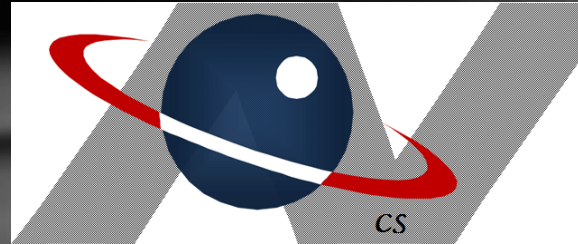


O(N) CS LESSONS

Lesson 1A - First Java Program

"HELLO WORLD"

With DEBUGGING examples



By John B. Owen

All rights reserved

©2011, revised 2015

Table of Contents



- [Objectives](#)
- ["Hello World" Lesson Sequence](#)
- [Compile Errors – Lexical](#)
- [Compile Errors – Syntax](#)
- [Four parts of an error message](#)
- [Lesson Summary / Labs](#)
- [Contact Information for supplementary materials](#)

Objective #1

- In this lesson the student will successfully enter, compile, and execute the source code for the "HELLO WORLD" program.
- *CS1 TEKS 126.33c2(A) create and properly display meaningful output;*



Objective #2

- You will also recognize and successfully DEBUG typical errors that occur during the programming process, using this first program example.
- *CSI TEKS 126.33c4(H) identify and debug errors;*

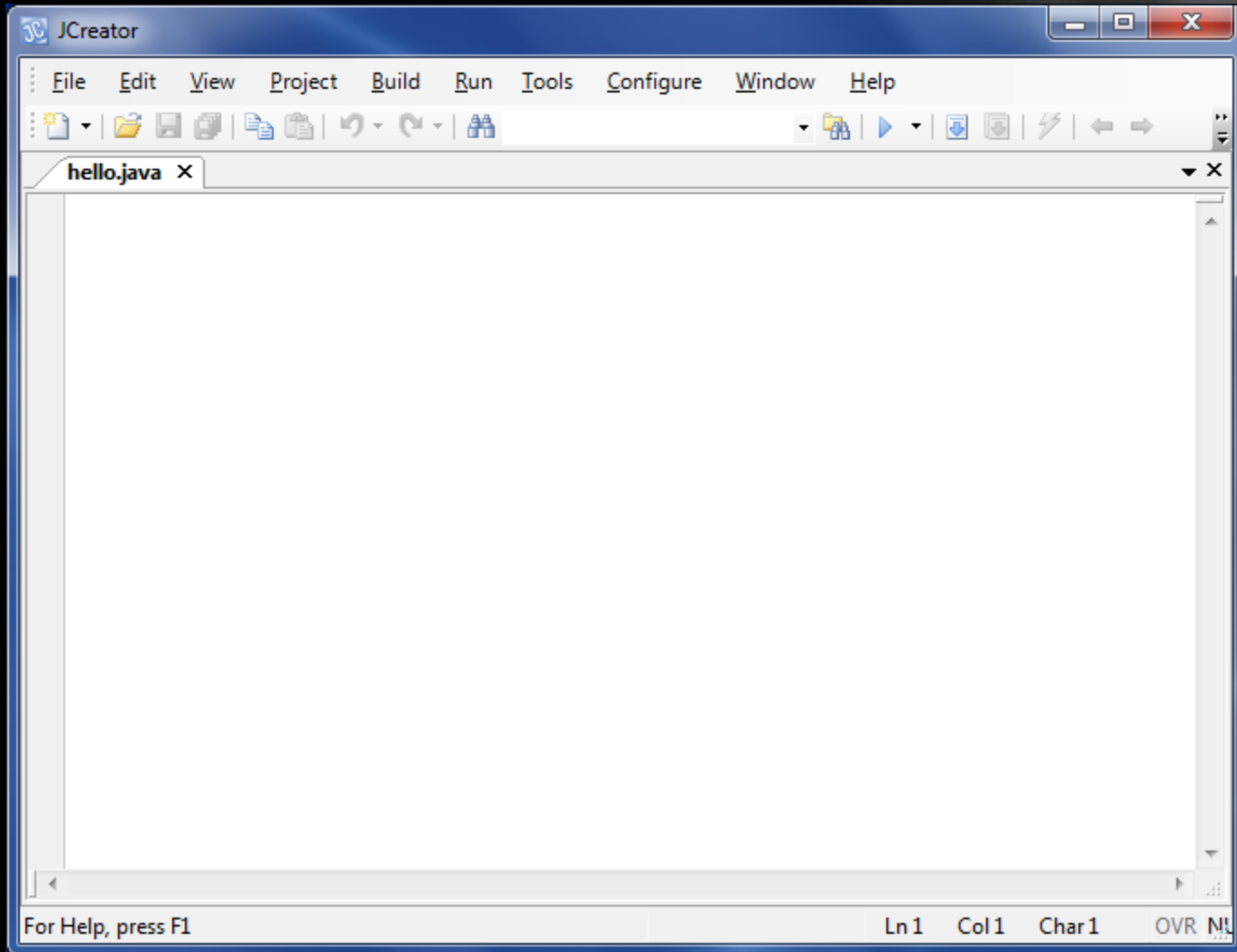


“Hello World” Lesson Sequence

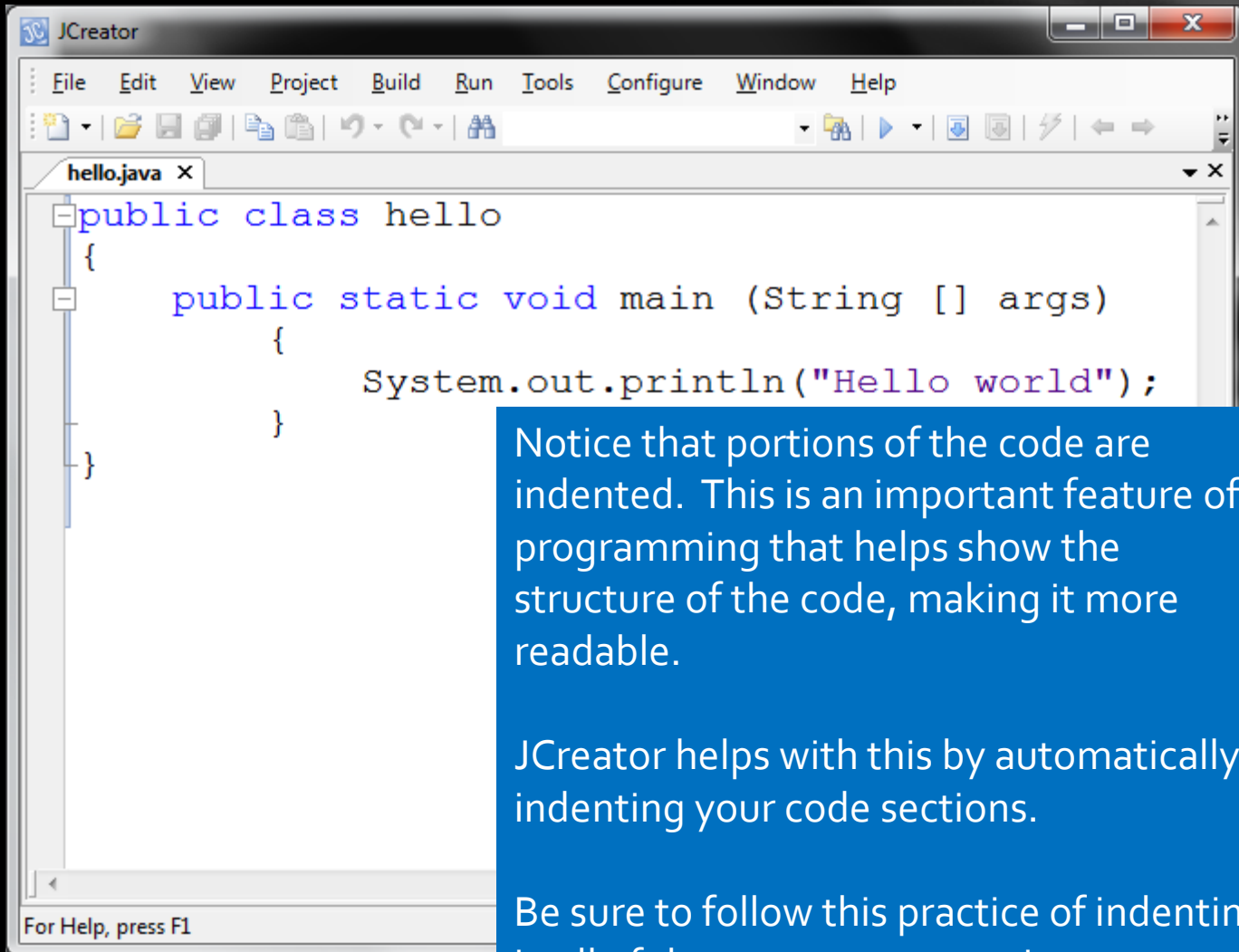
- The next six slides show this programming process done flawlessly – no errors
- After that, examples are shown of typical errors and how to fix them
- Fixing errors is called **DEBUGGING**, and is a normal and crucial part of the programming process.



Step 1 – Create and save a new file called "hello.java"



Step 2 – Type in this code and save it



```
public class hello
{
    public static void main (String [] args)
    {
        System.out.println("Hello world");
    }
}
```

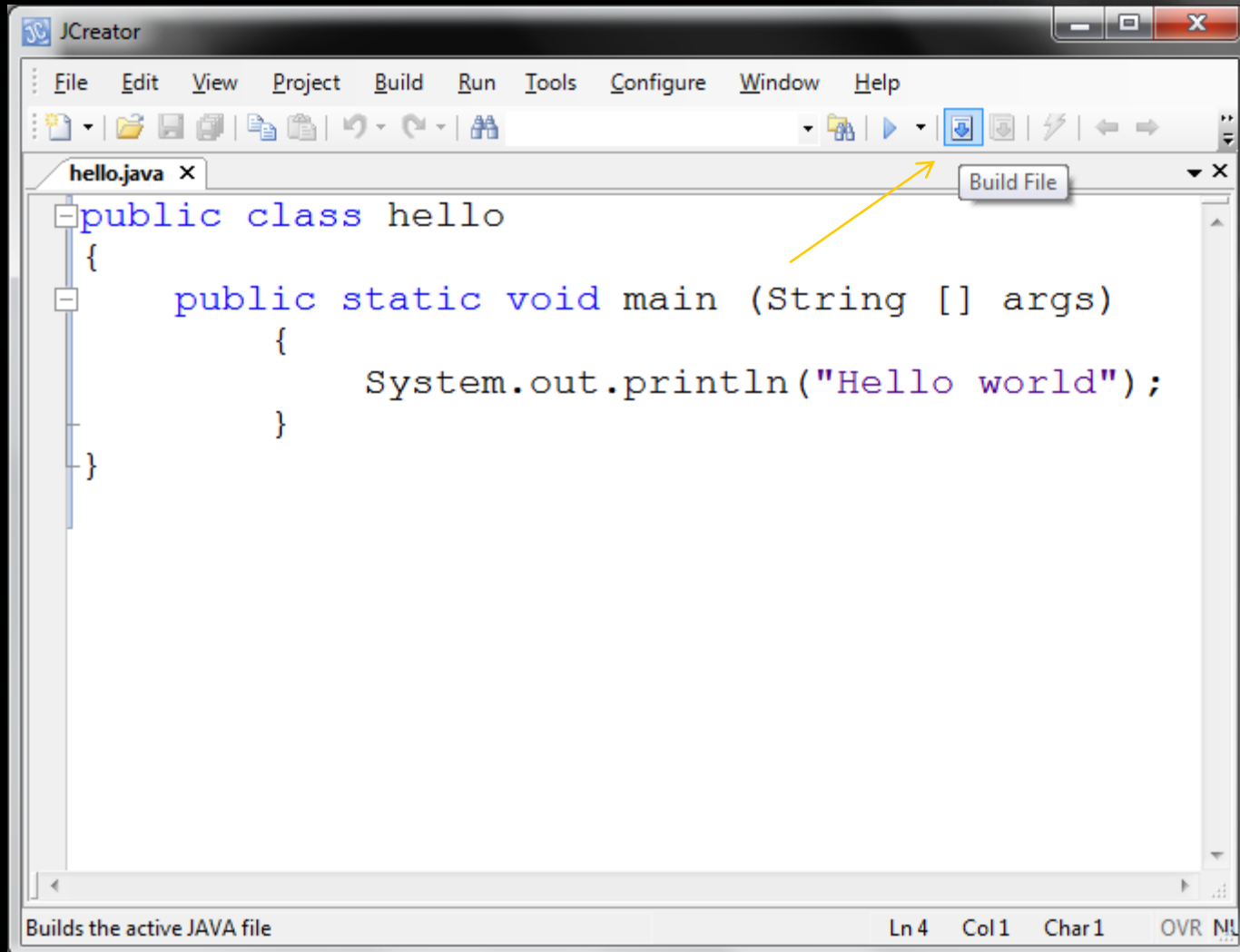
Notice that portions of the code are indented. This is an important feature of programming that helps show the structure of the code, making it more readable.

JCreator helps with this by automatically indenting your code sections.

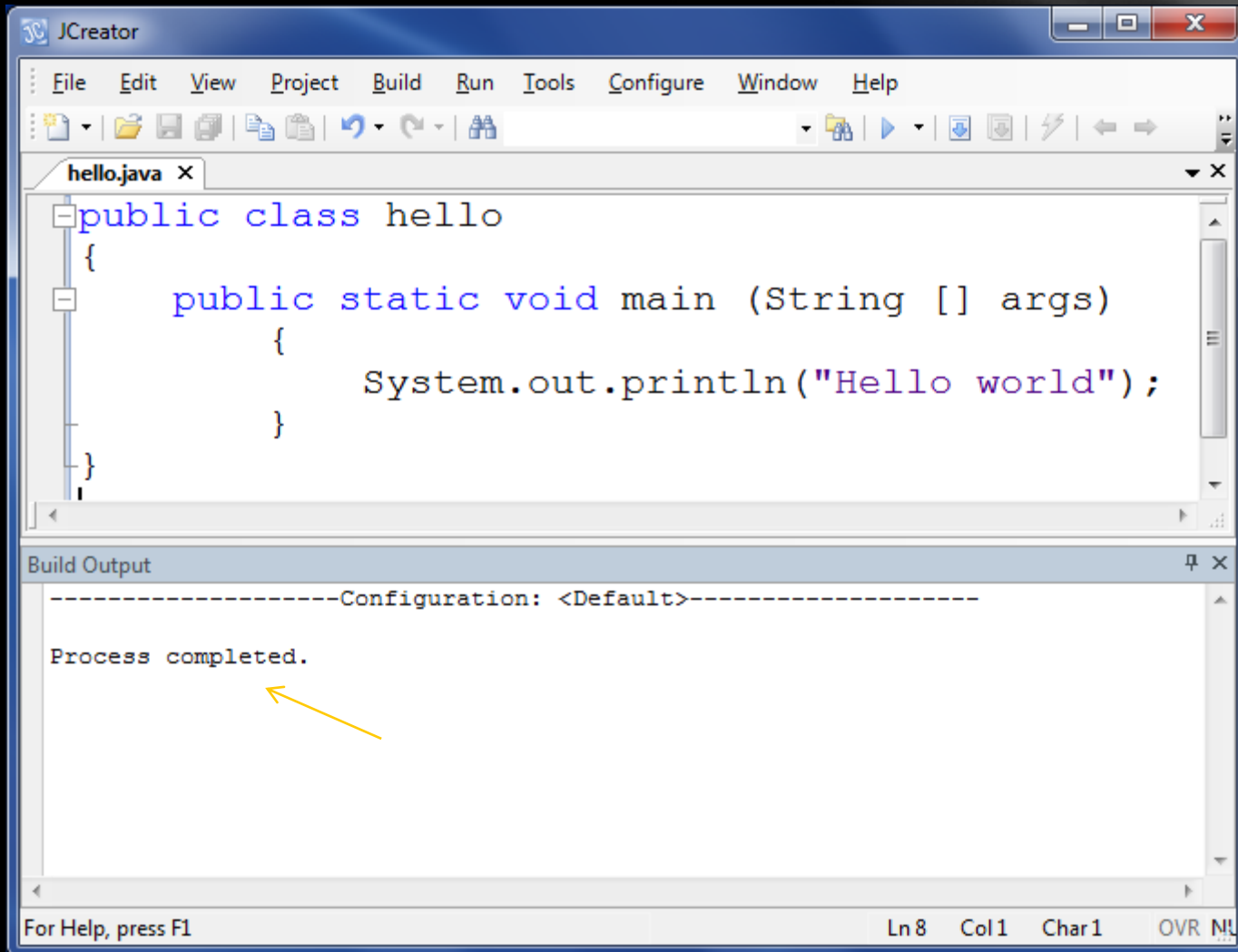
Be sure to follow this practice of indenting in all of the programs you write.



Step 3 – Compile the code by clicking “Build File”



Step 3 result – Clean compile



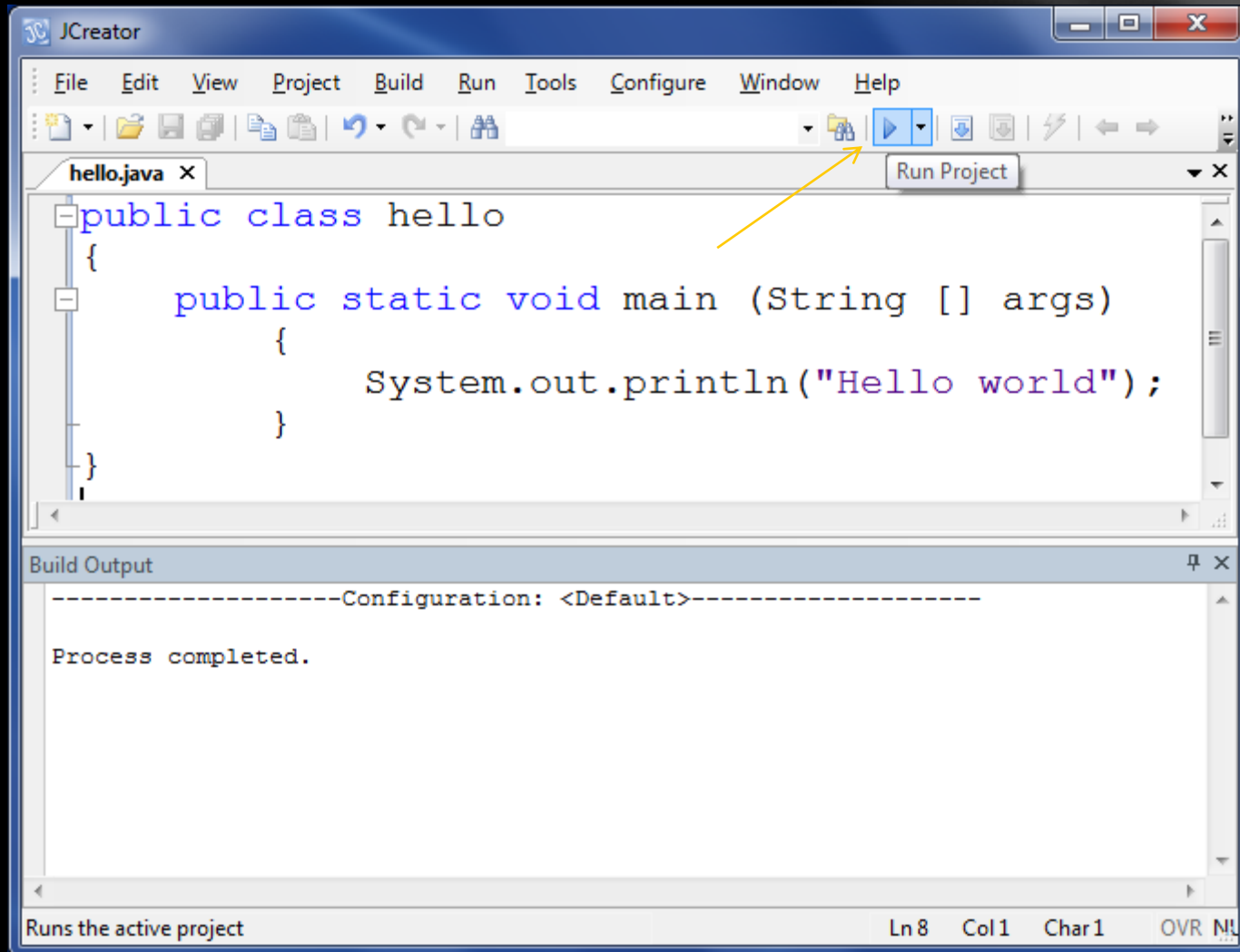
The screenshot shows the JCreator IDE with a Java file named 'hello.java' open. The code in the editor is:

```
public class hello
{
    public static void main (String [] args)
    {
        System.out.println("Hello world");
    }
}
```

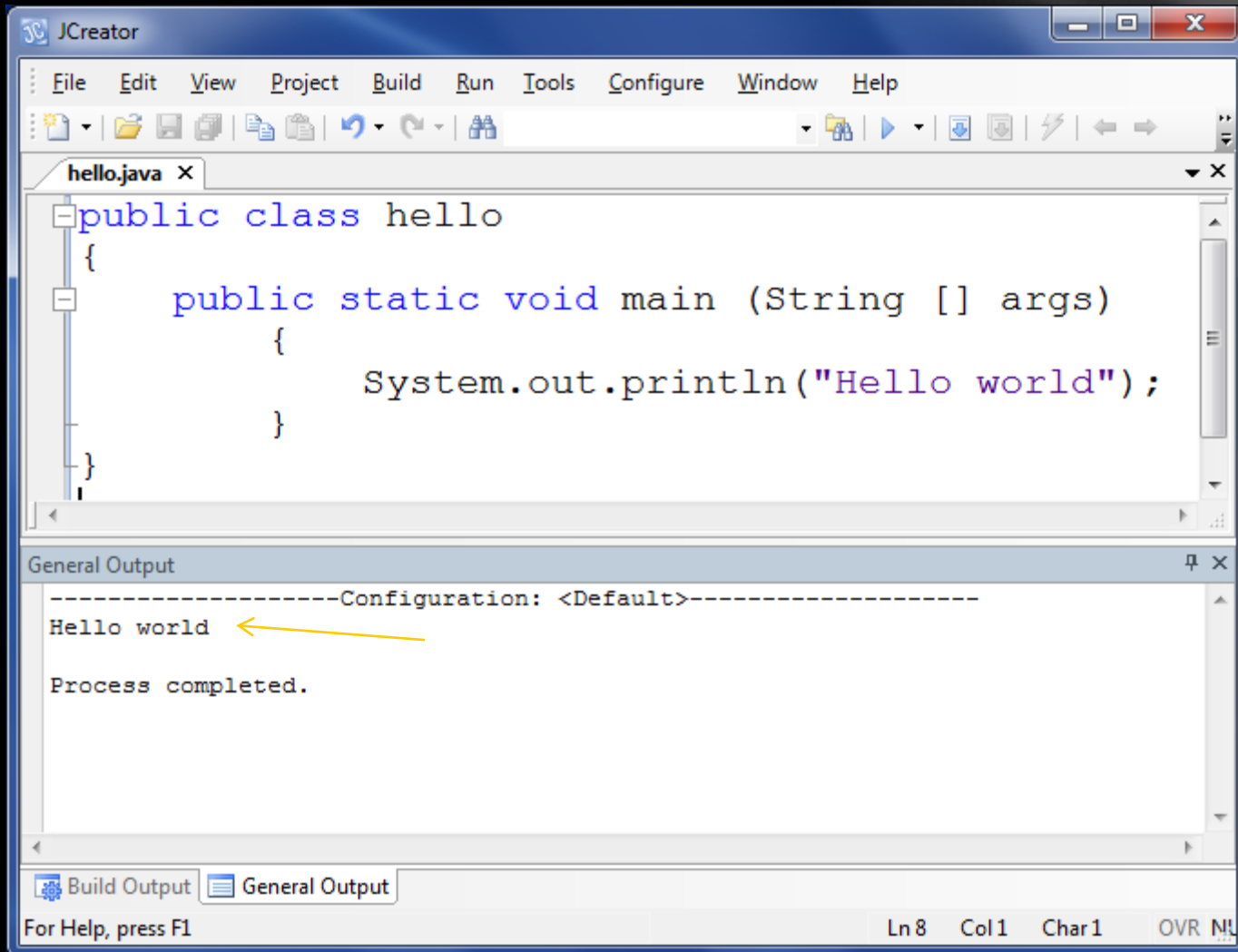
Below the code editor is the 'Build Output' window, which displays the message 'Process completed.' with a yellow arrow pointing to it. The status bar at the bottom indicates 'Ln 8 Col 1 Char 1 OVR NL'.



Step 4 – Execute the code by clicking “Run Project”



Step 4 result – Correct output



The screenshot shows the JCreator IDE with a Java file named 'hello.java'. The code in the editor is:

```
public class hello
{
    public static void main (String [] args)
    {
        System.out.println("Hello world");
    }
}
```

Below the code editor is the 'General Output' window, which displays the following text:

```
-----Configuration: <Default>-----
Hello world
Process completed.
```

A yellow arrow points to the 'Hello world' output line. The status bar at the bottom indicates 'Ln 8 Col 1 Char 1 OVR NL'.



SUCCESS!!!

- Good job!
- You just completed your first JAVA program!
- Now it's time to explore different types of errors that can occur, and how to fix them.



Compile time errors

- The first step in the execution process of any program is the compile, or build.
- This process translates the source code into *bytecode*, a kind of “universal language” of all JAVA programs.
- Two types of errors can occur during the compile process:
 - Lexical
 - Syntax

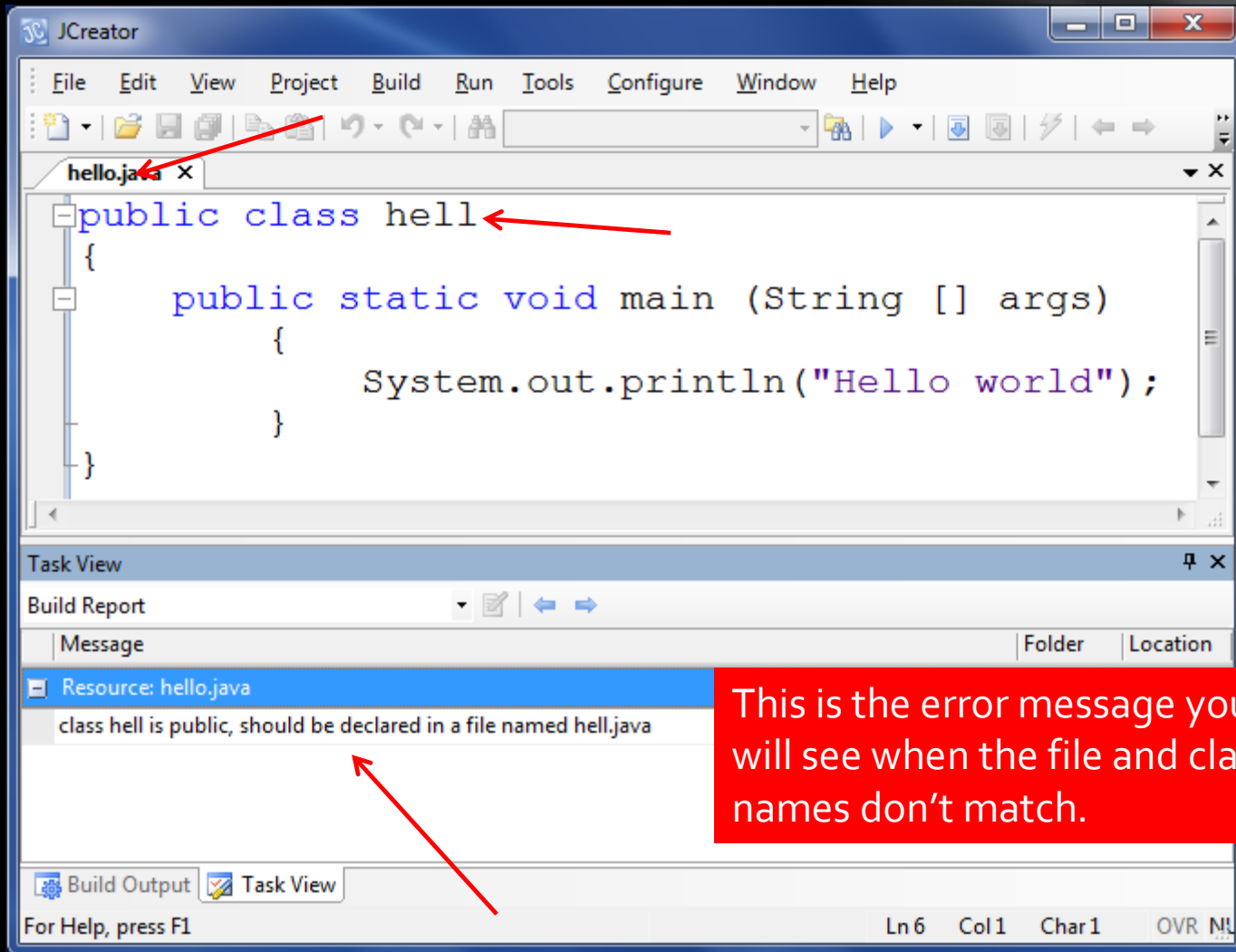


Lexical Errors

- An error in a program is most often a result of some important command mistyped, put in the wrong order, or completely omitted.
- This is called a LEXICAL error.
- To fix this type of error, simply retype the word correctly, or put the words in the correct order, then recompile.
- Here are some examples...



Lexical error example #1: Class name misspelled – **must match file name!**



The screenshot shows the JCreator IDE with a Java file named 'hello.java' open. The code in the editor is:

```
public class hell  
{  
    public static void main (String [] args)  
    {  
        System.out.println("Hello world");  
    }  
}
```

Two red arrows point to the class name 'hell' in the code and the error message in the Task View. The Task View shows the following error:

Message	Folder	Location
class hell is public, should be declared in a file named hell.java		

A red box highlights the error message with the text: "This is the error message you will see when the file and class names don't match."



Lexical error example #2: static and void reversed; **must be in correct order!**

```
1 public class hello
2 {
3     public void static main (String [] args)
4         {
5         System.out.println("Hello world");
6     }
7 }
```

Task View

Build Report

Message	Folder	Location
<identifier> expected		
'(' expected		
invalid method declaration; return type required		

Build Output Task View

For Help, press F1

Ln 7 Col 2 Char 2 OVR NL

This error generates three error messages, NONE of which tell you how to fix it!



Lexical error example #2: static and void reversed; **must be in correct order!**

```
1 public class hello
2 {
3     public void static main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
```

Task View

Build Report

Message	Folder	Location
Resource: hello.java		
<identifier> expected		
'(' expected		
invalid method declaration; return type required		

Build Output Task View

For Help, press F1

Quite often you have to look carefully and try to figure out what caused the error, especially when the error messages aren't very helpful!



LexErr Example #3: The words "String" and "System" must be capitalized!

```
1 public class hello
2 {
3     public static void main (string [] args)
4     {
5         system.out.println("Hello world");
6     }
7 }
```

Task View

Build Report

Message	Folder	Location
cannot find symbol class string		
package system does not exist		

Build Output Task View

For Help, press F1

Ln 11 Col 1 Char 1 OVR NL

"String" and "System" must be spelled with capital letters because they are special words in JAVA.



LexErr Example #3: The words "String" and "System" **must be capitalized!**

```
1 public class hello
2 {
3     public static void main (string [] args)
4     {
5         system.out.println("Hello world");
6     }
7 }
```

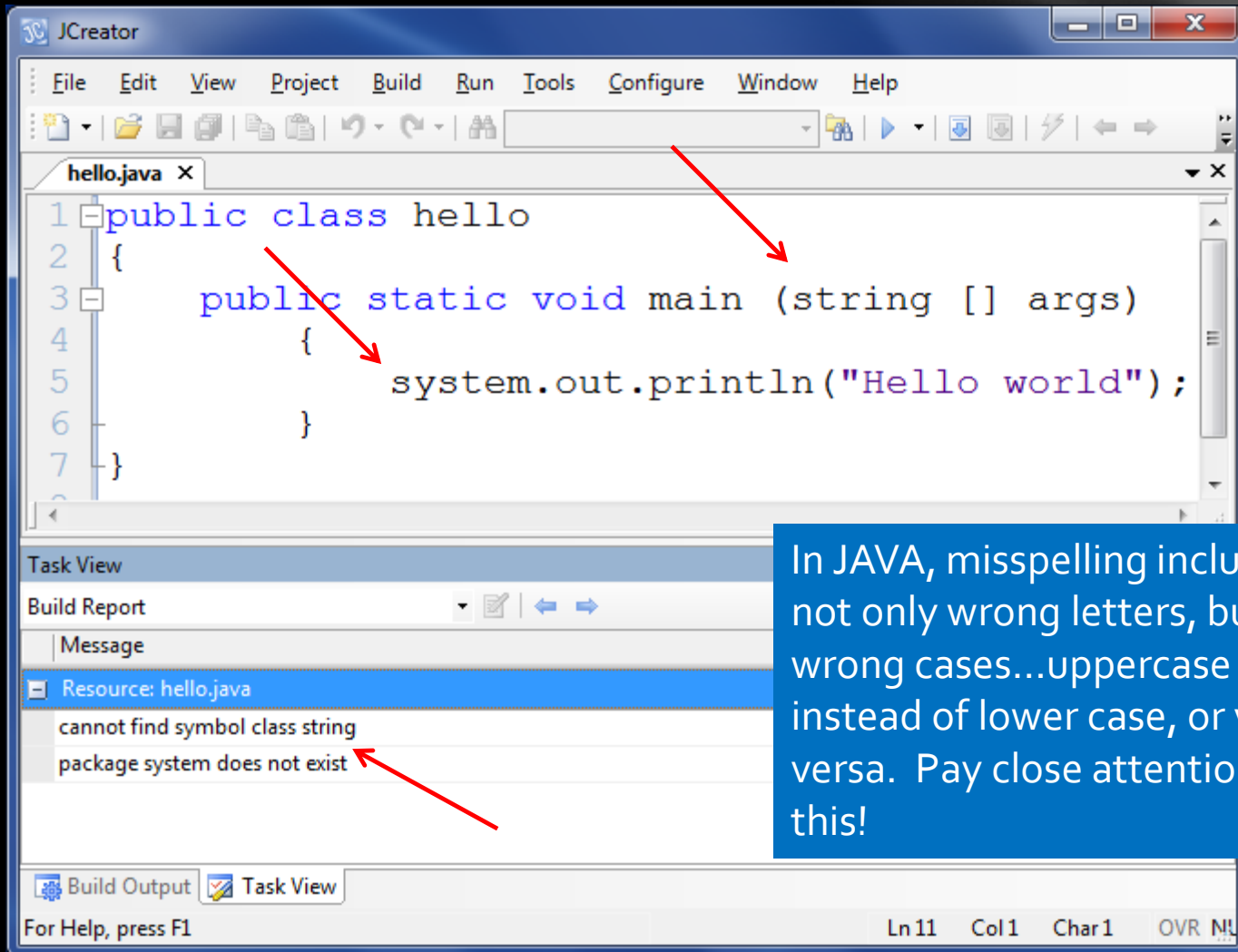
Task View

Message	Folder	Location
cannot find symbol class string		
package system does not exist		

Note that the error messages are different, even though the error is the same. This happens a lot.



LexErr Example #3: The words "String" and "System" **must be capitalized!**



```
1 public class hello
2 {
3     public static void main (string [] args)
4     {
5         system.out.println("Hello world");
6     }
7 }
```

Task View

Build Report

Message

Resource: hello.java

- cannot find symbol class string
- package system does not exist

Build Output Task View

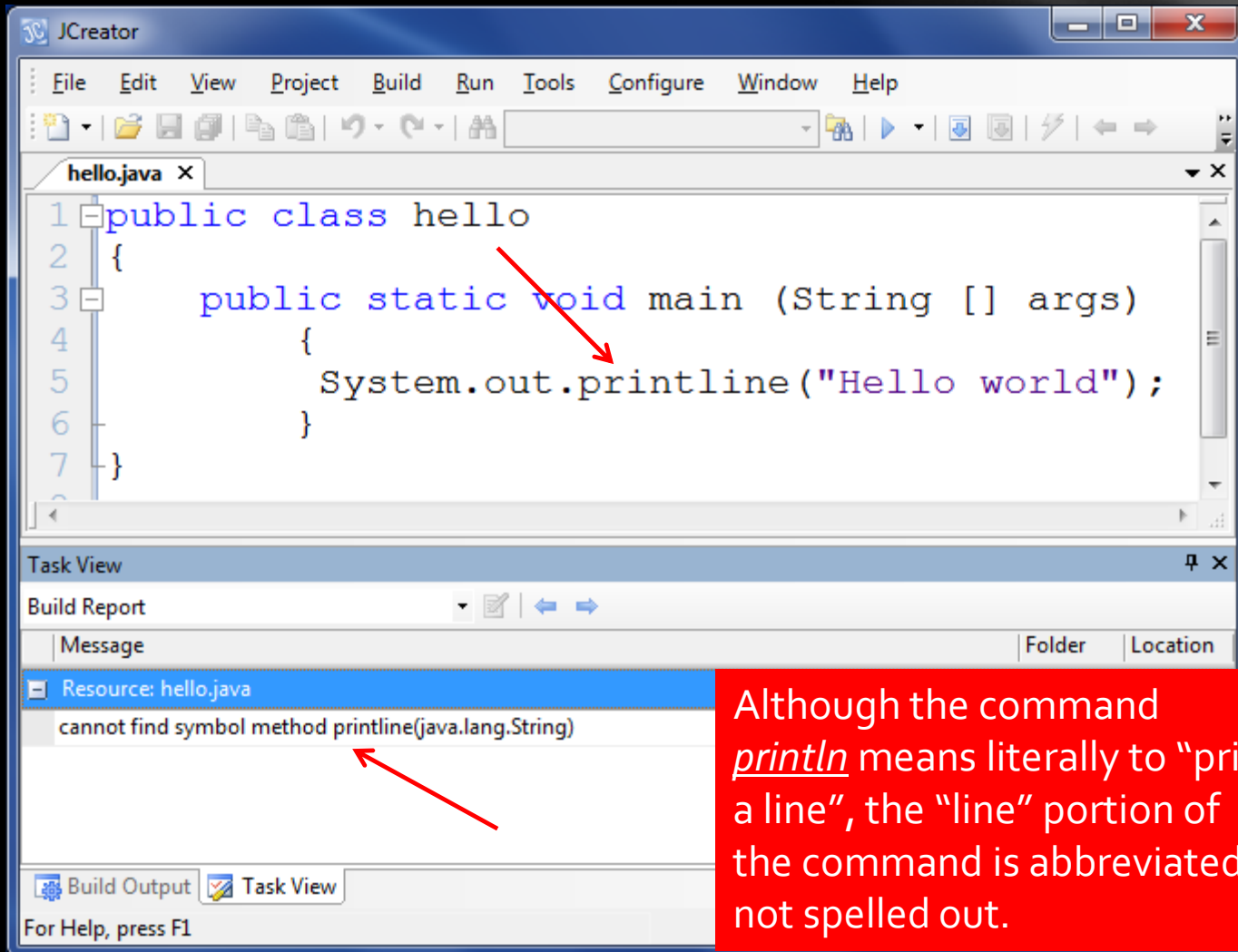
For Help, press F1

Ln 11 Col 1 Char 1 OVR NL

In JAVA, misspelling includes not only wrong letters, but wrong cases...uppercase instead of lower case, or vice versa. Pay close attention to this!



LexErr Example #4: The command *println* is misspelled!



The screenshot shows the JCreator IDE with a Java file named `hello.java`. The code is as follows:

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
```

A red arrow points from the word `println` in line 5 to the error message in the Task View:

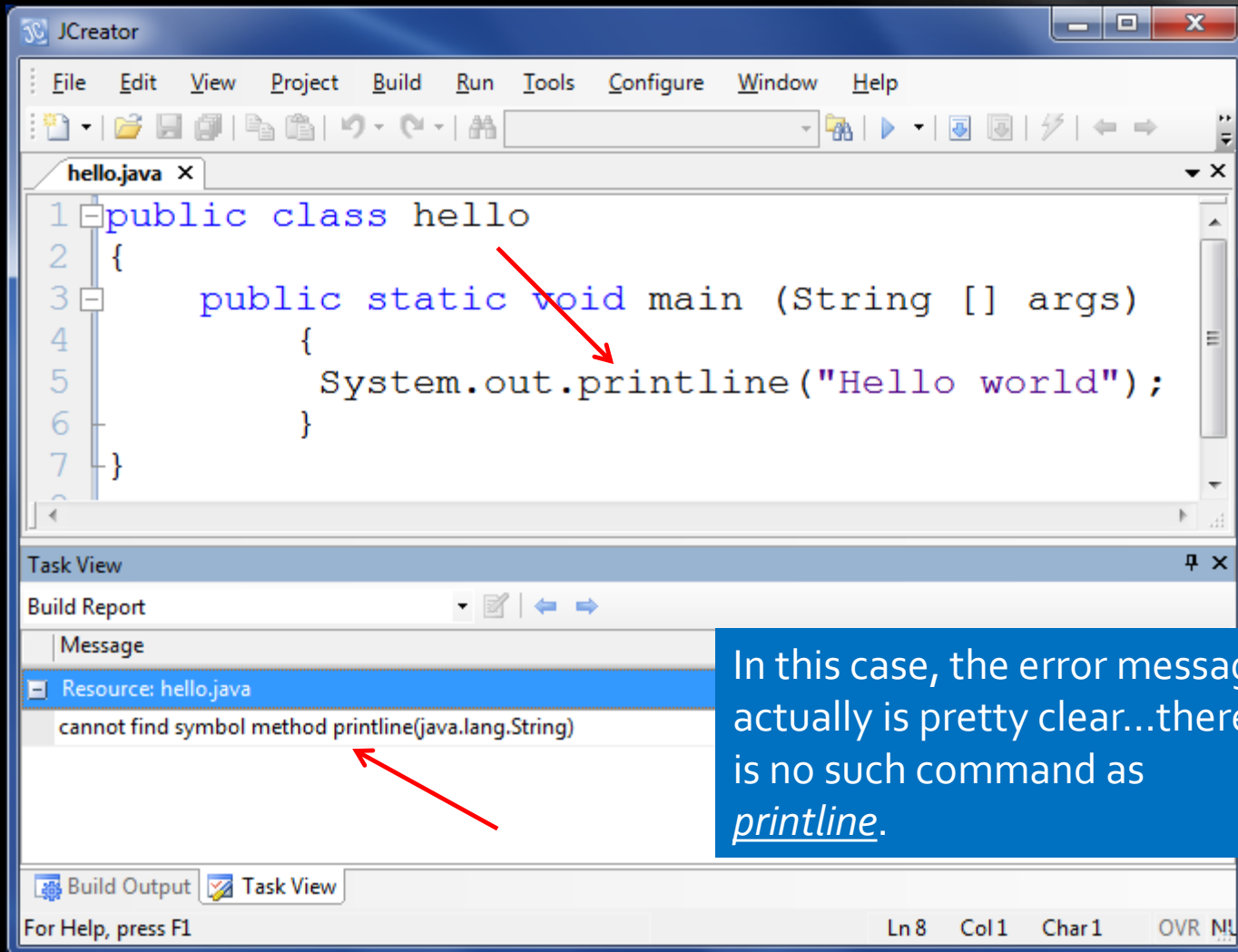
```
Resource: hello.java
cannot find symbol method println(java.lang.String)
```

Another red arrow points from the error message back to the `println` command in the code.

Although the command *println* means literally to "print a line", the "line" portion of the command is abbreviated, not spelled out.



LexErr Example #4: The command *println* is misspelled!



The screenshot shows the JCreator IDE with a Java file named `hello.java`. The code is as follows:

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
```

A red arrow points from the word `println` in the code to the error message in the Task View pane:

```
cannot find symbol method println(java.lang.String)
```

Another red arrow points from the error message back to the `println` in the code. A blue text box on the right contains the following text:

In this case, the error message actually is pretty clear...there is no such command as *println*.

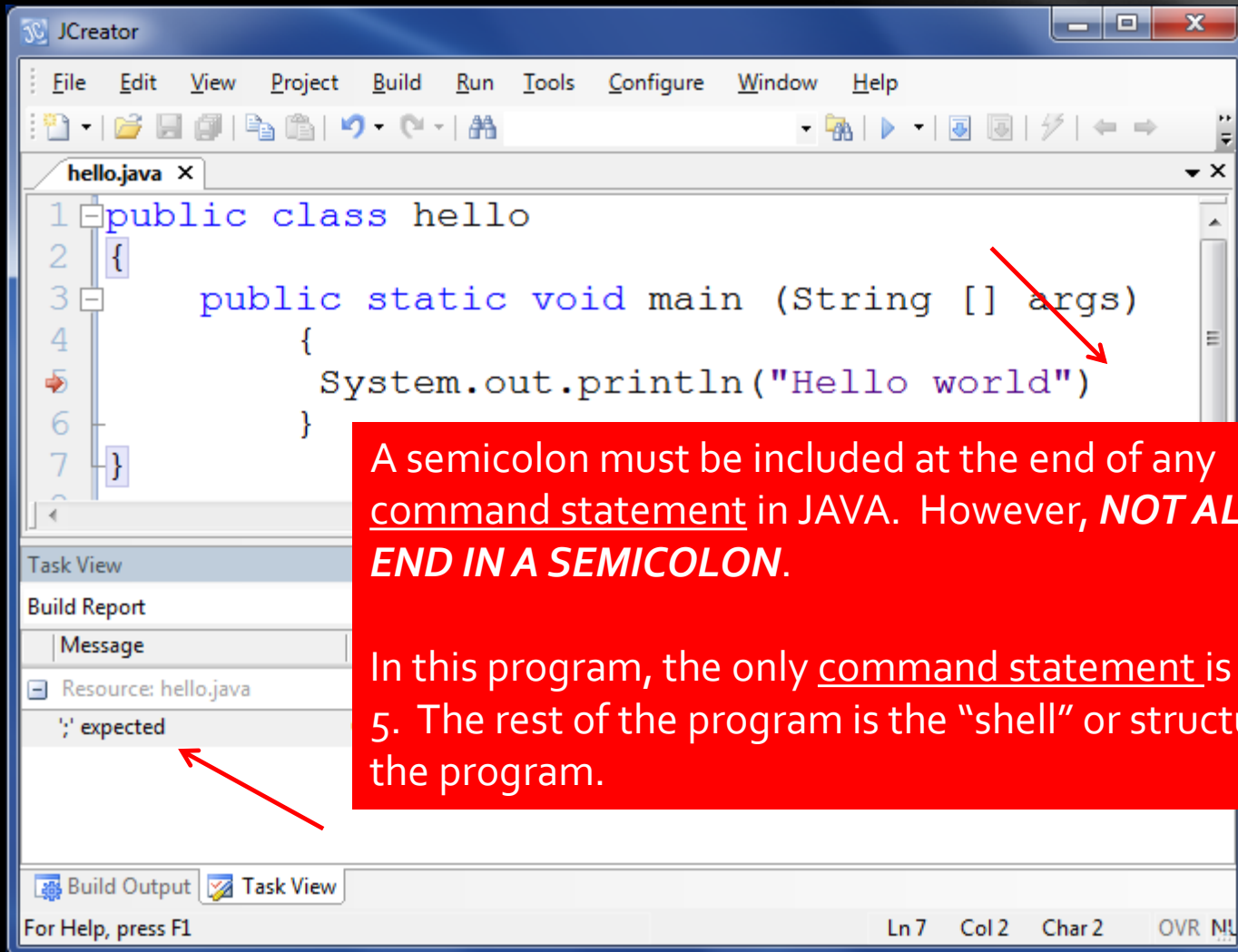


Syntax Errors

- Another type of error in a program deals with punctuation mistakes.
- This is called a SYNTAX error.
- **JAVA relies on punctuation to know when commands are finished, sections of code are complete, and many other situations.**
- Here are some examples...



Syntax Error Example #1: A semicolon is missing!



```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world")
6     }
7 }
```

A semicolon must be included at the end of any command statement in JAVA. However, ***NOT ALL LINES END IN A SEMICOLON.***

In this program, the only command statement is on line 5. The rest of the program is the "shell" or structure of the program.

Task View
Build Report
Message
Resource: hello.java
': expected

Build Output Task View
For Help, press F1
Ln 7 Col 2 Char 2 OVR NL



Syntax Error Example #2: A closing brace **is missing!**

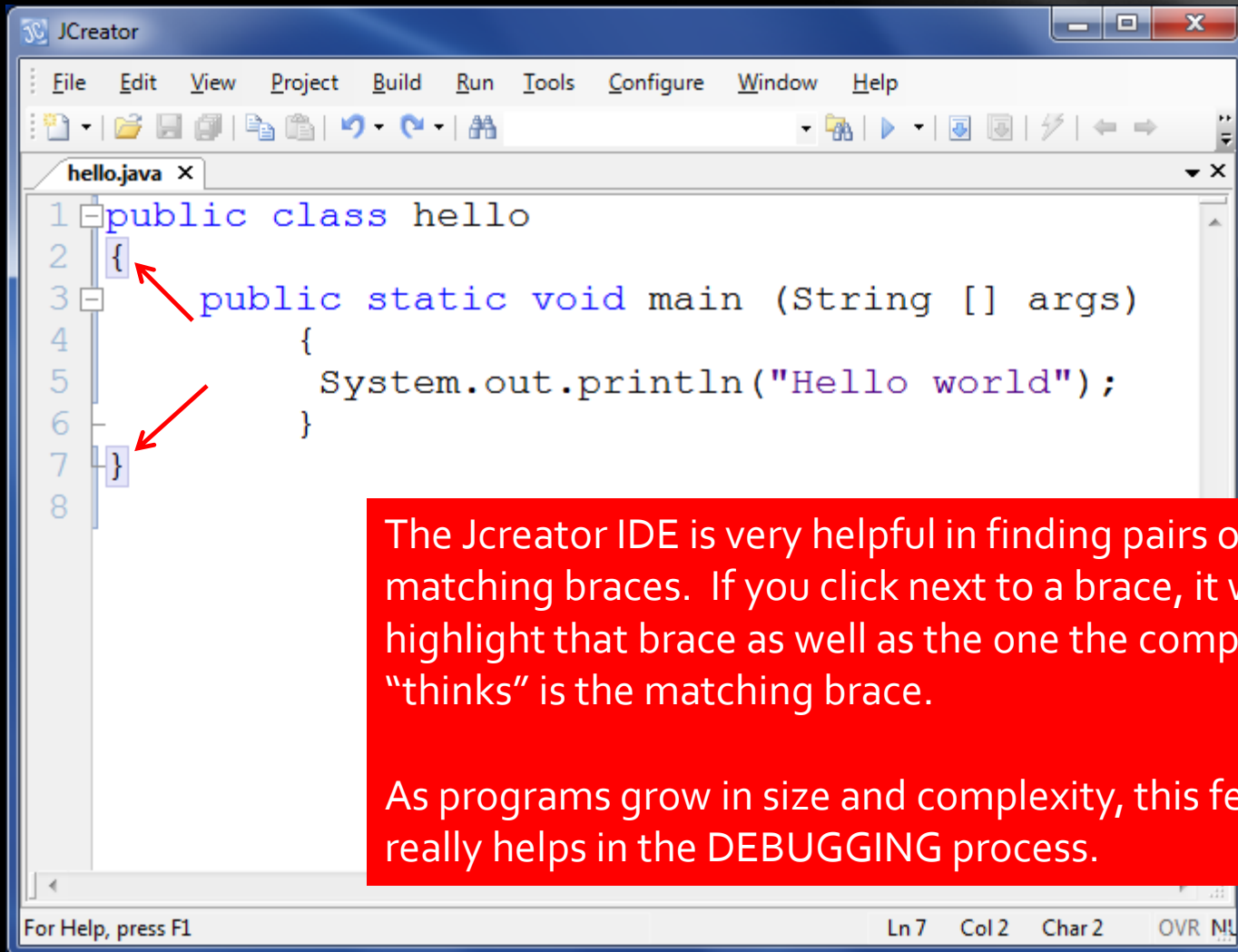
```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
```

Each open brace “{” marks the beginning of a block of code and must always be completed with a closing brace, “}”. In this example the closing brace for the main block is missing. The braces for the class block are OK.

Once again, the error message is not real helpful, but once you see it often enough, you will remember what it really means.



Matching braces highlighted



```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
8
```

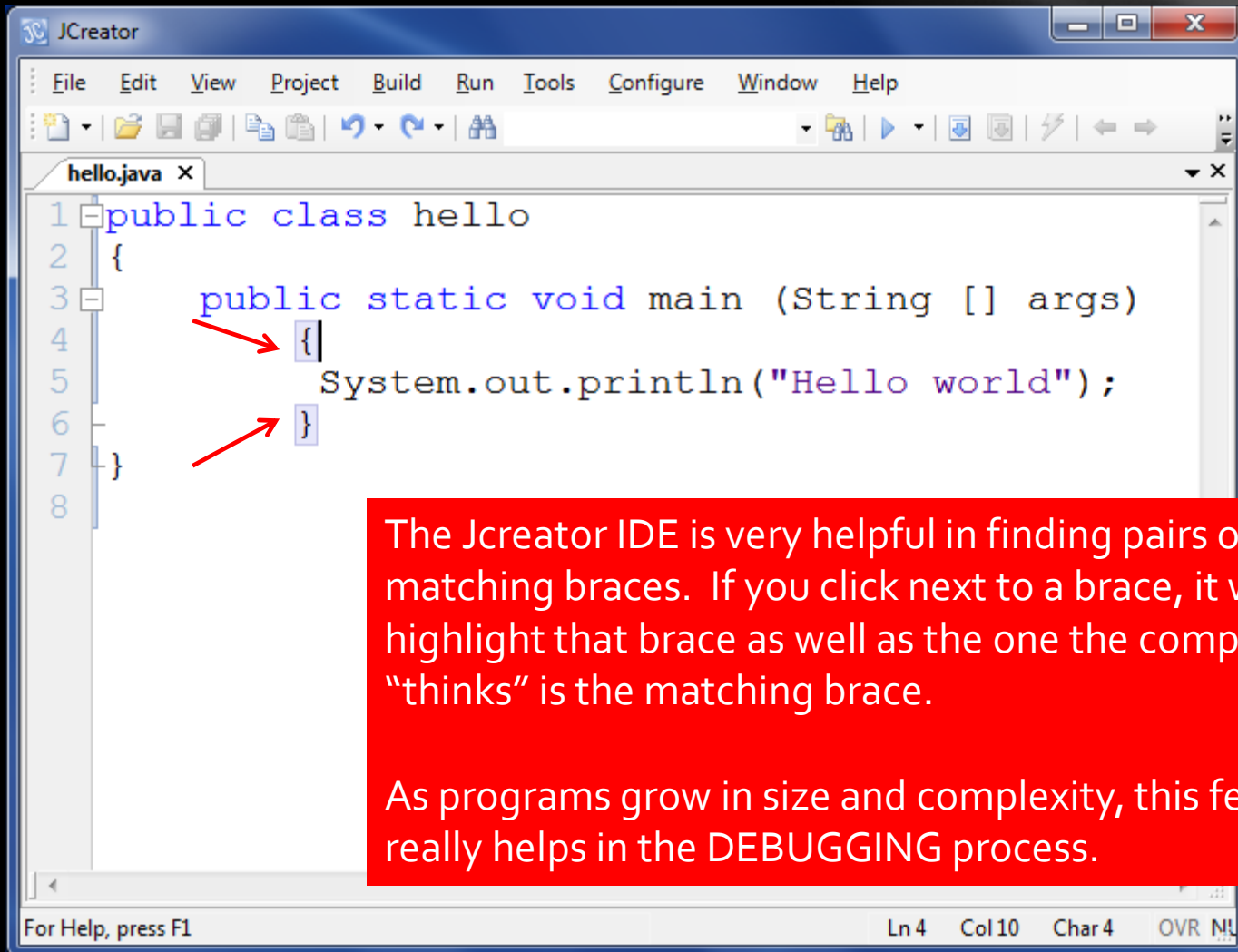
The Jcreator IDE is very helpful in finding pairs of matching braces. If you click next to a brace, it will highlight that brace as well as the one the compiler "thinks" is the matching brace.

As programs grow in size and complexity, this feature really helps in the DEBUGGING process.

For Help, press F1 Ln 7 Col 2 Char 2 OVR NL



Matching braces highlighted



```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
8
```

The Jcreator IDE is very helpful in finding pairs of matching braces. If you click next to a brace, it will highlight that brace as well as the one the compiler "thinks" is the matching brace.

As programs grow in size and complexity, this feature really helps in the DEBUGGING process.

For Help, press F1 Ln 4 Col 10 Char 4 OVR NL



Other types of errors

- There are two other general categories of errors which we will explore later in more detail.
- They are:
 - **Run time errors** – program compiles just fine, but encounters an error during execution
 - **Logical errors** – program runs fine, but doesn't do the right thing

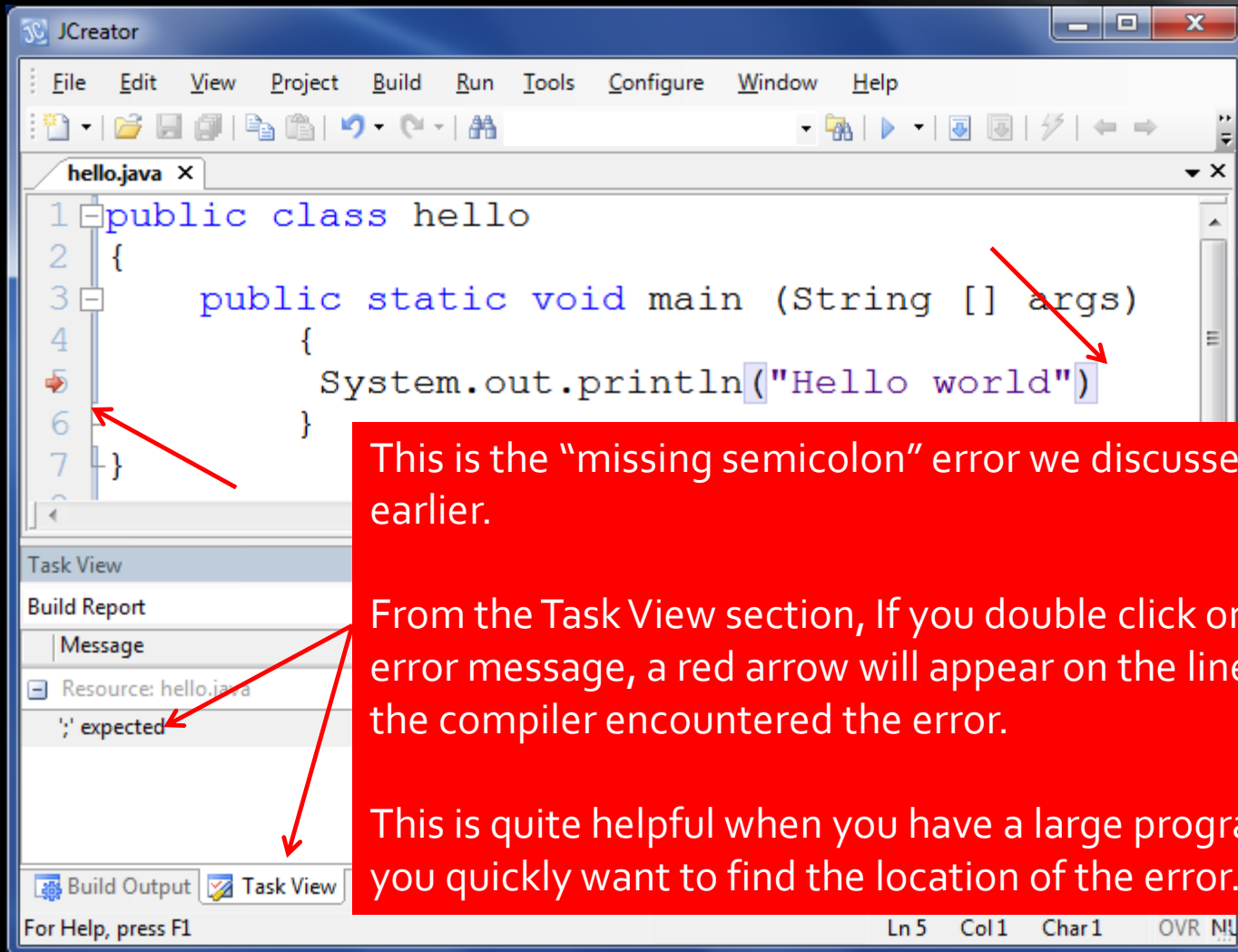


Four parts of an error message

- For the last part of this first lesson we'll talk about the four parts of an error message as shown by the JCreator IDE.



Red arrow on line of error



The screenshot shows the JCreator IDE with a Java file named `hello.java` open. The code is as follows:

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world")
6     }
7 }
```

The error message in the Task View is: `';' expected`. A red arrow points from this message to the closing curly brace on line 5 of the code.

This is the "missing semicolon" error we discussed earlier.

From the Task View section, if you double click on the error message, a red arrow will appear on the line where the compiler encountered the error.

This is quite helpful when you have a large program and you quickly want to find the location of the error.



Error message, Part One

The screenshot shows the JCreator IDE with a Java file named 'hello.java' open. The code is as follows:

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world")
6     }
7 }
```

The Build Output window shows the following error message:

```
-----Configuration: <Default>-----
C:\Users\John Owen\Desktop\OWEN CS Lessons 2011\hello.java:5: ';' expected
    System.out.println("Hello world")
                          ^
1 error
Process completed.
```

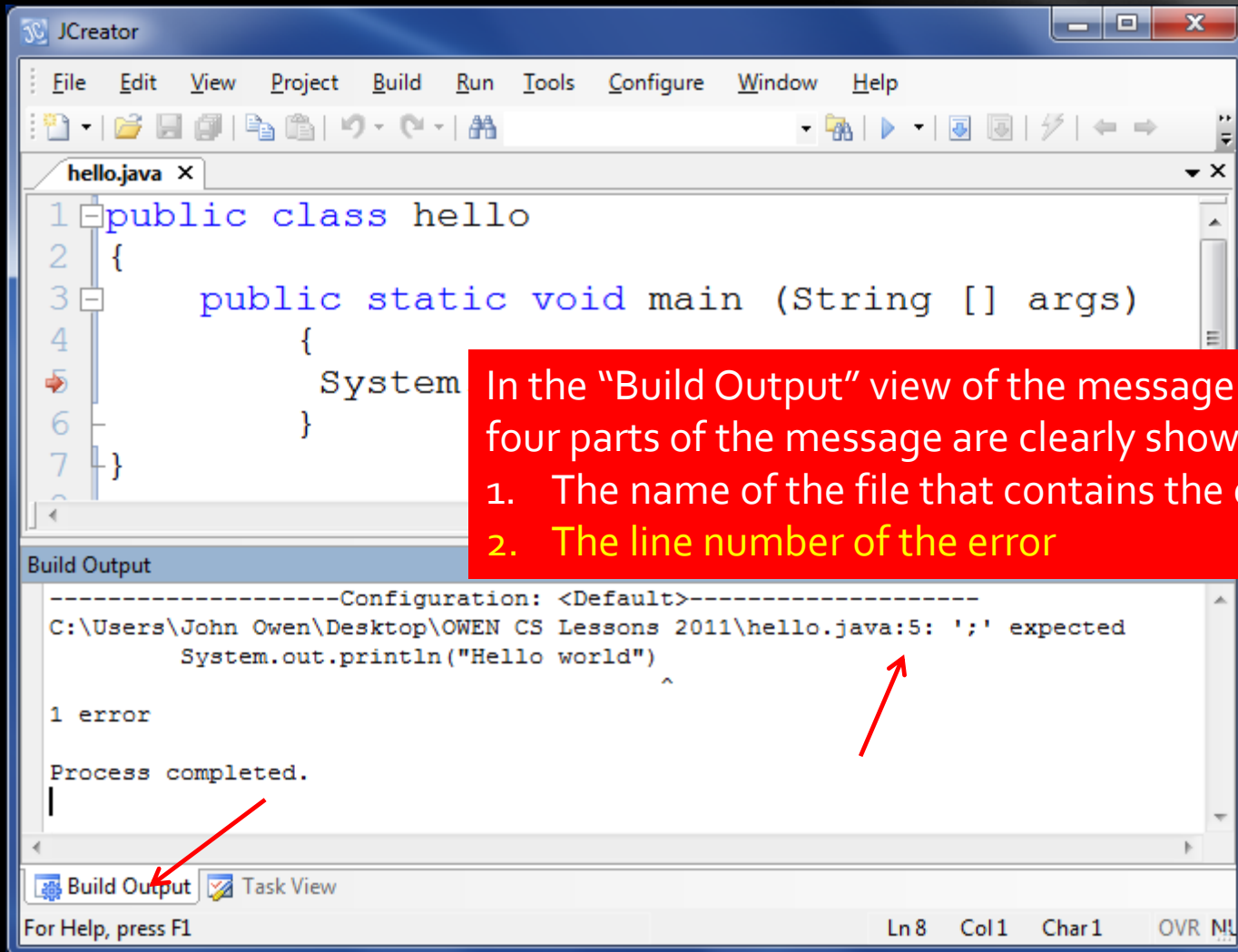
Red arrows in the image point to the error message in the Build Output window and the Build Output tab at the bottom of the IDE.

In the "Build Output" view of the message section, the four parts of the message are clearly shown:

1. The name of the file that contains the error



Error message, Part Two



The screenshot shows the JCreator IDE with a Java file named 'hello.java' open. The code is as follows:

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System
6     }
7 }
```

The 'Build Output' window at the bottom shows the following error message:

```
-----Configuration: <Default>-----
C:\Users\John Owen\Desktop\OWEN CS Lessons 2011\hello.java:5: ';' expected
    System.out.println("Hello world")
                        ^
1 error
Process completed.
```

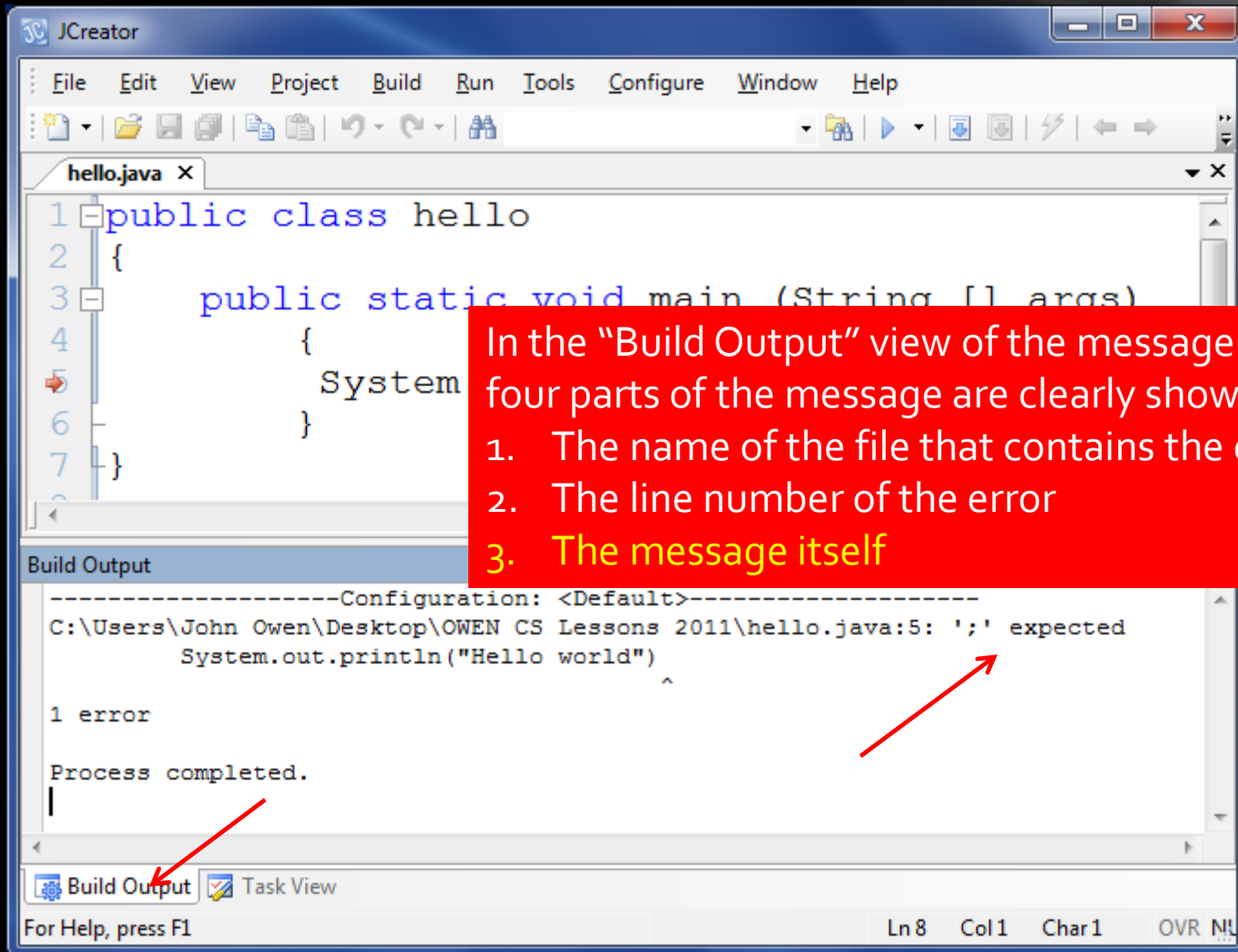
Two red arrows point to the error message: one points to the file path 'C:\Users\John Owen\Desktop\OWEN CS Lessons 2011\hello.java' and the other points to the line number '5'.

In the "Build Output" view of the message section, the four parts of the message are clearly shown:

1. The name of the file that contains the error
2. The line number of the error



Error message, Part Three



The screenshot shows the JCreator IDE with a Java file named `hello.java` open. The code is as follows:

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
7 }
```

The Build Output window at the bottom shows the following error message:

```
-----Configuration: <Default>-----
C:\Users\John Owen\Desktop\OWEN CS Lessons 2011\hello.java:5: ';' expected
    System.out.println("Hello world")
                        ^
1 error
Process completed.
```

Two red arrows point to the error message: one points to the file name and line number, and the other points to the error message text.

In the "Build Output" view of the message section, the four parts of the message are clearly shown:

1. The name of the file that contains the error
2. The line number of the error
3. The message itself



Error message, Part Four

The screenshot shows the JCreator IDE with a Java file named 'hello.java' open. The code in the editor is as follows:

```
1 public class h
2 {
3     public sta
4         {
5         System
6     }
7 }
```

The 'Build Output' window at the bottom displays the following error message:

```
-----Configuration: <Default>-----
C:\Users\John Owen\Desktop\OWEN CS Lessons 2011\hello.java:5: ';' expected
    System.out.println("Hello world")
                               ^
1 error
Process completed.
```

Two red arrows point to the error message: one points to the 'Build Output' tab, and the other points to the '^' symbol in the error message.

In the "Build Output" view of the message section, the four parts of the message are clearly shown:

1. The name of the file that contains the error
2. The line number of the error
3. The message itself
4. An arrow "^" pointing to where the compiler stopped when it encountered the error



Four parts, example #2

```
1 public class hello
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("Hello world");
6     }
}
```

Build Output

```
C:\Users\John Owen\Desktop\hello.java:5: error: file while parsing
^
1 error
Process completed.
```

This is the "missing brace" error message example.

The four parts of the message are still shown, but notice that the line number and "^" are actually on the line following the location of the missing brace.

This happens quite often, so look carefully!



Lesson Summary

- This lesson introduced the “Hello World” program using a typical indented program structure.



Lesson Summary

- It also pointed out different types of errors that can be encountered, the various features of the JCreator IDE error message system, and how to interpret the message parts in the **DEBUGGING** process.



Lesson 1A Lab

- And now, your first lab assignment is to complete a program that produces five lines of output:
 - Your name
 - Your favorite food
 - A celebrity or public figure, past or present, whom you would like to meet
 - Your least favorite chore at home
 - What you would rather be doing right now, other than completing this lab assignment



Lesson Lab Solution Example

```
Lesson1ALab.java
1 public class Lesson1ALab
2 {
3     public static void main(String [] args)
4     {
5         System.out.println("Name: John B Owen");
6         System.out.println("Favorite food: Chinese");
7         System.out.println("Want to meet: Carl Sagan");
8         System.out.println("Least favorite chore: Vacuuming");
9         System.out.println("Right now, I would rather be: Sailing my boat");
10    }
11 }
12 |
```

General Output

```
-----Configuration: <Default>-----
Name: John B Owen
Favorite food: Chinese
Want to meet: Carl Sagan
Least favorite chore: Vacuuming
Right now, I would rather be: Sailing my boat
```



CONGRATULATIONS!

- YOU JUST WROTE YOUR FIRST COMPUTER PROGRAM!
- *NOW GO ON TO LESSON 1B*



Thanks, and have fun!



To order supplementary materials for all the lessons in this package, including lesson examples, lab solutions, quizzes, tests, and unit reviews, visit the [O\(N\)CS Lessons](#) website, or contact me at

[John B. Owen](#)
captainjbo@gmail.com



8/21/2015