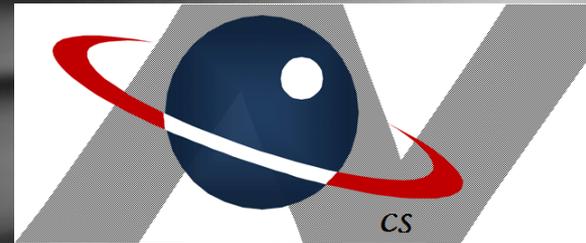


$O(N)$ CS LESSON 1B

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



By John B. Owen

All rights reserved

©2011, revised 2015

Table of Contents



- [Objectives](#)
- [Program Structure](#)
- [Output statement – four parts](#)
- [System.out.print and println](#)
- [Escape Sequences](#)
- [Outputting Data literals](#)
- [Lesson Summary / Labs](#)
- [Contact Information for supplementary materials](#)

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;

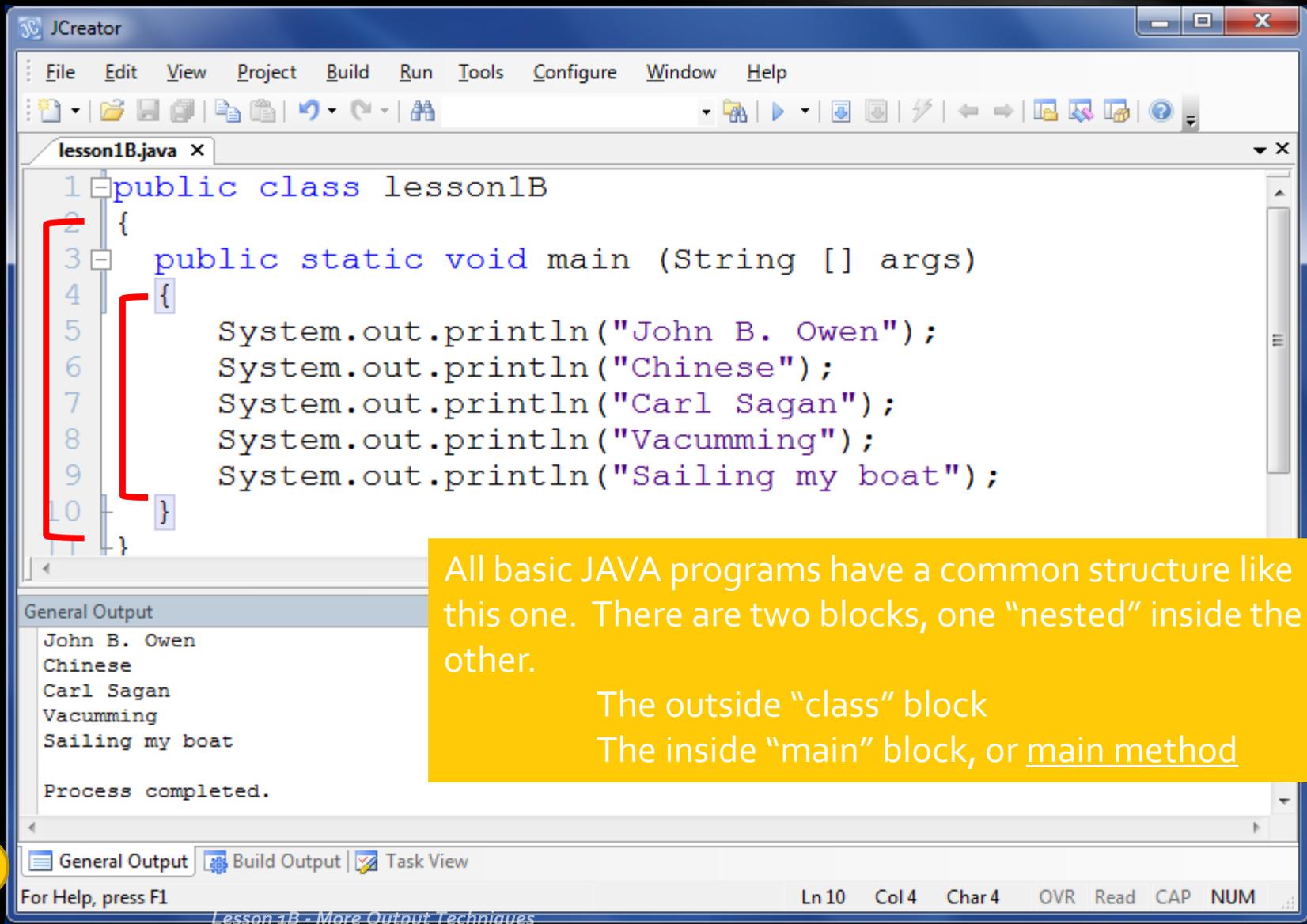


Program Structure

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



Structure of basic JAVA program



The screenshot shows the JCreator IDE with a Java file named 'lesson1B.java'. The code is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacumming");
9         System.out.println("Sailing my boat");
10    }
11 }
```

Red brackets highlight the 'main' method block (lines 3-10) and the 'class' block (lines 1-11). The 'General Output' window at the bottom shows the following text:

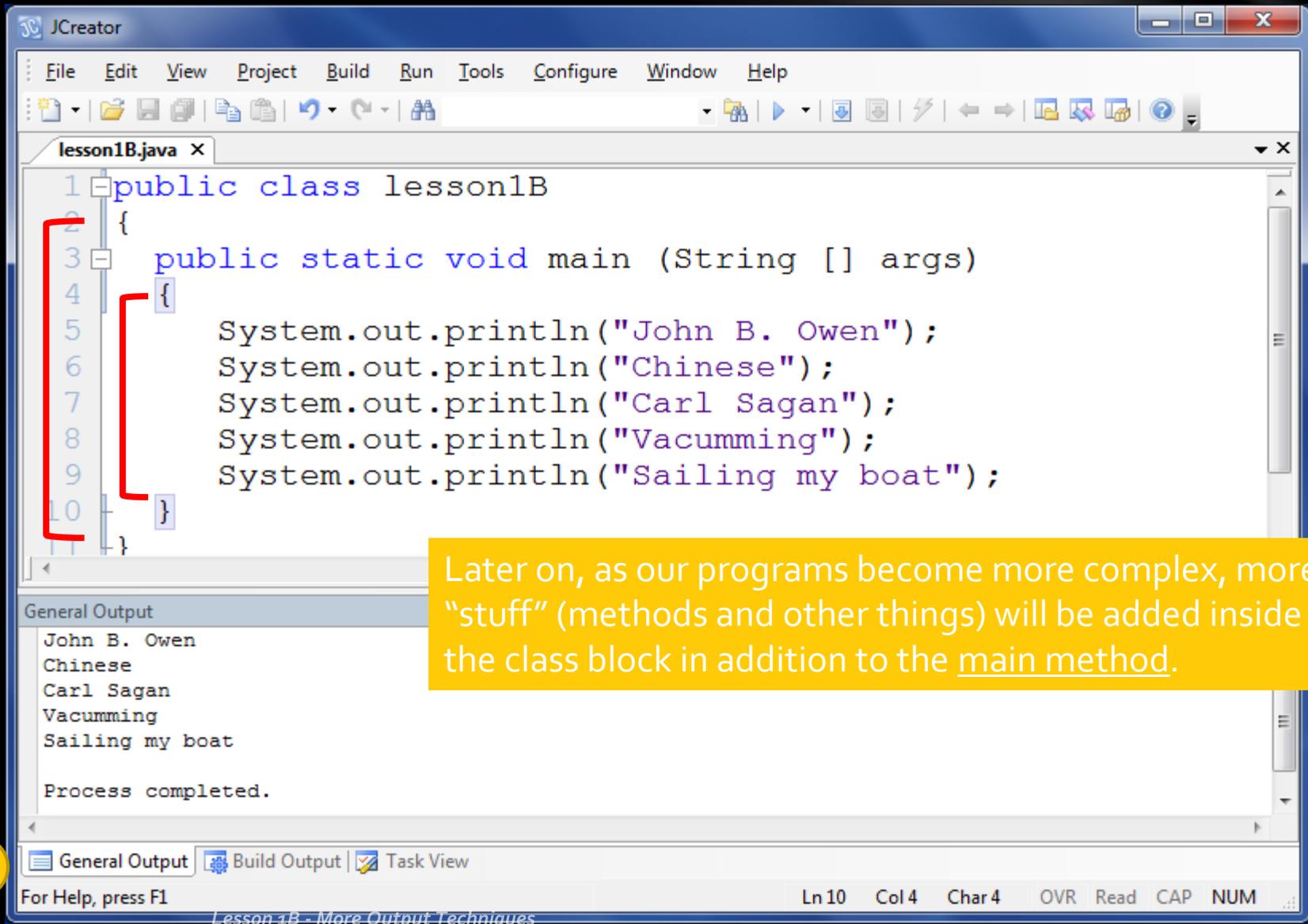
```
John B. Owen
Chinese
Carl Sagan
Vacumming
Sailing my boat
Process completed.
```

At the bottom of the IDE, there are tabs for 'General Output', 'Build Output', and 'Task View'. The status bar shows 'For Help, press F1' and 'Ln 10 Col 4 Char 4 OVR Read CAP NUM'. A yellow smiley face icon is in the bottom left corner.

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



The screenshot shows the JCreator IDE with a Java file named 'lesson1B.java'. The code is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacumming");
9         System.out.println("Sailing my boat");
10    }
11 }
```

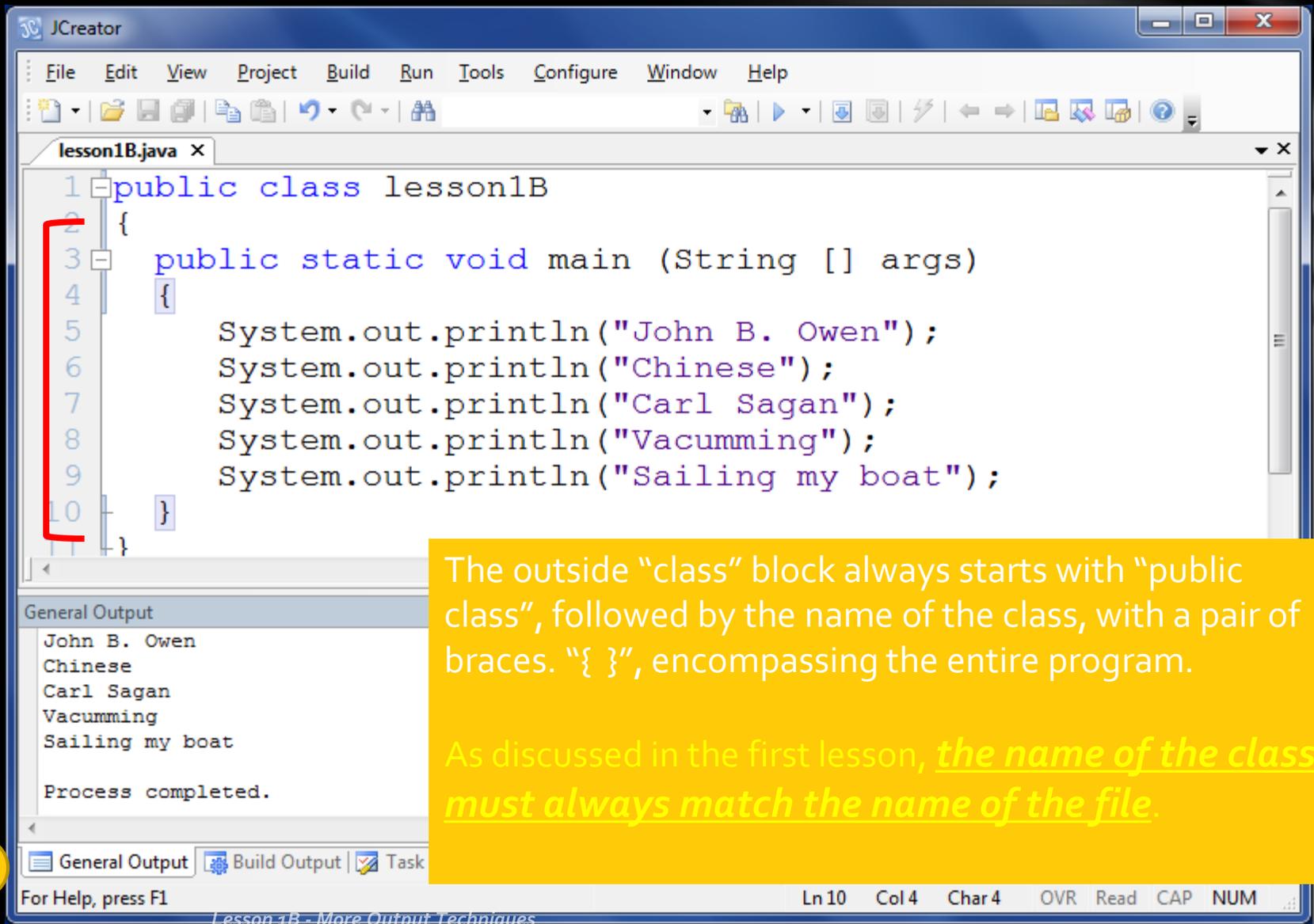
Red brackets highlight the class block (lines 1-11) and the main method block (lines 3-10). The 'General Output' window at the bottom shows the following output:

```
John B. Owen
Chinese
Carl Sagan
Vacumming
Sailing my boat
Process completed.
```

At the bottom of the IDE, the status bar shows 'Ln 10 Col 4 Char 4 OVR Read CAP NUM'. A yellow callout box contains the text: '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 screenshot shows the JCreator IDE with a Java file named 'lesson1B.java'. The code is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacumming");
9         System.out.println("Sailing my boat");
10    }
```

The 'General Output' window shows the following output:

```
John B. Owen
Chinese
Carl Sagan
Vacumming
Sailing my boat
Process completed.
```

A yellow callout box contains the following text:

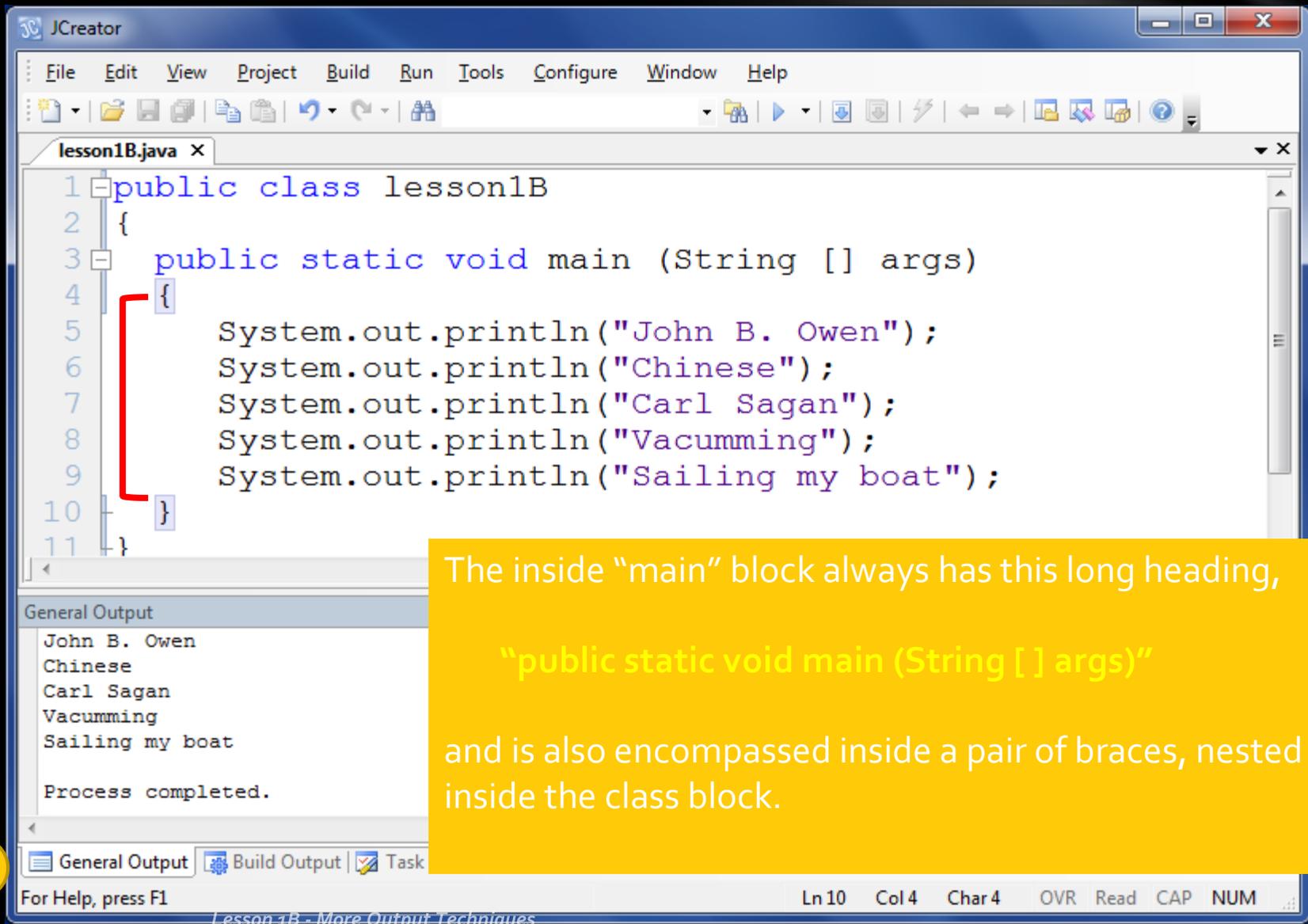
The outside "class" block always starts with "public class", followed by the name of the class, with a pair of braces. "{ }", encompassing the entire program.

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

At the bottom of the IDE, the status bar shows: Ln 10 Col 4 Char 4 OVR Read CAP NUM. The footer text reads: Lesson 1B - More Output Techniques.



Structure of basic JAVA program



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacumming");
9         System.out.println("Sailing my boat");
10    }
11 }
```

The `main` method block (lines 3-10) is highlighted with a red bracket. Below the code editor is the `General Output` window, which displays the following text:

```
John B. Owen
Chinese
Carl Sagan
Vacumming
Sailing my boat
Process completed.
```

The status bar at the bottom indicates the current position: `Ln 10 Col 4 Char 4 OVR Read CAP NUM`. The text `Lesson 1B - More Output Techniques` is visible in the bottom left corner of the IDE window.

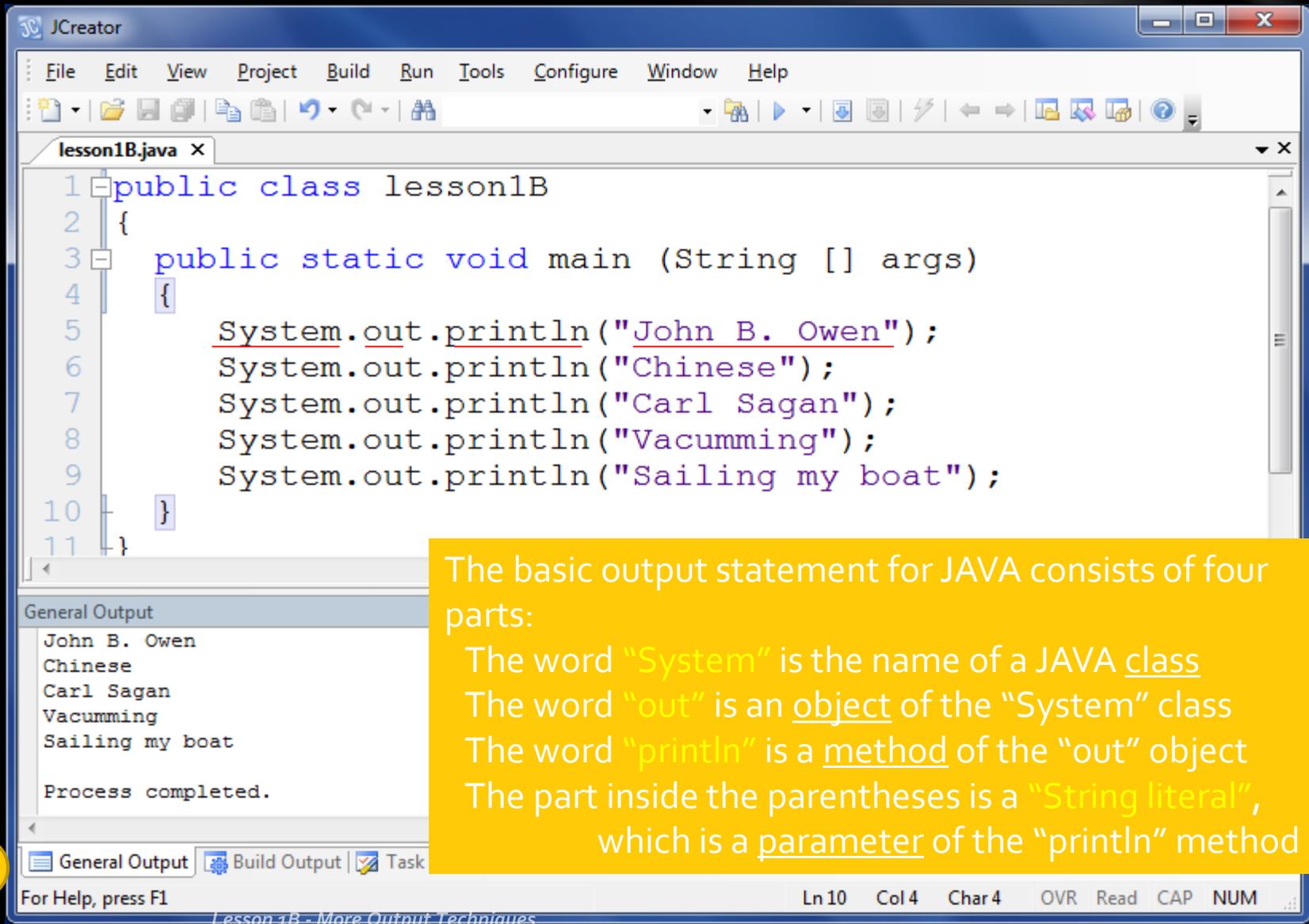
The inside "main" block always has this long heading,

"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 screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code defines a `public class lesson1B` with a `main` method. The `main` method contains five `System.out.println` statements, each printing a different string. Below the code editor, the `General Output` window displays the output of the program: `John B. Owen`, `Chinese`, `Carl Sagan`, `Vacuuming`, and `Sailing my boat`, followed by `Process completed.`

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacuuming");
9         System.out.println("Sailing my boat");
10    }
11 }
```

General Output

```
John B. Owen
Chinese
Carl Sagan
Vacuuming
Sailing my boat
Process completed.
```

For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM

Lesson 1B - More Output Techniques

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 – for 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.

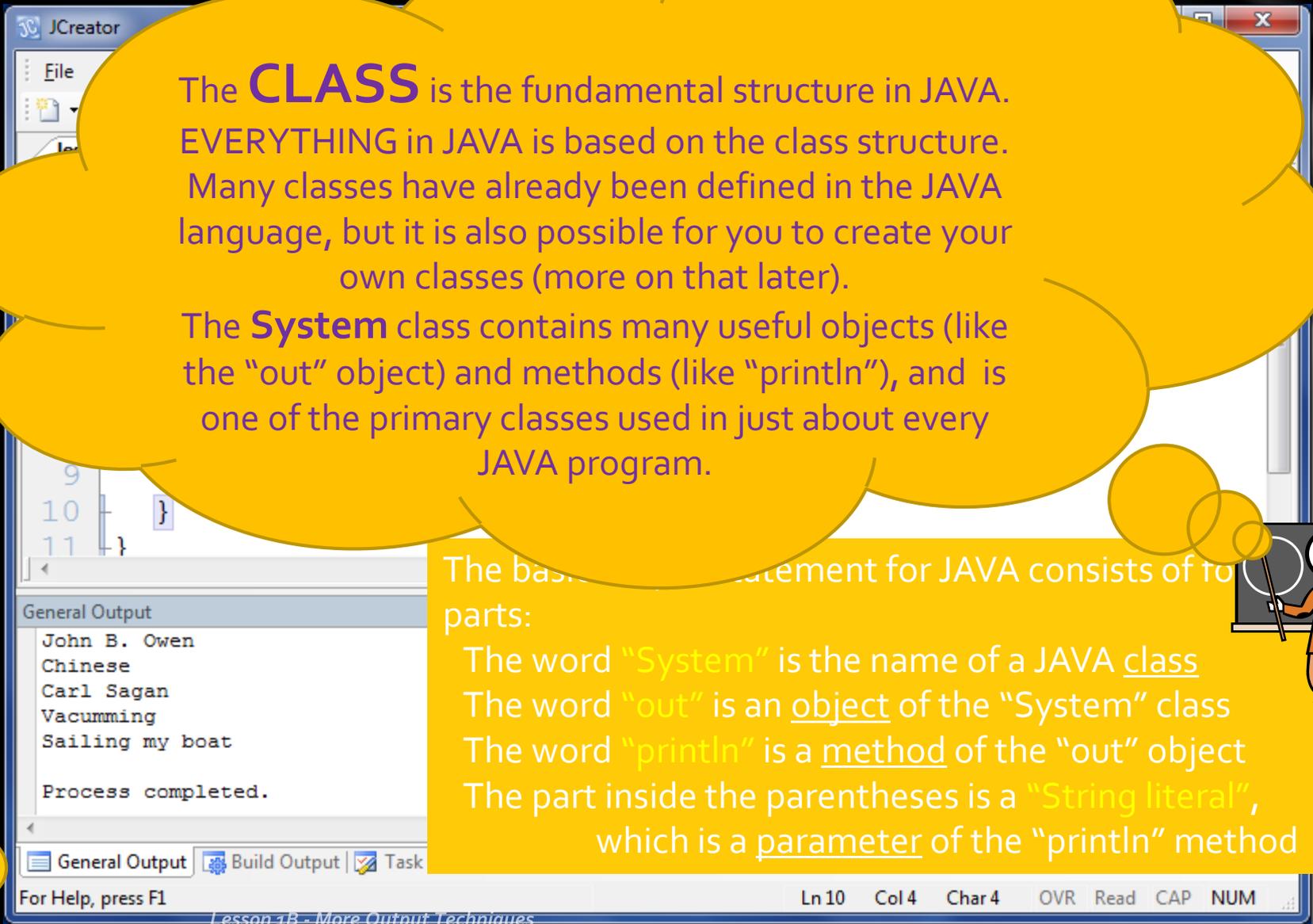
The basic output statement for JAVA consists of two parts:

The word “**System**” is the name of a JAVA class

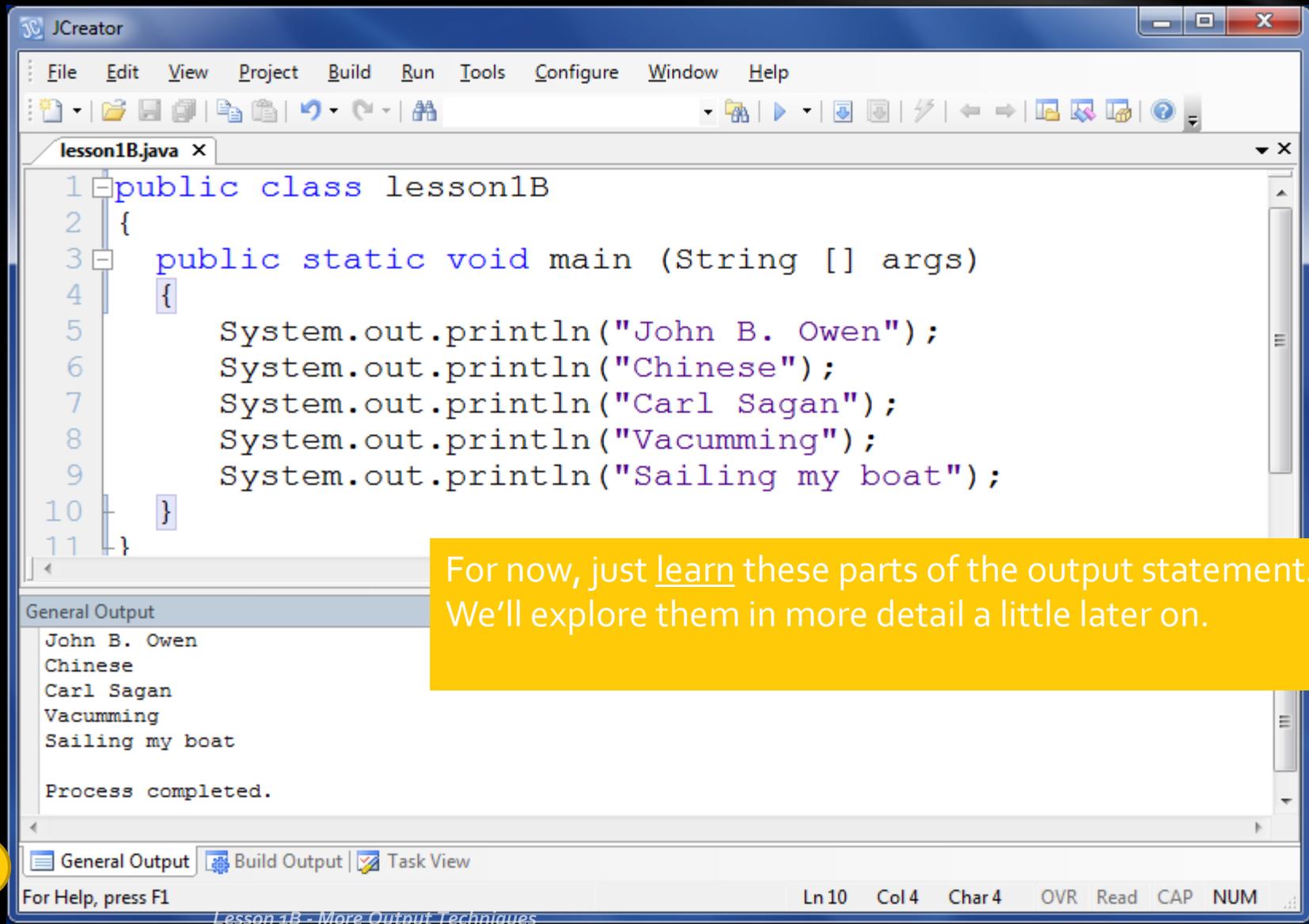
The word “**out**” is an object of the “System” class

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



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code defines a class `lesson1B` with a `main` method that prints five lines of text. Below the code, the 'General Output' window shows the exact output of the program, including a 'Process completed.' message. A yellow callout box highlights the output text.

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacuuming");
9         System.out.println("Sailing my boat");
10    }
11 }
```

General Output

```
John B. Owen
Chinese
Carl Sagan
Vacuuming
Sailing my boat
Process completed.
```

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

General Output | Build Output | Task View

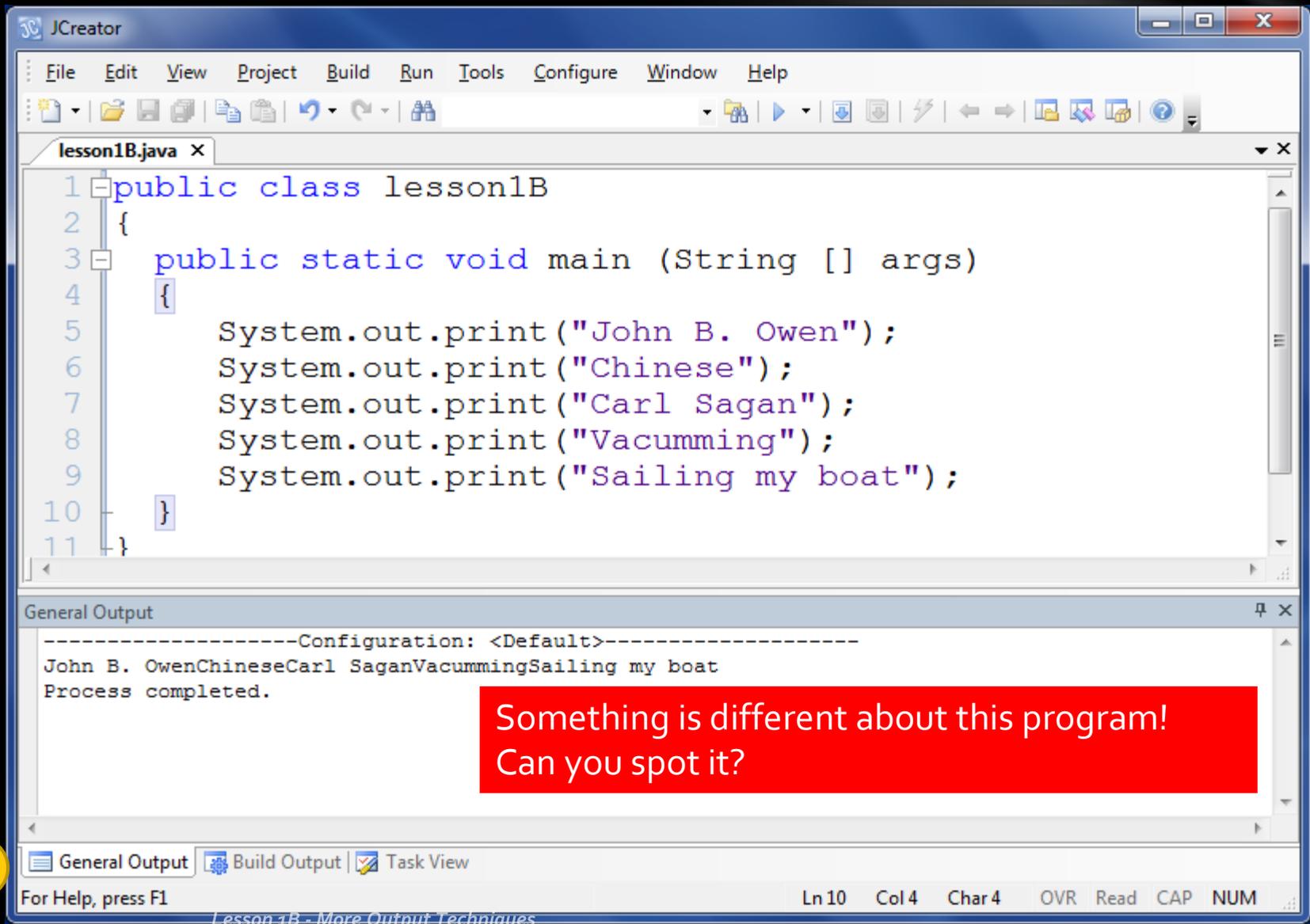
For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM

Lesson 1B - More Output Techniques



System.out.print



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code in the editor is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print ("John B. Owen");
6         System.out.print ("Chinese");
7         System.out.print ("Carl Sagan");
8         System.out.print ("Vacumming");
9         System.out.print ("Sailing my boat");
10    }
11 }
```

Below the editor, the General Output window shows the following text:

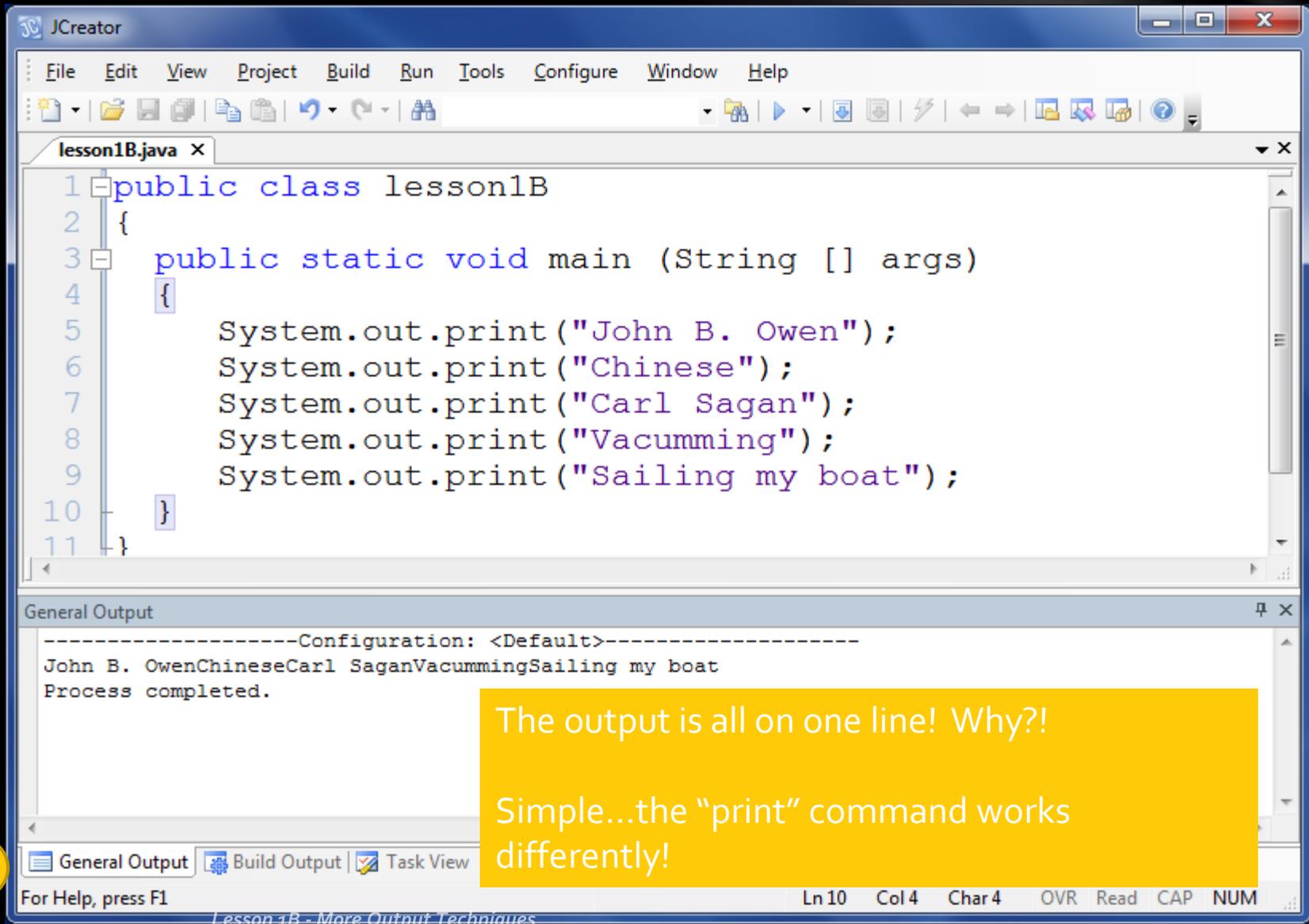
```
-----Configuration: <Default>-----
John B. OwenChineseCarl SaganVacummingSailing my boat
Process completed.
```

A red box is overlaid on the output window with the text: "Something is different about this program! Can you spot it?"

At the bottom of the IDE, the status bar shows "Ln 10 Col 4 Char 4 OVR Read CAP NUM".



System.out.print



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code defines a class `lesson1B` with a `main` method that prints five lines of text using `System.out.print`. The output window shows the result of the program execution, where all five lines of text are concatenated onto a single line. A yellow callout box explains that this happens because the `print` command does not add a newline character.

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print ("John B. Owen");
6         System.out.print ("Chinese");
7         System.out.print ("Carl Sagan");
8         System.out.print ("Vacumming");
9         System.out.print ("Sailing my boat");
10    }
11 }
```

General Output

-----Configuration: <Default>-----
John B. OwenChineseCarl SaganVacummingSailing my boat
Process completed.

The output is all on one line! Why?!

Simple...the "print" command works differently!

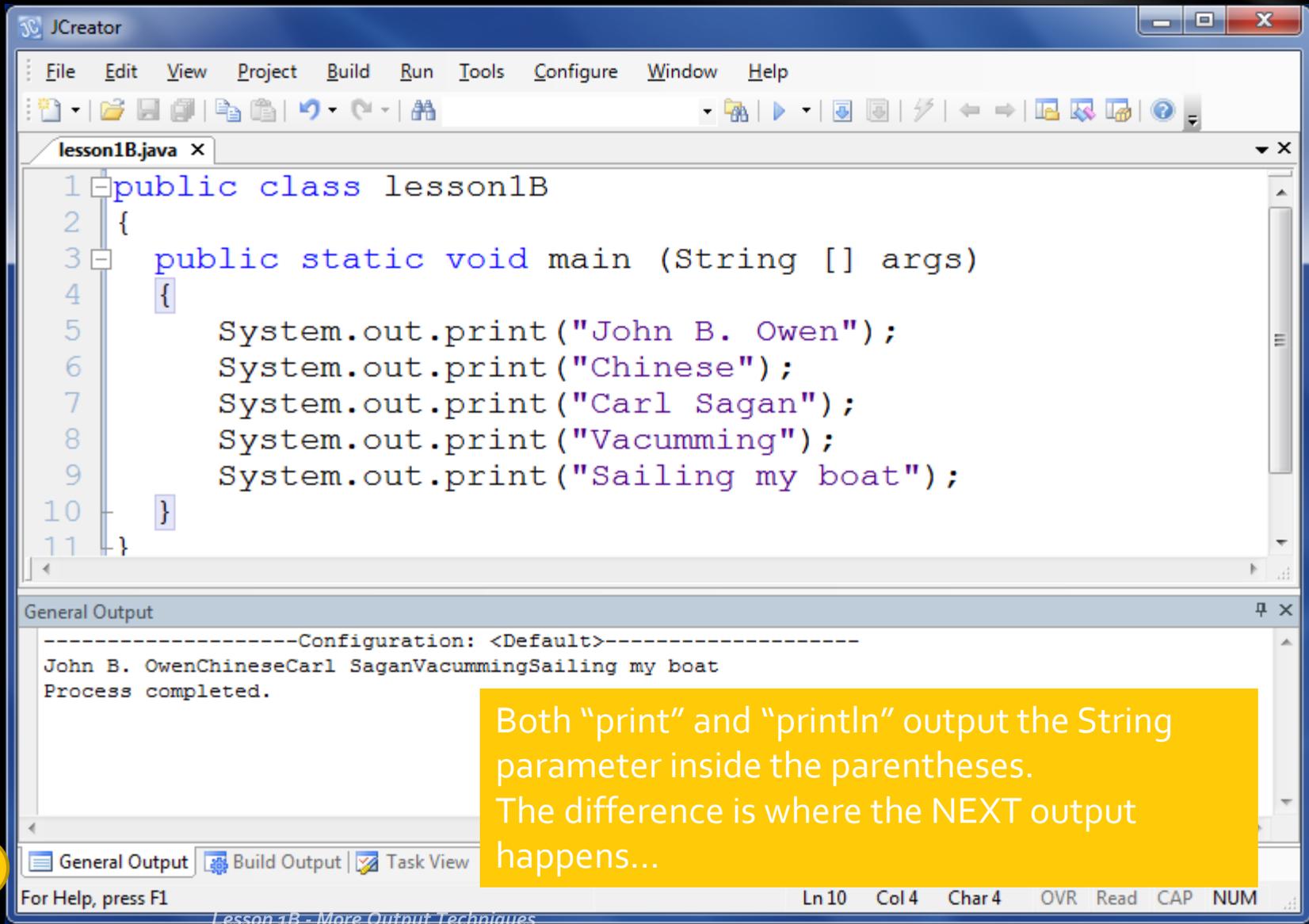
For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM

Lesson 1B - More Output Techniques



System.out.print



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print ("John B. Owen");
6         System.out.print ("Chinese");
7         System.out.print ("Carl Sagan");
8         System.out.print ("Vacumming");
9         System.out.print ("Sailing my boat");
10    }
11 }
```

The General Output window shows the following output:

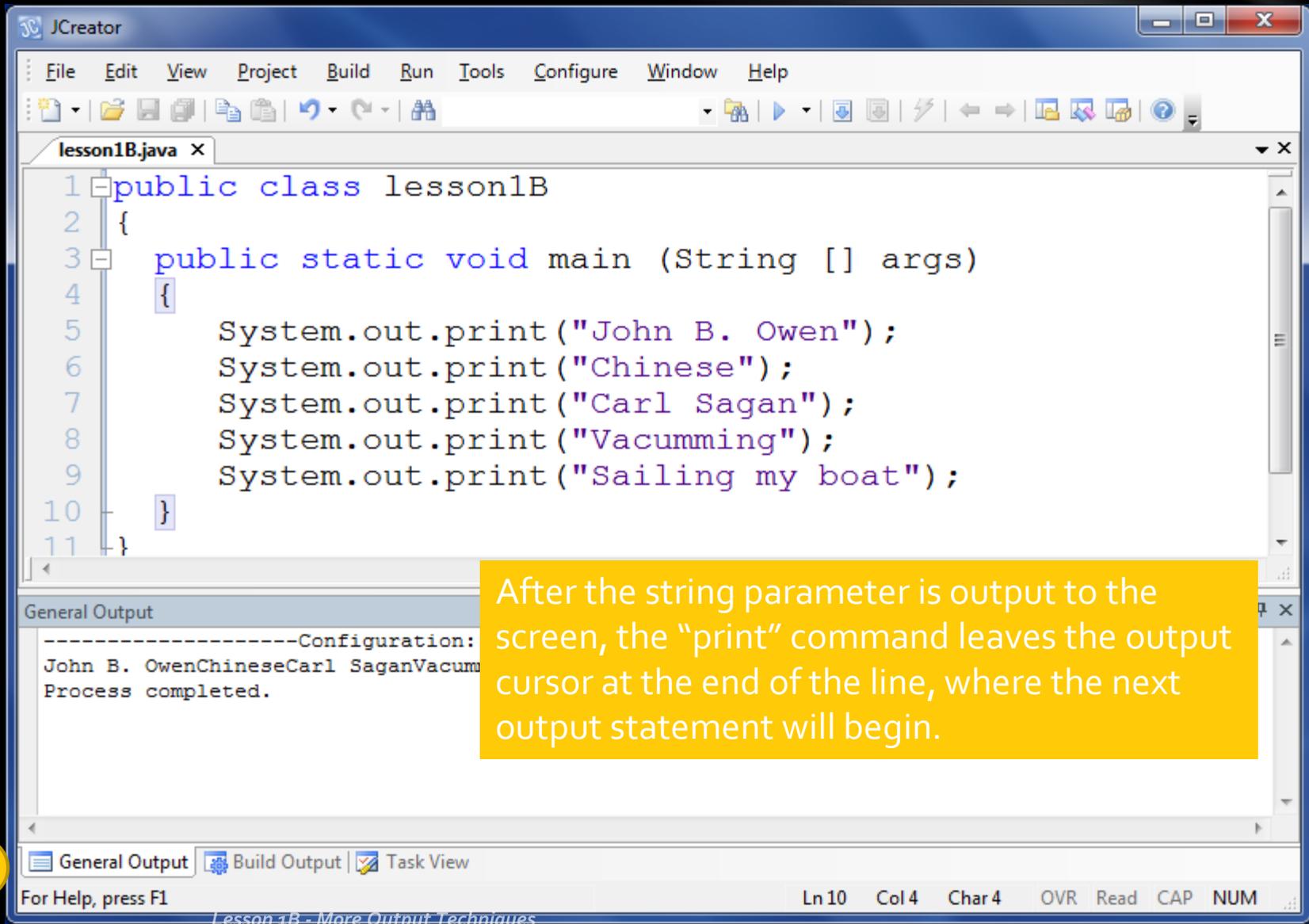
```
-----Configuration: <Default>-----
John B. OwenChineseCarl SaganVacummingSailing my boat
Process completed.
```

A yellow text box explains: Both "print" and "println" output the String parameter inside the parentheses. The difference is where the NEXT output happens...

At the bottom left, there is a yellow smiley face icon. The status bar at the bottom shows "For Help, press F1", "Ln 10 Col 4 Char 4", and "OVR Read CAP NUM". The footer text is "Lesson 1B - More Output Techniques".



System.out.print



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code defines a class `lesson1B` with a `main` method that prints five strings: "John B. Owen", "Chinese", "Carl Sagan", "Vacumming", and "Sailing my boat". The output window shows the concatenated result of these prints: "John B. OwenChineseCarl SaganVacumm".

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print ("John B. Owen");
6         System.out.print ("Chinese");
7         System.out.print ("Carl Sagan");
8         System.out.print ("Vacumming");
9         System.out.print ("Sailing my boat");
10    }
11 }
```

General Output

```
-----Configuration:
John B. OwenChineseCarl SaganVacumm
Process completed.
```

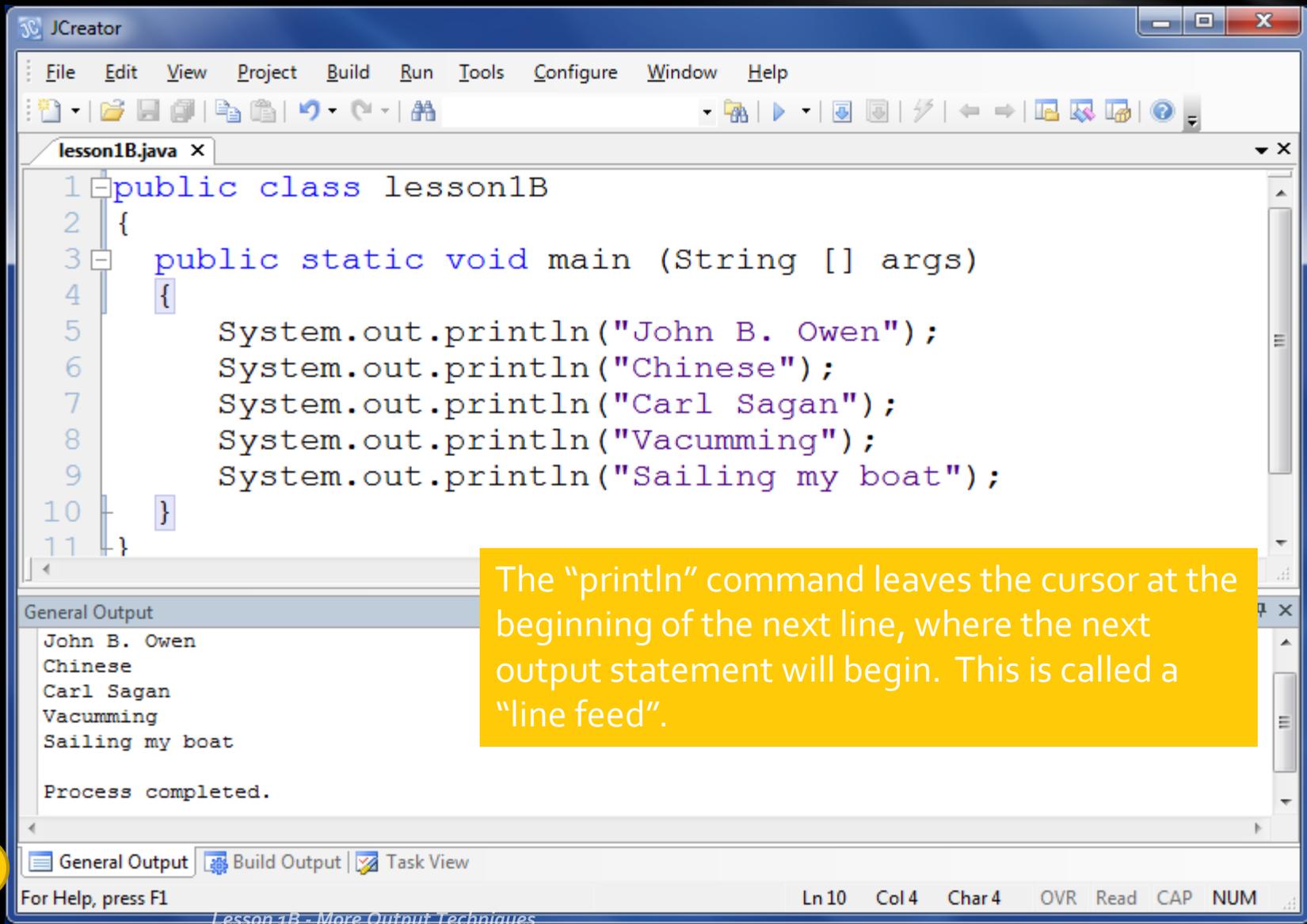
After the string parameter is output to the screen, the "print" command leaves the output cursor at the end of the line, where the next output statement will begin.

For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM



System.out.println



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code defines a `public class lesson1B` with a `main` method that prints five lines of text. The output window shows the text printed by the program, followed by "Process completed." A yellow callout box explains that the `println` command leaves the cursor at the beginning of the next line, which is called a "line feed".

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("John B. Owen");
6         System.out.println("Chinese");
7         System.out.println("Carl Sagan");
8         System.out.println("Vacumming");
9         System.out.println("Sailing my boat");
10    }
11 }
```

General Output

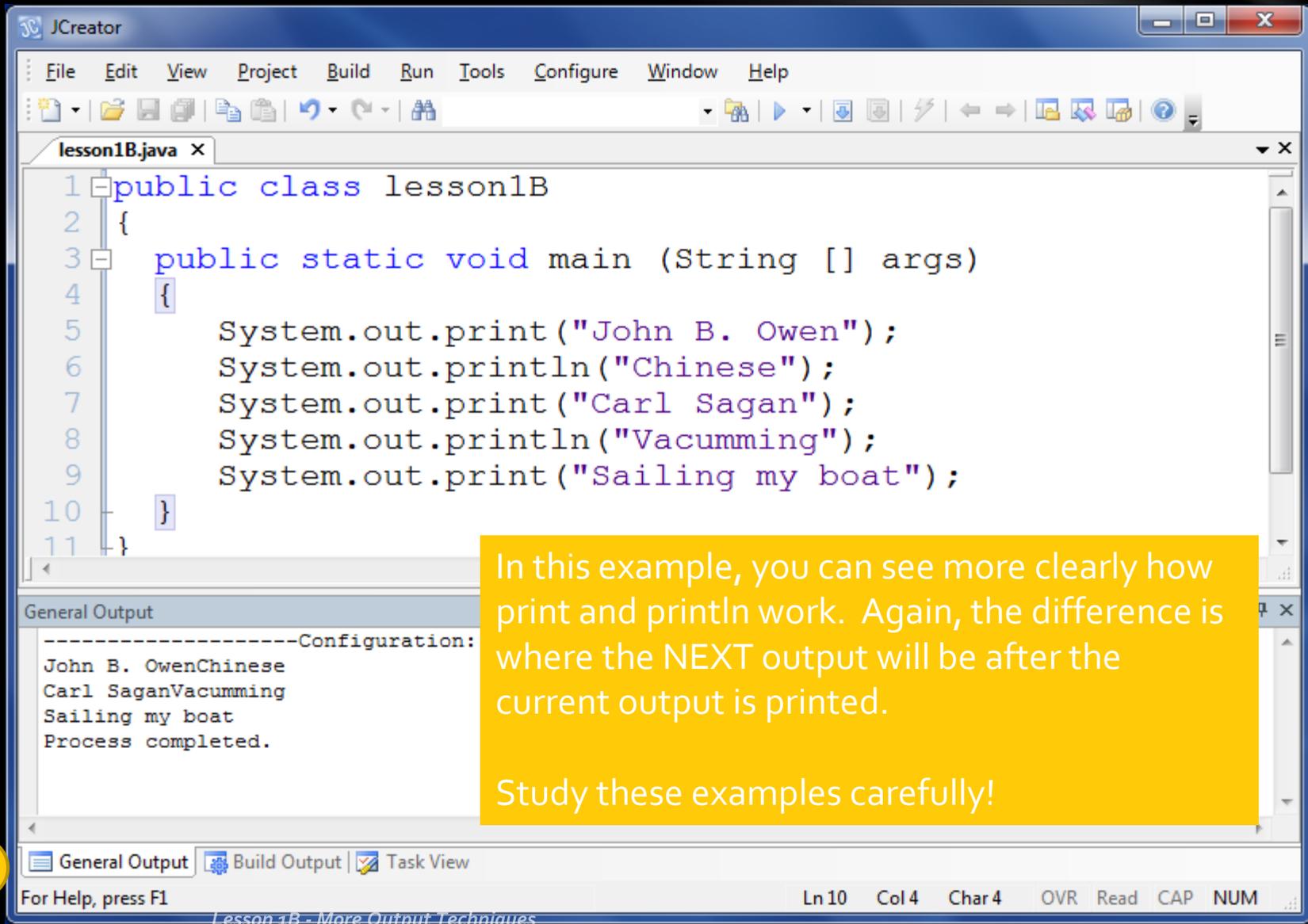
```
John B. Owen
Chinese
Carl Sagan
Vacumming
Sailing my boat
Process completed.
```

For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM



Mixing it up!



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code defines a class `lesson1B` with a `main` method. The `main` method contains five print statements: `System.out.print("John B. Owen");`, `System.out.println("Chinese");`, `System.out.print("Carl Sagan");`, `System.out.println("Vacumming");`, and `System.out.print("Sailing my boat");`. The `General Output` window at the bottom shows the execution results: `-----Configuration: John B. OwenChinese Carl SaganVacumming Sailing my boat Process completed.`

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print("John B. Owen");
6         System.out.println("Chinese");
7         System.out.print("Carl Sagan");
8         System.out.println("Vacumming");
9         System.out.print("Sailing my boat");
10    }
11 }
```

General Output

```
-----Configuration:
John B. OwenChinese
Carl SaganVacumming
Sailing my boat
Process completed.
```

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!

For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM

Lesson 1B - More Output Techniques



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.



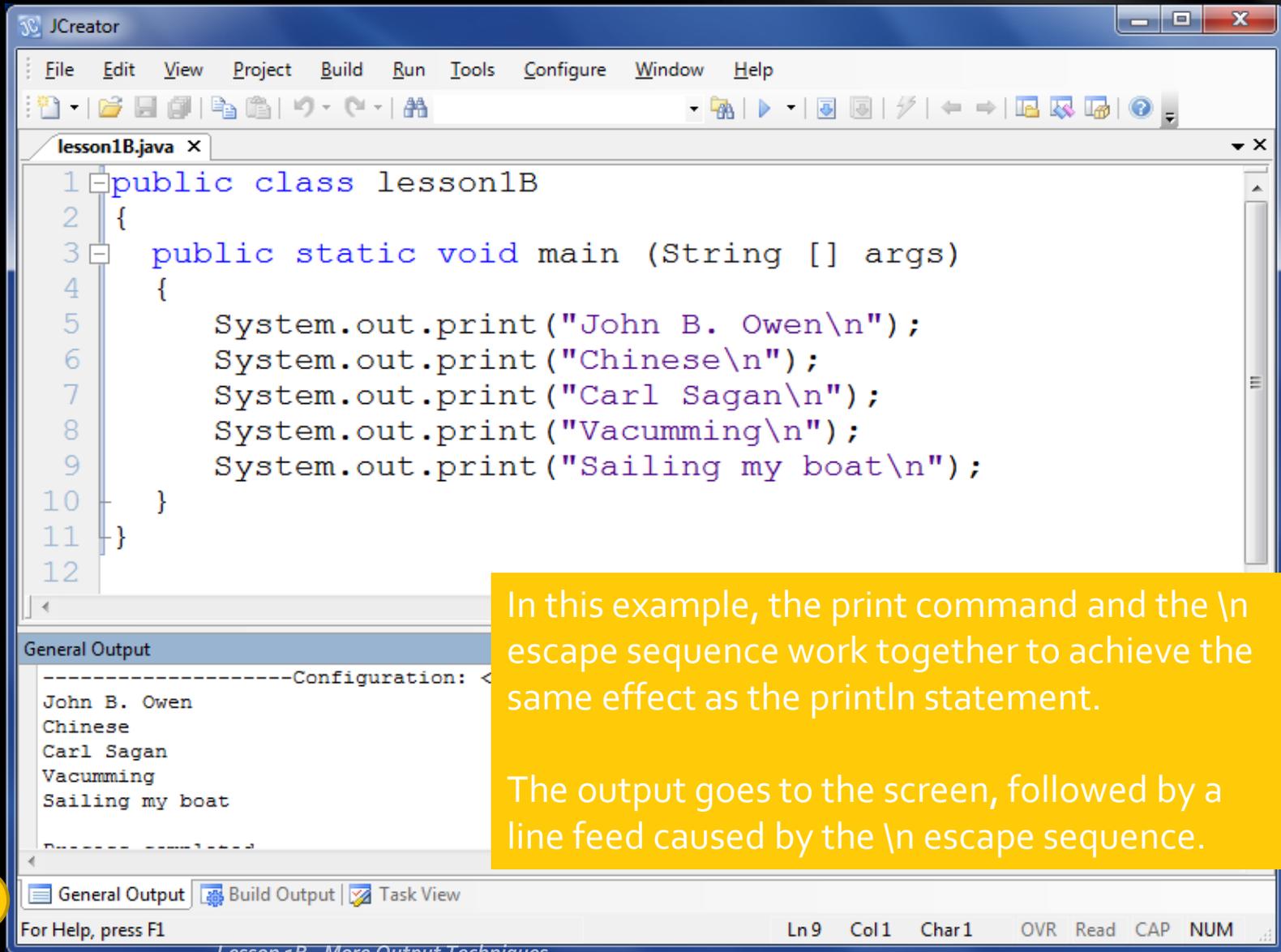
Escape Sequences

- 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)



```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print("John B. Owen\n");
6         System.out.print("Chinese\n");
7         System.out.print("Carl Sagan\n");
8         System.out.print("Vacuuming\n");
9         System.out.print("Sailing my boat\n");
10    }
11 }
12
```

General Output

```
-----Configuration: <
John B. Owen
Chinese
Carl Sagan
Vacuuming
Sailing my boat
----- completed
```

General Output | Build Output | Task View

For Help, press F1

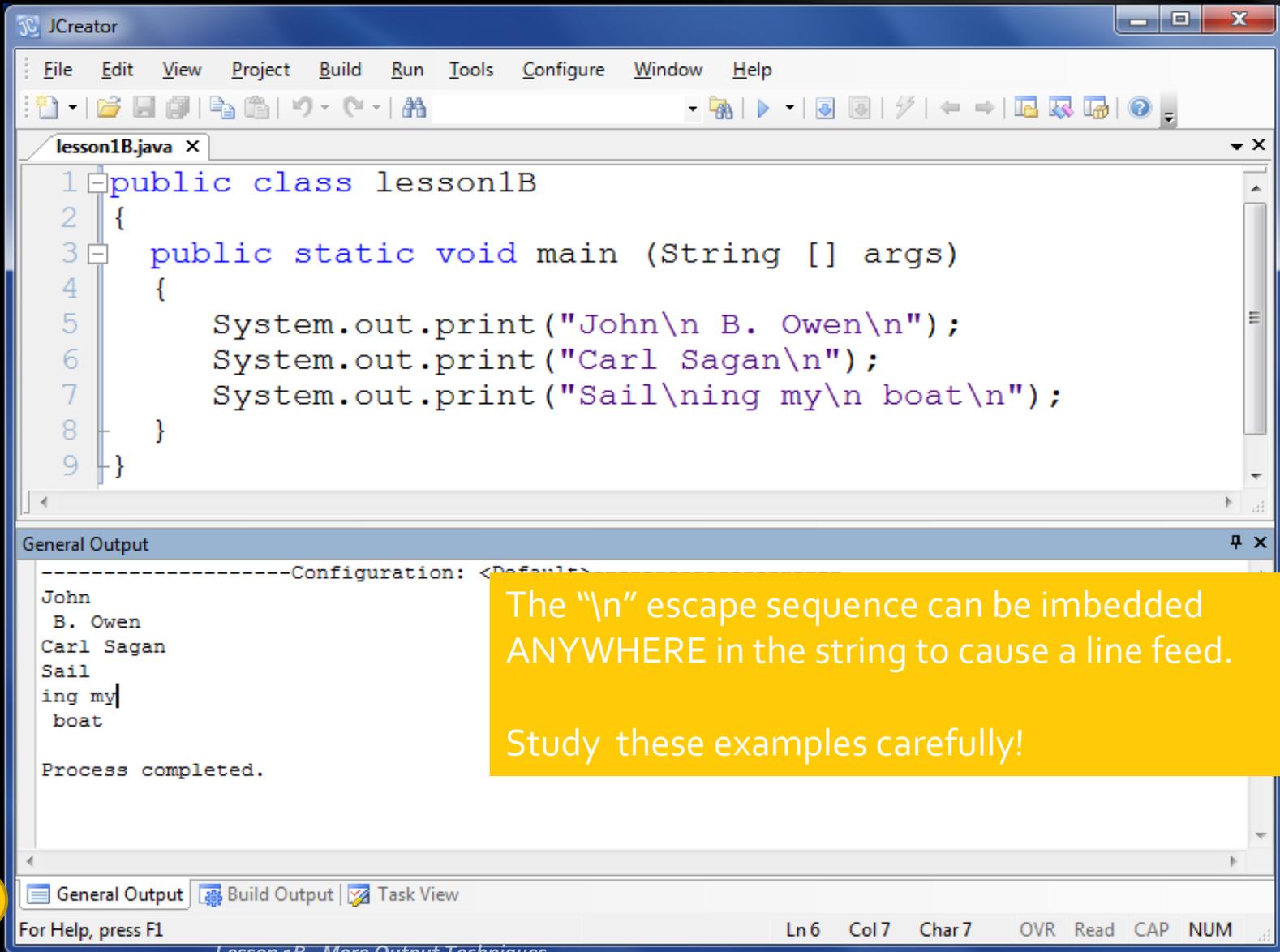
Ln9 Col1 Char1 OVR Read CAP NUM

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 screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code in the editor is as follows:

```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.print("John\n B. Owen\n");
6         System.out.print("Carl Sagan\n");
7         System.out.print("Sail\nning my\n boat\n");
8     }
9 }
```

Below the editor, the General Output window shows the execution results:

```
-----Configuration: <Default>-----
John
 B. Owen
Carl Sagan
Sail
ing my|
 boat
Process completed.
```

A yellow callout box contains the following text:

The "\n" escape sequence can be imbedded ANYWHERE in the string to cause a line feed.
Study these examples carefully!

At the bottom of the IDE, the status bar shows "Ln 6 Col 7 Char 7" and "OVR Read CAP NUM".



\t (tab)

```
JCreator
File Edit View Project Build Run Tools Configure Window Help
lesson1B.java x lesson1B.java
1 public class lesson1B_
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("123456781234567812345678");
6         System.out.println("*\t*\t*\t*");
7     }
8 }
```

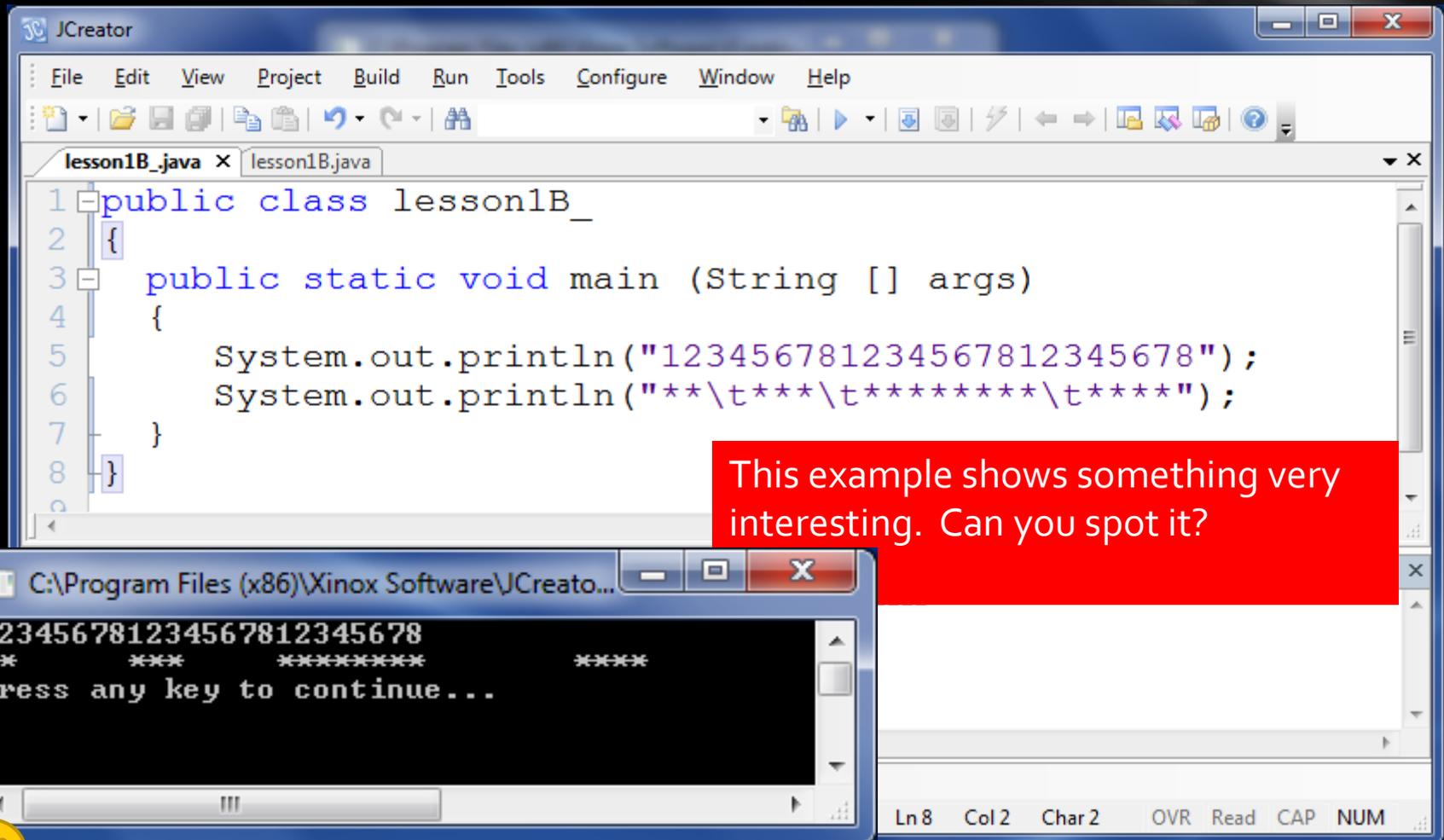
The tab escape sequence, `\t`, causes the output to be tabbed to the next tab location, eight steps beyond the previous one.

The output example above shows this very clearly.

```
C:\Program Files (x86)\Xinox Software\JCreator...
123456781234567812345678
*      *      *      *
Press any key to continue..._
```



\t (tab)



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java`. The code is as follows:

```
1 public class lesson1B_  
2 {  
3     public static void main (String [] args)  
4     {  
5         System.out.println("123456781234567812345678");  
6         System.out.println("**\t**\t*****\t**");  
7     }  
8 }
```

The output window shows the following text:

```
123456781234567812345678  
**      **      *****      **  
Press any key to continue...
```

A red text box overlaid on the IDE says: "This example shows something very interesting. Can you spot it?"



\t (tab)

```
JCreator
File Edit View Project Build Run Tools Configure Window Help
lesson1B.java x lesson1B.java
1 public class lesson1B_
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("123456781234567812345678");
6         System.out.println("**\t**\t*****\t****");
7     }
8 }
```

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?

```
C:\Program Files (x86)\Xinox Software\JCreato...
123456781234567812345678
**      ***      *****      ****
Press any key to continue...
```



\t (tab)

```
JCreator
File Edit View Project Build Run Tools Configure Window Help
lesson1B.java x lesson1B.java
1 public class lesson1B_
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("123456781234567812345678");
6         System.out.println("**\t**\t*****\t****");
7     }
8 }
```

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.

```
C:\Program Files (x86)\Xinox Software\JCreato...
123456781234567812345678
**      ***      *****      ****
Press any key to continue...
```



\\ 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



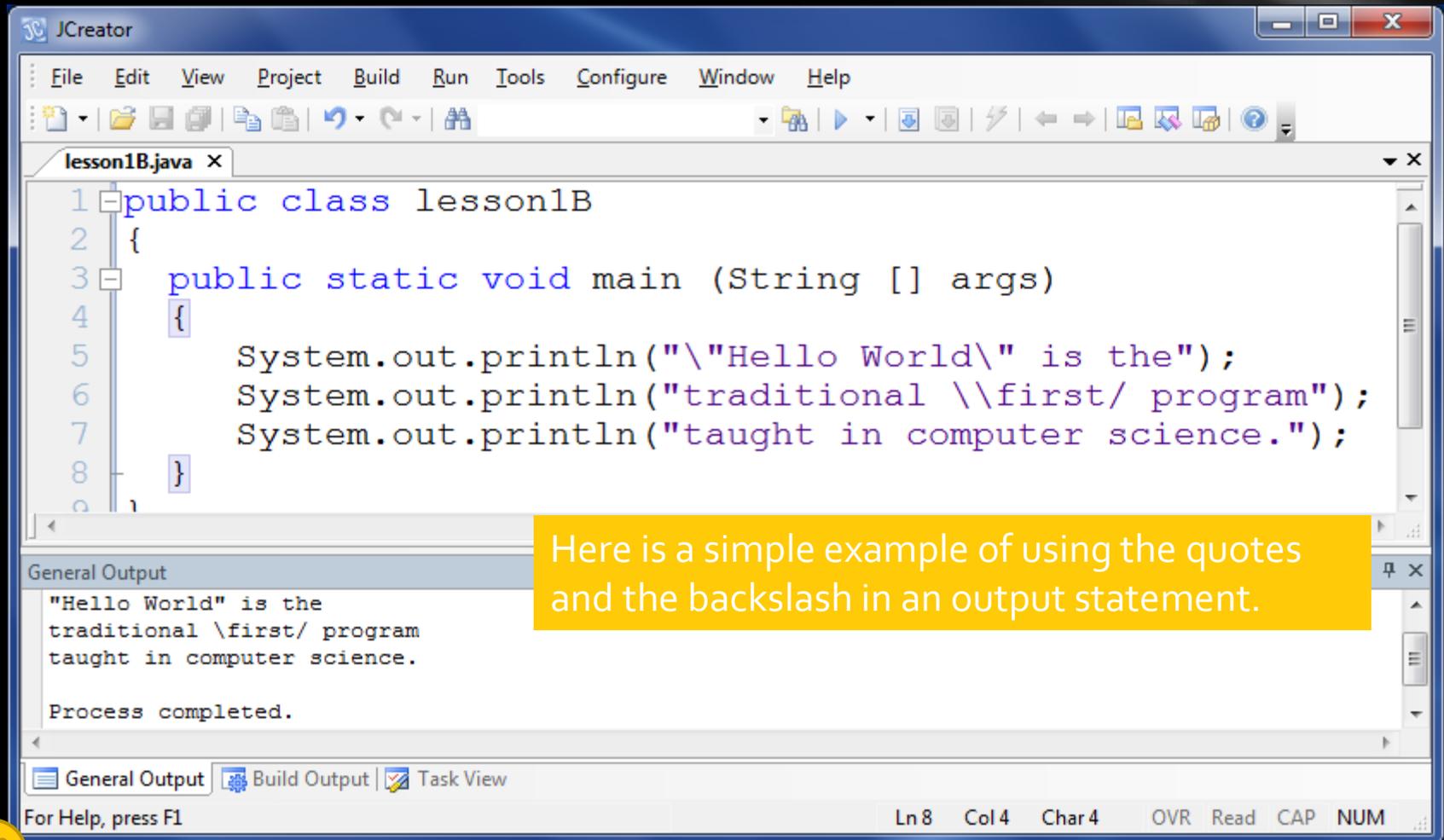
\\ and \"

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 \"



```
1 public class lesson1B
2 {
3     public static void main (String [] args)
4     {
5         System.out.println("\"Hello World\" is the");
6         System.out.println("traditional \\first/ program");
7         System.out.println("taught in computer science.");
8     }
9 }
```

General Output

```
"Hello World" is the
traditional \\first/ program
taught in computer science.

Process completed.
```

General Output | Build Output | Task View

For Help, press F1

Ln 8 Col 4 Char 4 OVR Read CAP NUM

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

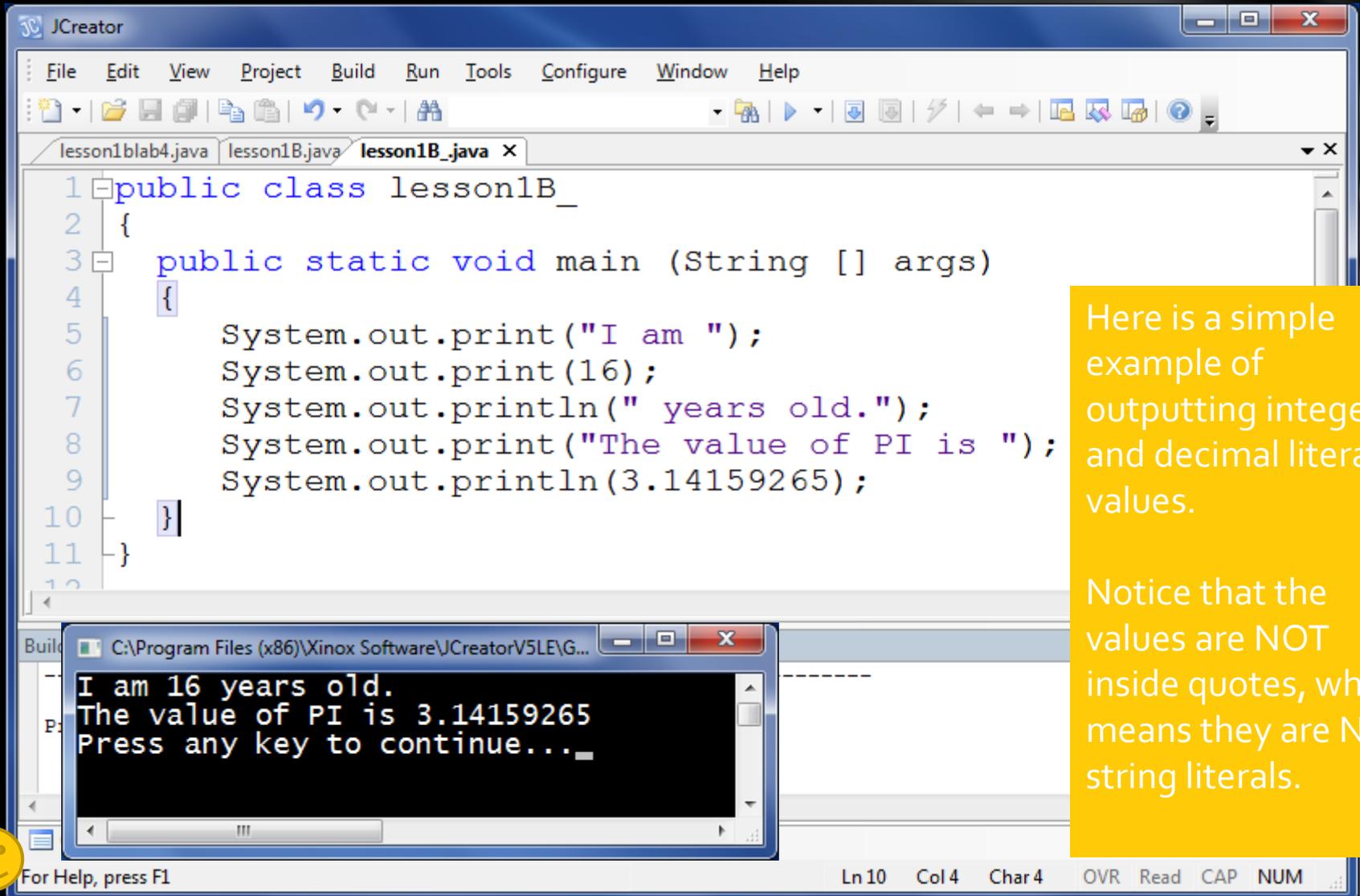


Data literals – string, integer, decimal

- 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



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java` open. The code defines a `main` method that prints a string, an integer, and a decimal value. A separate console window shows the output of the program, which matches the code's output.

```
1 public class lesson1B_  
2 {  
3     public static void main (String [] args)  
4     {  
5         System.out.print("I am ");  
6         System.out.print(16);  
7         System.out.println(" years old.");  
8         System.out.print("The value of PI is ");  
9         System.out.println(3.14159265);  
10    }  
11 }
```

Output:

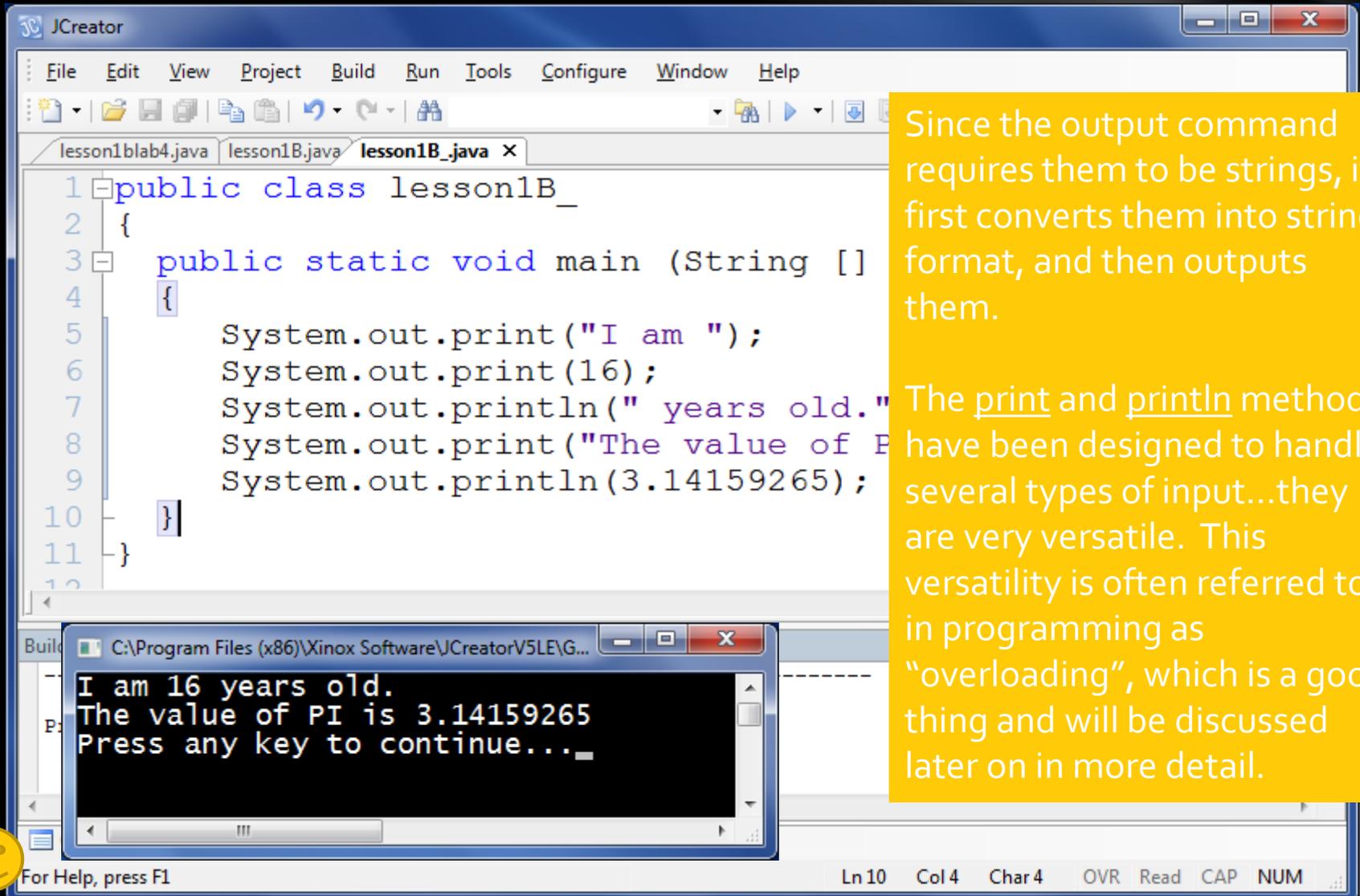
```
I am 16 years old.  
The value of PI is 3.14159265  
Press any key to continue..._
```

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



The screenshot shows the JCreator IDE with a Java file named `lesson1B.java` open. The code defines a class `lesson1B` with a `main` method. The `main` method contains five lines of code that use `System.out` to print various outputs: a string, an integer, a string with a newline, a string with a space, and a decimal value. Below the code editor, a console window displays the output of the program: "I am 16 years old.", "The value of PI is 3.14159265", and "Press any key to continue...".

```
1 public class lesson1B_  
2 {  
3     public static void main (String []  
4     {  
5         System.out.print("I am ");  
6         System.out.print(16);  
7         System.out.println(" years old.");  
8         System.out.print("The value of PI ");  
9         System.out.println(3.14159265);  
10    }  
11 }
```

I am 16 years old.
The value of PI is 3.14159265
Press any key to continue..._

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.



For Help, press F1

Ln 10 Col 4 Char 4 OVR Read CAP NUM

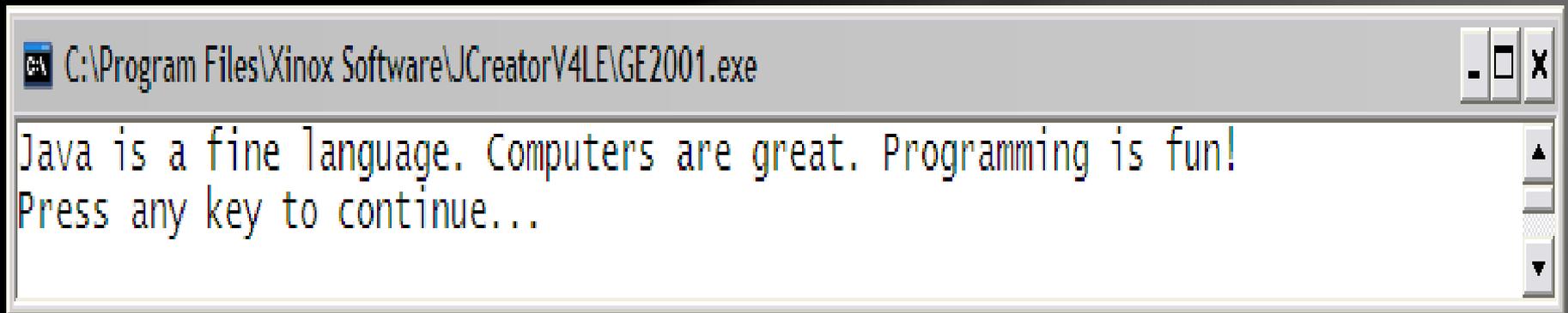
Lesson Summary

- 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.

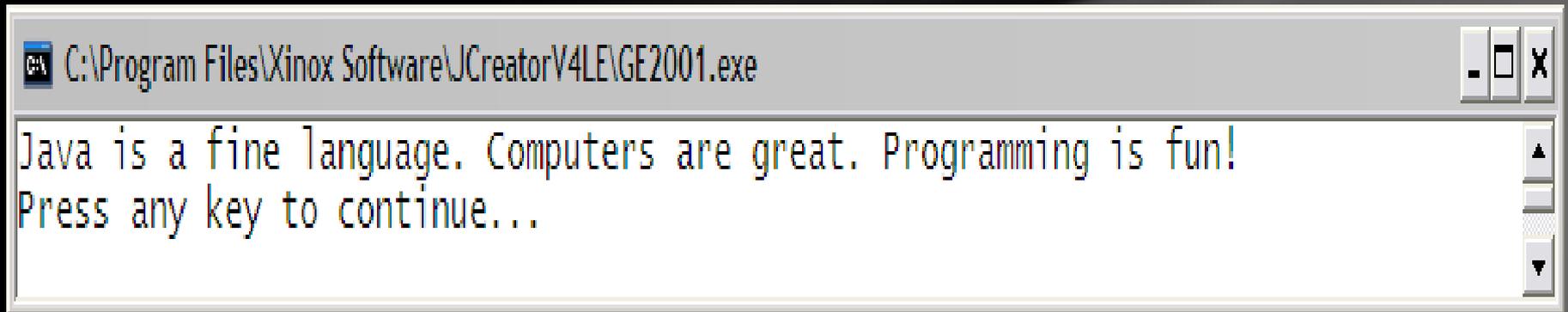


The screenshot shows a window titled "C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe". The window contains a text area with the following text: "Java is a fine language. Computers are great. Programming is fun!" followed by "Press any key to continue...". The text is displayed in a monospaced font. The window has standard Windows window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side.



Labs for Lesson 1B

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



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe. The window contains the following text: "Java is a fine language. Computers are great. Programming is fun!" followed by "Press any key to continue...". The text is displayed in a monospaced font. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.



Labs for Lesson 1B

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

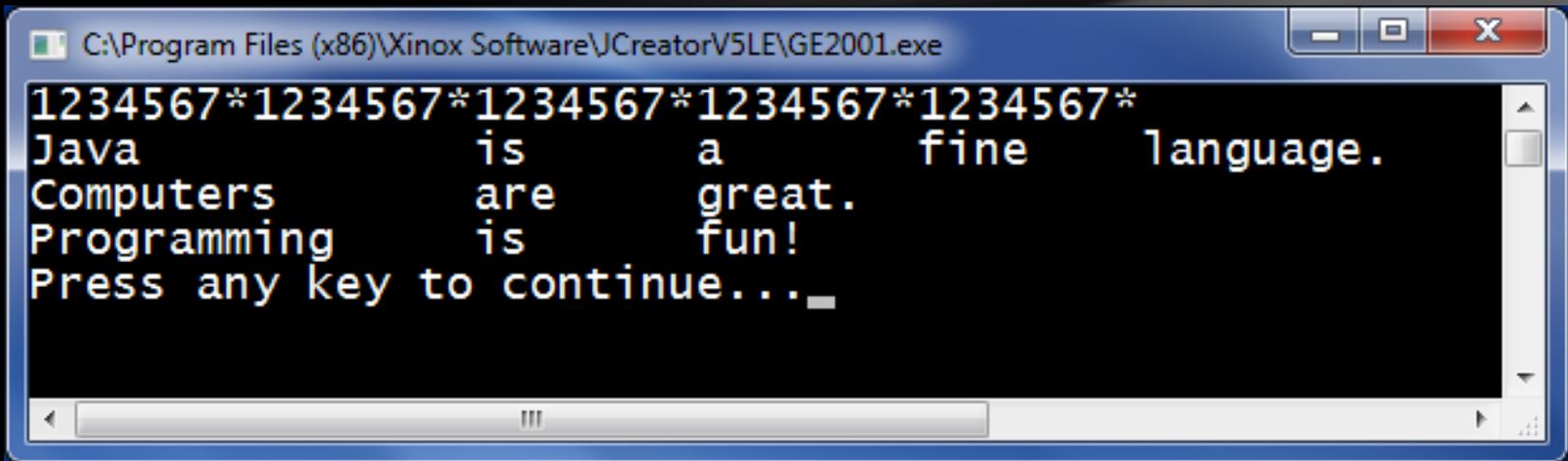
```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
```

```
Java is a fine language.  
Computers are great.  
Programming is fun!  
Press any key to continue...
```



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.*

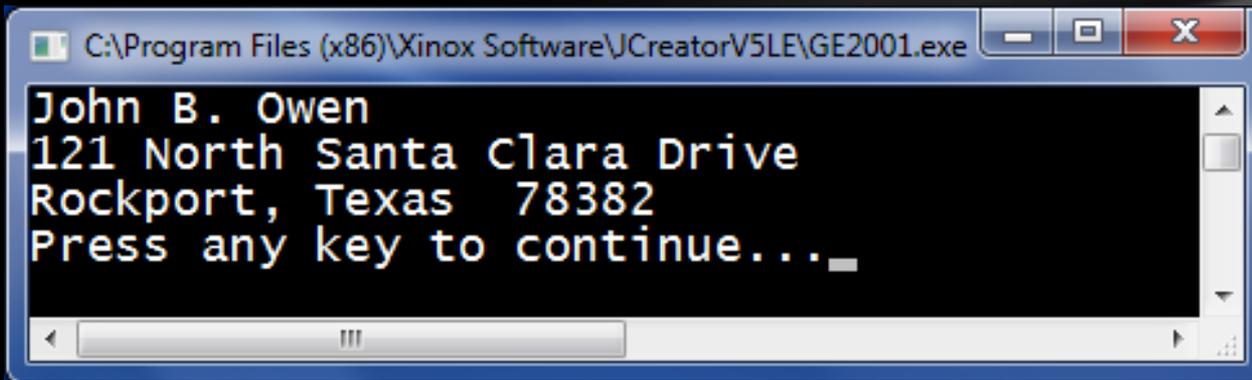


```
C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe
1234567*1234567*1234567*1234567*1234567*
Java      is      a      fine      language.
Computers are    great.
Programming is    fun!
Press any key to continue..._
```



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!



A screenshot of a Java application window titled "C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe". The window contains a text area with the following output:
John B. Owen
121 North Santa Clara Drive
Rockport, Texas 78382
Press any key to continue..._

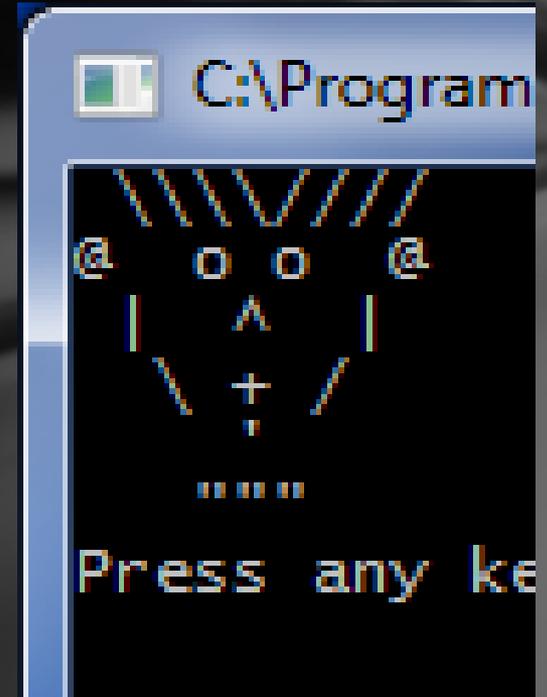


```
System.out.println("John B. Owen");  
System.out.print(121);  
System.out.println(" North Santa Cla  
System.out.print("Rockport, Texas 78382");
```



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



8/21/2015