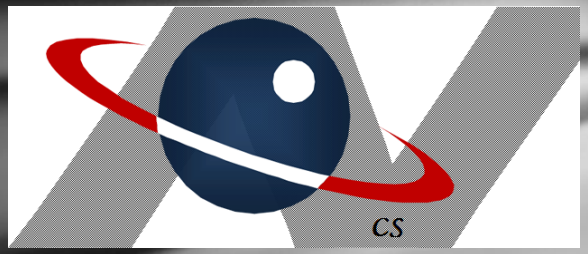
O(N) CS LESSON 1B

Lesson 1B – More Simple Output Techniques using Escape Sequences and Literal Values



By John B. Owen

Table of Contents

Objective #1

- In this lesson the student will further understand the basic JAVA program structure, using the lab program from Lesson 1A.
- CS1 TEKS 126.33c2(A) create and properly display meaningful output;

Objective #2

- The student will understand and apply more output techniques, also using the lab from Lesson 1A, specifically exploring the difference between using and *print* and *println*.
- CS1 TEKS 126.33c2(A) create and properly display meaningful output;

Objective #3

- Student will also understand and apply special escape sequences to further control output, including `"\n"`, `"\t"`, `"\\"`, and `"\""`, using more sample programs in the lesson.
- CS1 TEKS 126.33c2(A) create and properly display meaningful output;

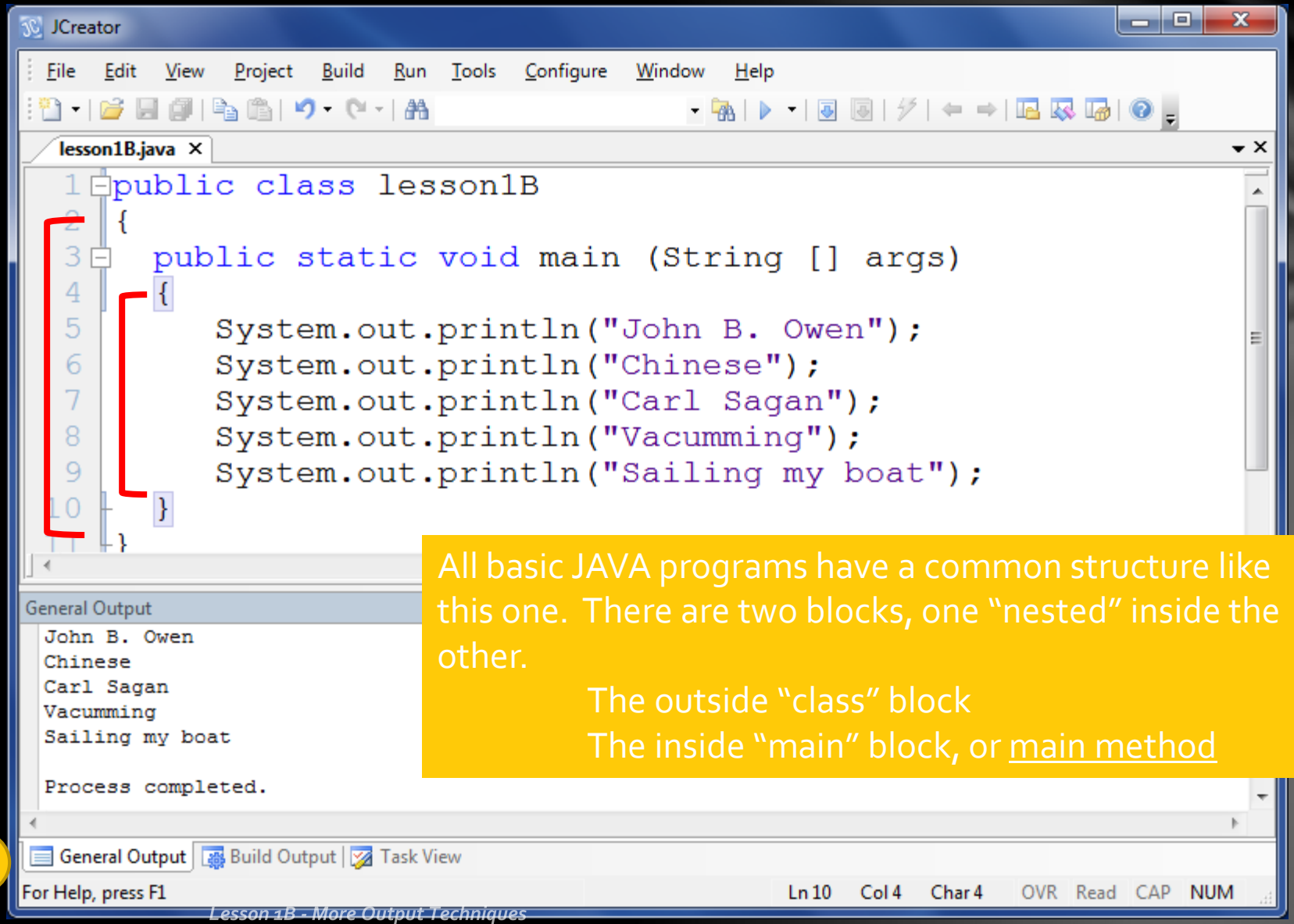
Objective #4

- Finally, the student will understand what "literal values" are, and how to output three basic types:
  - String literals
  - Integer literals
  - Decimal literals
  - CS1 TEKS 126.33c4(D) identify the data types and objects needed to solve a problem;

Program Structure

- The next few slides show and discuss the basic program structure of the typical JAVA program.

Structure of basic JAVA program

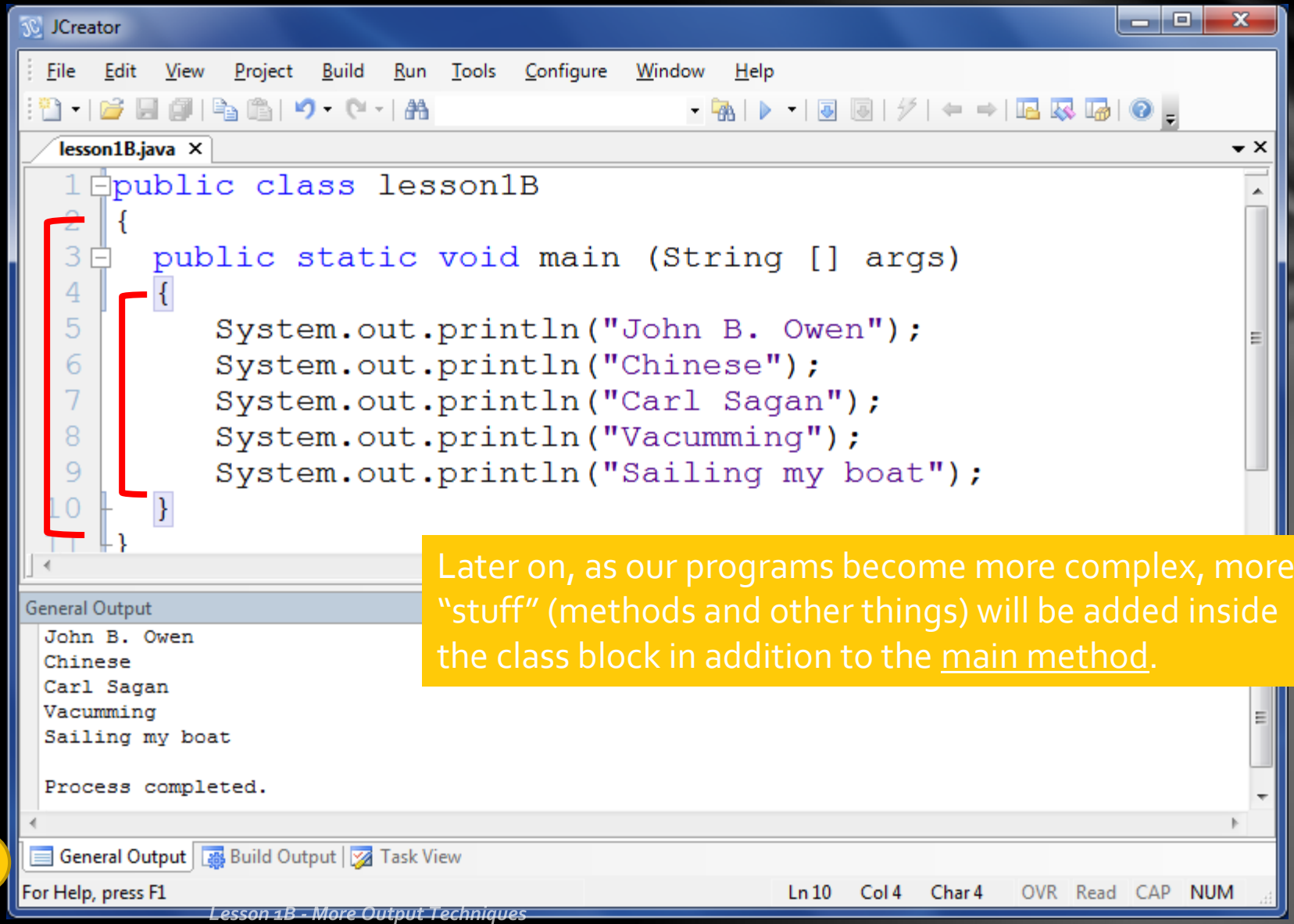


All basic JAVA programs have a common structure like this one. There are two blocks, one "nested" inside the other.

The outside "class" block

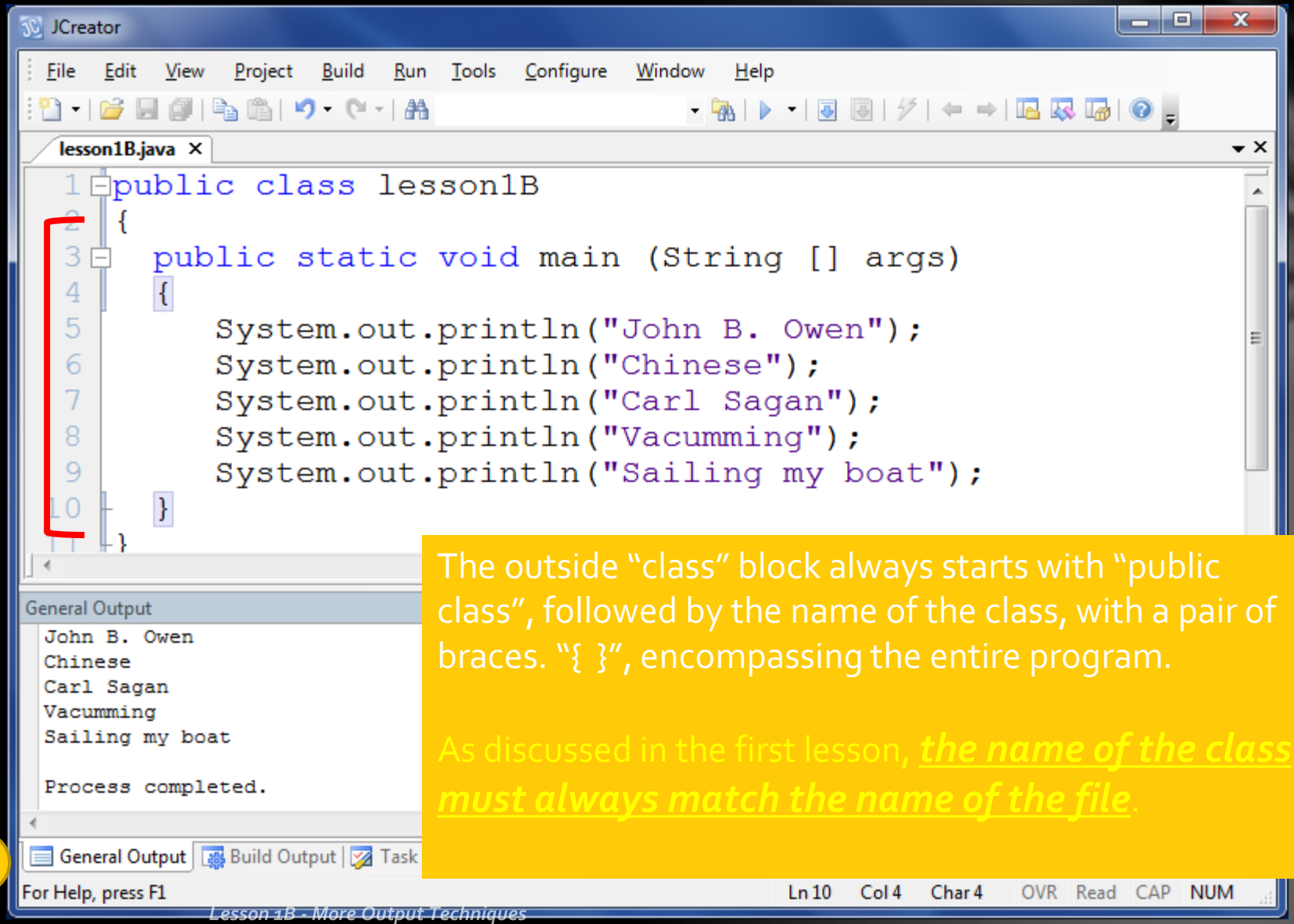
The inside "main" block, or main method

Structure of basic JAVA program



Later on, as our programs become more complex, more "stuff" (methods and other things) will be added inside the class block in addition to the main method.

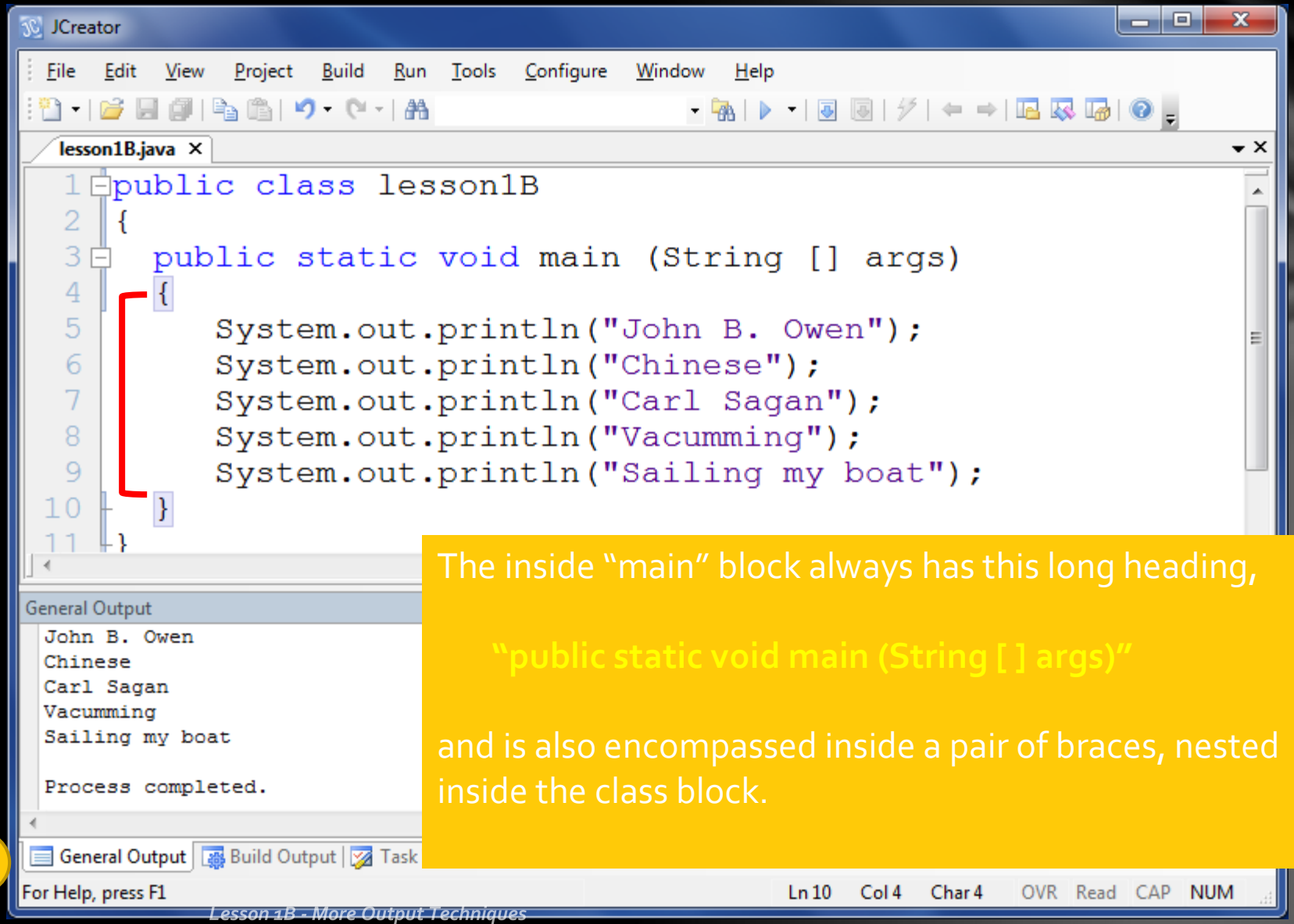
Structure of basic JAVA program



The outside "class" block always starts with "public class", followed by the name of the class, with a pair of braces. "{ }", encompassing the entire program.

As discussed in the first lesson, ***the name of the class must always match the name of the file***.

Structure of basic JAVA program

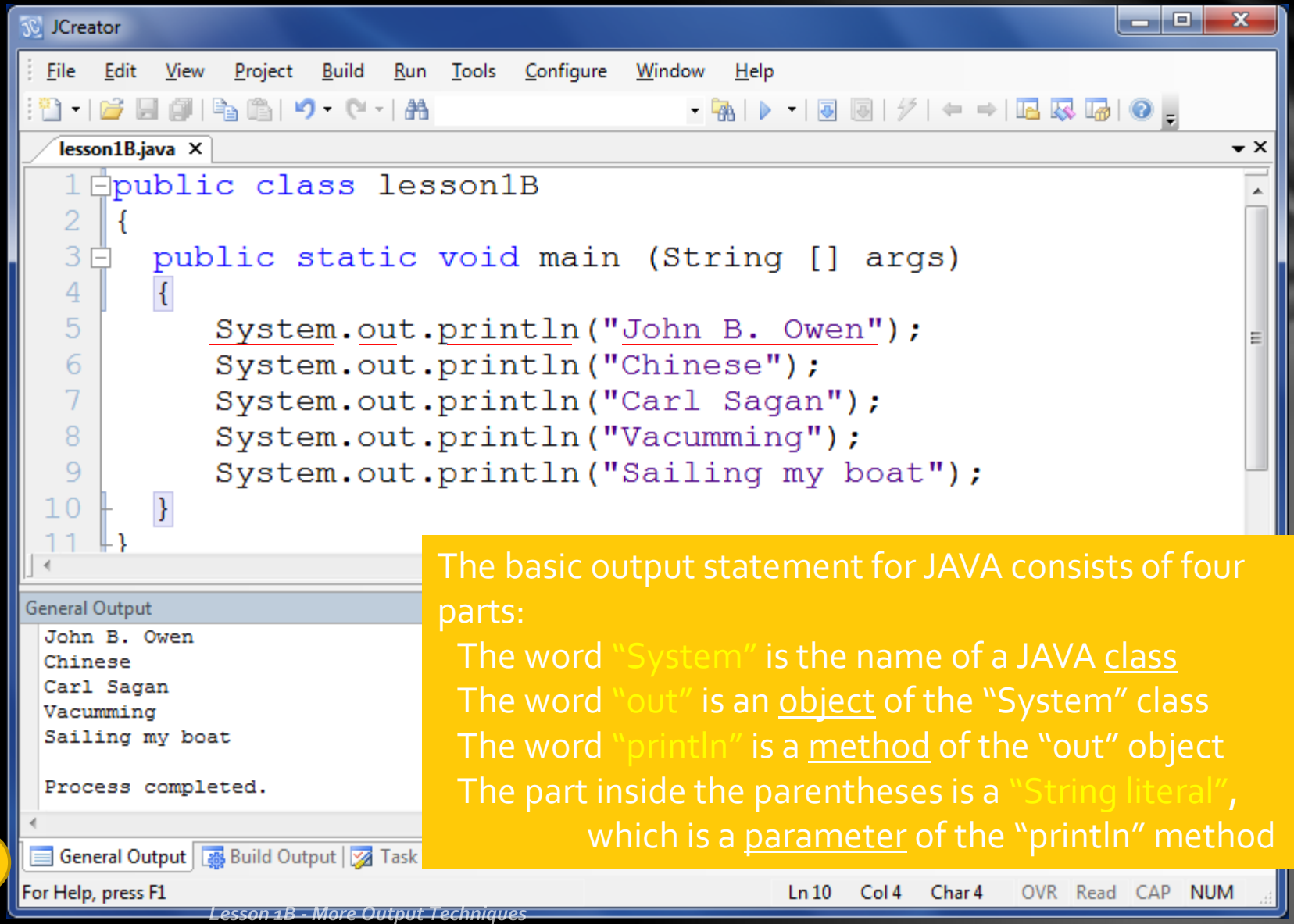


The inside "main" block always has this long heading,

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

and is also encompassed inside a pair of braces, nested inside the class block.

Output statement – four parts



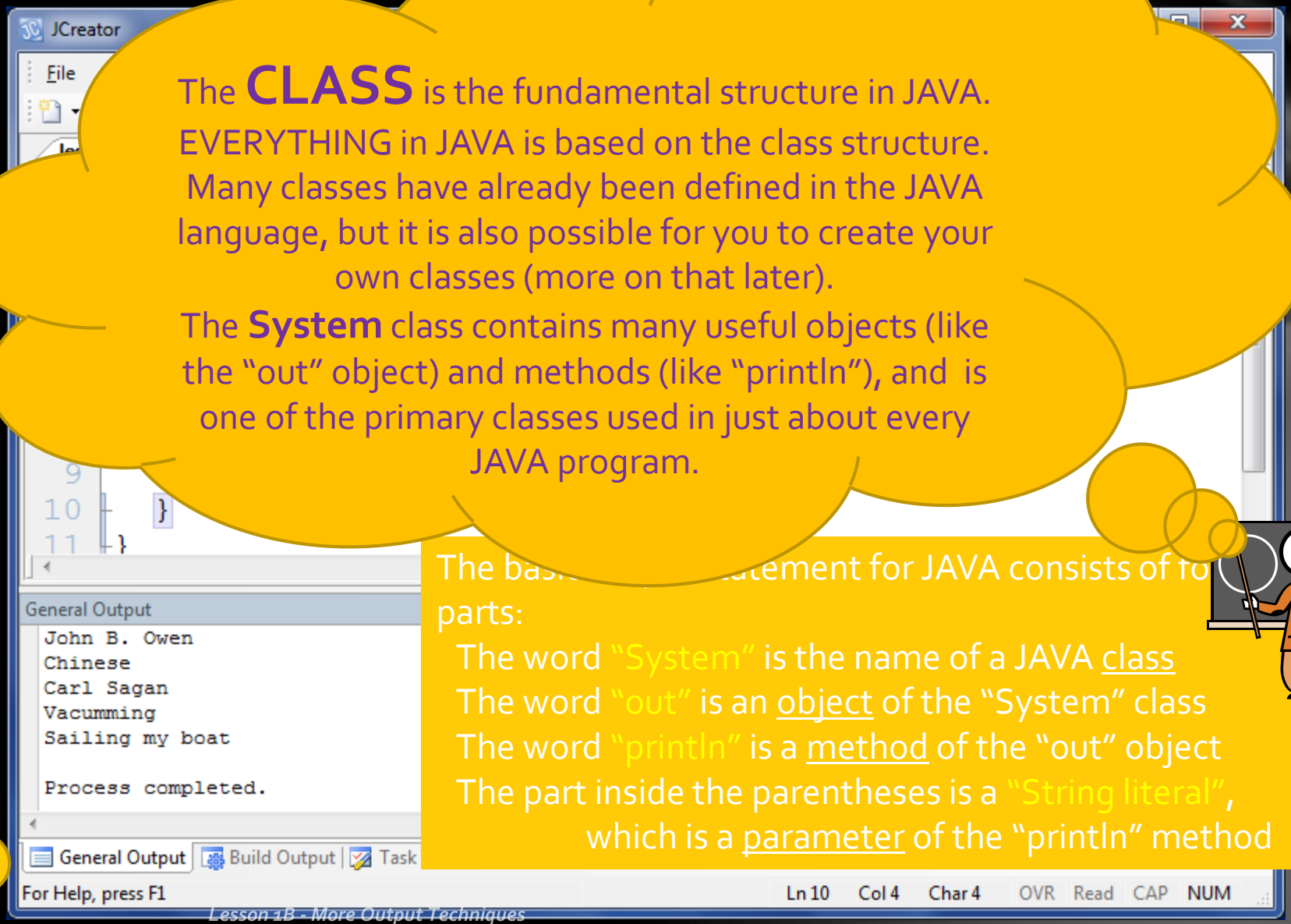
The basic output statement for JAVA consists of four parts:

- The word "System" is the name of a JAVA class
- The word "out" is an object of the "System" class
- The word "println" is a method of the "out" object
- The part inside the parentheses is a "String literal", which is a parameter of the "println" method

Output statement – fo[illegible] part

The **CLASS** is the fundamental structure in JAVA. EVERYTHING in JAVA is based on the class structure. Many classes have already been defined in the JAVA language, but it is also possible for you to create your own classes (more on that later).

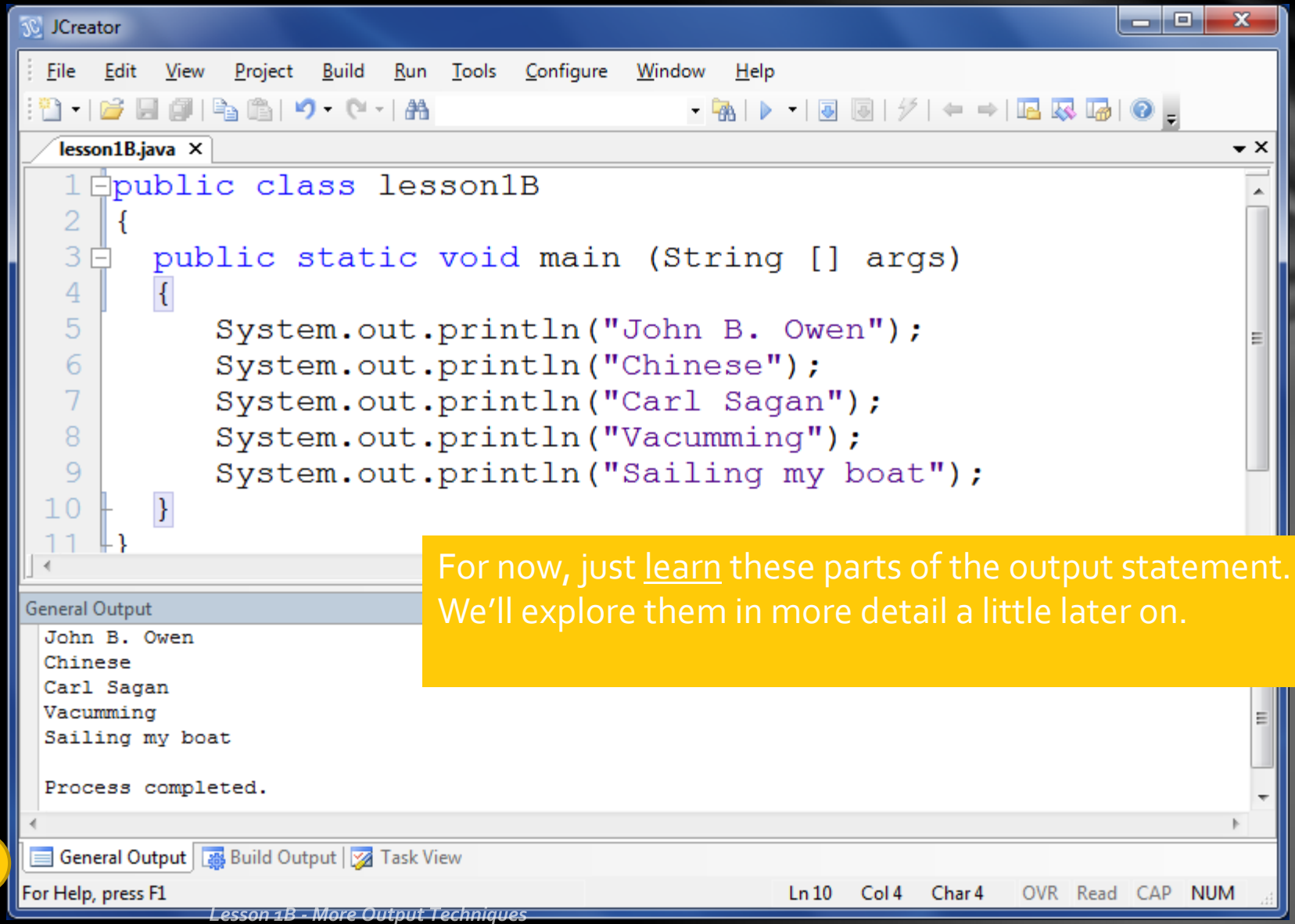
The **System** class contains many useful objects (like the "out" object) and methods (like "println"), and is one of the primary classes used in just about every JAVA program.



The bas[illegible]tement for JAVA consists of fo[illegible] parts:

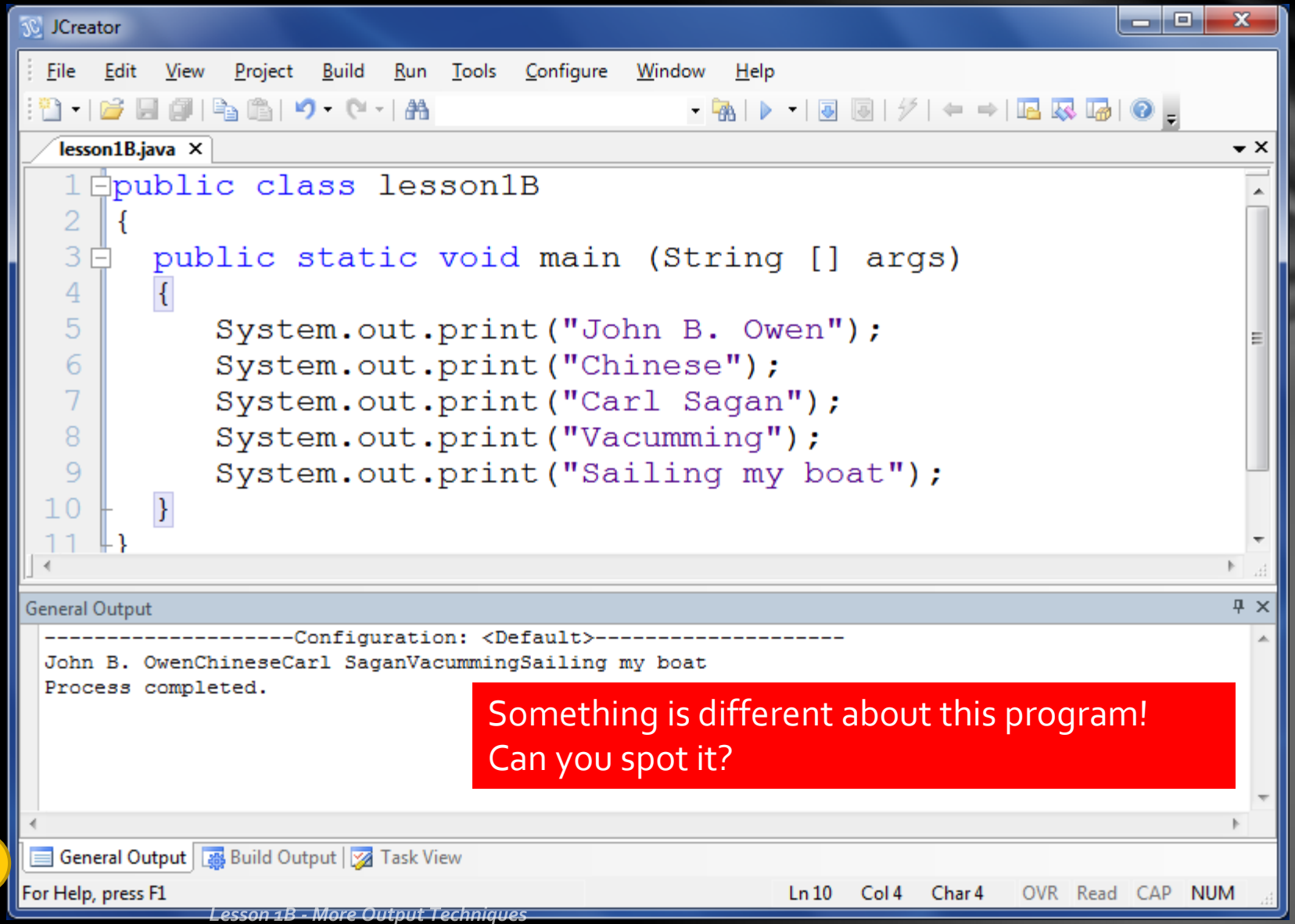
- The word "System" is the name of a JAVA class
- The word "out" is an object of the "System" class
- The word "println" is a method of the "out" object
- The part inside the parentheses is a "String literal", which is a parameter of the "println" method

Output statement – four parts

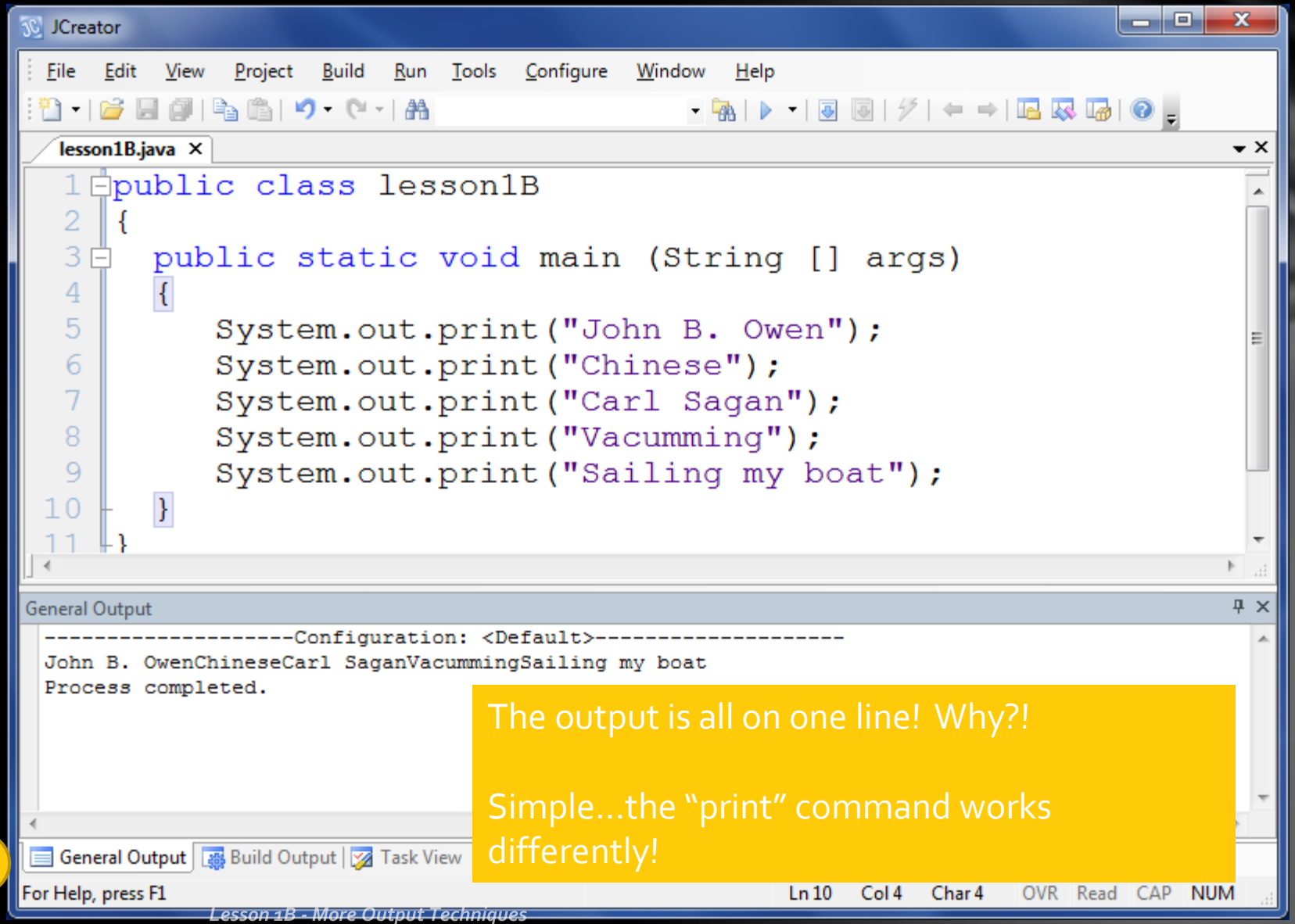


For now, just learn these parts of the output statement. We'll explore them in more detail a little later on.

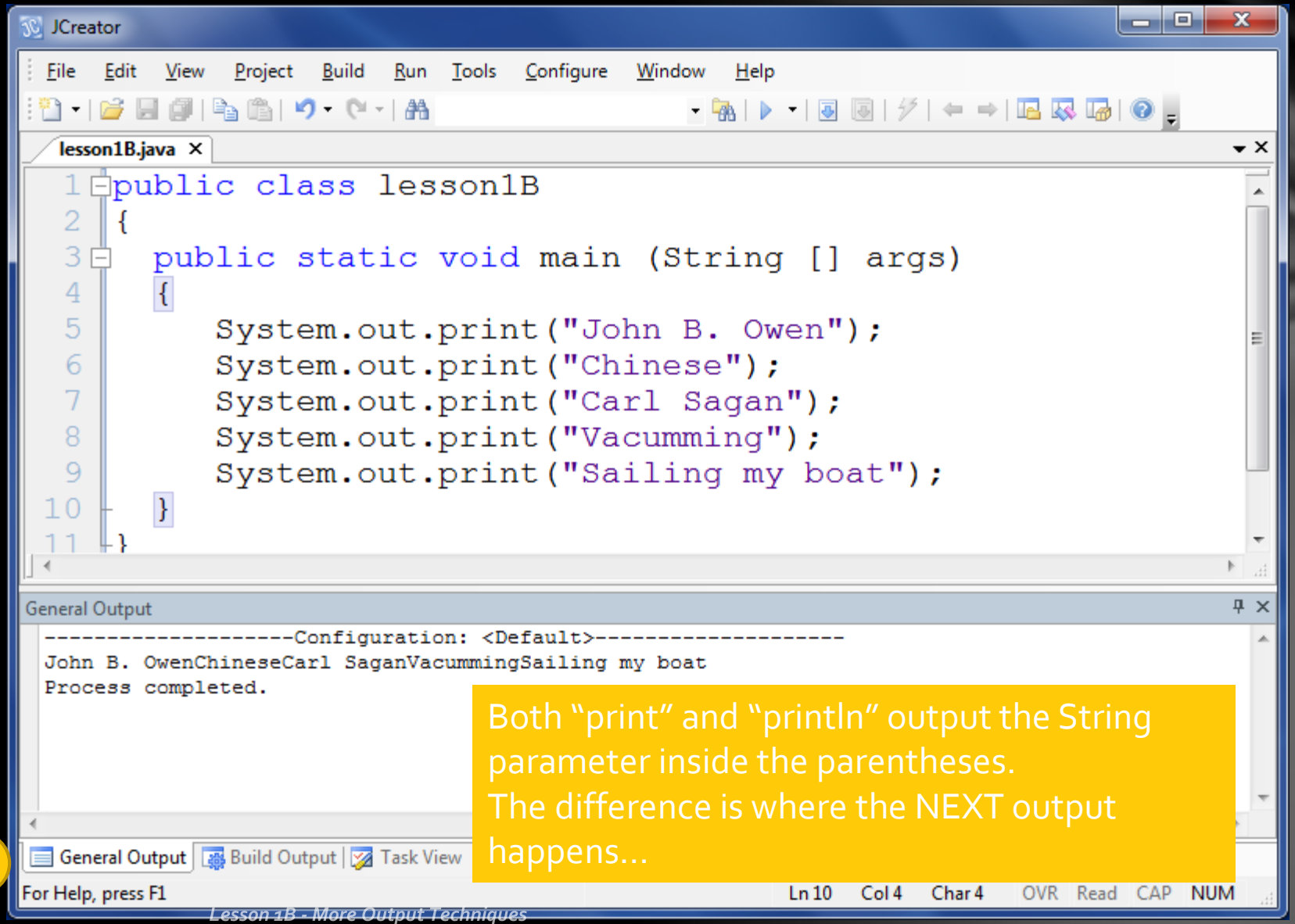
System.out.print



System.out.print

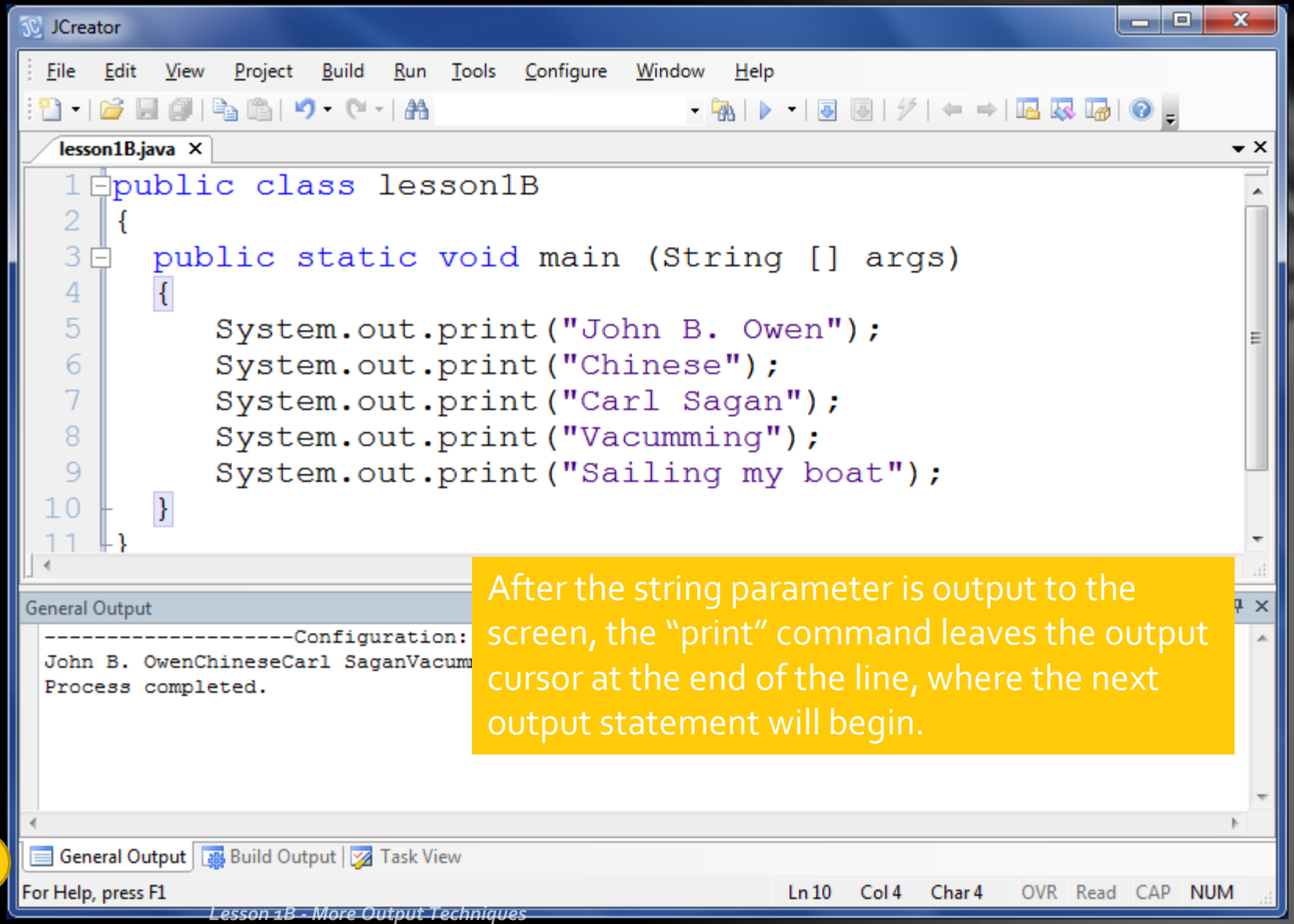


System.out.print



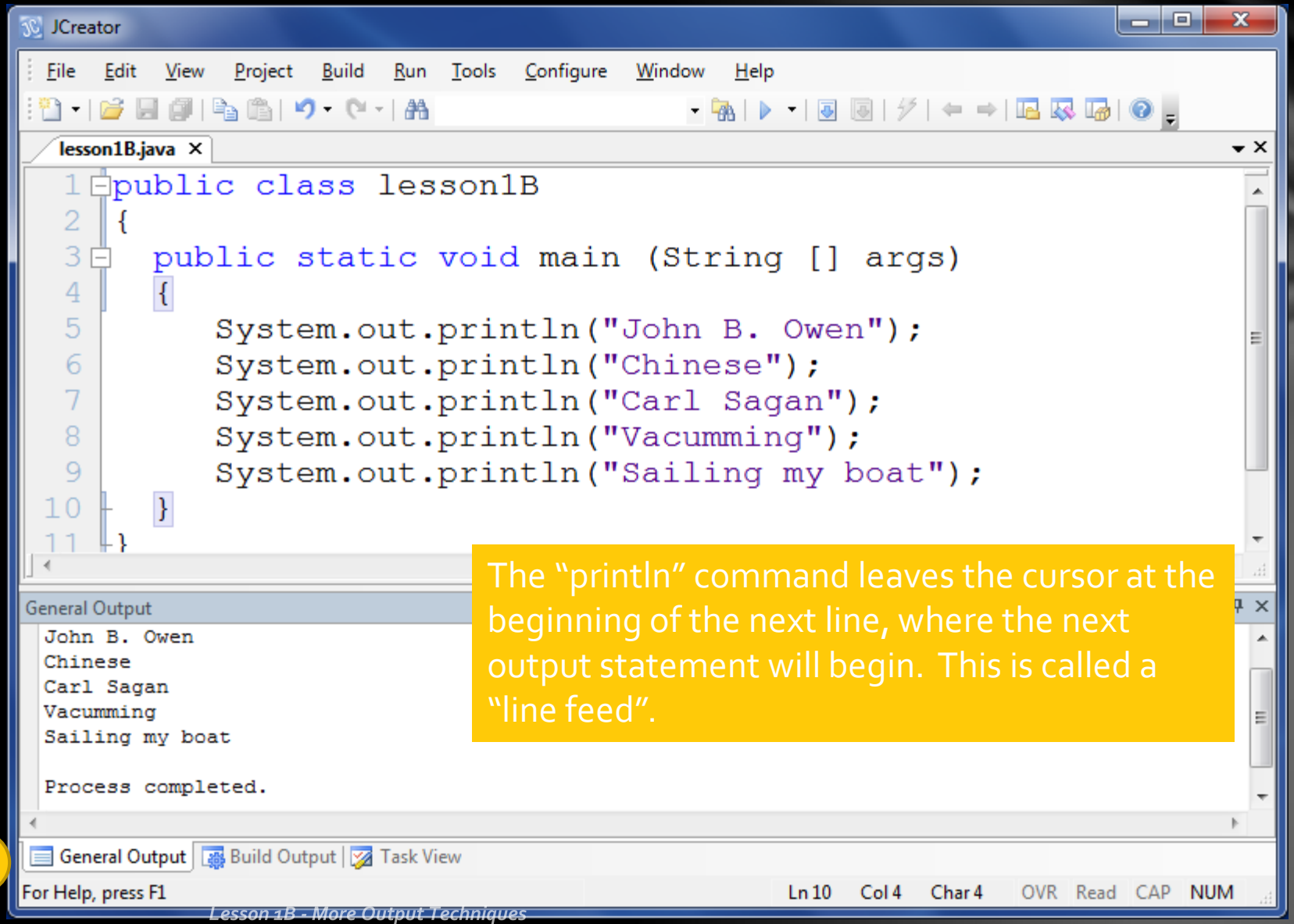
Both "print" and "println" output the String parameter inside the parentheses.
The difference is where the NEXT output happens…

System.out.print



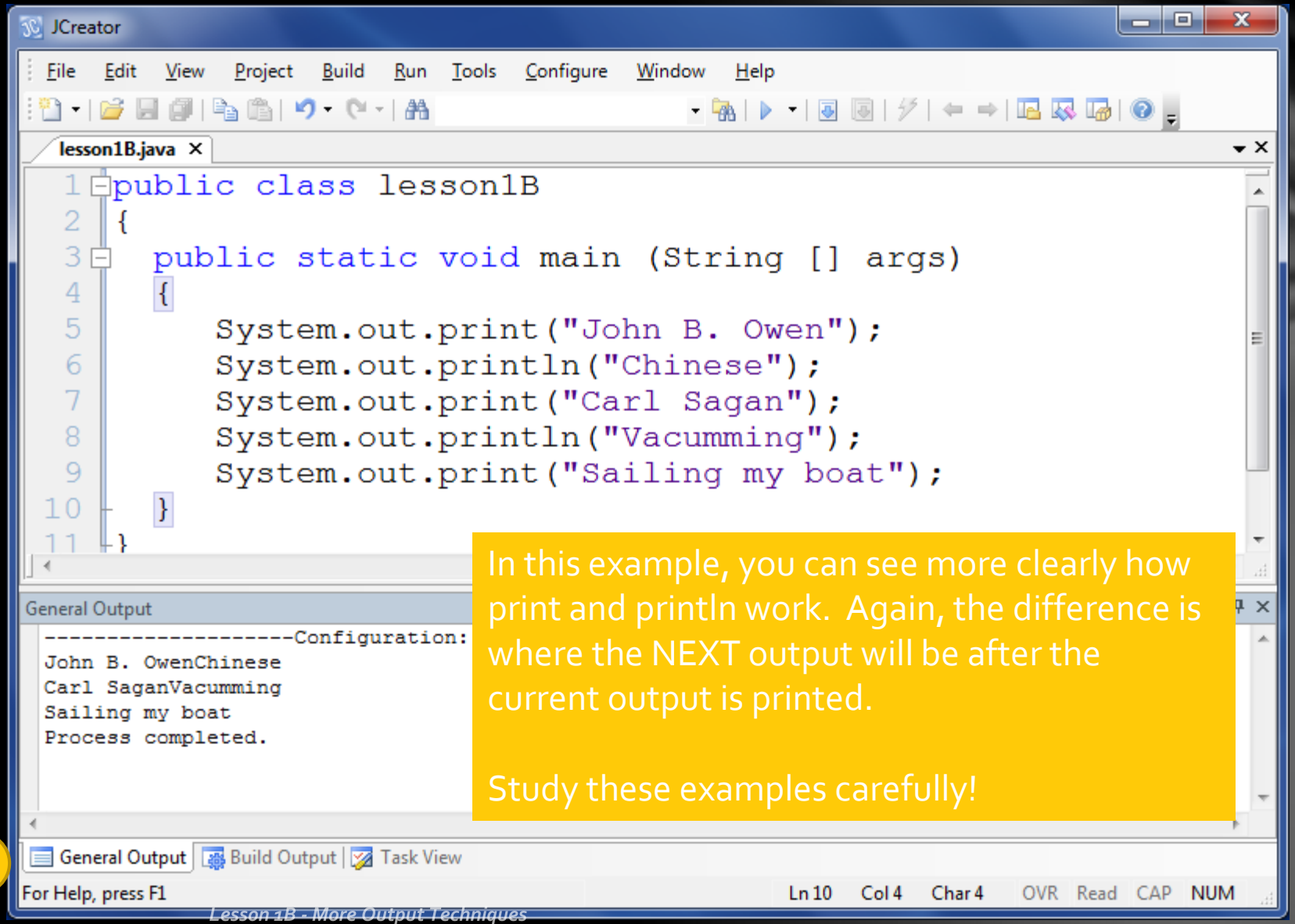
After the string parameter is output to the screen, the "print" command leaves the output cursor at the end of the line, where the next output statement will begin.

System.out.println



The "println" command leaves the cursor at the beginning of the next line, where the next output statement will begin. This is called a "line feed".

Mixing it up!



In this example, you can see more clearly how print and println work. Again, the difference is where the NEXT output will be after the current output is printed.

Study these examples carefully!

Escape Sequences

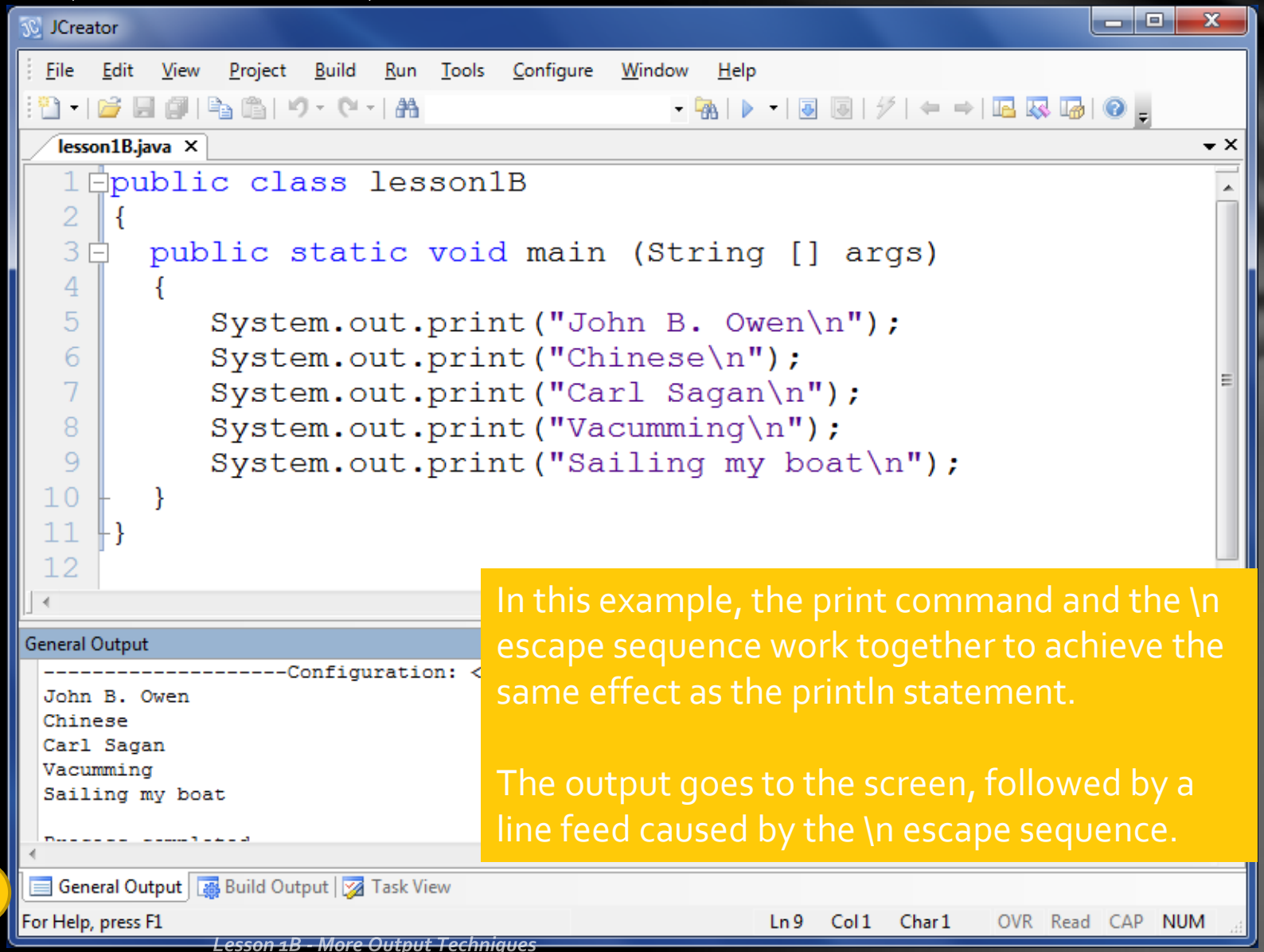
- There are special things called "escape sequences" that can be imbedded inside the string parameter of the print or println commands.
- These help provide extra ways to format, or control, the output of these commands.

Escape Sequences

- The most commonly used ones are:
  - \n – creates a line feed
  - \t – creates a tab jump
  - \\ – outputs the \ character
  - \" – outputs the " character

Let's look at some examples…

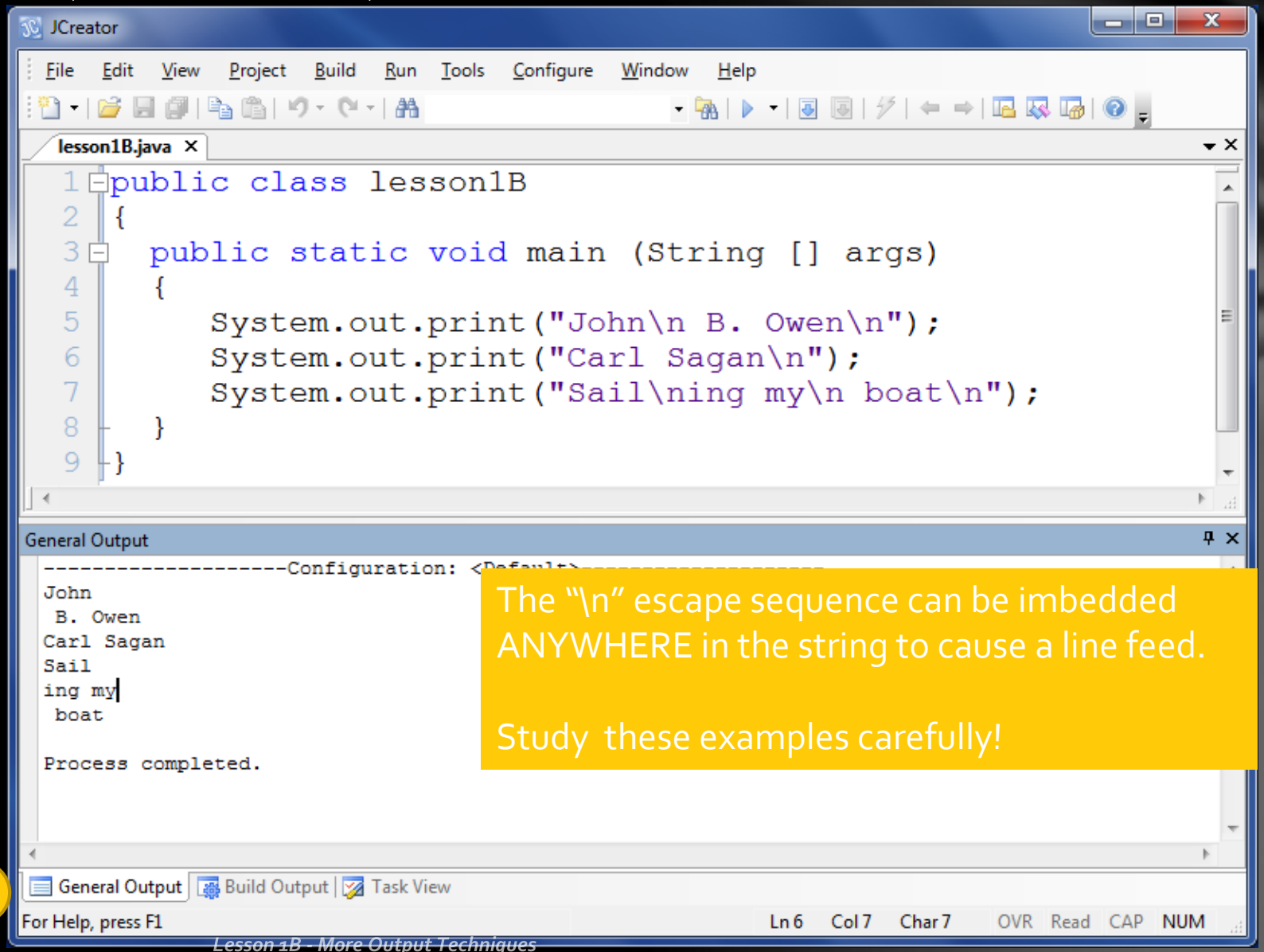
\n (linefeed)



In this example, the print command and the \n escape sequence work together to achieve the same effect as the println statement.

The output goes to the screen, followed by a line feed caused by the \n escape sequence.

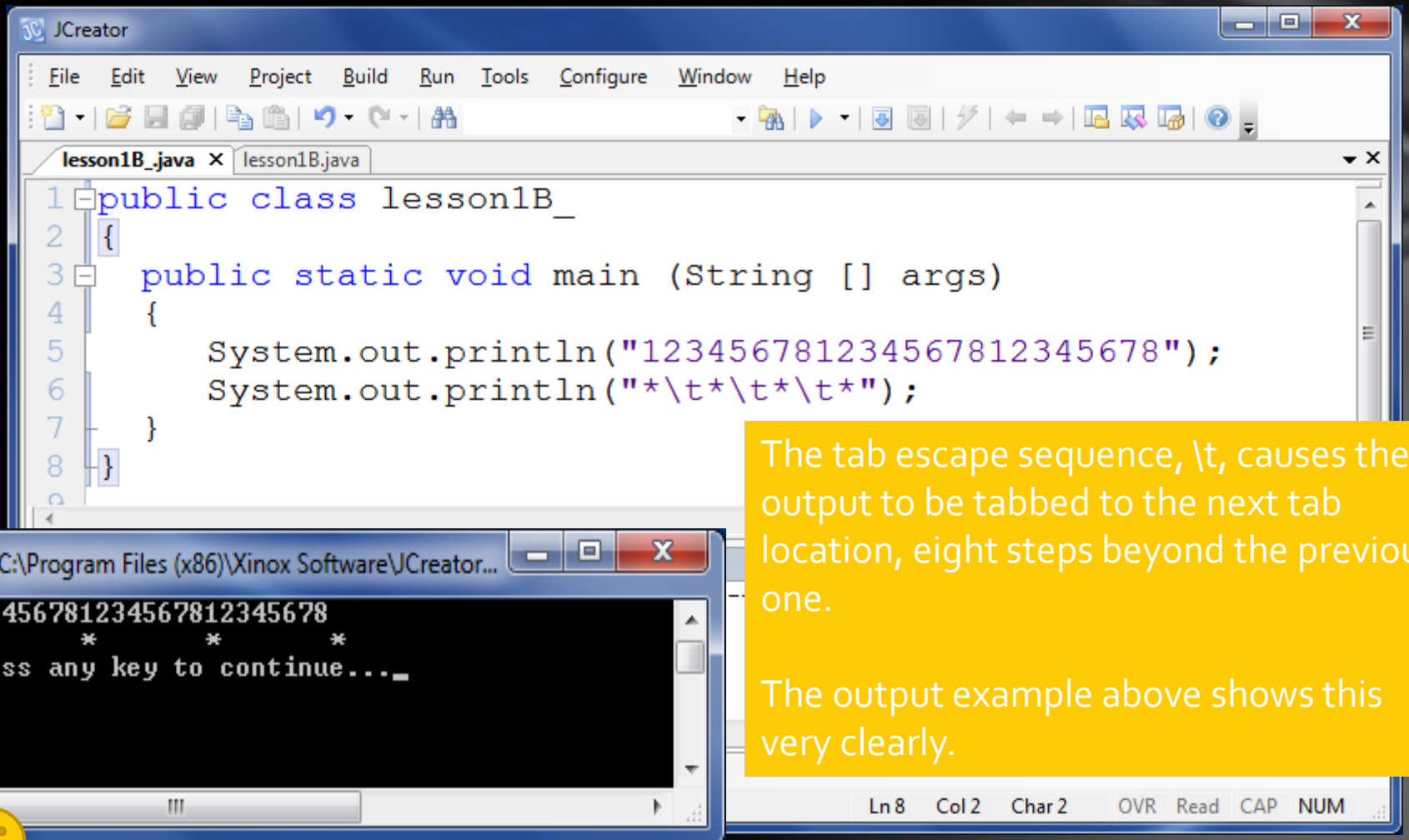
\n (linefeed)



The "\n" escape sequence can be imbedded ANYWHERE in the string to cause a line feed.

Study these examples carefully!

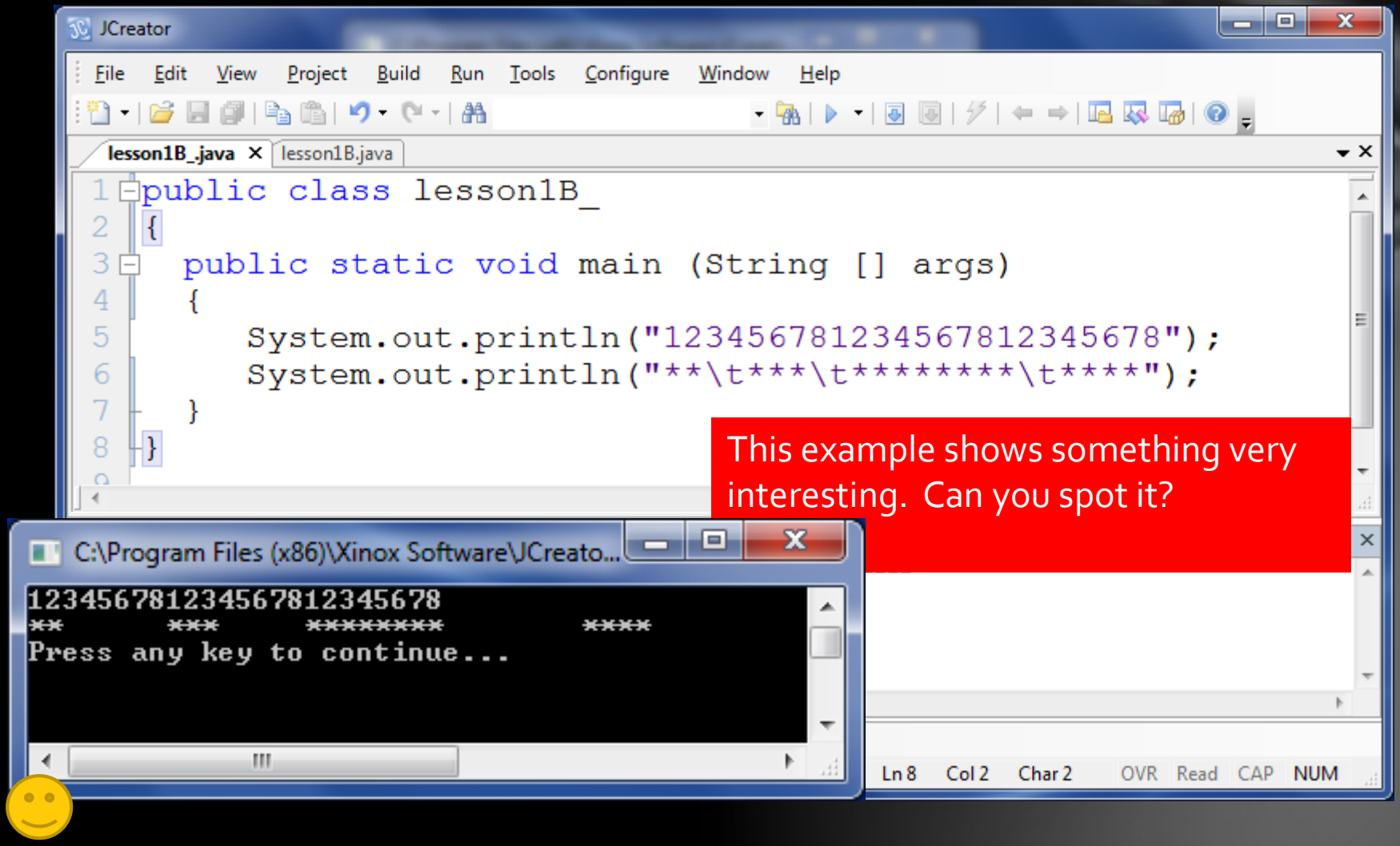
\t (tab)



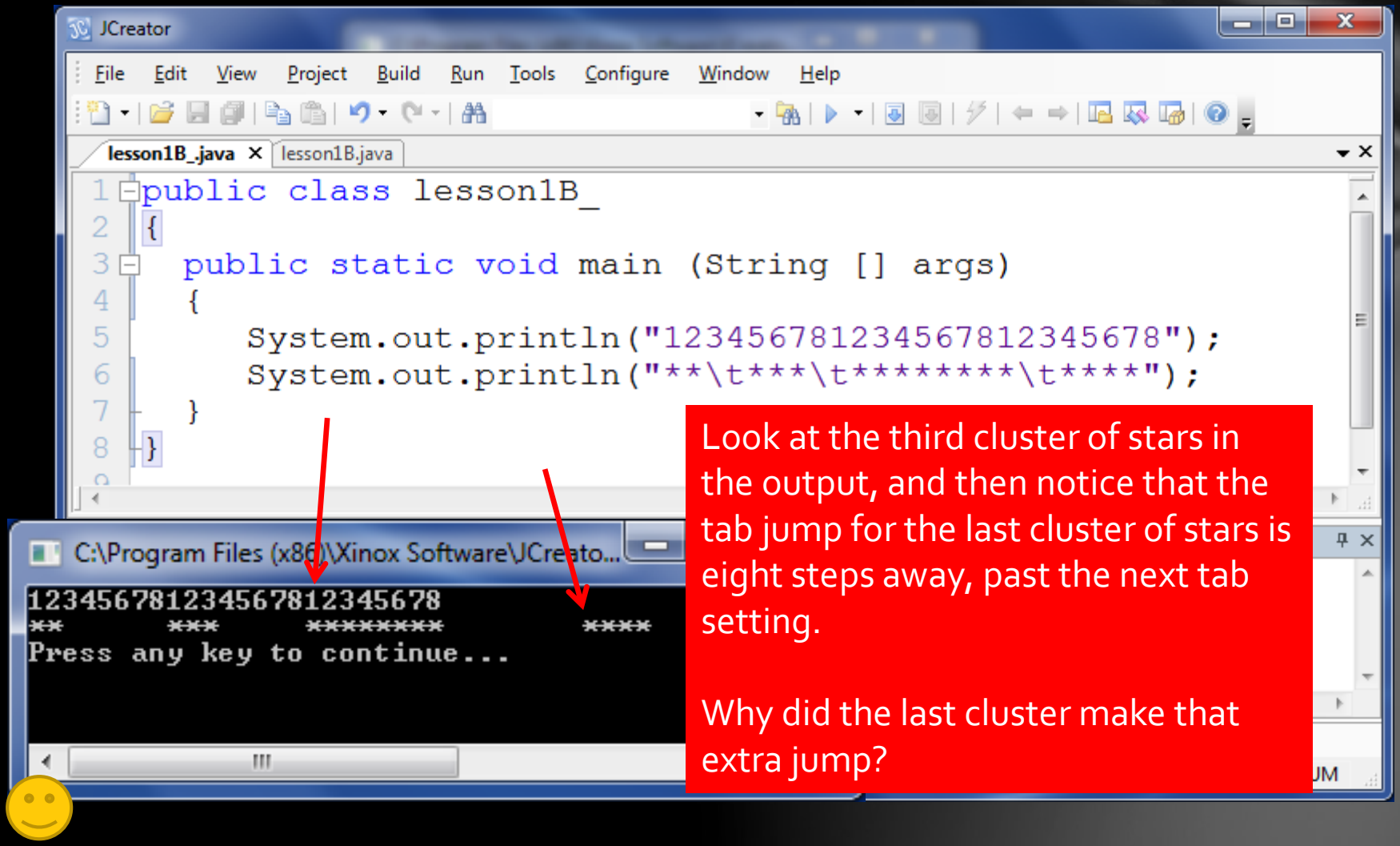
The tab escape sequence, \t, causes the output to be tabbed to the next tab location, eight steps beyond the previous one.

The output example above shows this very clearly.

\t (tab)


JCreator
File Edit View Project Build Run Tools Configure Window Help
lesson1B_.java
lesson1B.java
1 public class lesson1B_
2 {
3 public static void main (String [] args)
4 {
5 System.out.println("12345678123456781234567 8");
6 System.out.println("**\t***\t********\t****");
7 }
8 }
This example shows something very interesting. Can you spot it?
C:\Program Files (x86)\Xinox Software\JCreato...
12345678123456781234567 8
** *** ********* ****
Press any key to continue...
Ln 8 Col 2 Char 2 OVR Read CAP NUM

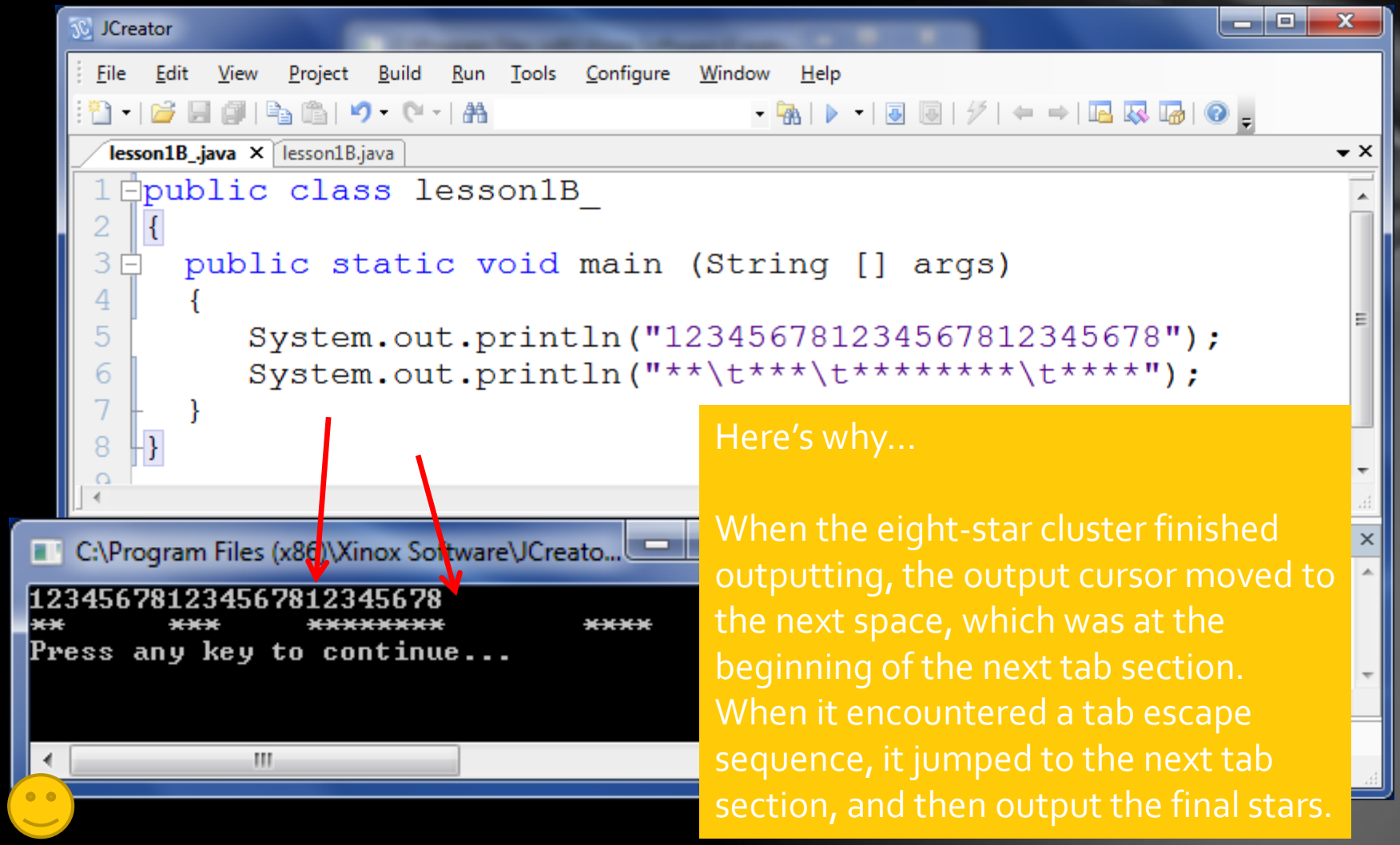
\t (tab)



Look at the third cluster of stars in the output, and then notice that the tab jump for the last cluster of stars is eight steps away, past the next tab setting.

Why did the last cluster make that extra jump?

\t (tab)



Here's why…

When the eight-star cluster finished outputting, the output cursor moved to the next space, which was at the beginning of the next tab section. When it encountered a tab escape sequence, it jumped to the next tab section, and then output the final stars.

\\ and \”

- These last two escape sequences are special cases
  - “\\” – outputs the \ character
  - “\”” – outputs the “ character

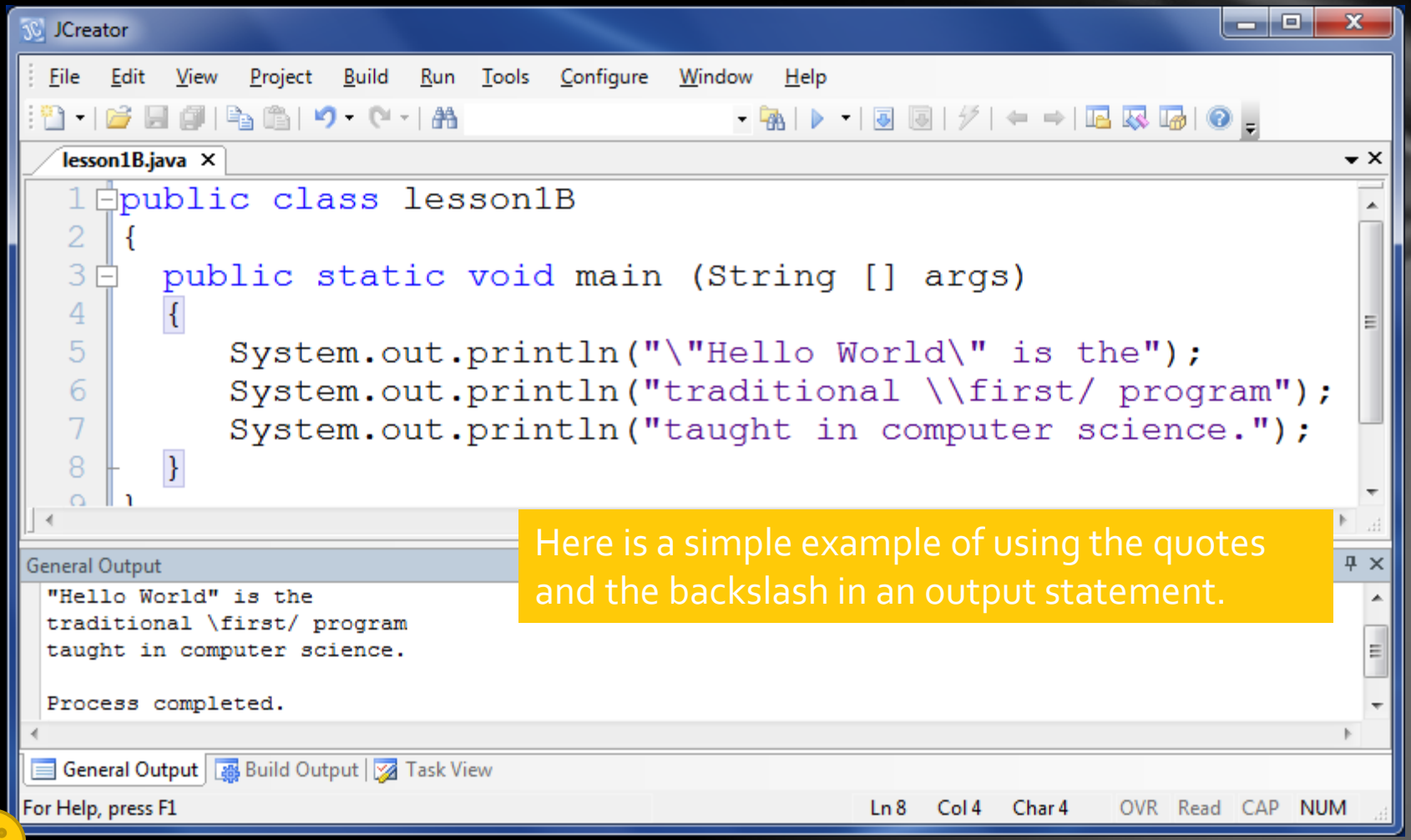
Since the \ (backslash) is used as the first character of every escape sequence, it cannot be used normally as an actual output character

\\ and \"

The same holds true for the " (double quote). It is used as the *delimiter*, or boundary character, for string literals, and cannot be used normally in output statements.

To include either the \ or " as actual characters that appear in program output, these escape sequences must be used.

\\ and \”

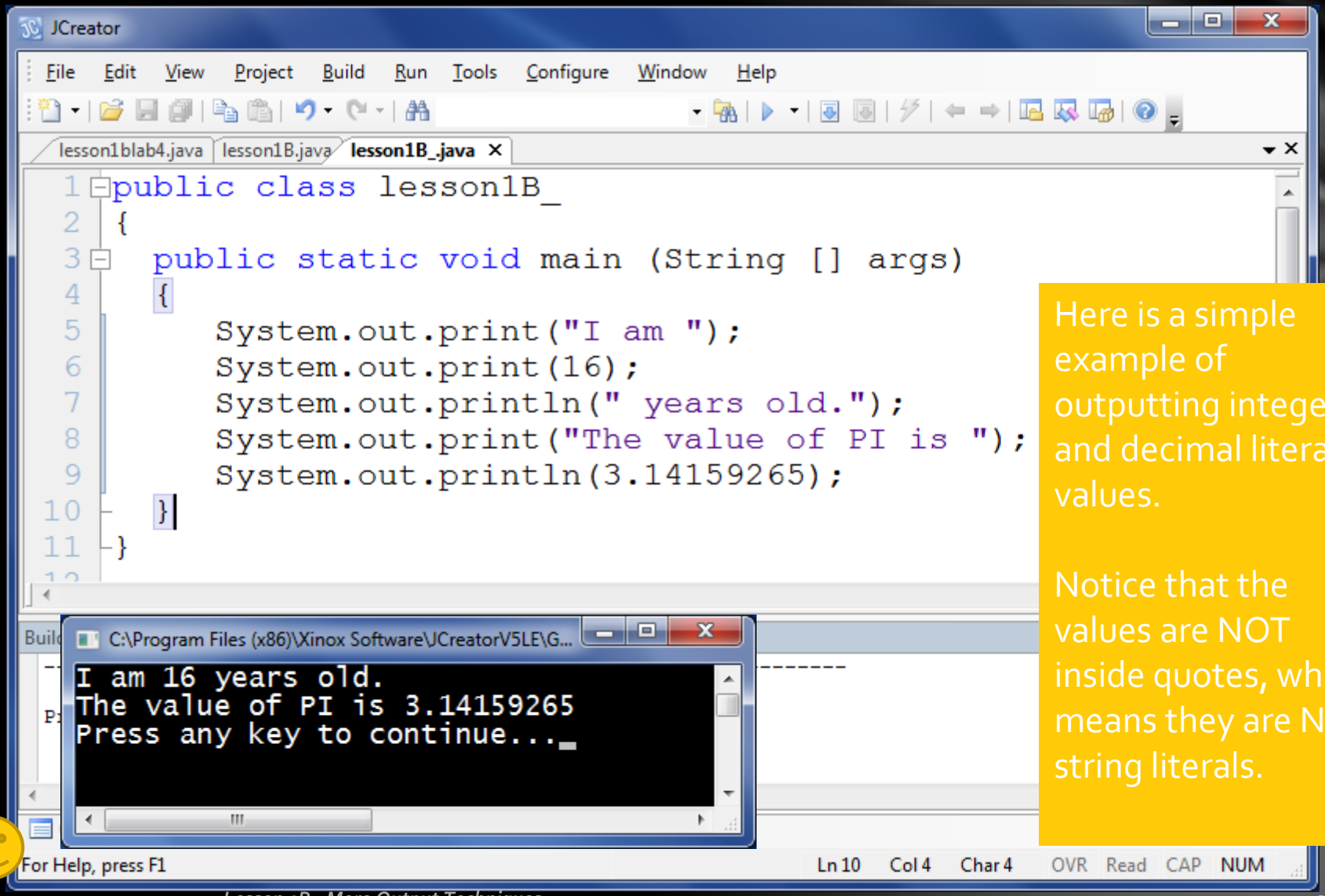


Here is a simple example of using the quotes and the backslash in an output statement.

Data literals – string, integer, decimal

- So far in these first two lessons, every output parameter for the **System.out.println** command has been a String literal, or *a string of characters inside quotes*
- It is also possible to use integer literals and decimal literals as parameters for this command

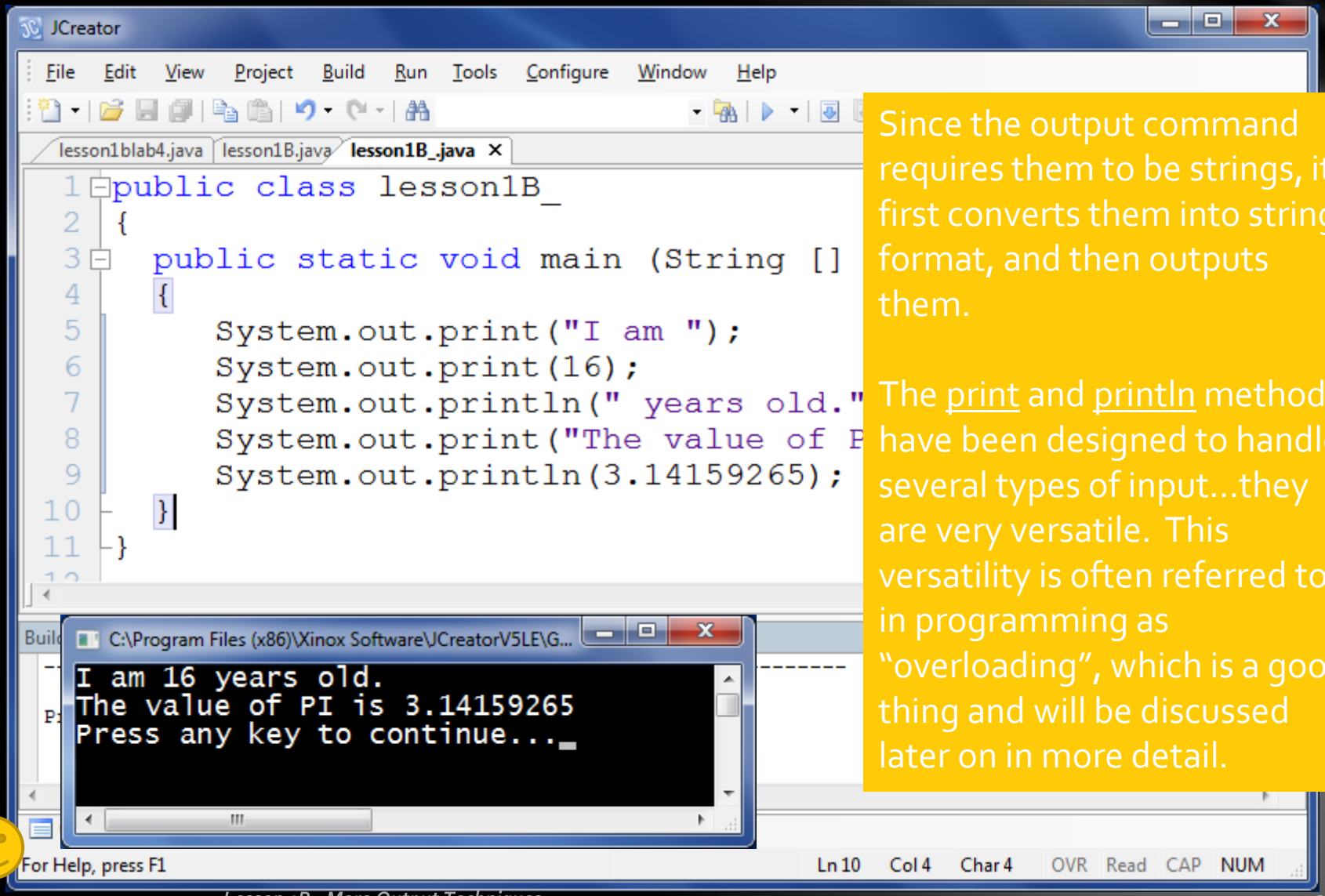
Outputting integer and decimal literals



Here is a simple example of outputting integer and decimal literal values.

Notice that the values are NOT inside quotes, which means they are NOT string literals.

Outputting integer and decimal literals



Since the output command requires them to be strings, it first converts them into string format, and then outputs them.

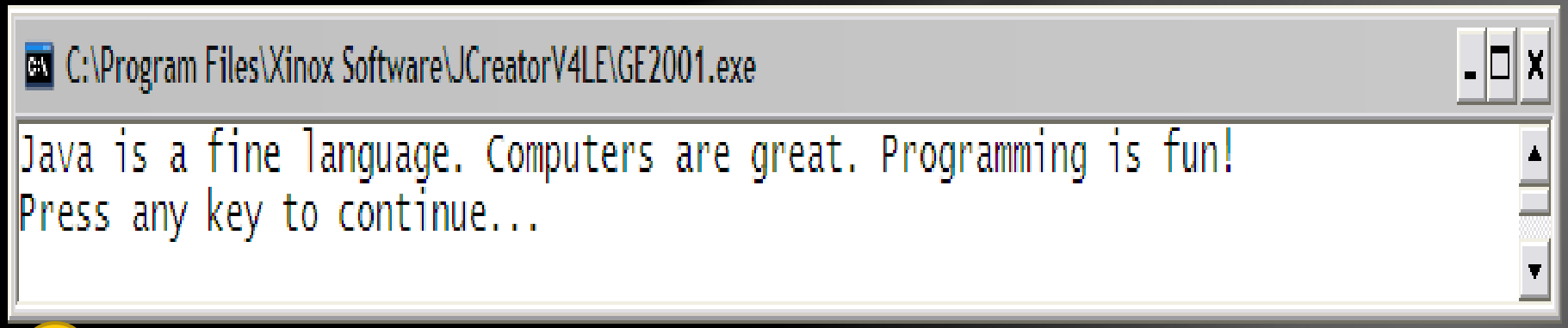
The print and println methods have been designed to handle several types of input…they are very versatile. This versatility is often referred to in programming as "overloading", which is a good thing and will be discussed later on in more detail.

Lesson Summary

- This lesson discussed the basic program structure, showing how the main block is always nested inside the class block.
- It introduced the *print* command, which leaves the output cursor immediately after it, unlike the *println* command causes a line feed after it.
- It also introduced four escape sequences: \n, \t, \\, and \"
- Finally it showed that in addition to string literals, integer and decimal literals can also be output using the System.out.println command.

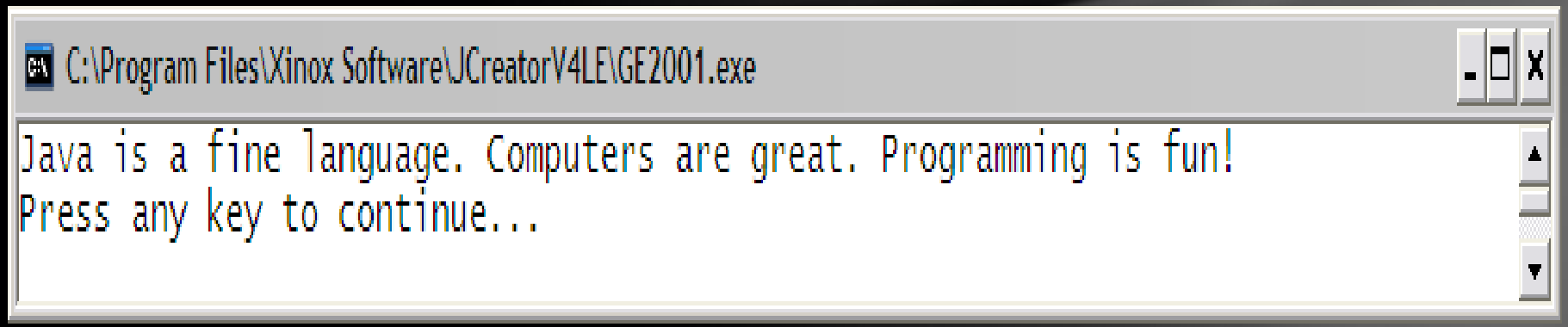
Labs for Lesson 1B

- For this lesson you will do six very easy labs.
  - Lab 1B-1 Create a program that produces the output below using only one output statement.



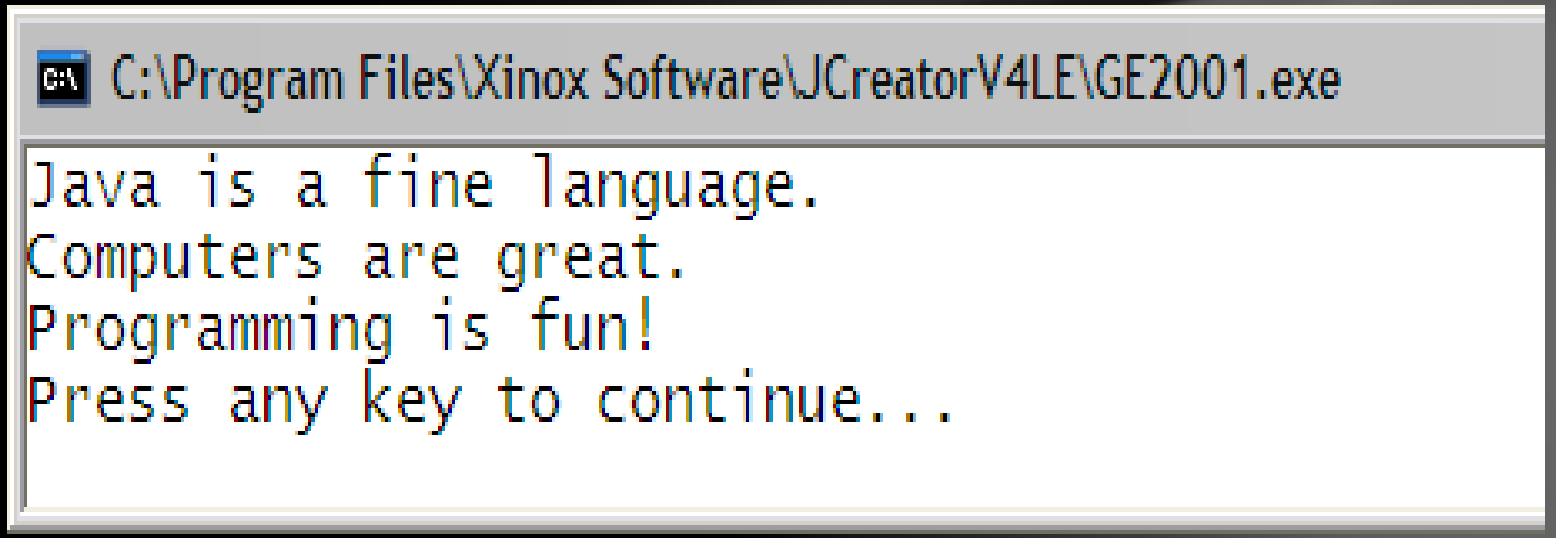
Labs for Lesson 1B

- Lab 1B-2 Create a program that produces the output below using three separate output statements, one for each sentence.



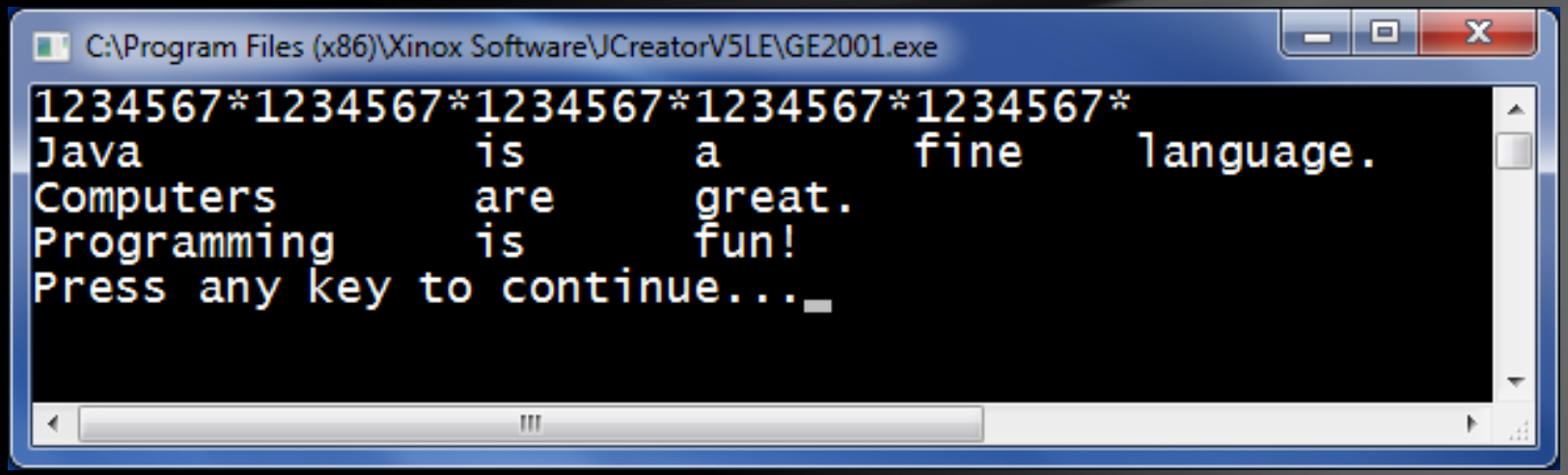
Labs for Lesson 1B

- Lab 1B-3 Create a program that produces the output below using ONLY ONE output statement.



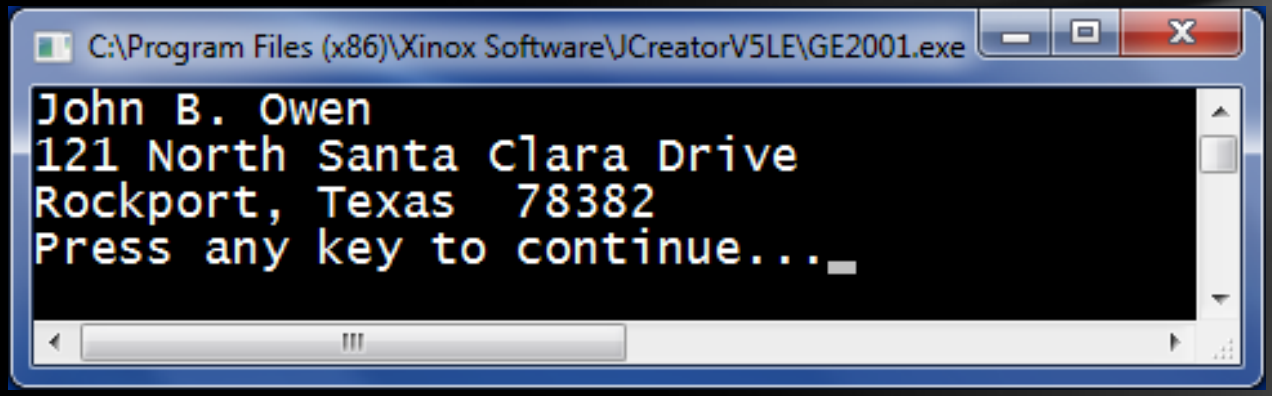
Labs for Lesson 1B

- Lab 1B-4 Create a program that produces the EXACT output below, complete with the "metric" line at the top. *Hint: Use tabs.*



Labs for Lesson 1B

- Lab 1B-5 Create a program that outputs your address on three lines. However, the numbers must be output as integer literals, not strings!

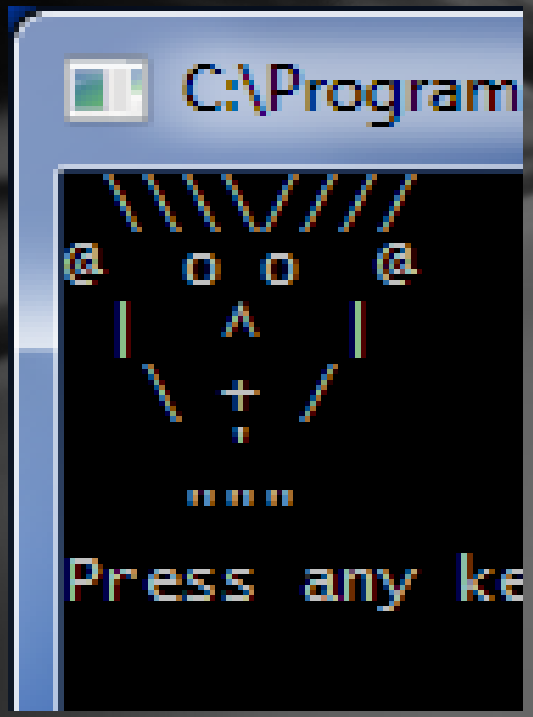


This should help you get started...

```
System.out.println("John B. Owen");
System.out.print(121);
System.out.println(" North Santa Cla
```

Labs for Lesson 1B

- Lab 1B-6 Write a program that outputs a "face" similar to the one shown, but of your own creation. It must contain at least one backslash character, and at least one double-quote character.



CONGRATULATIONS!

- You have just learned some more cool stuff about JAVA, including how to use escape sequences, the difference between print and println, and about three different data literals – Strings, integers, and decimals.
- *Now go on to Lesson 1C*

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