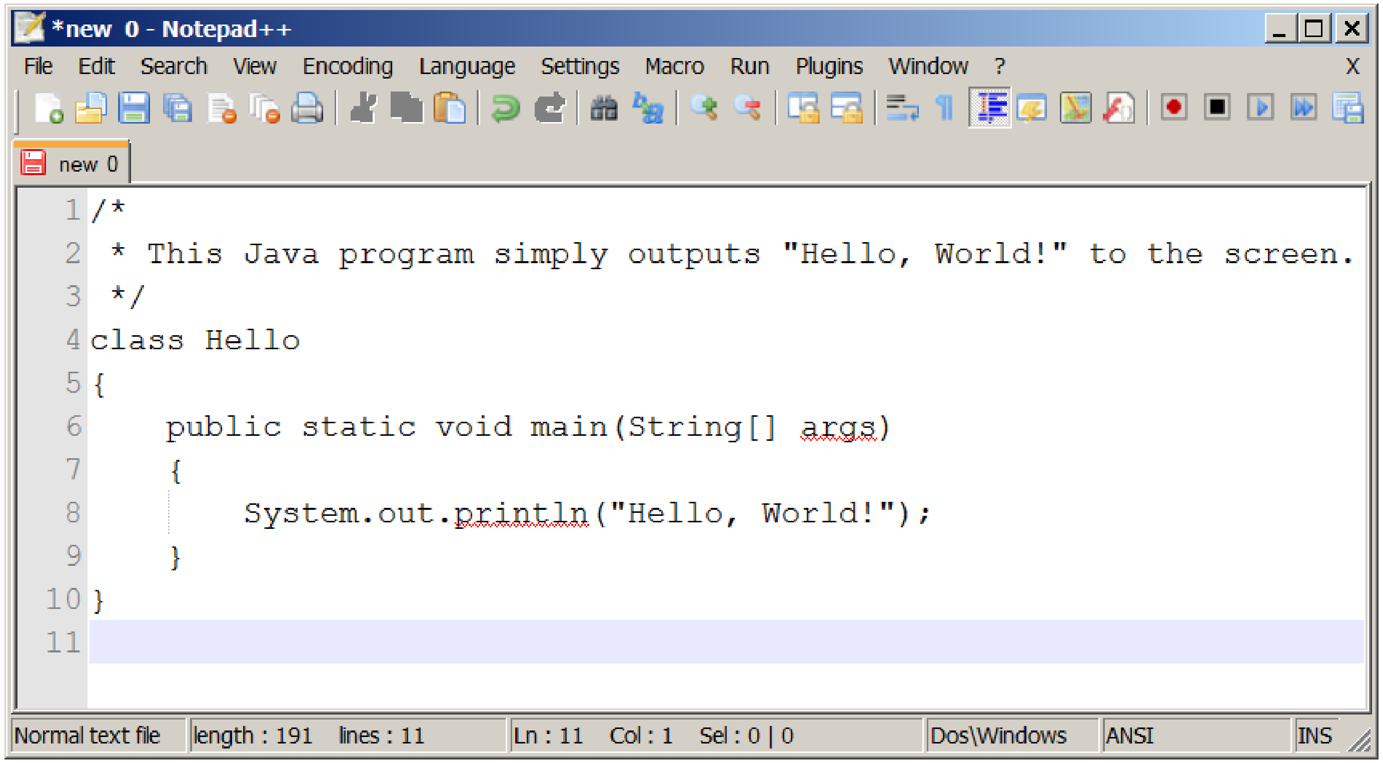
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Rollins>mkdir MyApp

C:\Users\Rollins>cd MyApp

C:\Users\Rollins\MyApp>dir
 Volume in drive C has no label.
 Volume Serial Number is 8E4A-9445

 Directory of C:\Users\Rollins\MyApp

12/06/2015  09:27 PM    <DIR>          .
12/06/2015  09:27 PM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  11,405,324,288 bytes free

C:\Users\Rollins\MyApp>
```

Remember this path.

Open Notepad++


new 0 - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? X
new 0
1
Normal text file
length : 0 lines : 1
Ln : 1 Col : 1 Sel : 0 | 0
Dos\Windows
ANSI
INS

Type the program

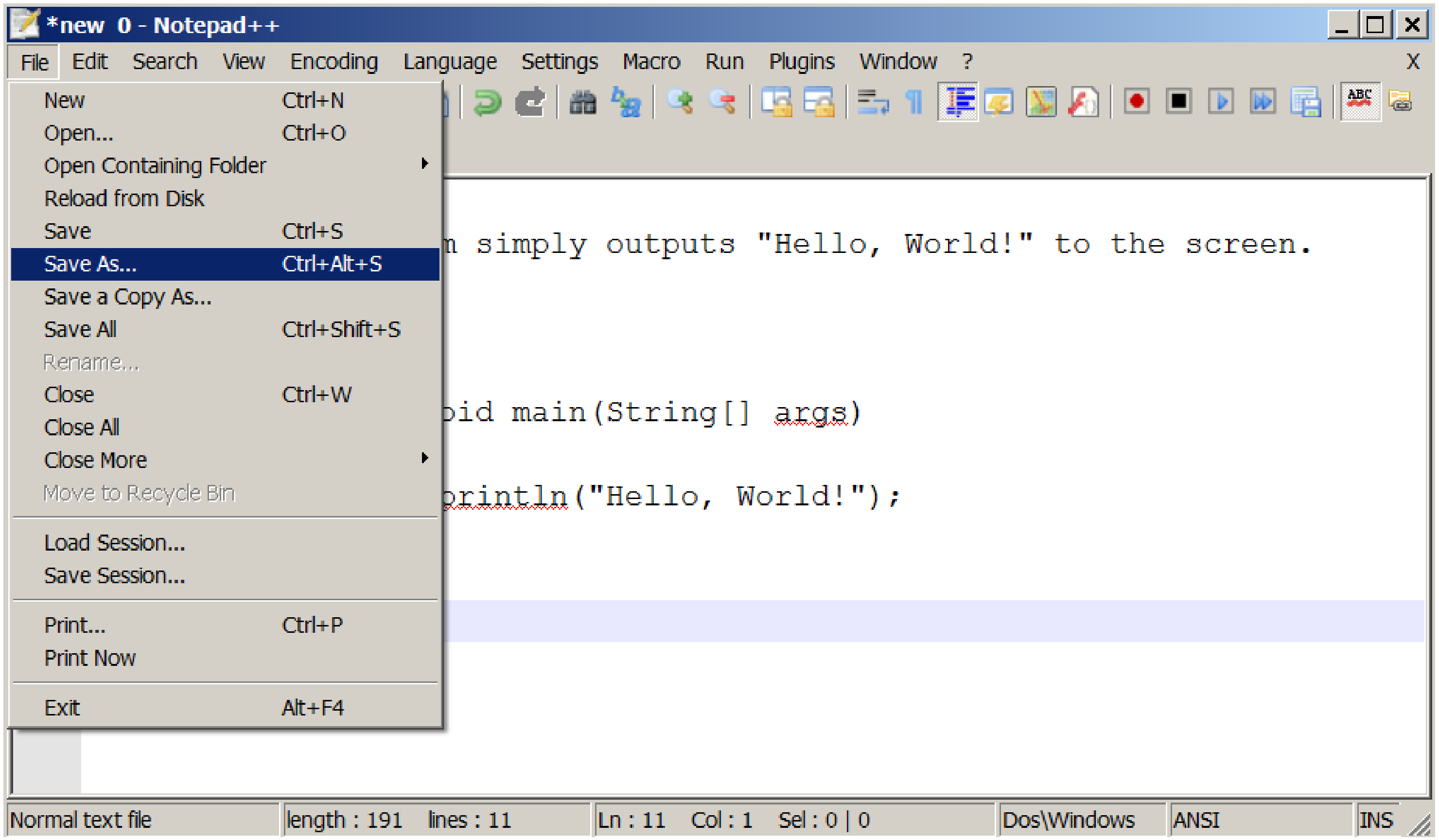

*new 0 - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
new 0

```
/*
 * This Java program simply outputs "Hello, World!" to the screen.
 */
class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}

```

Normal text file | length : 191 lines : 11 | Ln : 11 Col : 1 Sel : 0 | 0 | Dos\Windows | ANSI | INS

Save the file on disk

Click on File, in the menu bar, then select Save As...

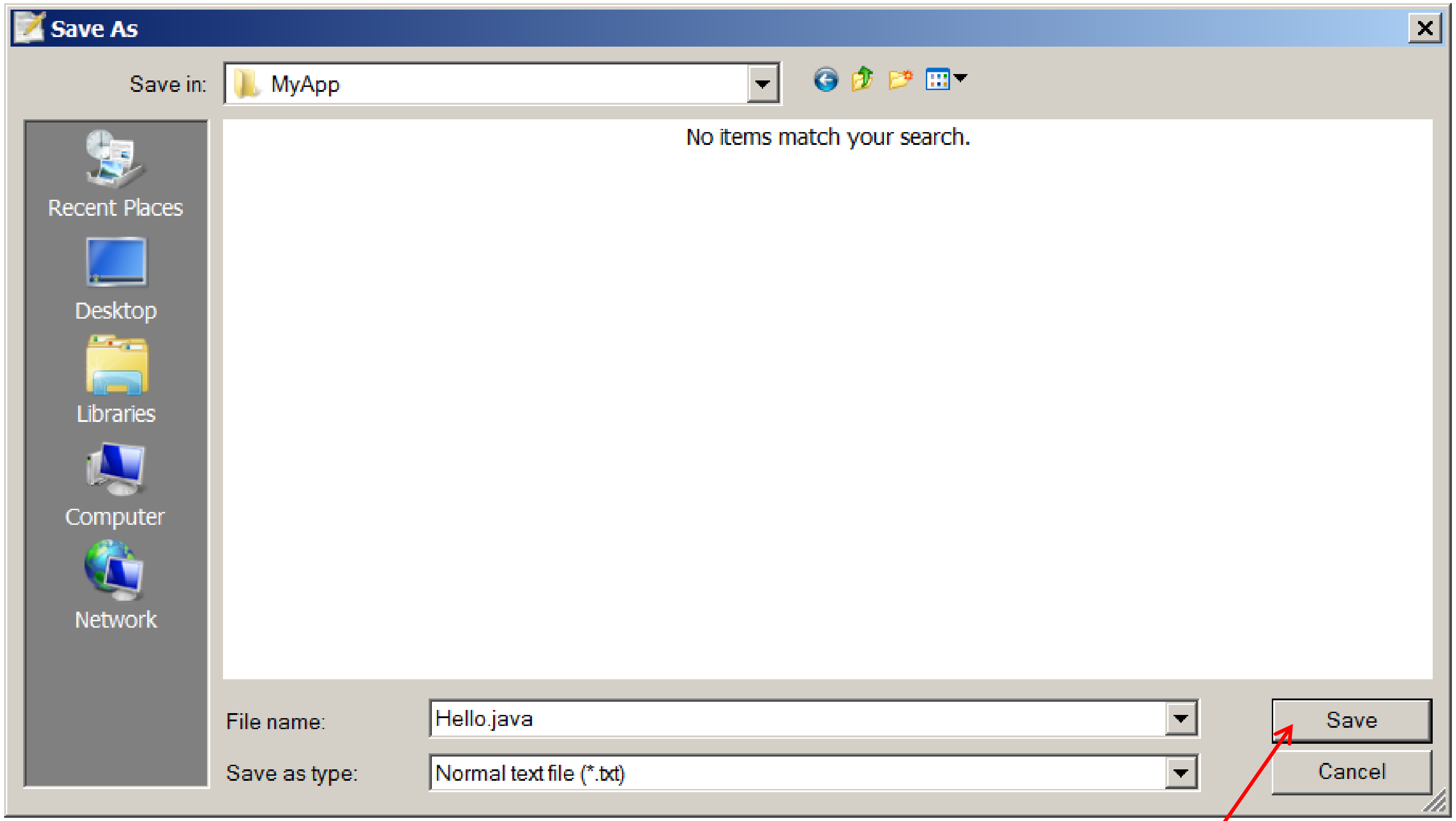


```
m simply outputs "Hello, World!" to the screen.

oid main(String[] args)

println("Hello, World!");
```

Normal text file | length : 191 lines : 11 | Ln : 11 Col : 1 Sel : 0 | 0 | Dos\Windows | ANSI | INS

Save As ...

- Navigate to the directory that you created earlier.
- Set the file name to match your class name, with the extension java.



Save in: MyApp

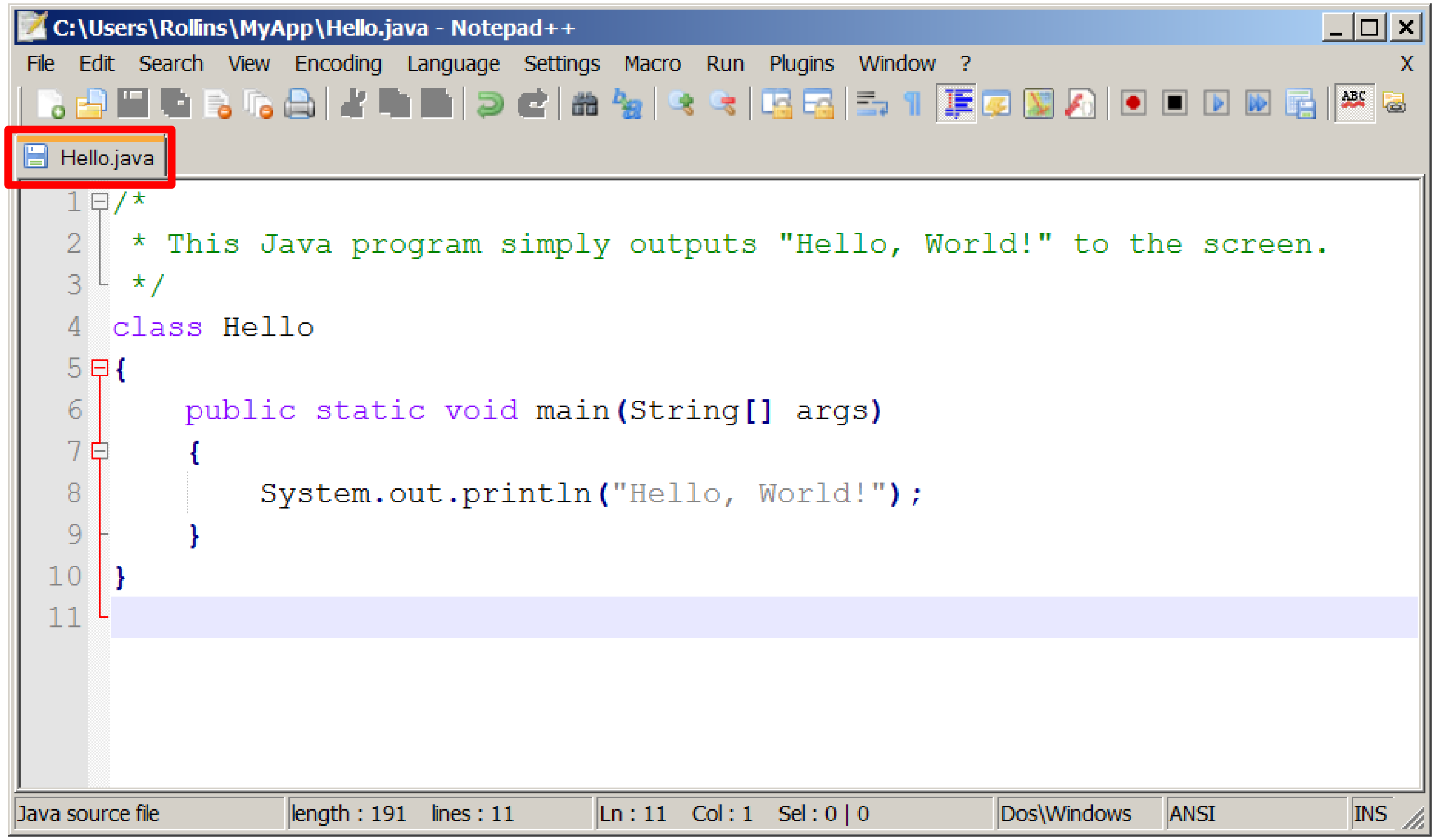
No items match your search.

File name: Hello.java

Save as type: Normal text file (*.txt)

Click Save

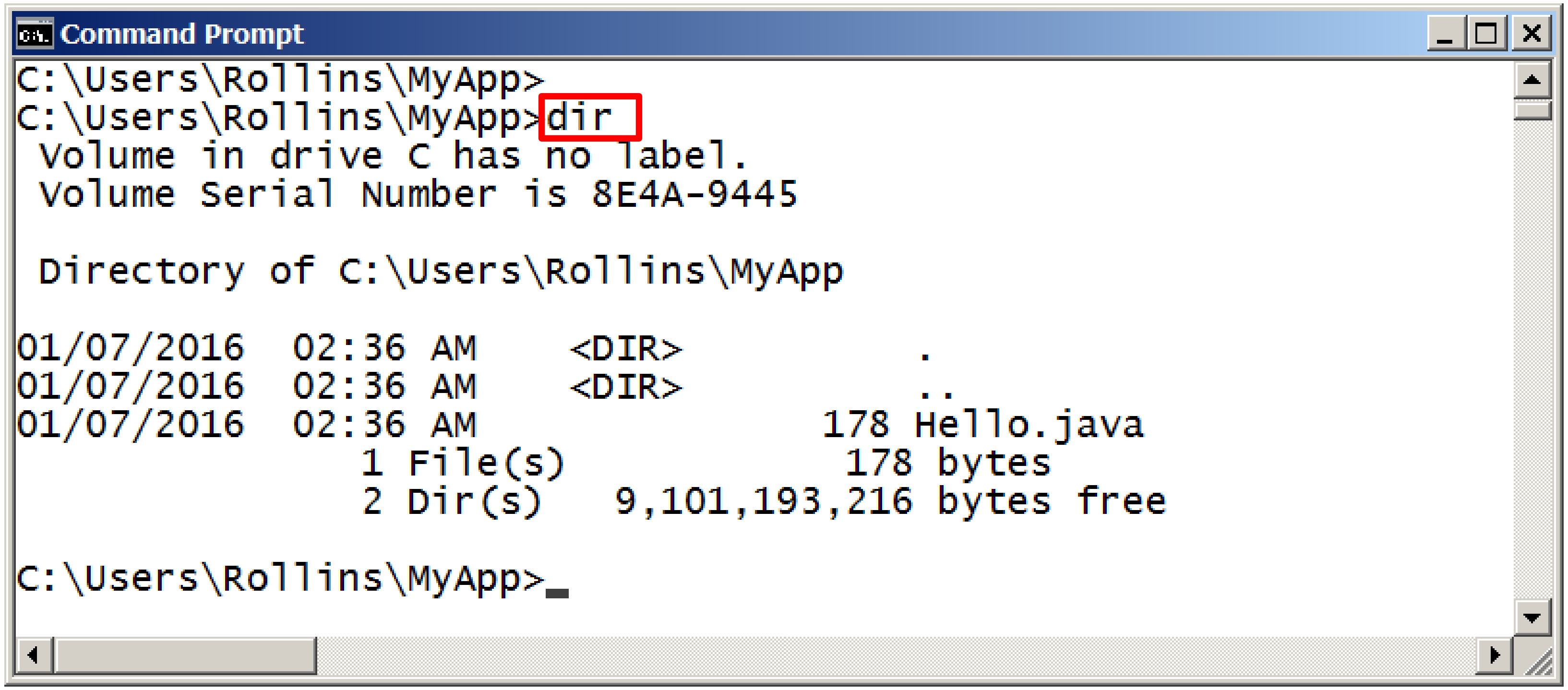
The file has been written to disk.


C:\Users\Rollins\MyApp\Hello.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Hello.java

```java
/*
 * This Java program simply outputs "Hello, World!" to the screen.
 */
class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

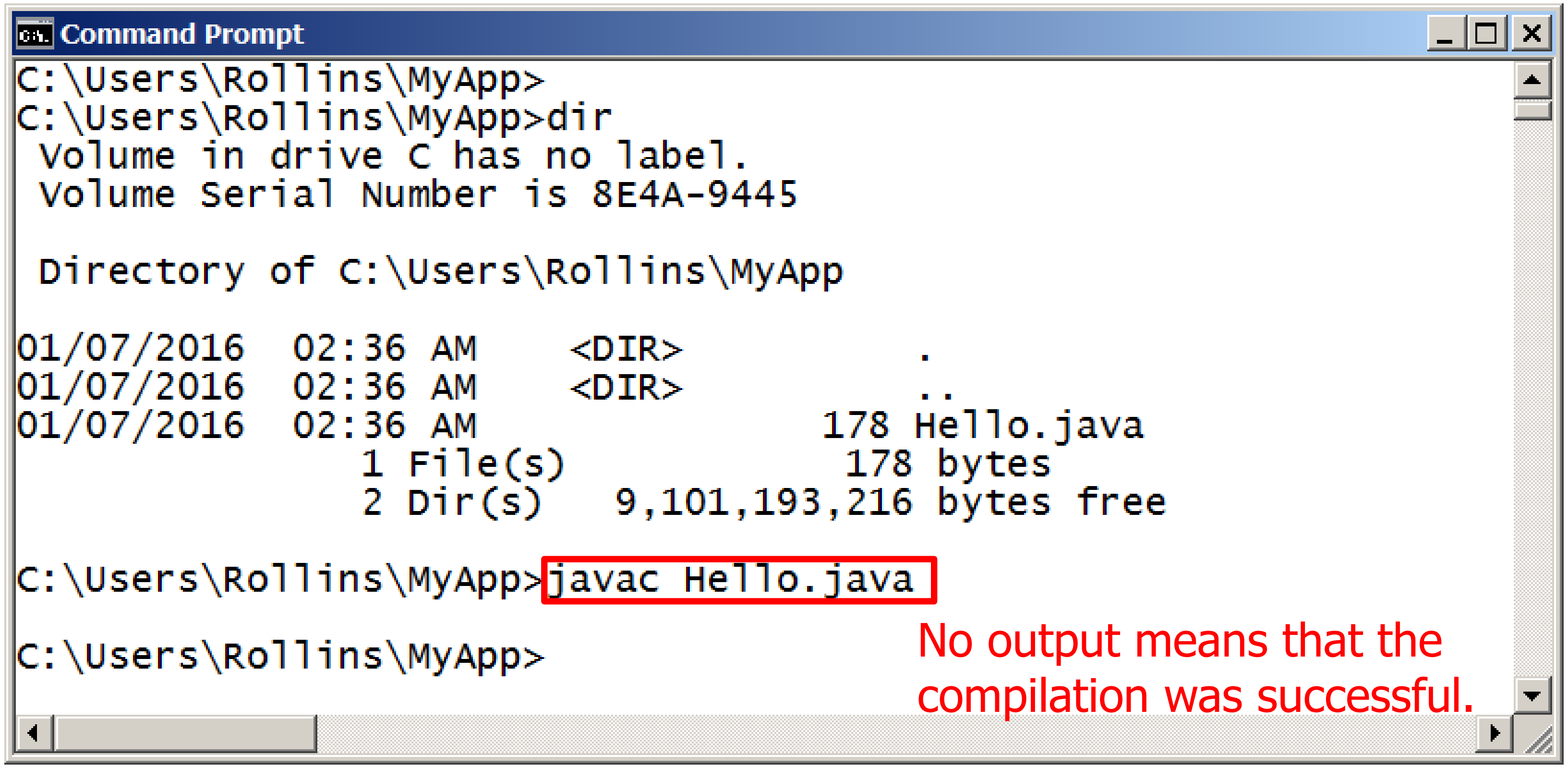
```
Java source file | length : 191    lines : 11 | Ln : 11    Col : 1    Sel : 0 | 0 | Dos\Windows | ANSI | INS
```

Back in Command Window



```
C:\Users\Rollins\MyApp>
C:\Users\Rollins\MyApp>dir
 Volume in drive C has no label.
 Volume Serial Number is 8E4A-9445

 Directory of C:\Users\Rollins\MyApp

01/07/2016  02:36 AM    <DIR>          .
01/07/2016  02:36 AM    <DIR>          ..
01/07/2016  02:36 AM               178 Hello.java
               1 File(s)            178 bytes
               2 Dir(s)   9,101,193,216 bytes free

C:\Users\Rollins\MyApp>_
```

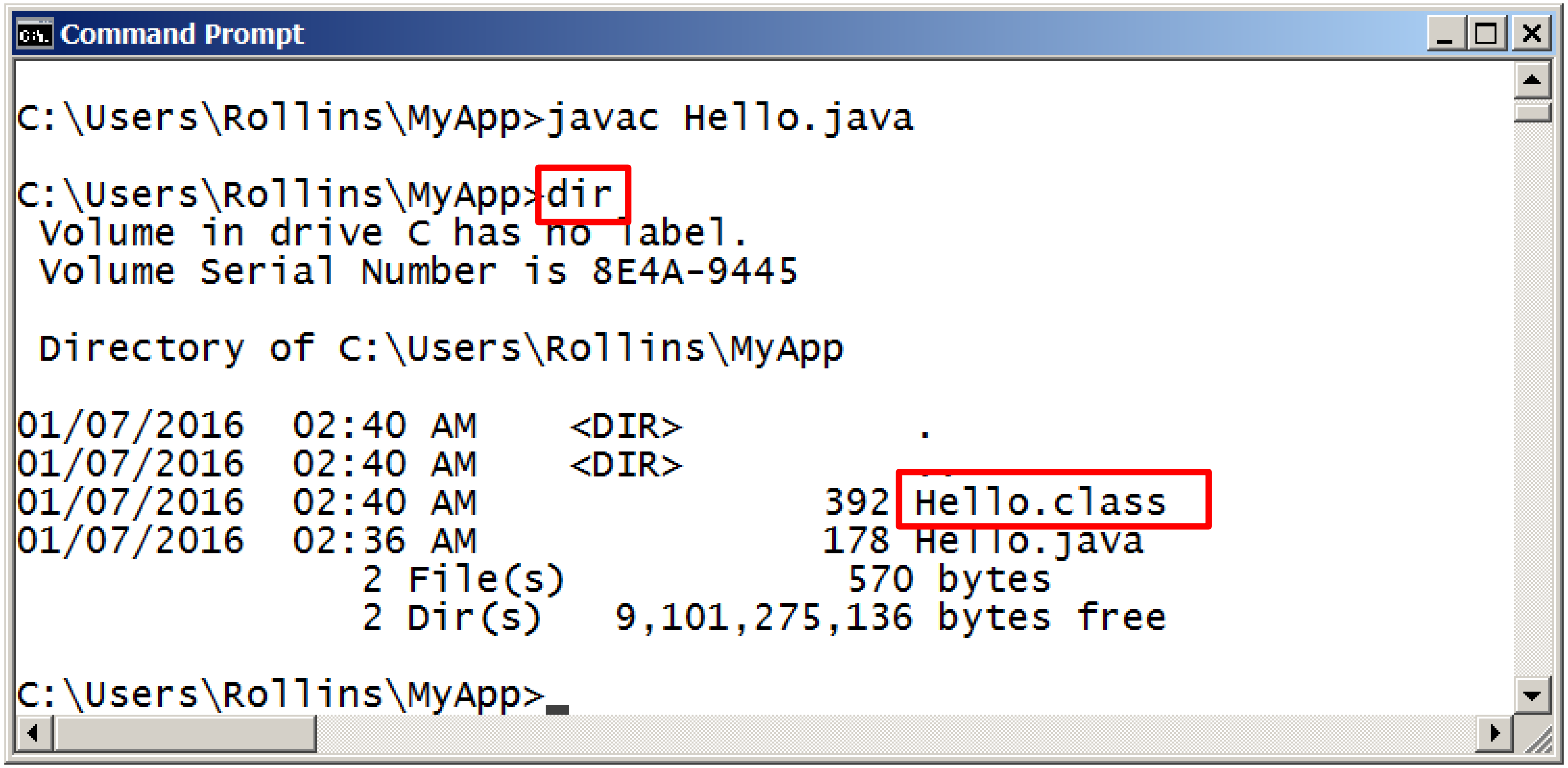
Compile the Program



```
C:\Users\Rollins\MyApp>
C:\Users\Rollins\MyApp>dir
 Volume in drive C has no label.
 Volume Serial Number is 8E4A-9445

 Directory of C:\Users\Rollins\MyApp

01/07/2016  02:36 AM    <DIR>          .
01/07/2016  02:36 AM    <DIR>          ..
01/07/2016  02:36 AM               178 Hello.java
               1 File(s)            178 bytes
               2 Dir(s)   9,101,193,216 bytes free

C:\Users\Rollins\MyApp>javac Hello.java

C:\Users\Rollins\MyApp>
```

No output means that the compilation was successful.

Here is our Java Bytecode File

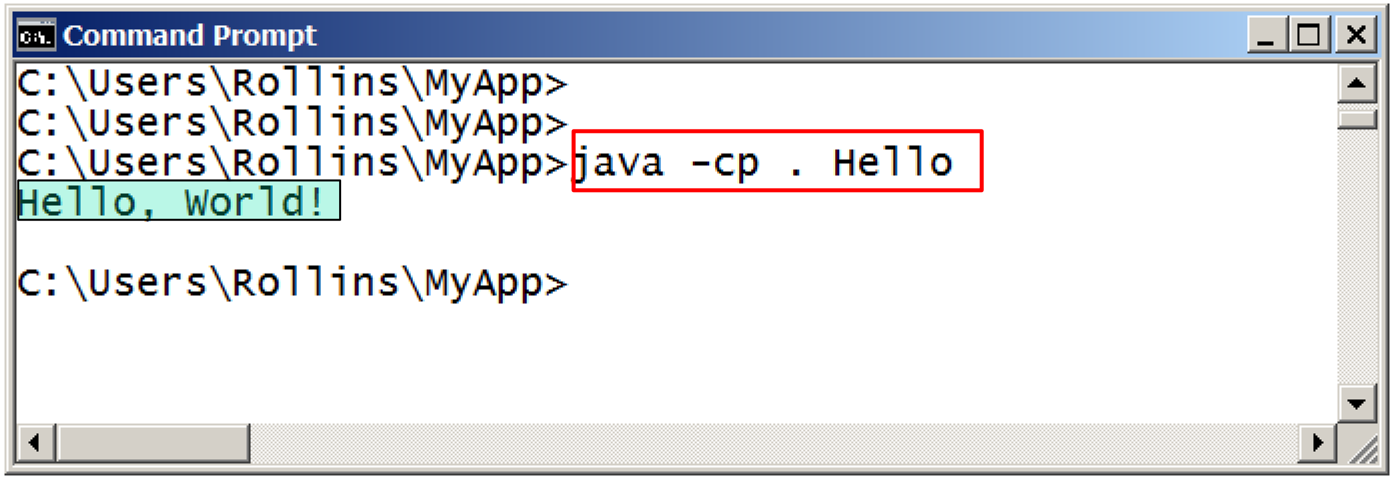


```
C:\Users\Rollins\MyApp>javac Hello.java

C:\Users\Rollins\MyApp>dir
 Volume in drive C has no label.
 Volume Serial Number is 8E4A-9445

 Directory of C:\Users\Rollins\MyApp

01/07/2016  02:40 AM    <DIR>          .
01/07/2016  02:40 AM    <DIR>          ..
01/07/2016  02:40 AM               392 Hello.class
01/07/2016  02:36 AM               178 Hello.java
               2 File(s)            570 bytes
               2 Dir(s)   9,101,275,136 bytes free

C:\Users\Rollins\MyApp>_
```

The complier created file Hello.class

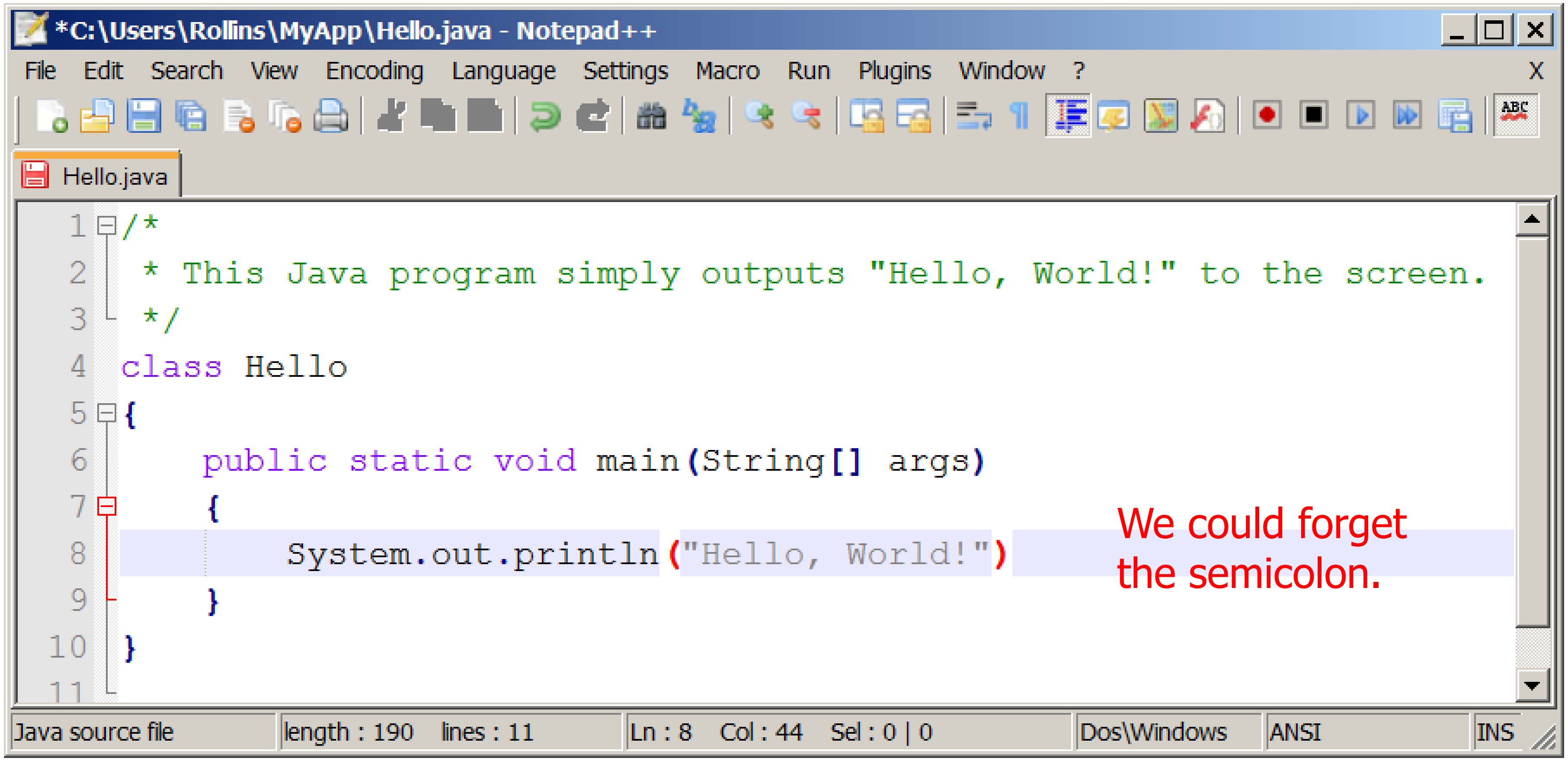
Run the Program

Our program's output



```
C:\Users\Rollins\MyApp>
C:\Users\Rollins\MyApp>
C:\Users\Rollins\MyApp>java -cp . Hello
Hello, World!

C:\Users\Rollins\MyApp>
```

What could possibly go wrong?

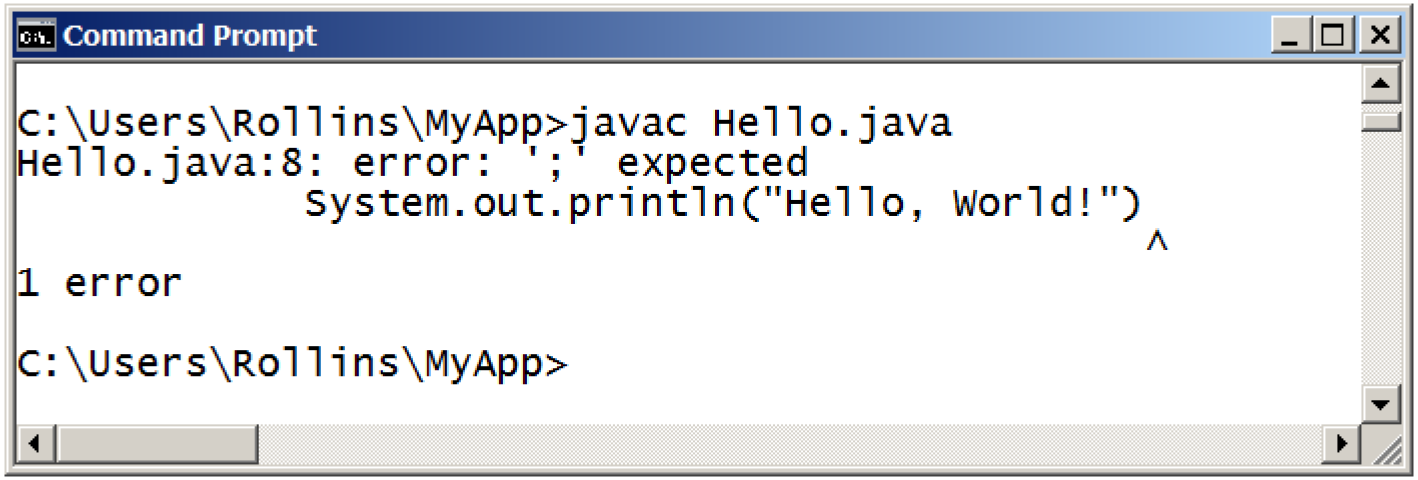

*C:\Users\Rollins\MyApp\Hello.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Hello.java

```
/*
 * This Java program simply outputs "Hello, World!" to the screen.
 */
class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!")
    }
}
```

Java source file | length : 190 lines : 11 | Ln : 8 Col : 44 Sel : 0 | 0 | Dos\Windows | ANSI | INS

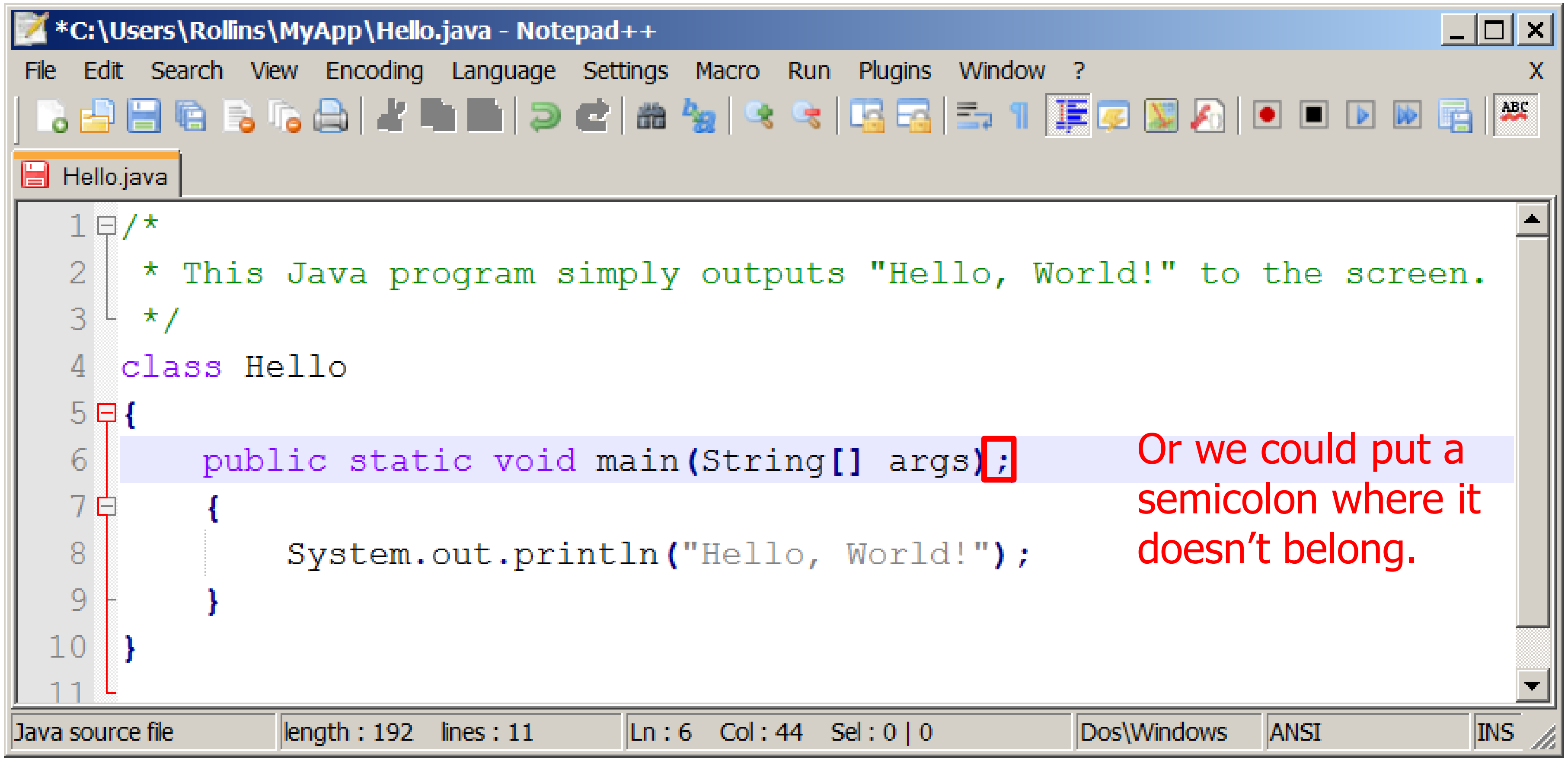
We could forget the semicolon.

A Syntax Error



```
C:\Users\Rollins\MyApp>javac Hello.java
Hello.java:8: error: ';' expected
            System.out.println("Hello, World!")
                                               ^
1 error

C:\Users\Rollins\MyApp>
```

Another Syntax Error

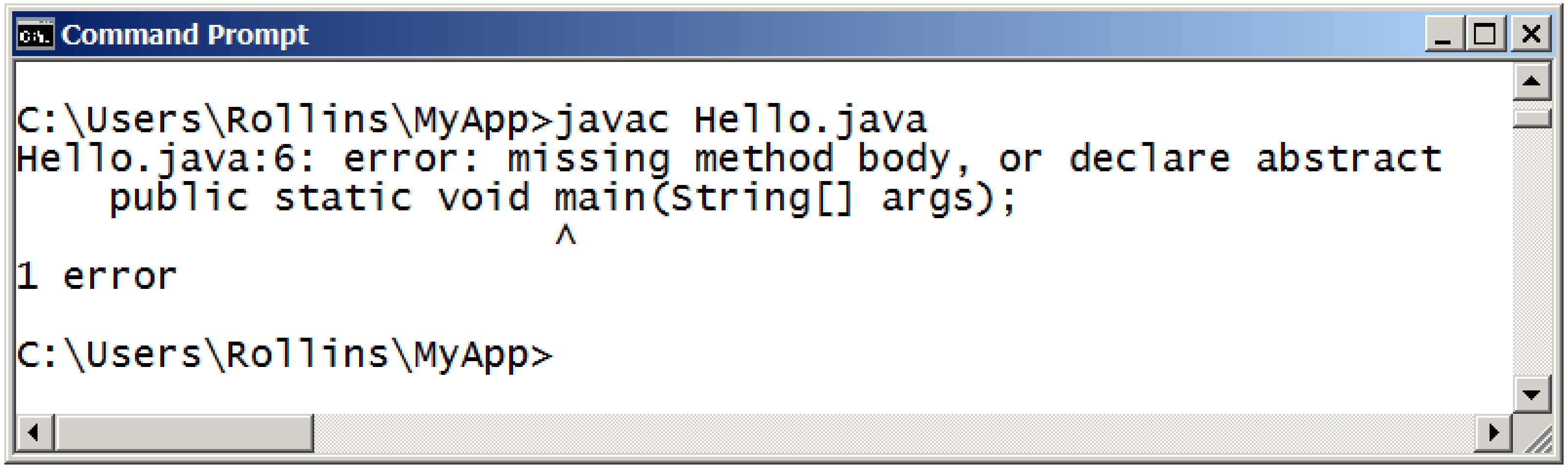

*C:\Users\Rollins\MyApp\Hello.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Hello.java

```
/*
 * This Java program simply outputs "Hello, World!" to the screen.
 */
class Hello
{
    public static void main(String[] args);
    {
        System.out.println("Hello, World!");
    }
}
```

Java source file | length : 192 lines : 11 | Ln : 6 Col : 44 Sel : 0 | 0 | Dos\Windows | ANSI | INS

Or we could put a semicolon where it doesn't belong.

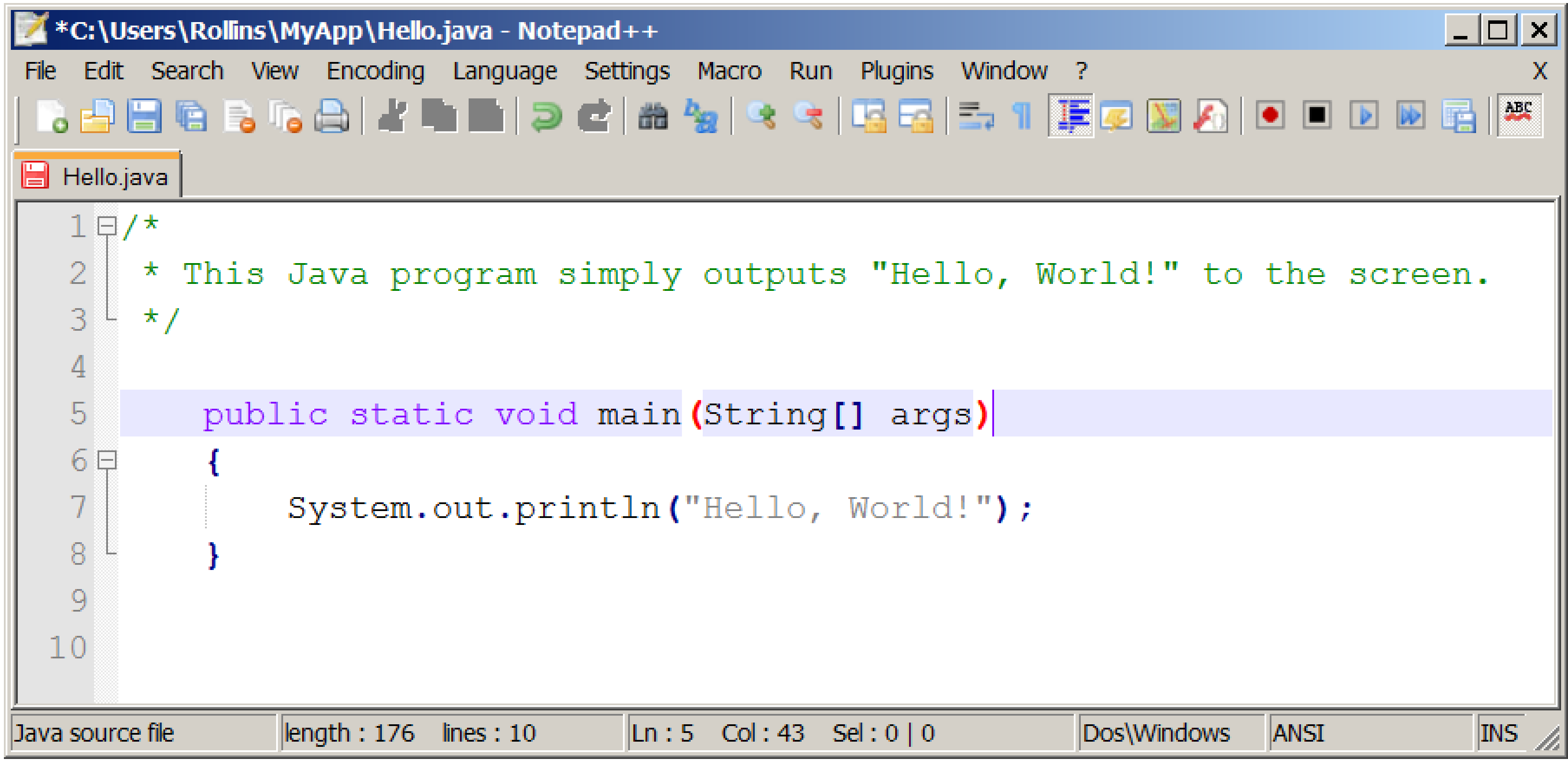
A rather obscure error message



```
C:\Users\Rollins\MyApp>javac Hello.java
Hello.java:6: error: missing method body, or declare abstract
    public static void main(String[] args);
                       ^
1 error

C:\Users\Rollins\MyApp>
```

Look at the line with the error and try to determine what is wrong with it.

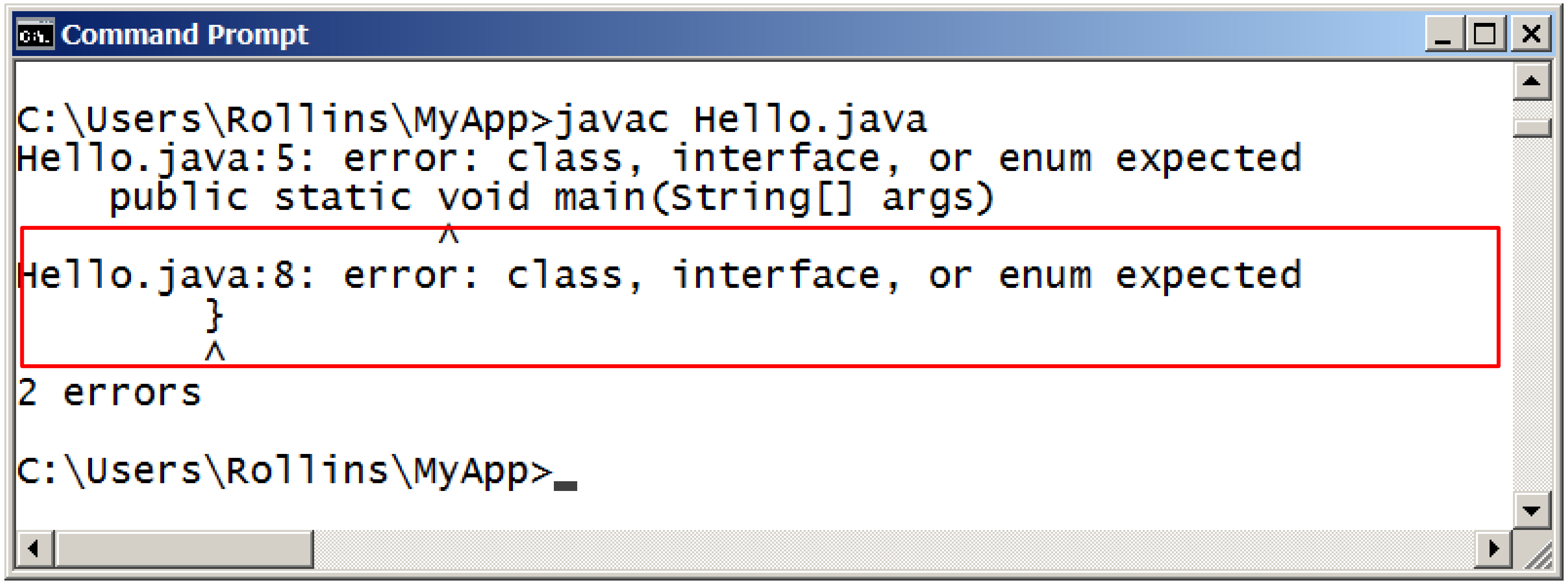
Forgot the "class"


*C:\Users\Rollins\MyApp\Hello.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Hello.java

```
/*
 * This Java program simply outputs "Hello, World!" to the screen.
 */

    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }


```

Java source file | length : 176 lines : 10 | Ln : 5 Col : 43 Sel : 0 | 0 | Dos\Windows | ANSI | INS

Another compilation error

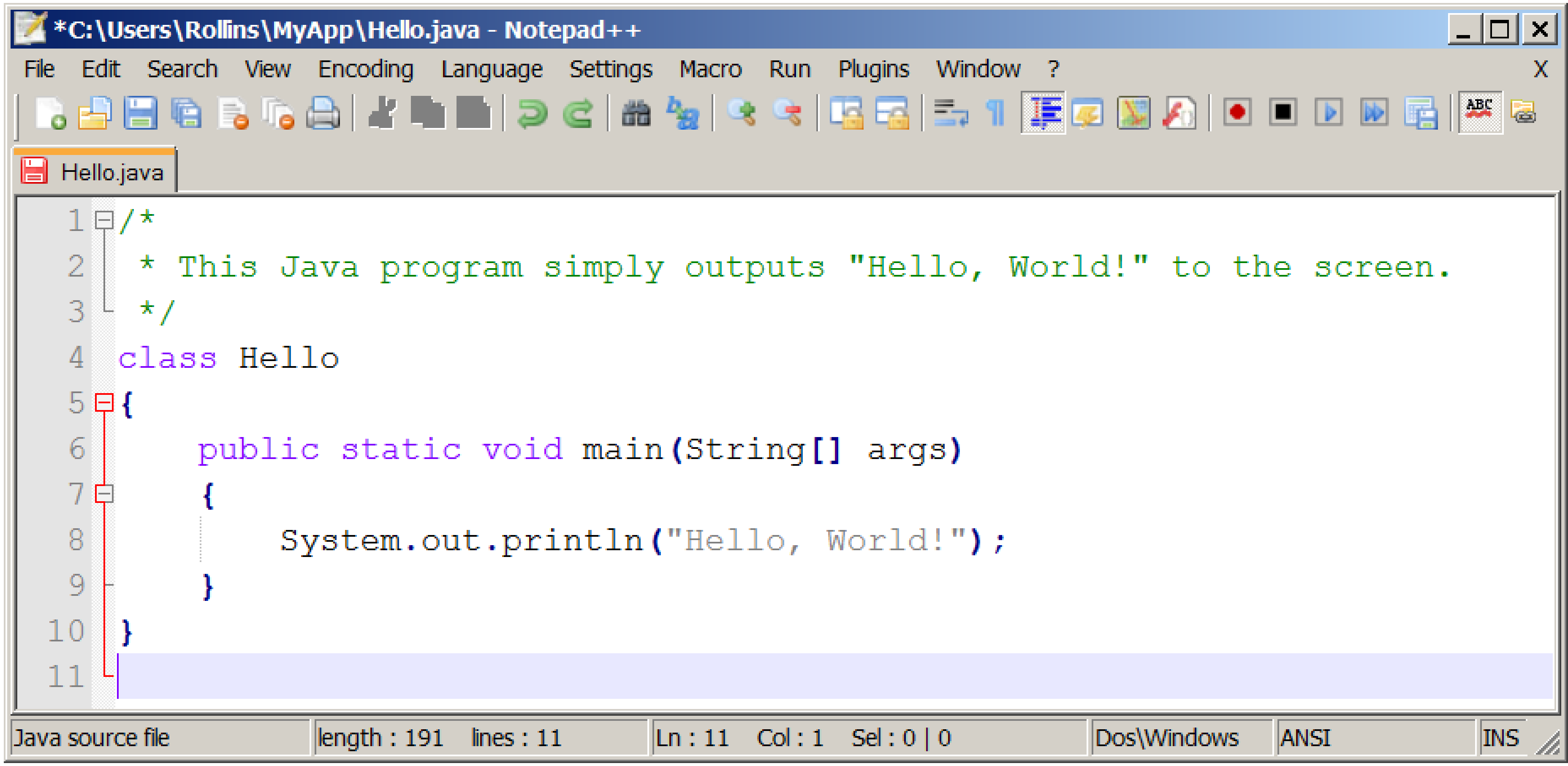


```
C:\Users\Rollins\MyApp>javac Hello.java
Hello.java:5: error: class, interface, or enum expected
    public static void main(String[] args)
                  ^
Hello.java:8: error: class, interface, or enum expected
        }
        ^
2 errors

C:\Users\Rollins\MyApp>_
```

The line 8 error is bogus. (Byproduct of the first error.)

In general, if an error after the first is not obvious ignore it.
Fix the first error and recompile.

Fix the Error

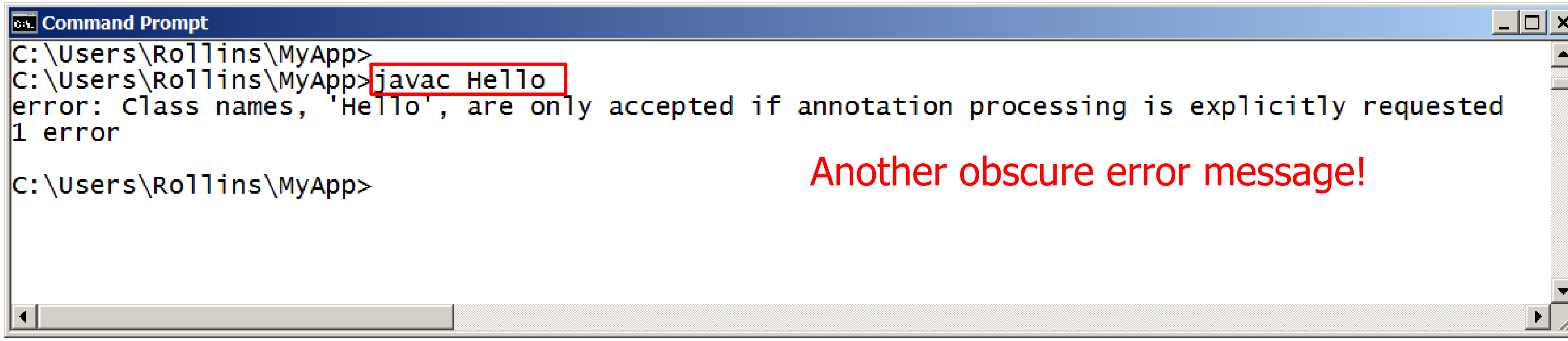


*C:\Users\Rollins\MyApp\Hello.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Hello.java

```
/*
 * This Java program simply outputs "Hello, World!" to the screen.
 */
class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}

```

```
Java source file    length : 191    lines : 11    Ln : 11    Col : 1    Sel : 0 | 0    Dos\Windows    ANSI    INS
```

Try Again



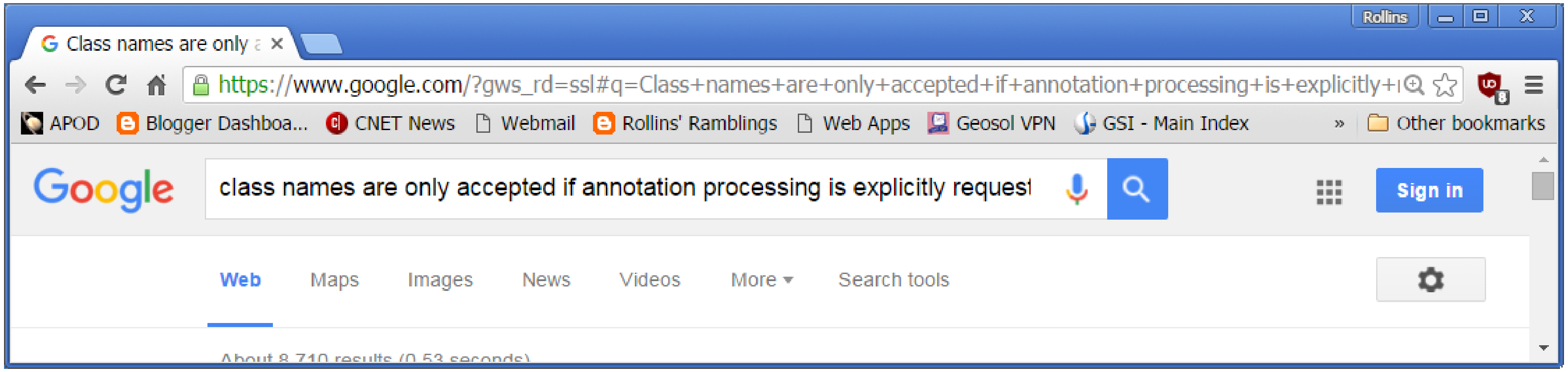
```
C:\Users\Rollins\MyApp>
C:\Users\Rollins\MyApp>javac Hello
error: Class names, 'Hello', are only accepted if annotation processing is explicitly requested
1 error

C:\Users\Rollins\MyApp>
```

Another obscure error message!

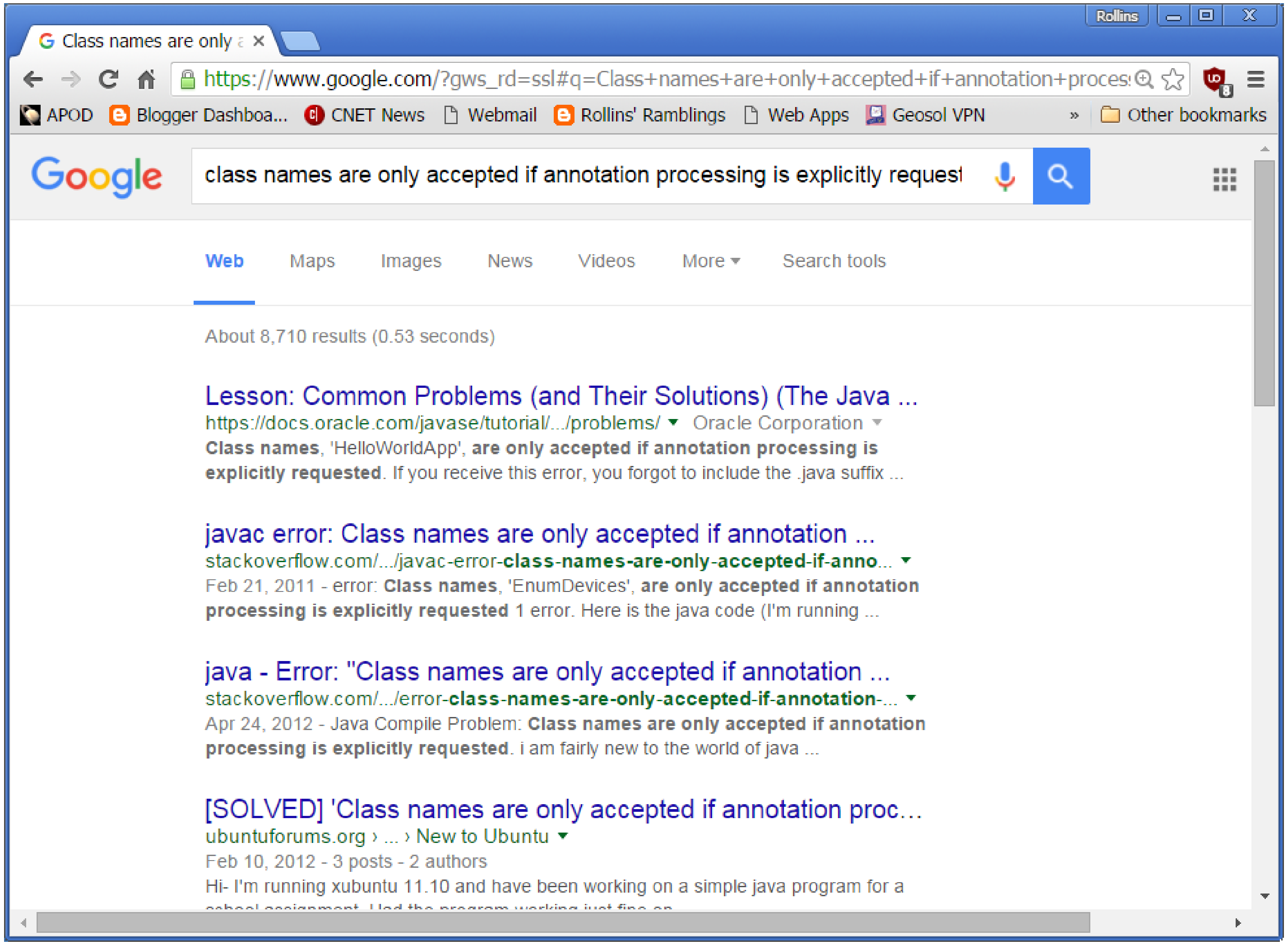
Now what's wrong?

We just corrected the error.

Ask Google!


Class names are only a
https://www.google.com/?gws_rd=ssl#q=Class+names+are+only+accepted+if+annotation+processing+is+explicitly+r
APOD
Blogger Dashboa...
CNET News
Webmail
Rollins' Ramblings
Web Apps
Geosol VPN
GSI - Main Index
Other bookmarks
class names are only accepted if annotation processing is explicitly request
Sign in
Web
Maps
Images
News
Videos
More
Search tools
About 8,710 results (0.53 seconds)

Google knows everything!



Class names are only a

https://www.google.com/?gws_rd=ssl#q=Class+names+are+only+accepted+if+annotation+proces

APOD | Blogger Dashboa... | CNET News | Webmail | Rollins' Ramblings | Web Apps | Geosol VPN | Other bookmarks

class names are only accepted if annotation processing is explicitly request

Web | Maps | Images | News | Videos | More | Search tools

About 8,710 results (0.53 seconds)

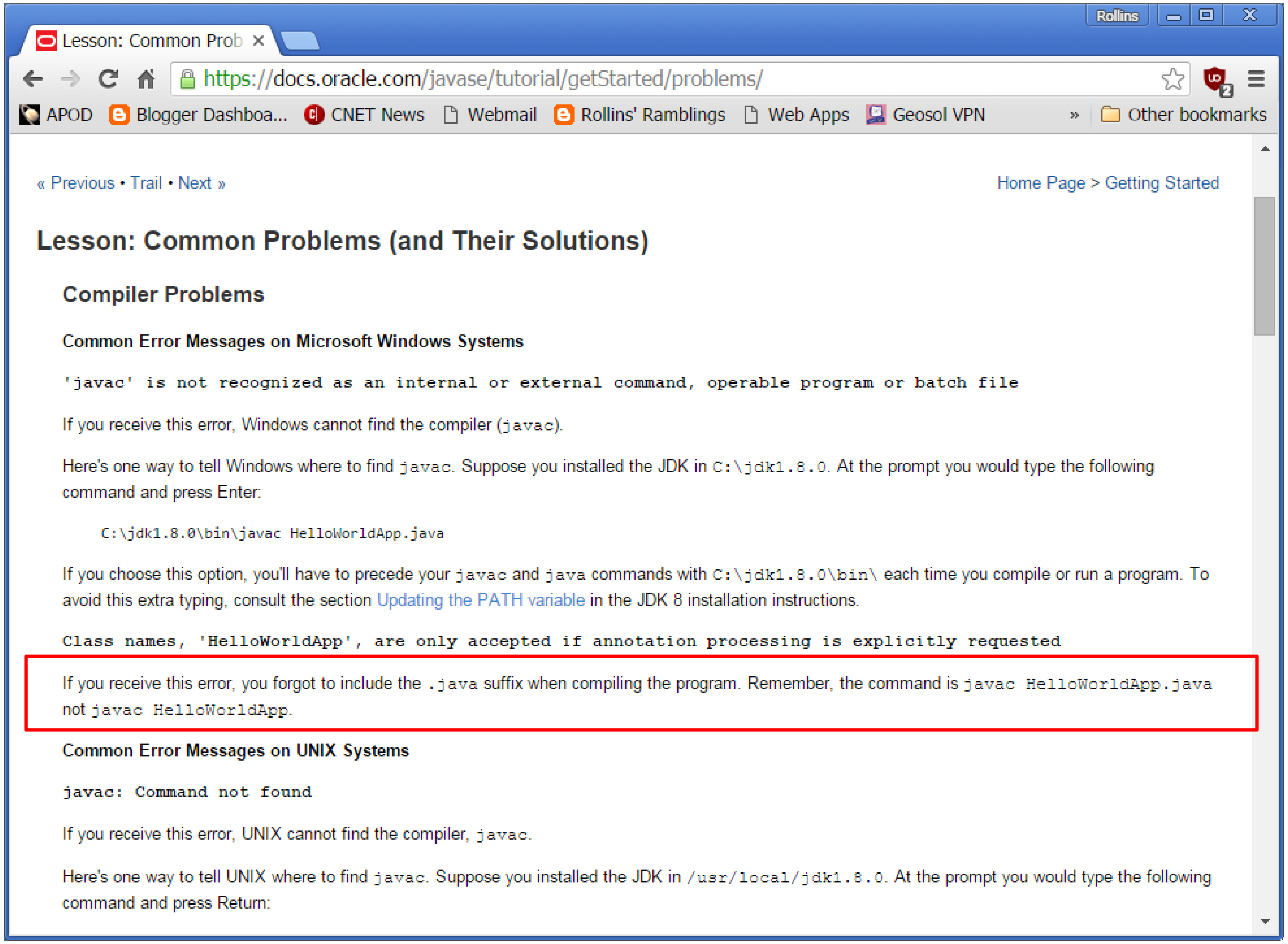
Lesson: Common Problems (and Their Solutions) (The Java ...
https://docs.oracle.com/javase/tutorial/.../problems/ Oracle Corporation
Class names, 'HelloWorldApp', **are only accepted if annotation processing is explicitly requested**. If you receive this error, you forgot to include the .java suffix ...

javac error: Class names are only accepted if annotation ...
stackoverflow.com/.../javac-error-**class-names-are-only-accepted-if-anno**...
Feb 21, 2011 - error: **Class names**, 'EnumDevices', **are only accepted if annotation processing is explicitly requested** 1 error. Here is the java code (I'm running ...

java - Error: "Class names are only accepted if annotation ...
stackoverflow.com/.../error-**class-names-are-only-accepted-if-annotation**-...
Apr 24, 2012 - Java Compile Problem: **Class names are only accepted if annotation processing is explicitly requested**. i am fairly new to the world of java ...

[SOLVED] 'Class names are only accepted if annotation proc...
ubuntuforums.org › ... › New to Ubuntu
Feb 10, 2012 - 3 posts - 2 authors
Hi- I'm running xubuntu 11.10 and have been working on a simple java program for a

Someone has seen this error previously



Lesson: Common Prob ×

https://docs.oracle.com/javase/tutorial/getStarted/problems/

APOD | Blogger Dashboa... | CNET News | Webmail | Rollins' Ramblings | Web Apps | Geosol VPN | » | Other bookmarks

« Previous • Trail • Next »

Home Page > Getting Started

Lesson: Common Problems (and Their Solutions)

Compiler Problems

Common Error Messages on Microsoft Windows Systems

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

If you receive this error, Windows cannot find the compiler (`javac`).

Here's one way to tell Windows where to find `javac`. Suppose you installed the JDK in `C:\jdk1.8.0`. At the prompt you would type the following command and press Enter:

```
C:\jdk1.8.0\bin\javac HelloWorldApp.java
```

If you choose this option, you'll have to precede your `javac` and `java` commands with `C:\jdk1.8.0\bin\` each time you compile or run a program. To avoid this extra typing, consult the section Updating the PATH variable in the JDK 8 installation instructions.

```
Class names, 'HelloWorldApp', are only accepted if annotation processing is explicitly requested
```

If you receive this error, you forgot to include the `.java` suffix when compiling the program. Remember, the command is `javac HelloWorldApp.java` not `javac HelloWorldApp`.

Common Error Messages on UNIX Systems

```
javac: Command not found
```

If you receive this error, UNIX cannot find the compiler, `javac`.

Here's one way to tell UNIX where to find `javac`. Suppose you installed the JDK in `/usr/local/jdk1.8.0`. At the prompt you would type the following command and press Return:

Readings and Assignments

- Reading: Chapter 1
- Lab Assignment: Java Lab 1
  - Assignment on class web site
  - Due in one week.
    - Best bet: do it in your scheduled lab session.
- Self-Assessment Exercises (not submitted)
  - Self-Review Questions Section 1.4
    - SR1.21, SR1.23, SR1.24, SR1.25
  - After Chapter Exercises
    - EX1.3, EX1.16, EX1.17, EX1.20

Lab Submission

- Submit your project in Canvas.

- Make sure you attach all the files for the lab before you click “submit”.
- Be sure to click “submit” after attaching your files
- Labs are due by midnight one week from the day preceding the class in which the assignment was given.

Grading Guidelines for Java Labs and Lab Exams

- 80% - Functionality: Runs correctly, generating correct outputs. A program that does not compile will result in a zero.
- 10% - Style: Use consistent indentation to emphasize block structure; variables have meaningful names.
- 10% - Comments: Comment major code segments adequately

Getting Help

Work on the project in your lab session or weekly help session.

A TA will be present to help with any problems.

After the lab session

- Ask a TA during office hours
- Ask your Instructor
- Email your TA or Instructor

About Next Week

- Monday is a USF holiday.
  - Sections 001 and 002 will have no lecture.

- In order to keep the sections together, I will post next week's lecture on the class web site as PowerPoint slides.
  - View the slides at your convenience.
  - No lecture on Tuesday for sections 003 and 004.
    - This will be a help session in SOC 150.
    - All sections are free to attend.
- Lab sessions will take place as scheduled.

First Day Attendance