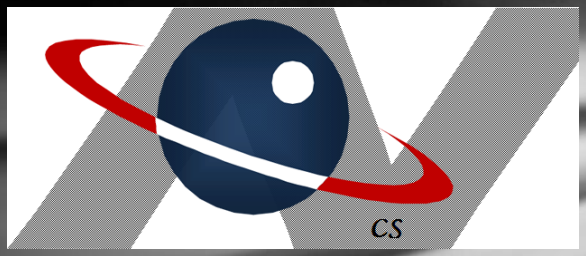
# O(N) CS LESSONS




*Lesson 1A - First Java Program*

*"HELLO WORLD"*

*With DEBUGGING examples*

*By John B. Owen*

# Table of Contents

# 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;*

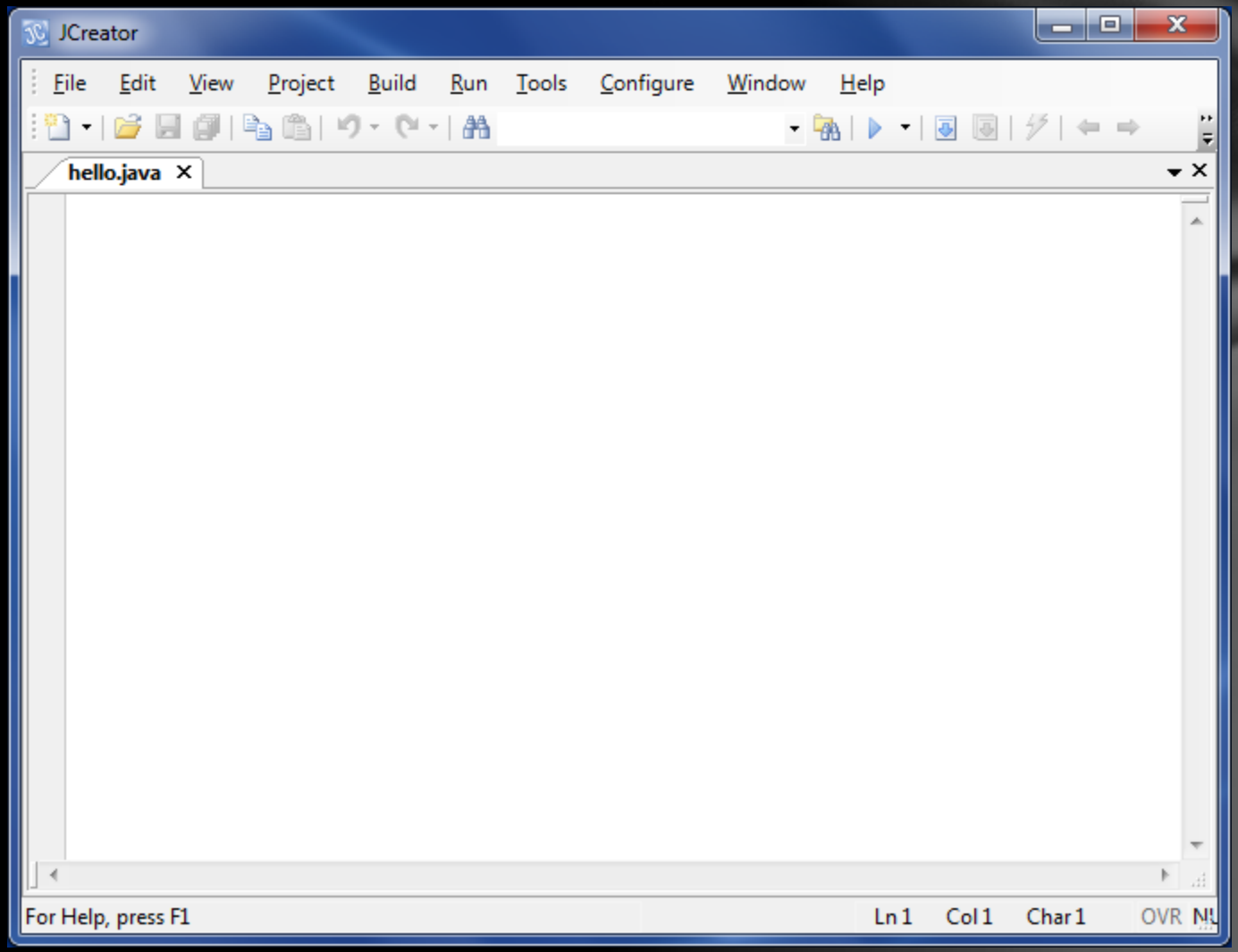
# Objective #2

- You will also recognize and successfully DEBUG typical errors that occur during the programming process, using this first program example.
- *CS1 TEKS 126.33c4(H) identify and debug errors;*

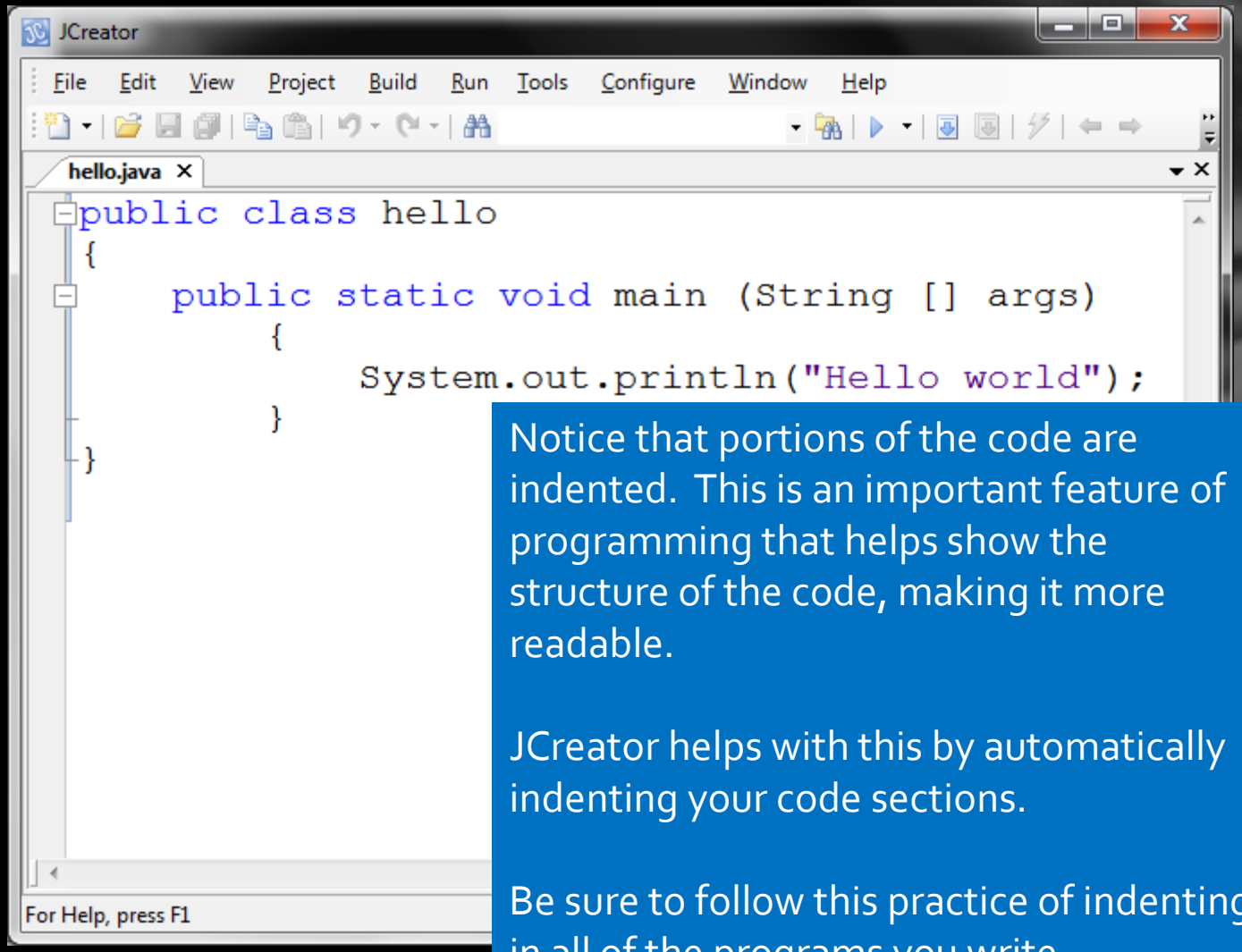
# "Hello World" Lesson Sequence

- 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

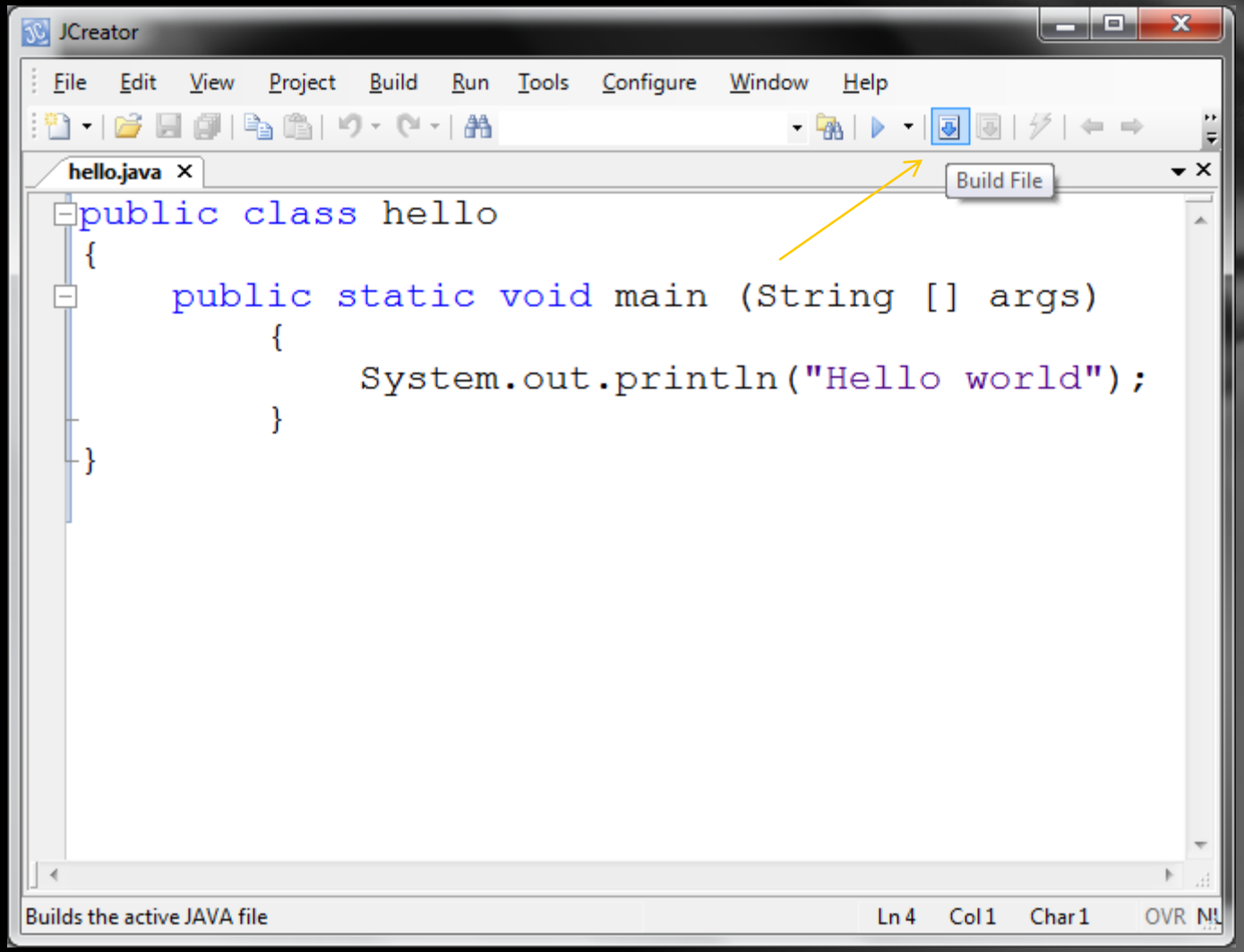



Notice that portions of the code are indented. This is an important feature of programming that helps show the structure of the code, making it more readable.

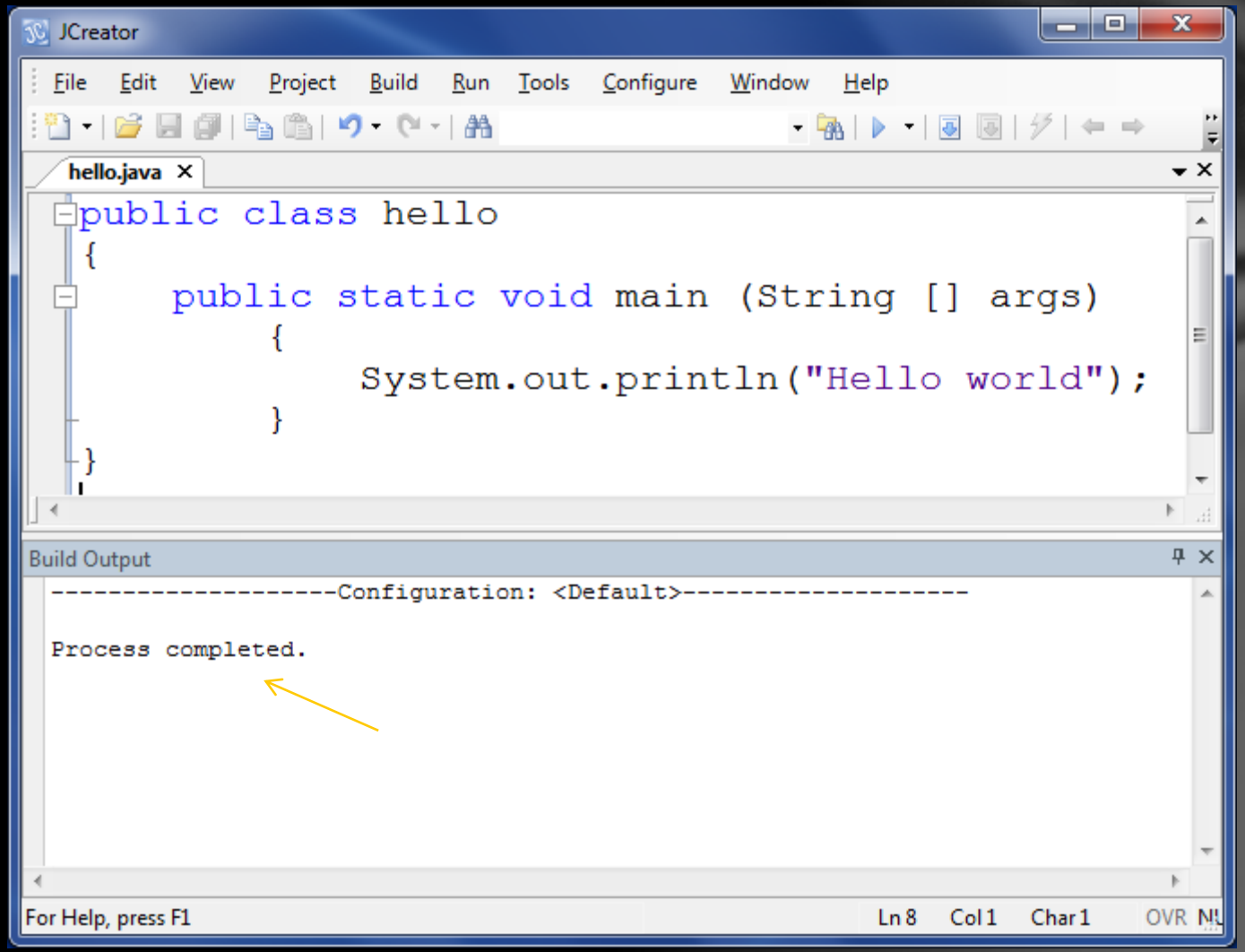
JCreator helps with this by automatically indenting your code sections.

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

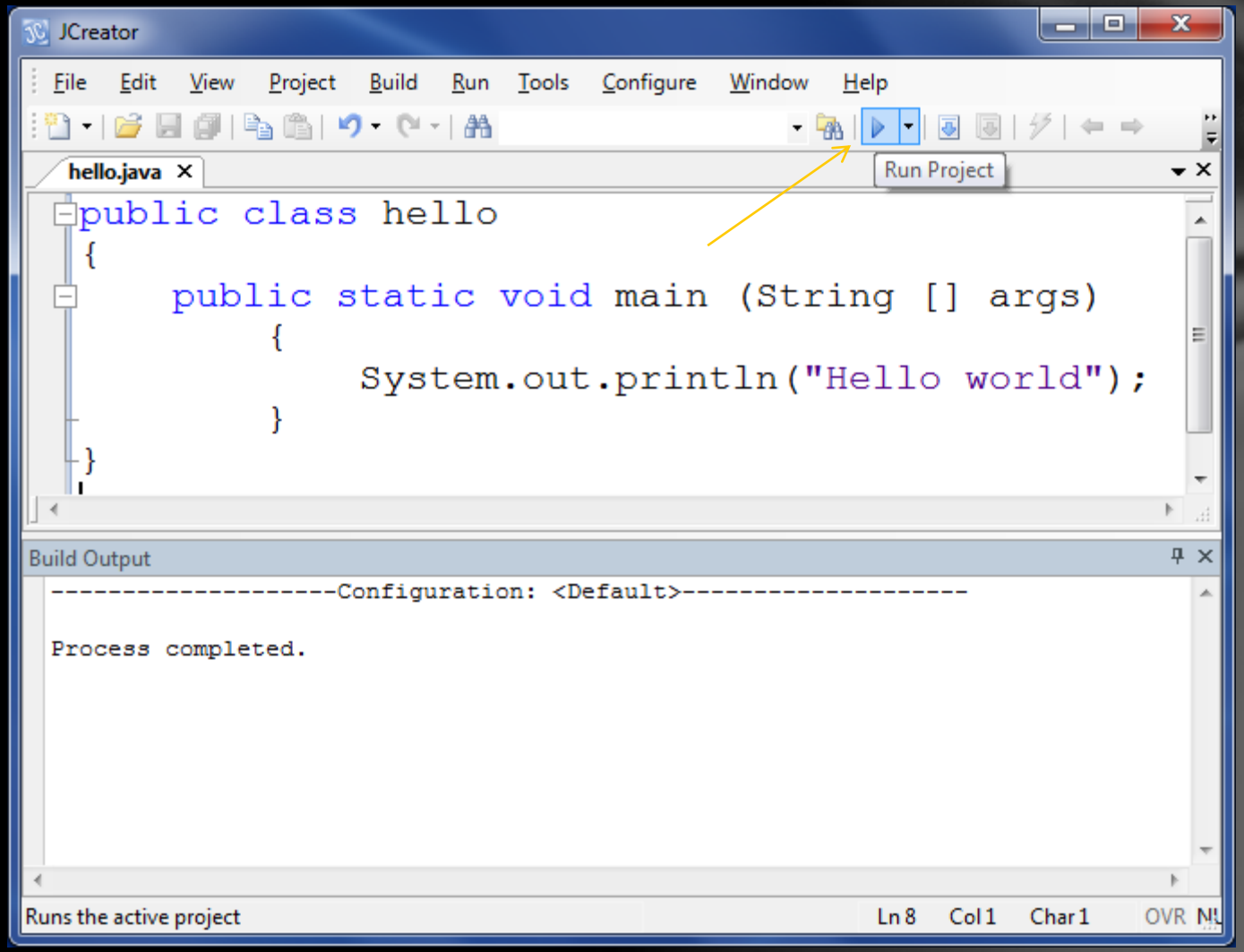
# Step 3 – Compile the code by clicking "Build File"



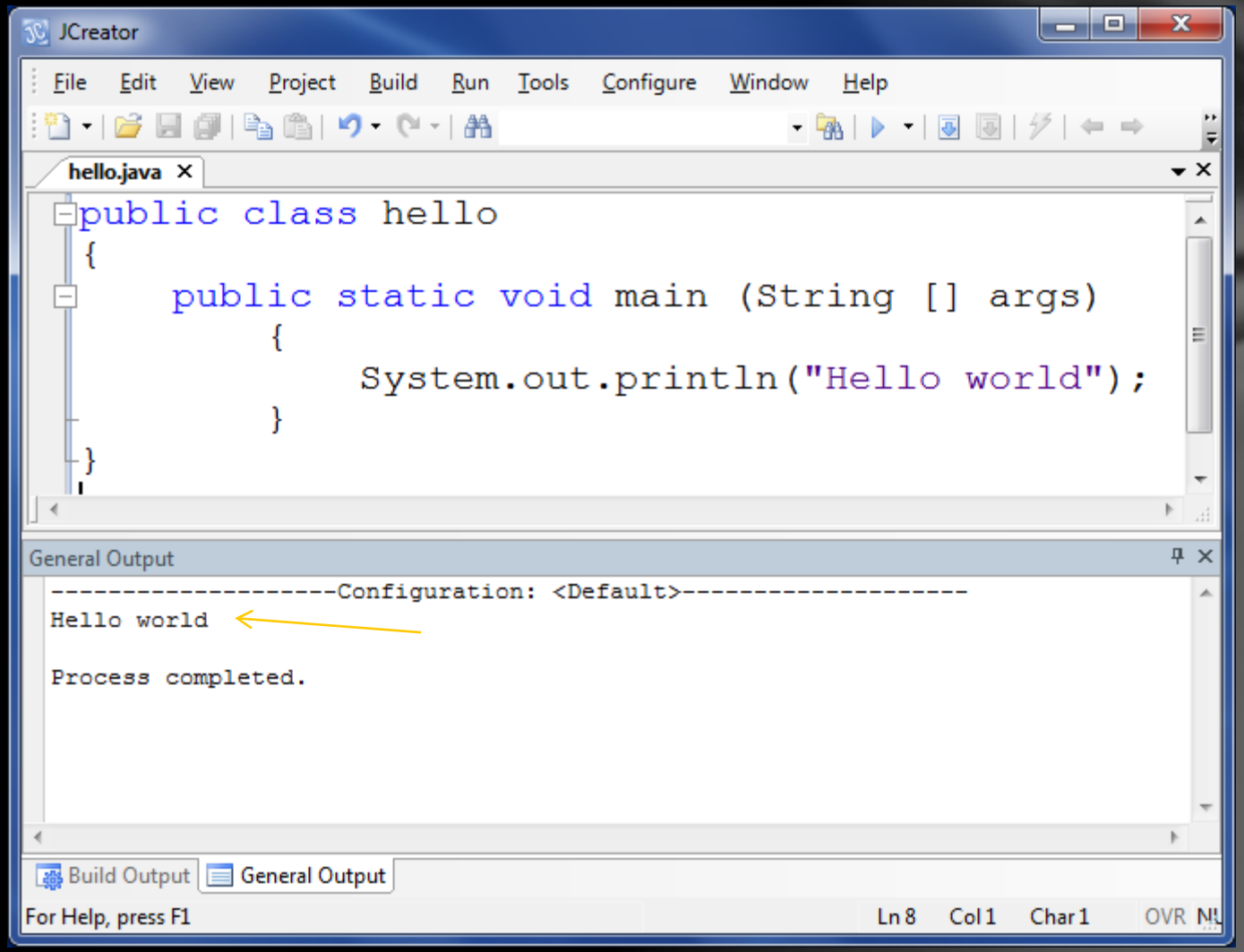
# Step 3 result – Clean compile



# Step 4 – Execute the code by clicking "Run Project"



# Step 4 result – Correct output



# 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.

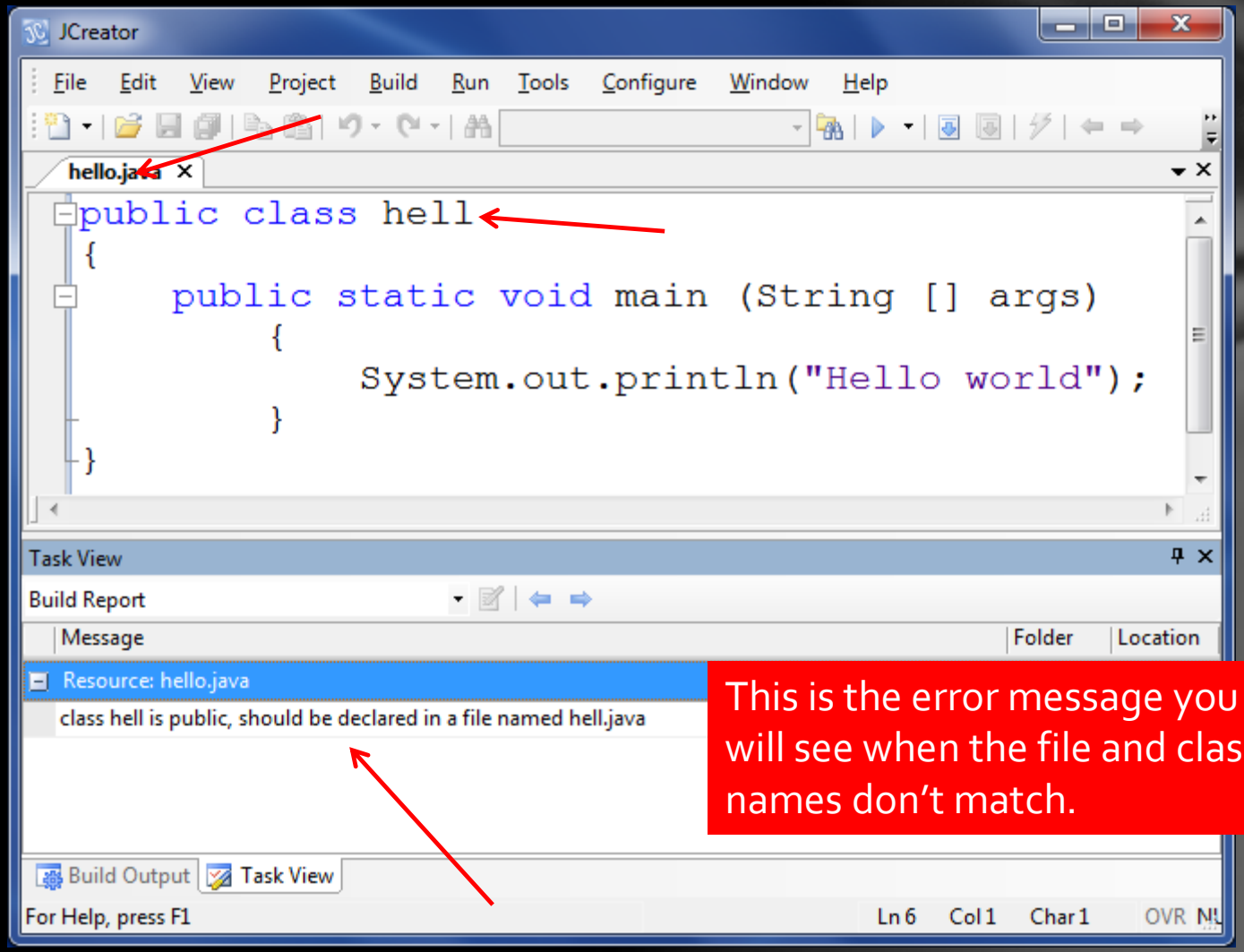
# Compile time errors

- 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

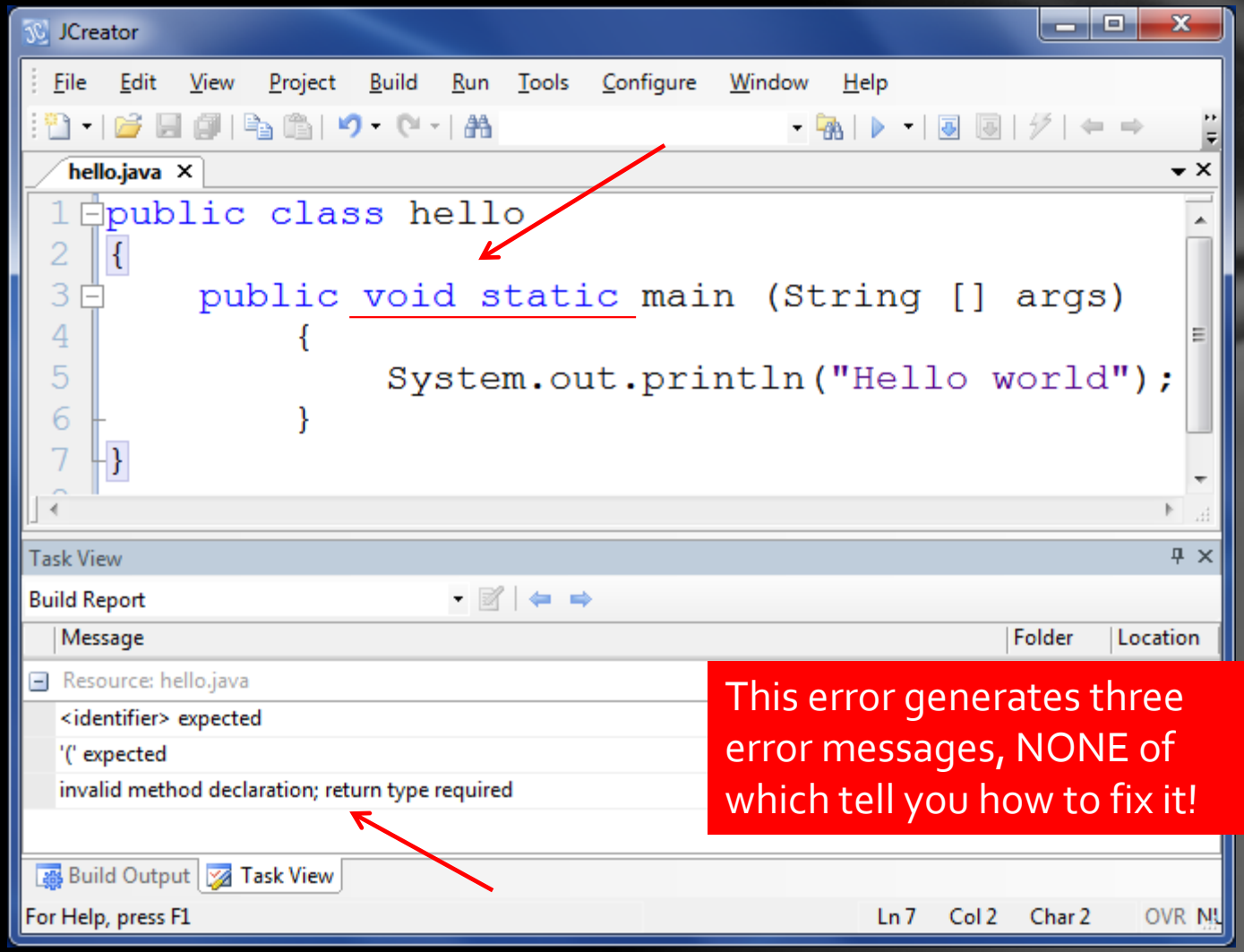
# Lexical Errors

- 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!

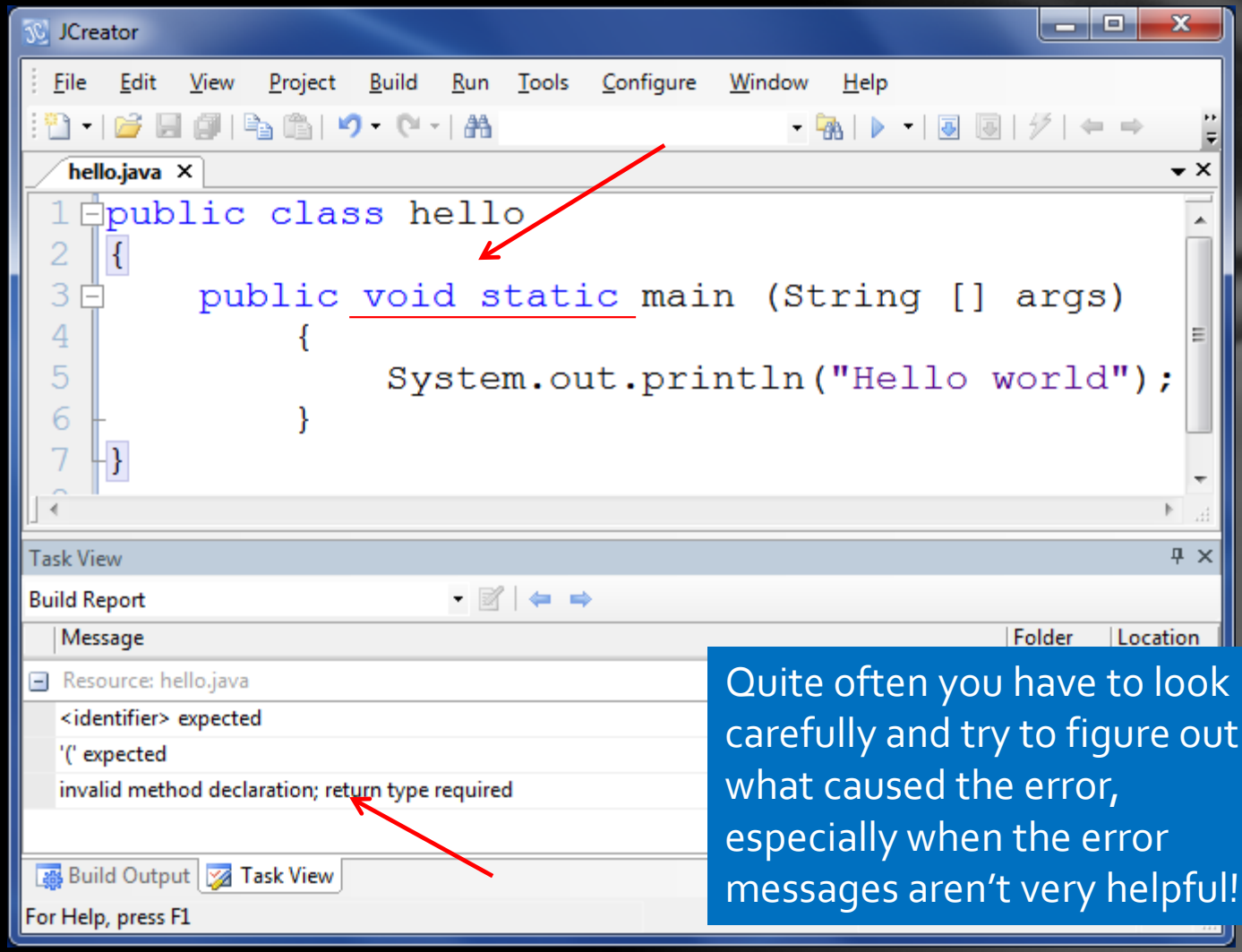


# Lexical error example #2: static and void reversed; must be in correct order!



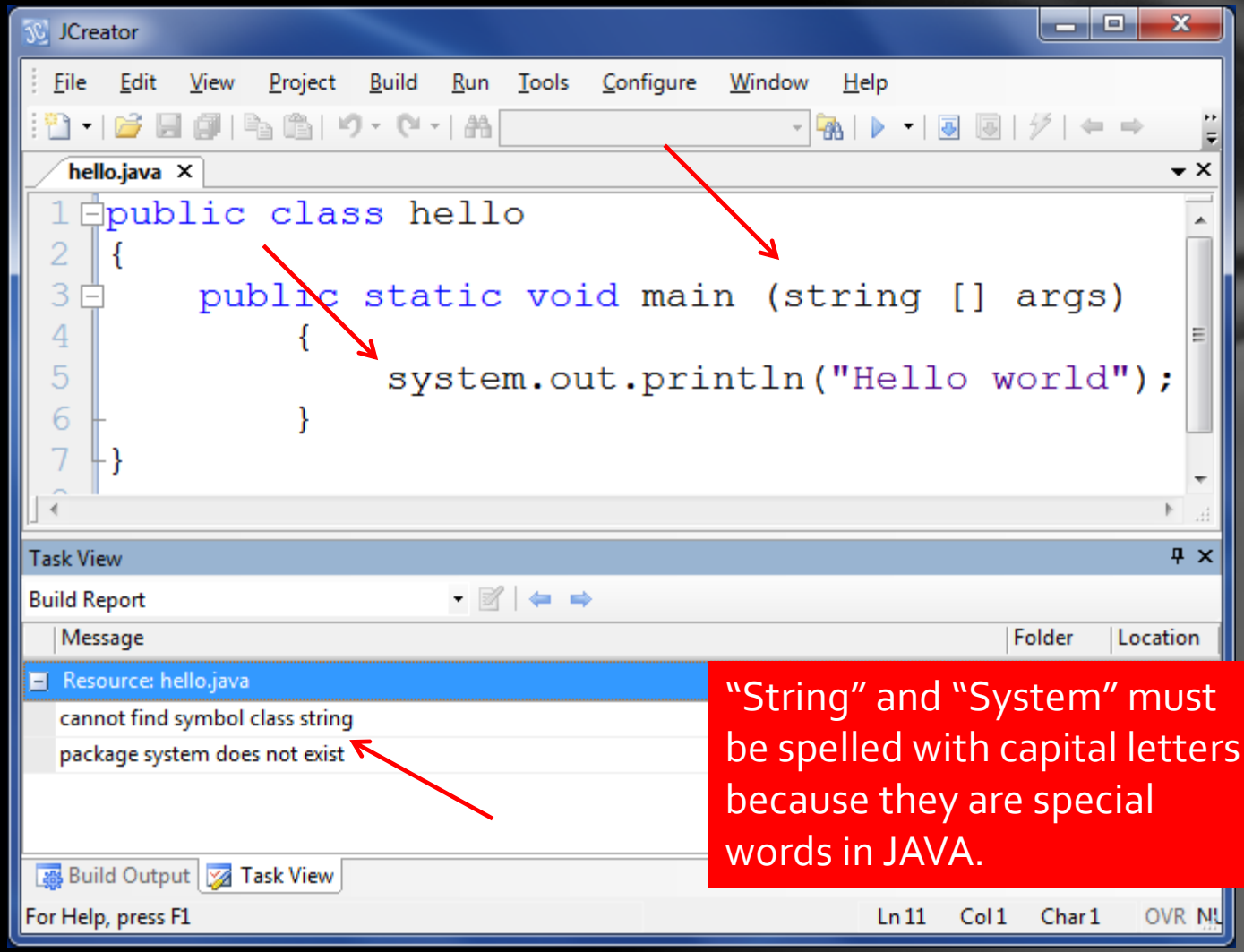

This error generates three error messages, NONE of which tell you how to fix it!

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



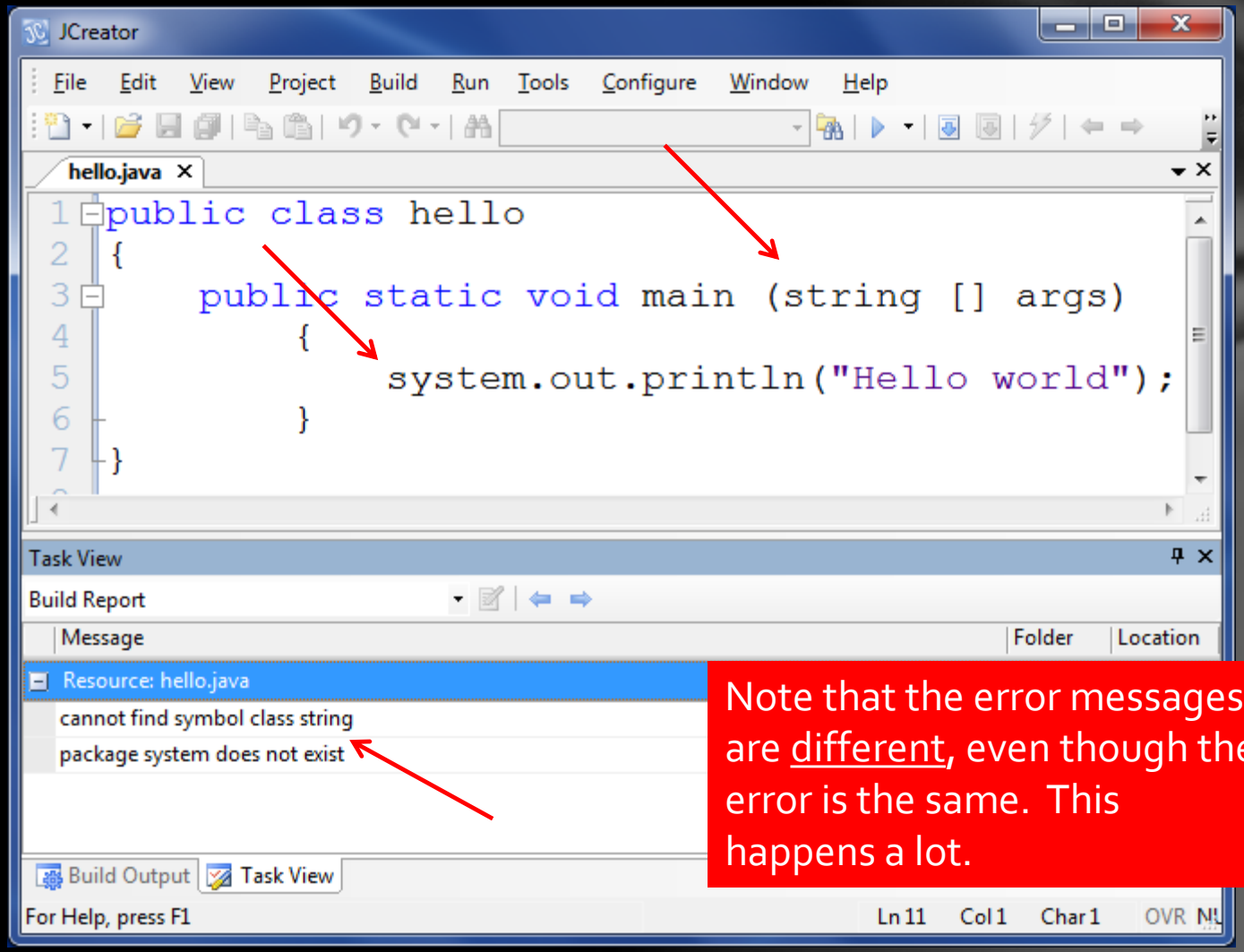

Quite often you have to look carefully and try to figure out what caused the error, especially when the error messages aren't very helpful!

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



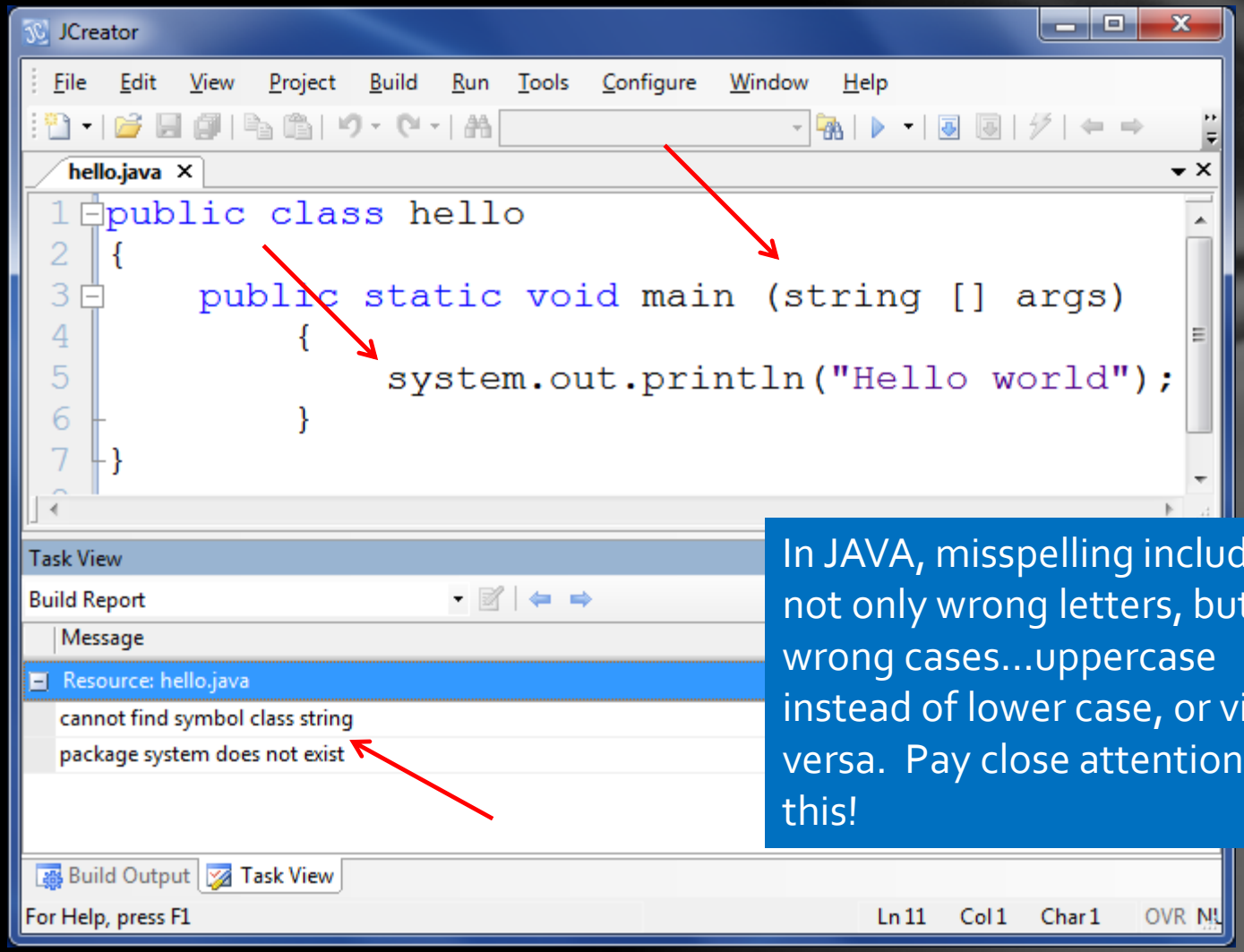

"String" and "System" must be spelled with capital letters because they are special words in JAVA.

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



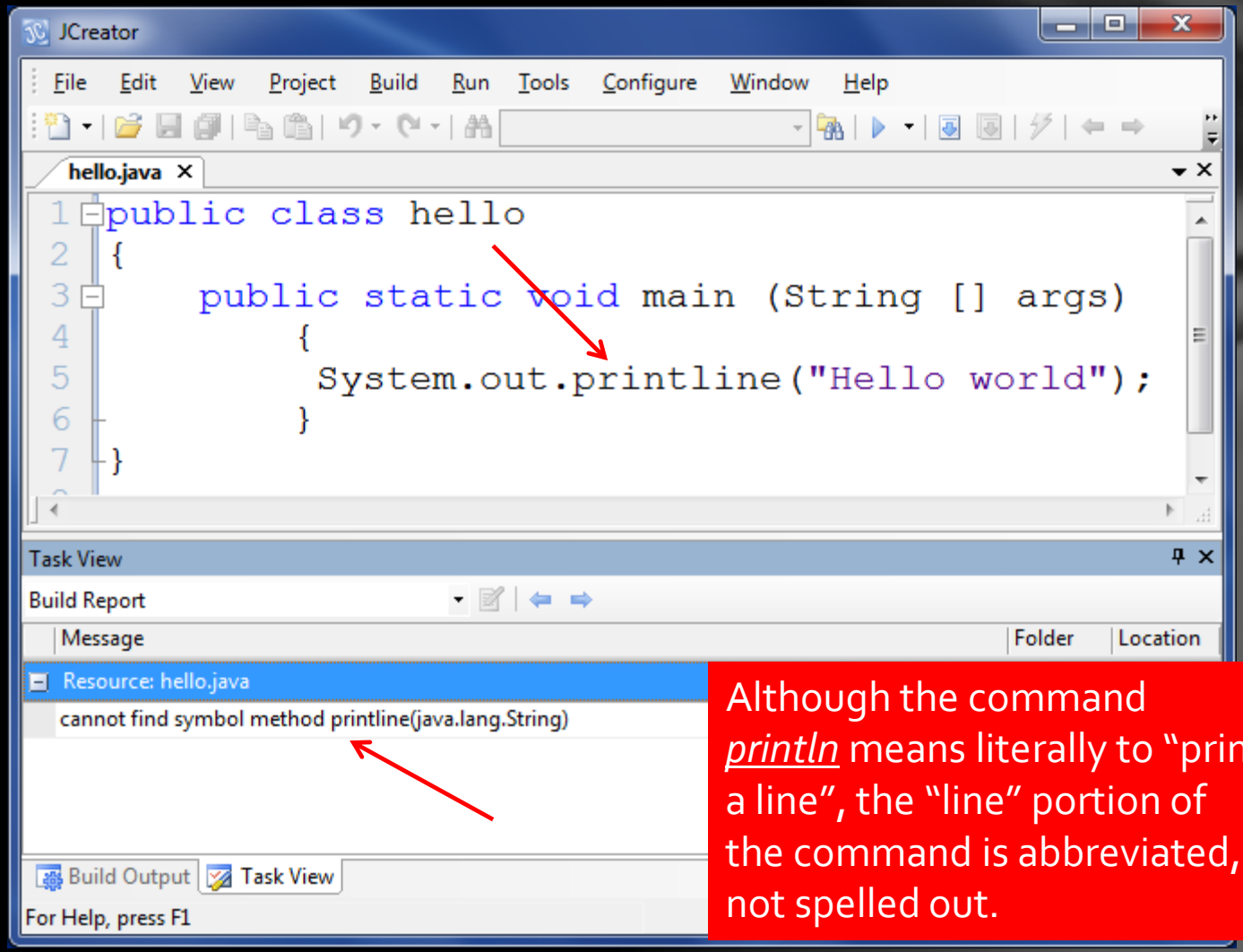

Note that the error messages are different, even though the error is the same. This happens a lot.

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



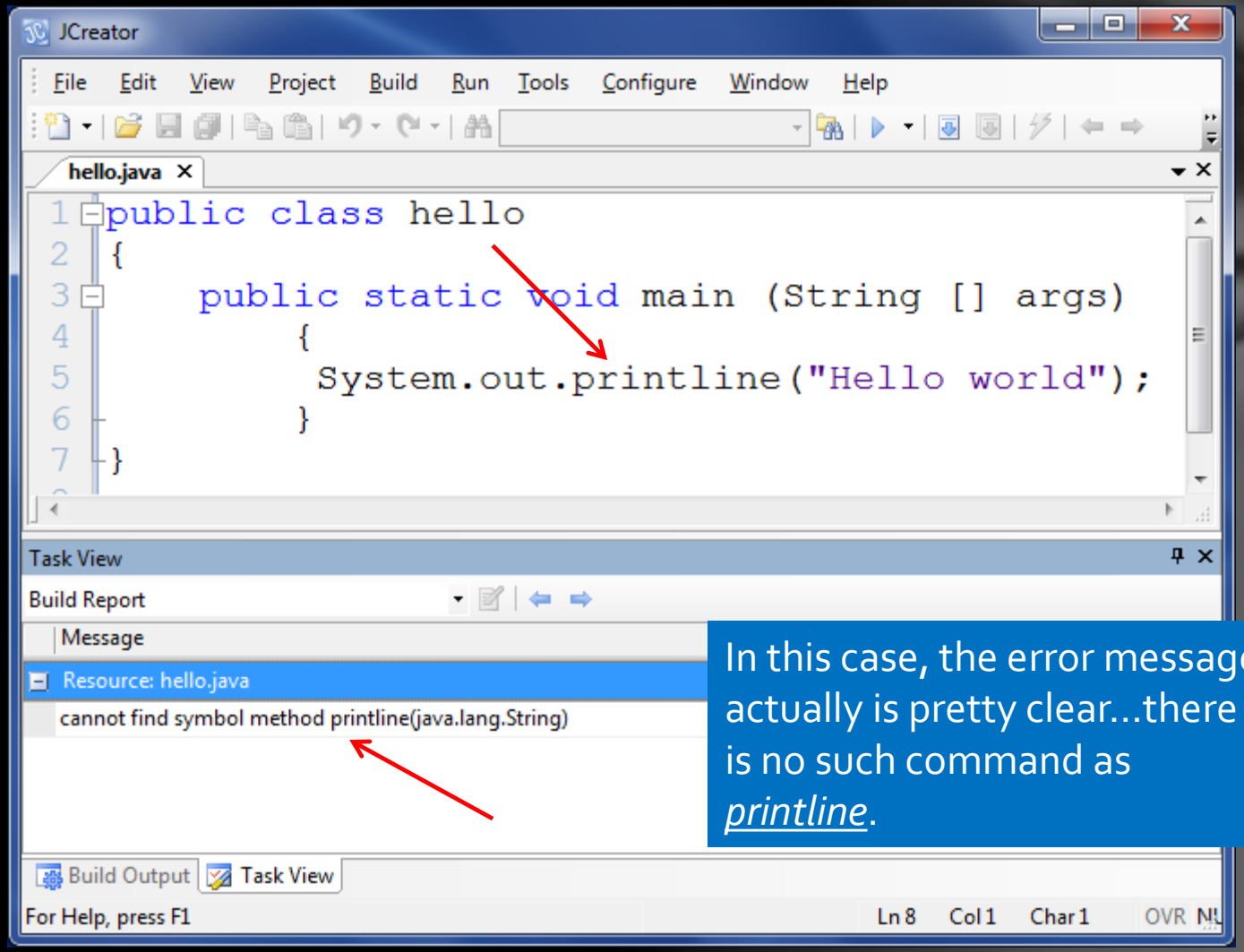

In JAVA, misspelling includes not only wrong letters, but wrong cases…uppercase instead of lower case, or vice versa. Pay close attention to this!

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




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!

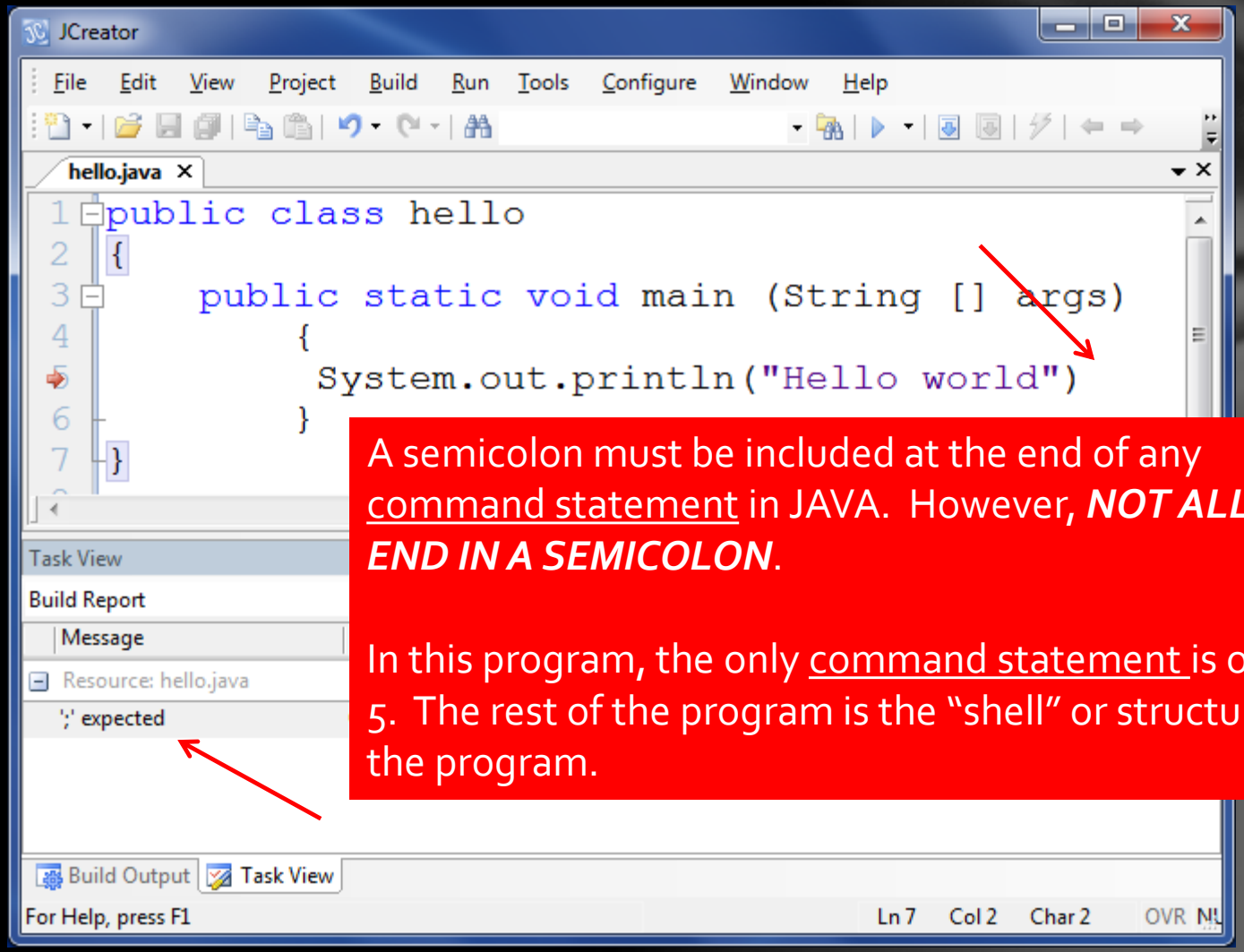



In this case, the error message actually is pretty clear…there is no such command as *printline*.

# 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…

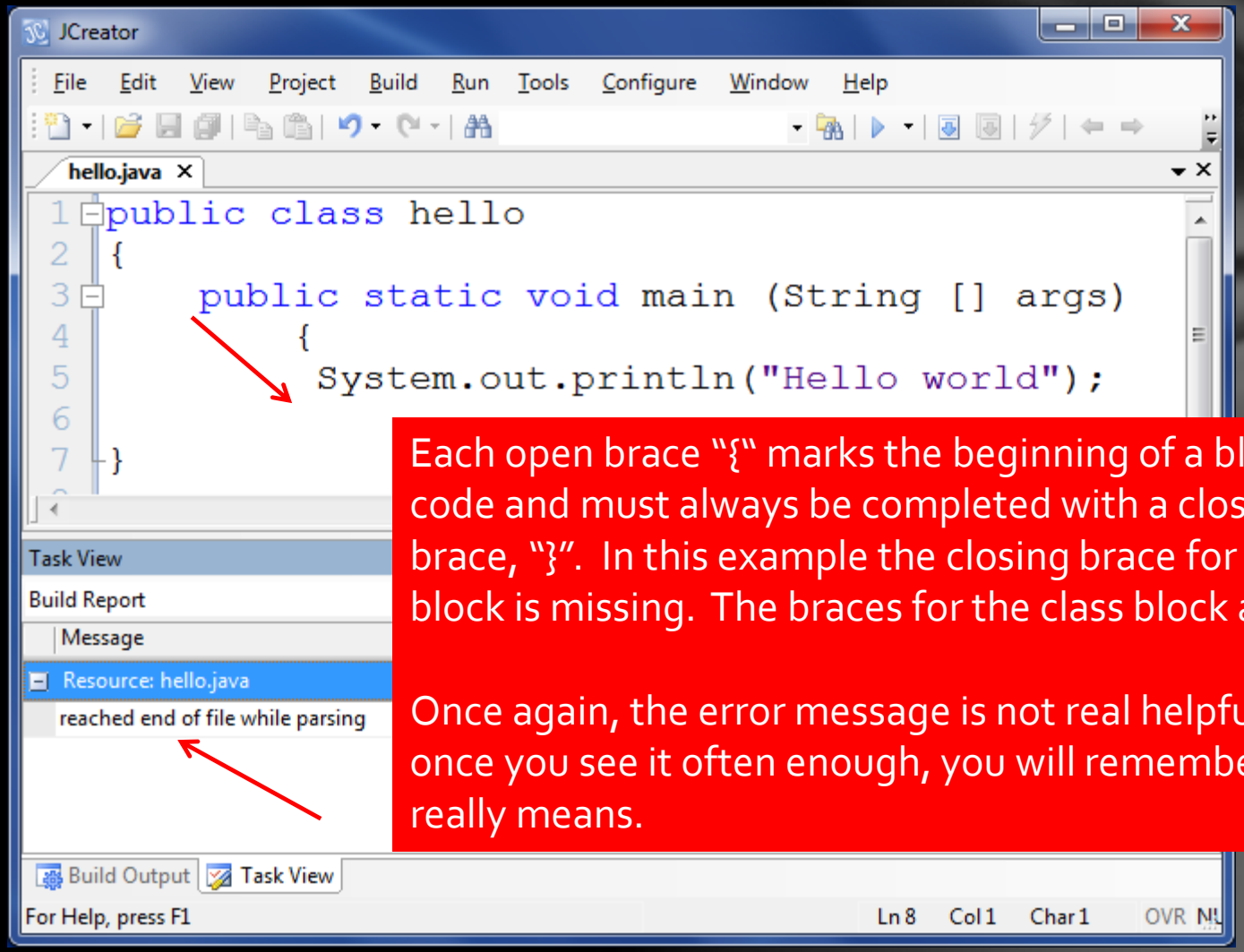
# Syntax Error Example #1: A semicolon is missing!




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.

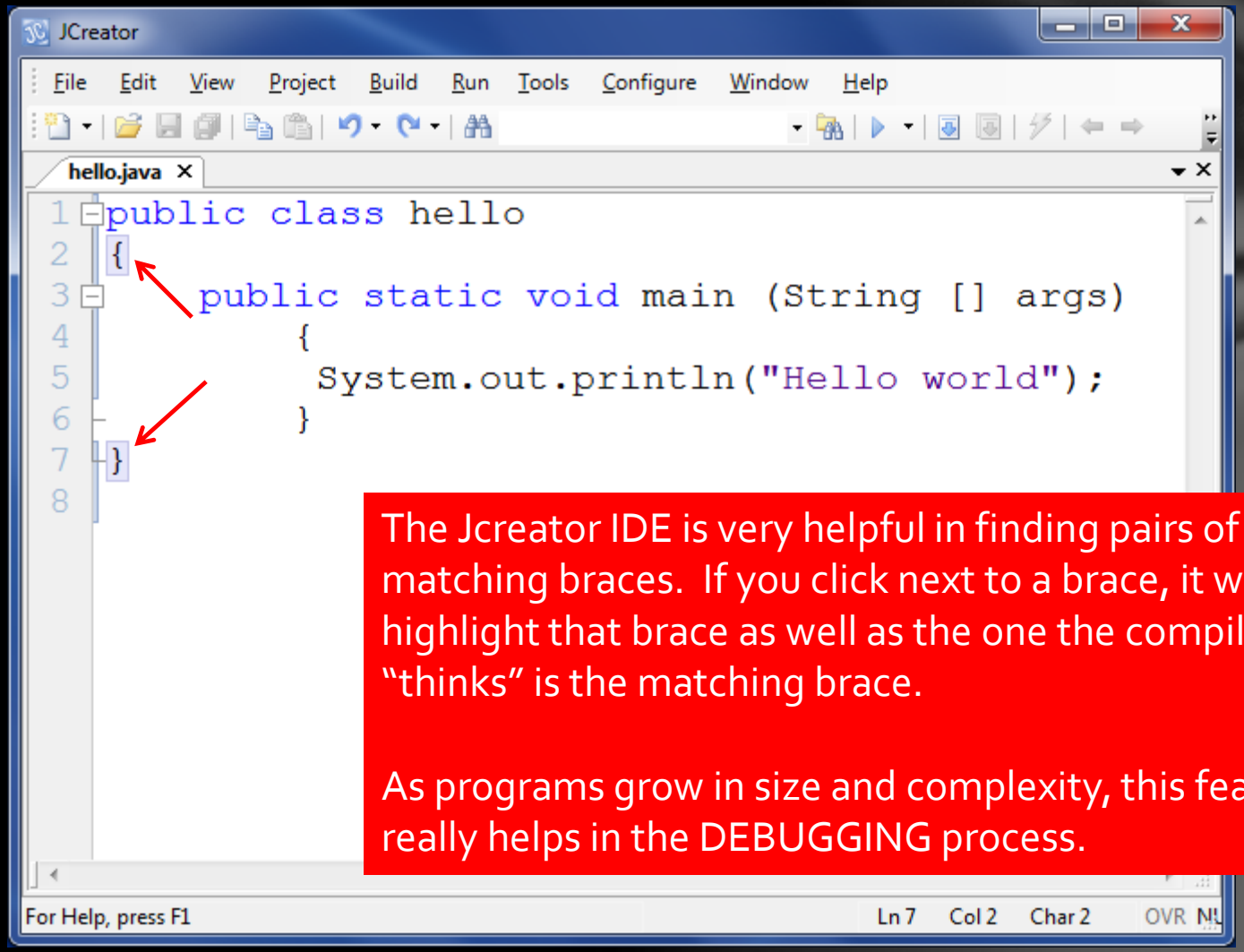
# Syntax Error Example #2: A closing brace is missing!




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.

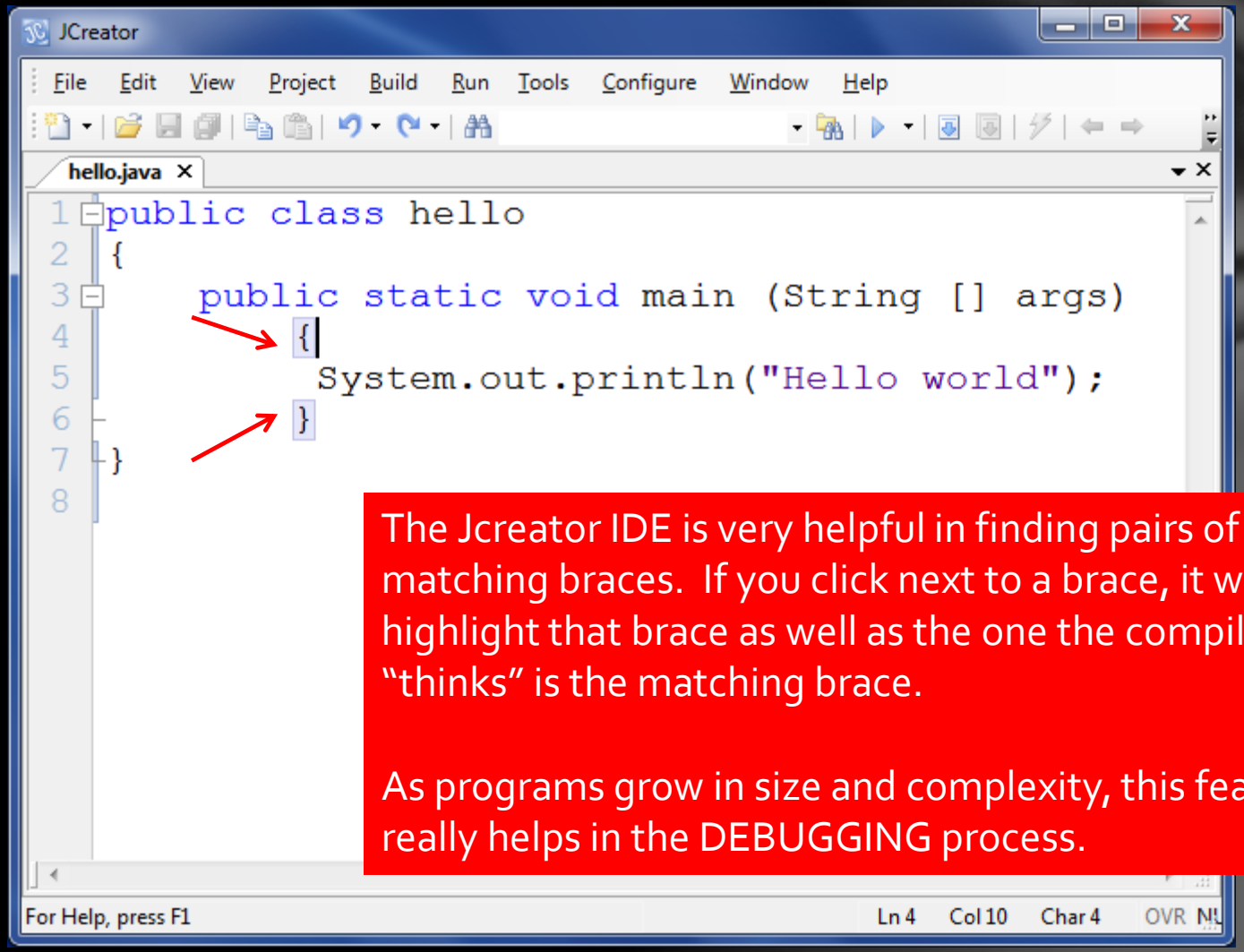
# Matching braces highlighted




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.

# Matching braces highlighted




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.

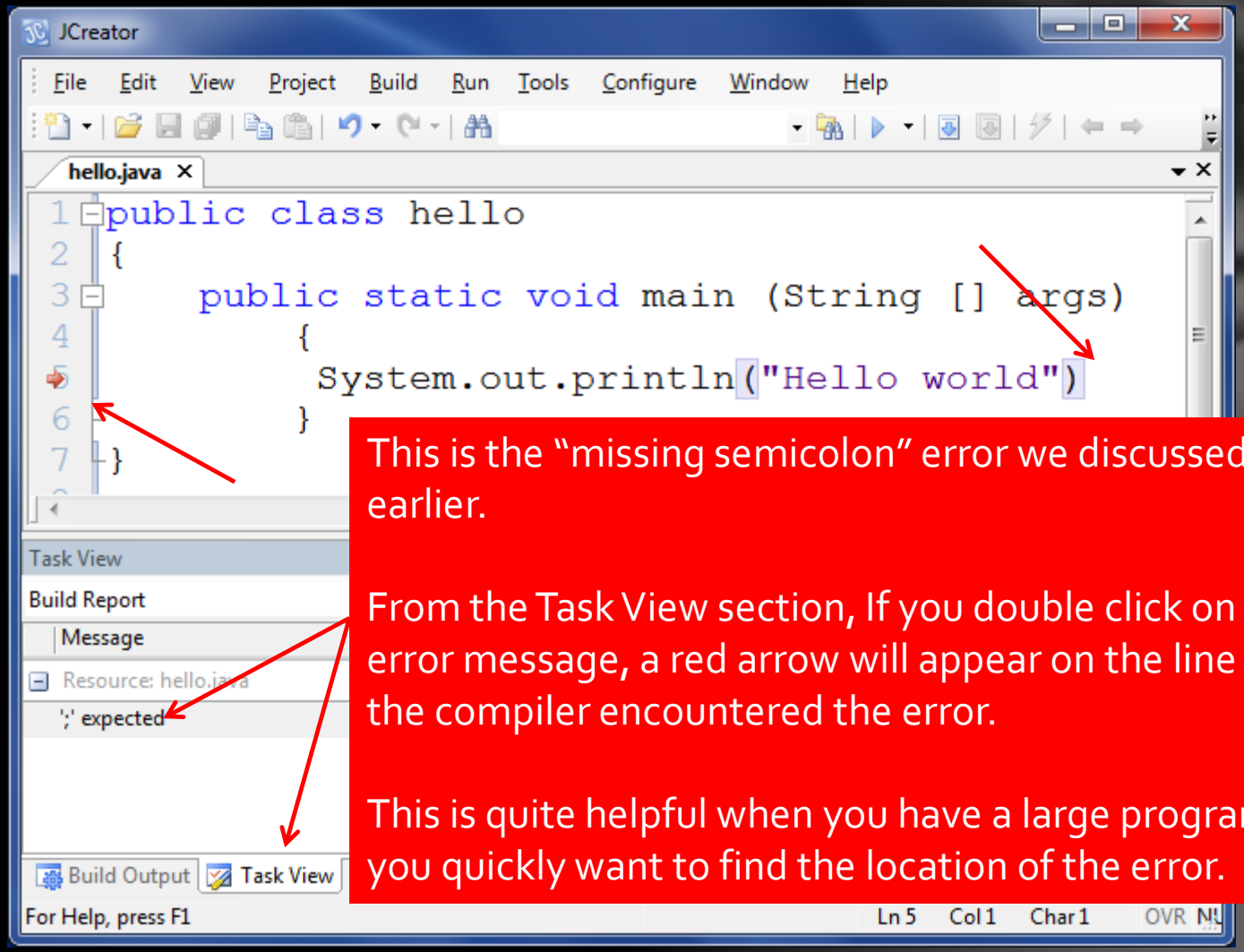
# Other types of errors

- 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

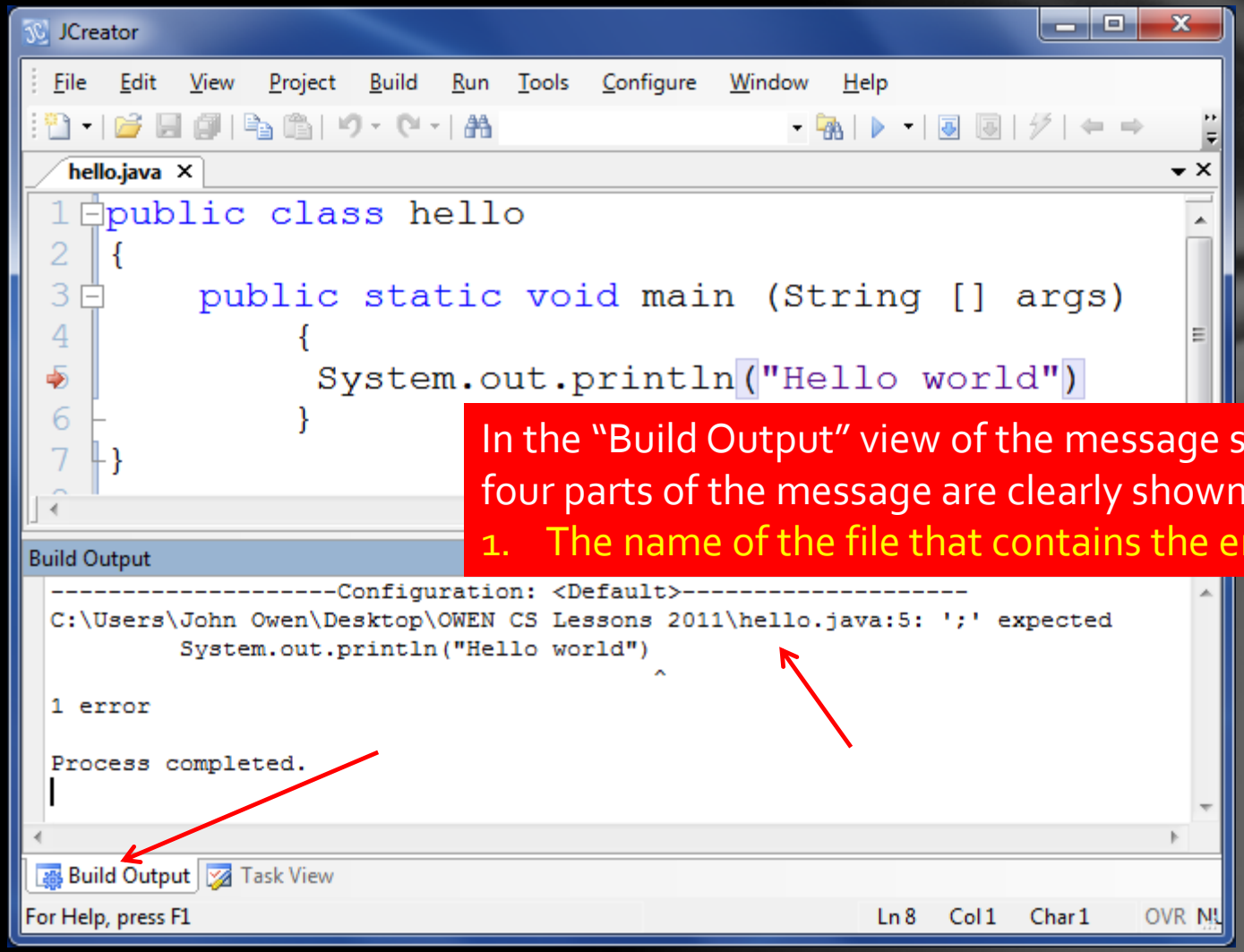



This is the "missing semicolon" error we discussed earlier.

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.

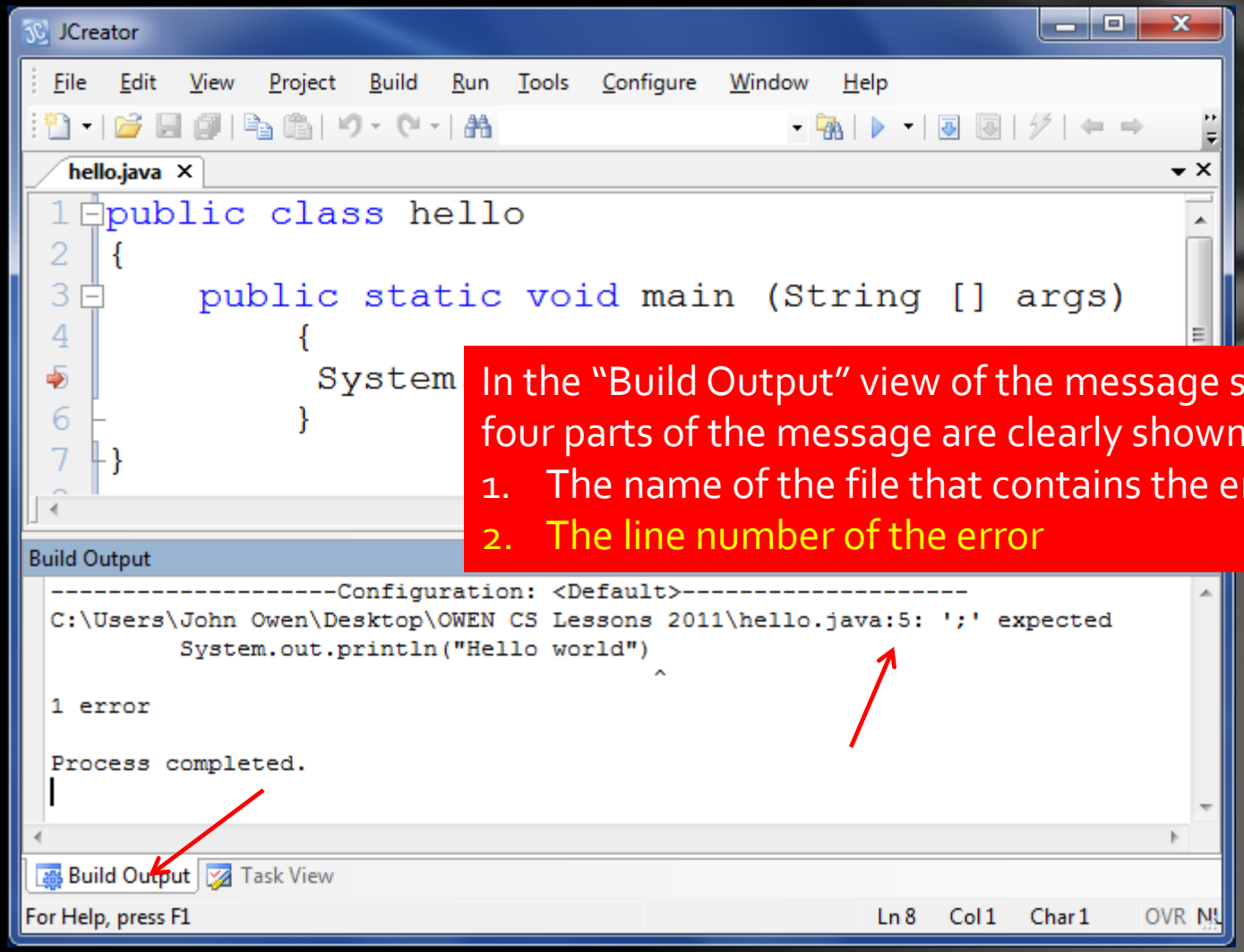
# Error message, Part One




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

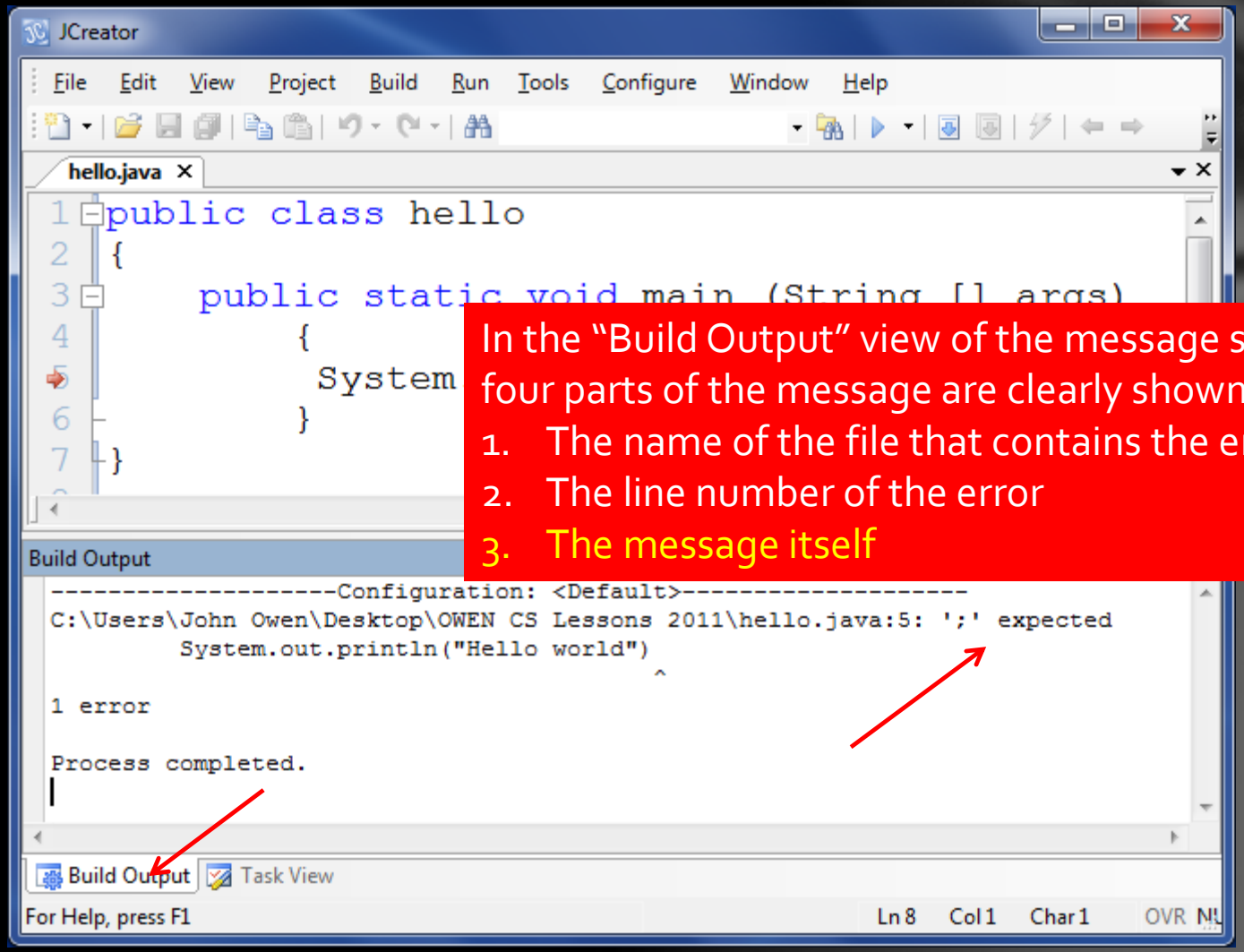
# Error message, Part Two




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

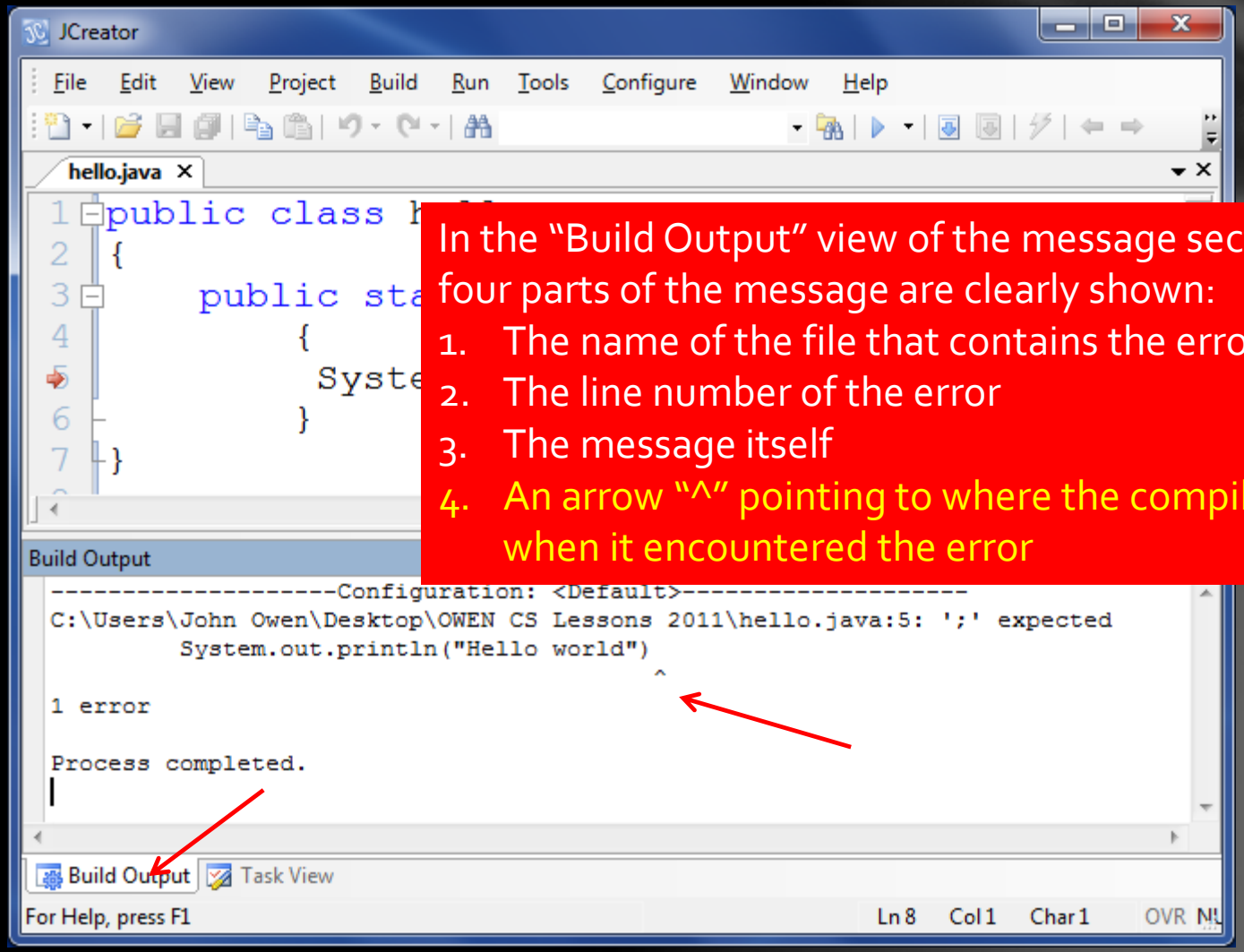
# Error message, Part Three




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

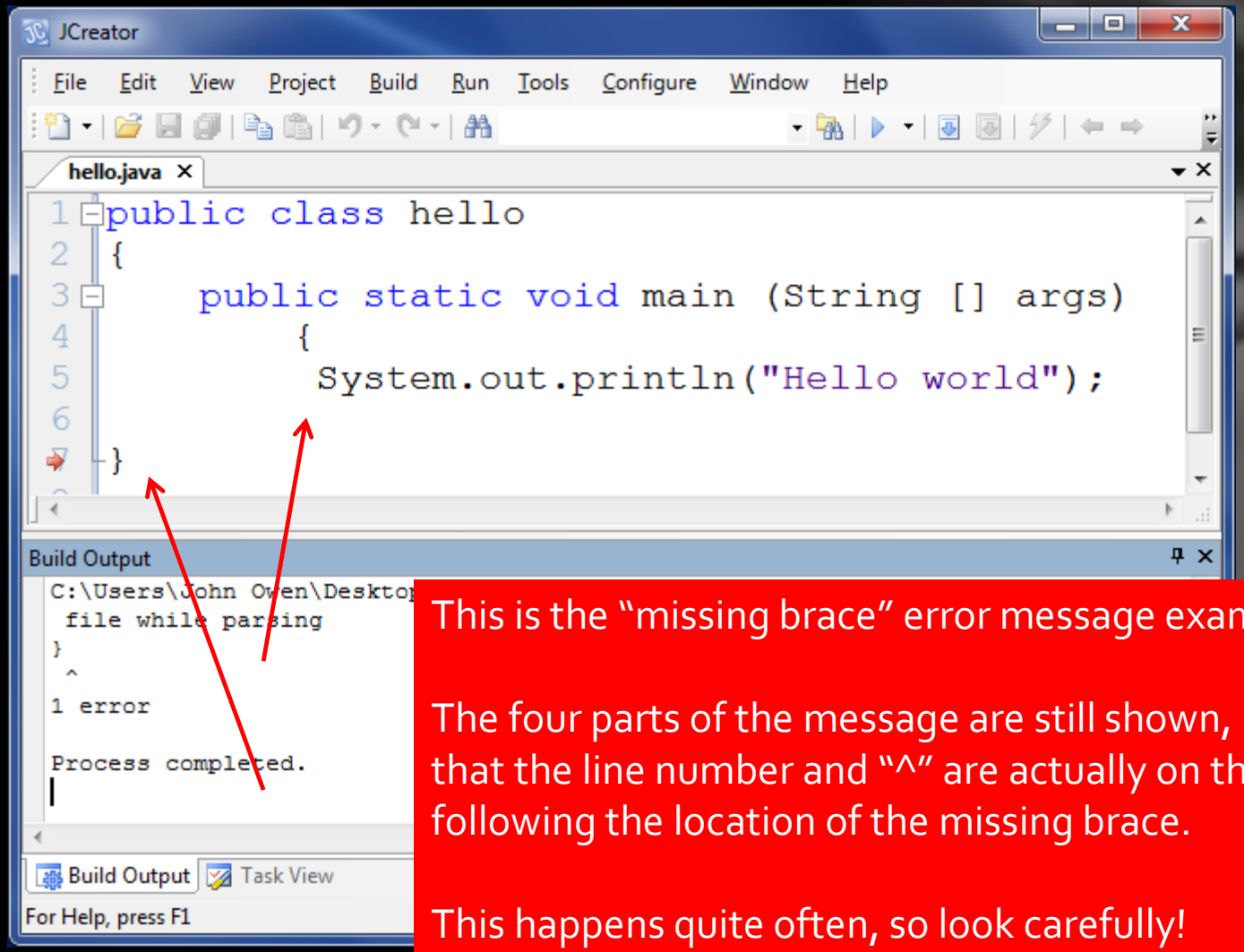
# Error message, Part Four




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




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
```

eneral 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