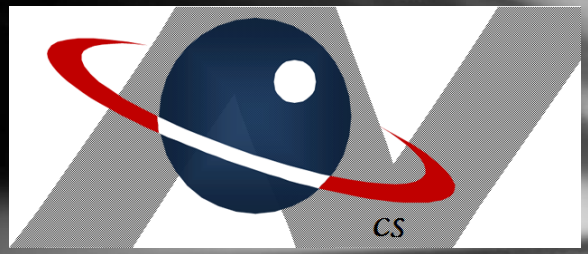
O(N) CS LESSONS

*Lesson 1D – More **printf** formatting features*



By John B. Owen

Table of Contents

Objective #1

- The student will understand and apply advanced ***printf*** features dealing with field width and format flags using JAVA programming examples.
- CS1 TEKS 126.33c2(E) improve numeric display by optimizing data visualization;

Objective #2

- The student will understand and apply the various single character suffixes that are appended to the %t Calendar format specifier to output different aspects of dates and times.
- The student will understand how to use the set command to change the value of the current date and time.

Example #1 -Field width specifier

printf includes several advanced features that help further control output formatting.
In this example, a value is included within the %s format specifier. It works like a tab, but can be customized to whatever size you need.
Study it carefully.

JCreator

File Edit View Project Build Run Tools Confic

printf.java

```
1  import java.util.*;
2  public class printf
3  {
4    public static void main (String [] args)
5    {
6      System.out.println("123456789*123456789*123456789*");
7      System.out.printf ("%10s","H");
8      System.out.printf ("%10s","EL");
9      System.out.printf ("%10s%n","LO");
10   }
11 }
12
```

General Output

For Help, press F1

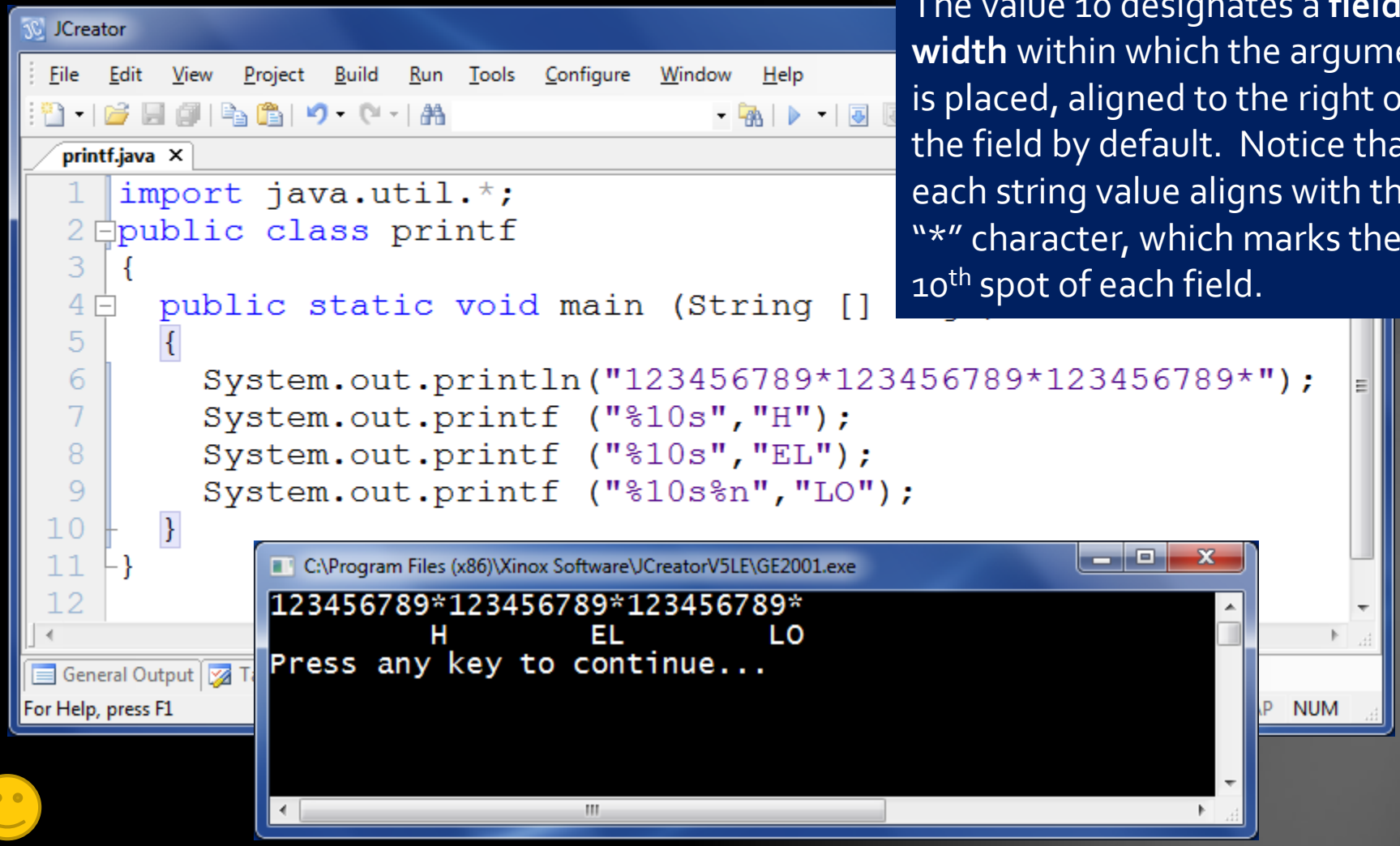
NUM

C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe

```
123456789*123456789*123456789*
         H        EL        LO
Press any key to continue...
```

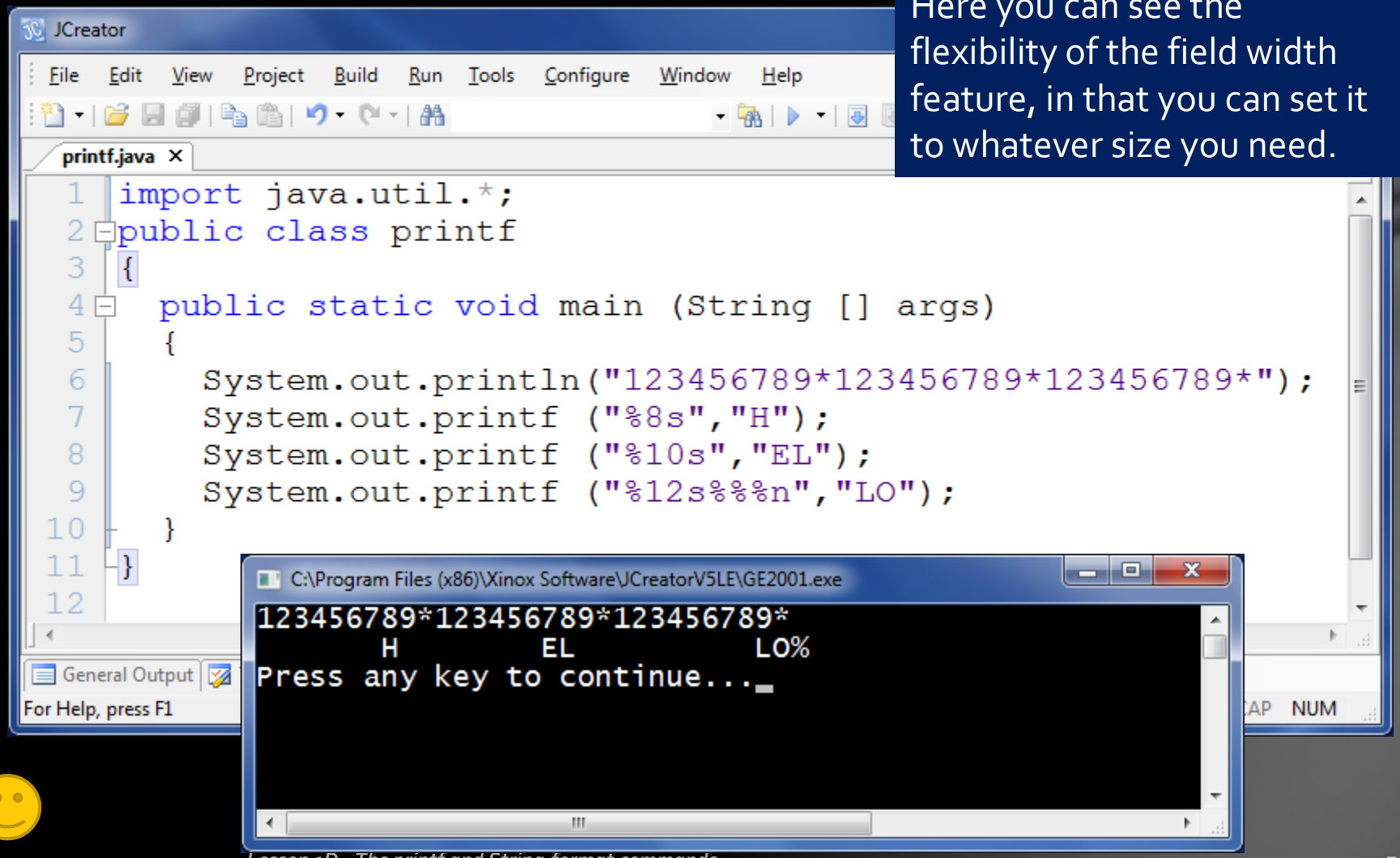
Example #1 -Field width specifier

The value 10 designates a **field width** within which the argument is placed, aligned to the right of the field by default. Notice that each string value aligns with the "*" character, which marks the 10th spot of each field.



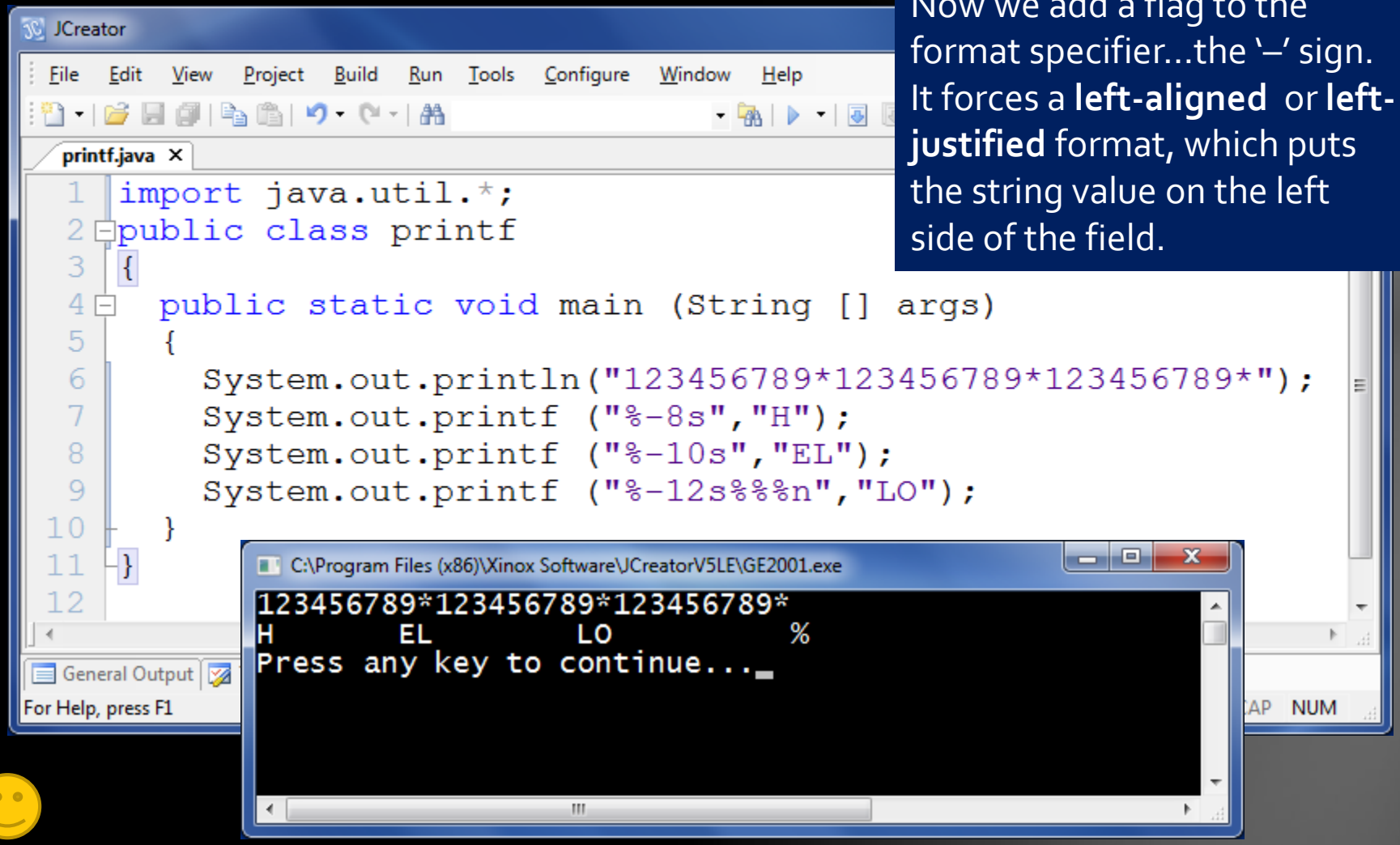
Example #2 -Different field widths

Here you can see the flexibility of the field width feature, in that you can set it to whatever size you need.



Example #3 - Left-align flag '-'

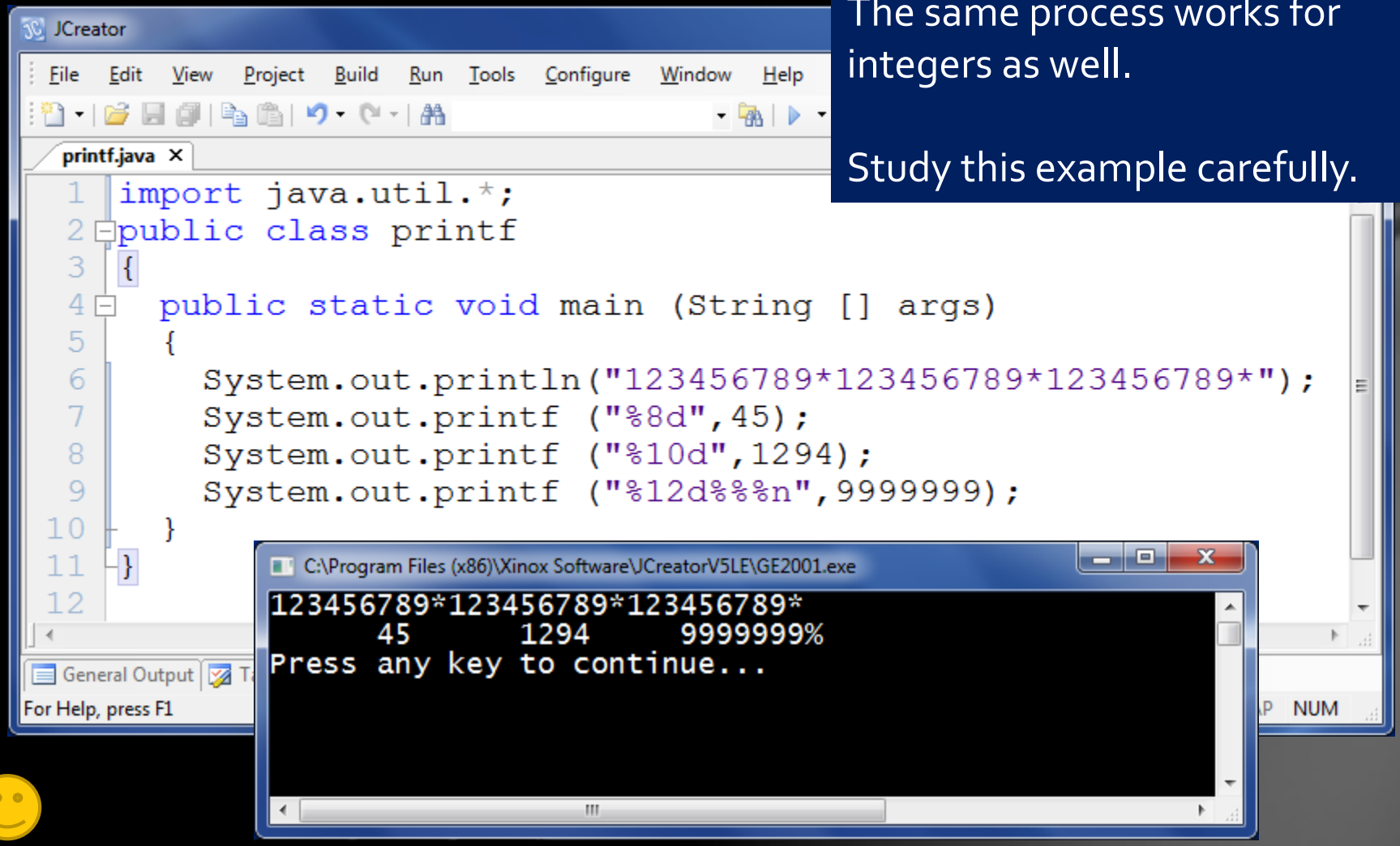
Now we add a flag to the format specifier…the '–' sign. It forces a **left-aligned** or **left-justified** format, which puts the string value on the left side of the field.



Example #4 - Field width with integers

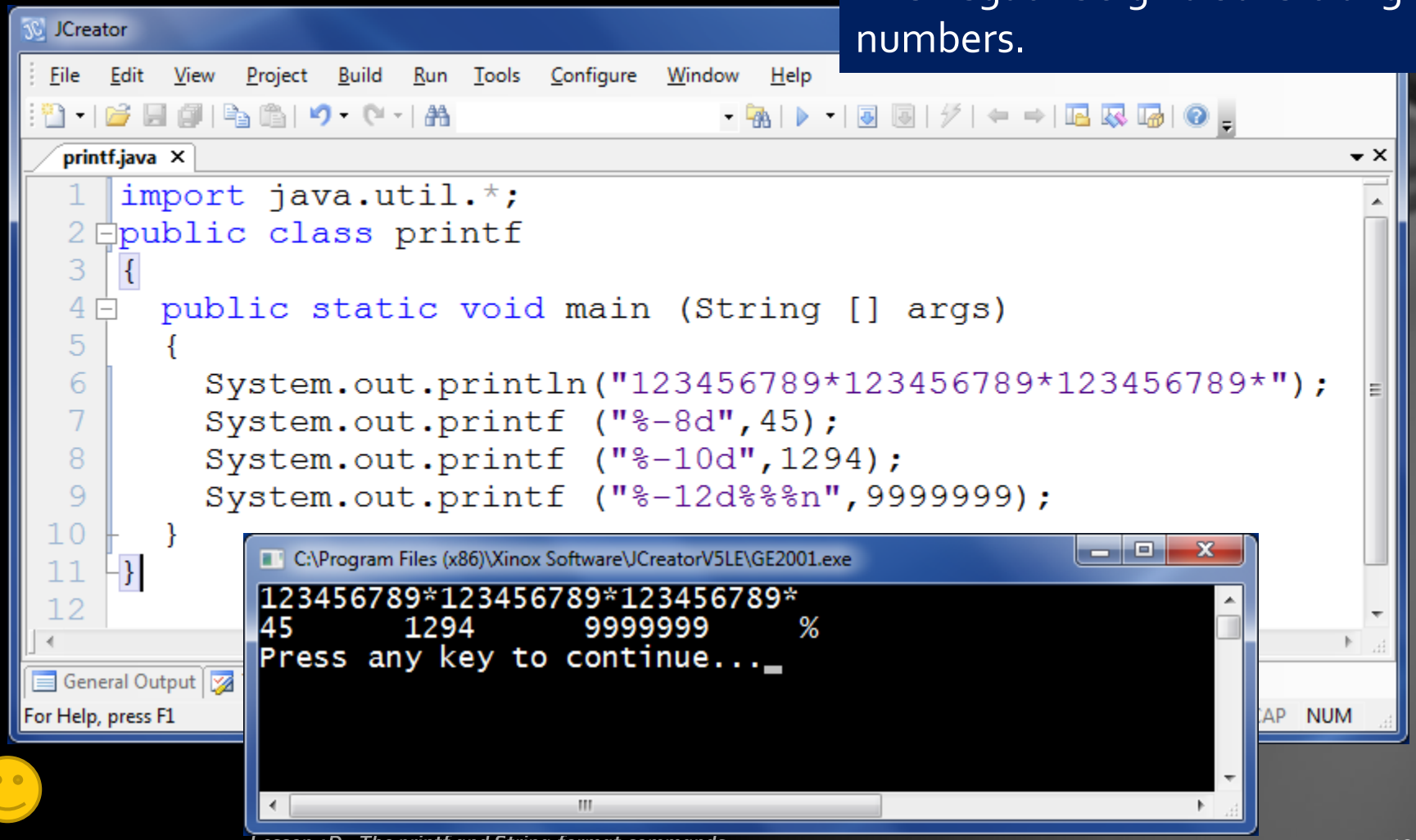
The same process works for integers as well.

Study this example carefully.

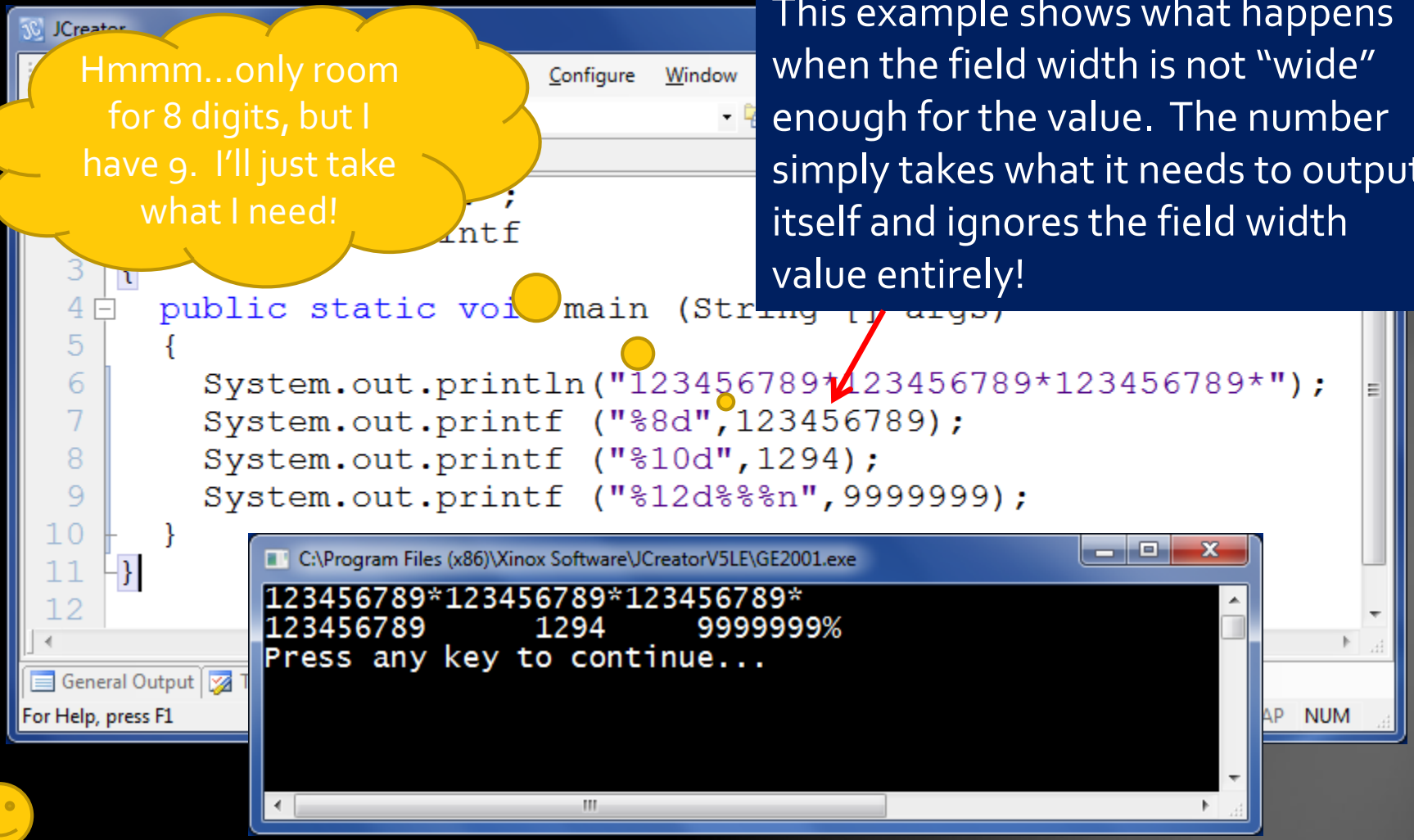


Example #5 - Left-aligned works too

The negative sign also left-aligns numbers.

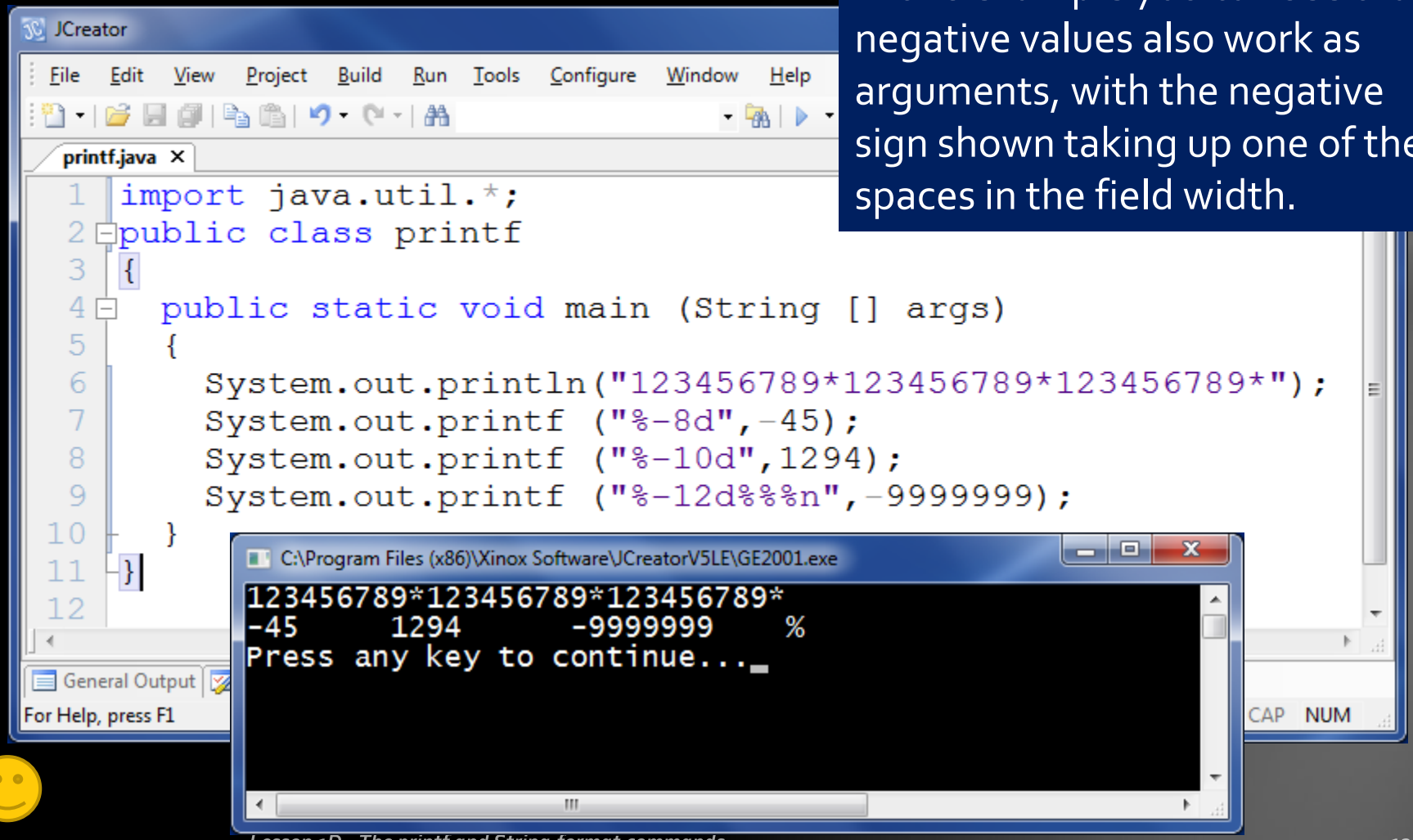


Example #6 - Field width overflow



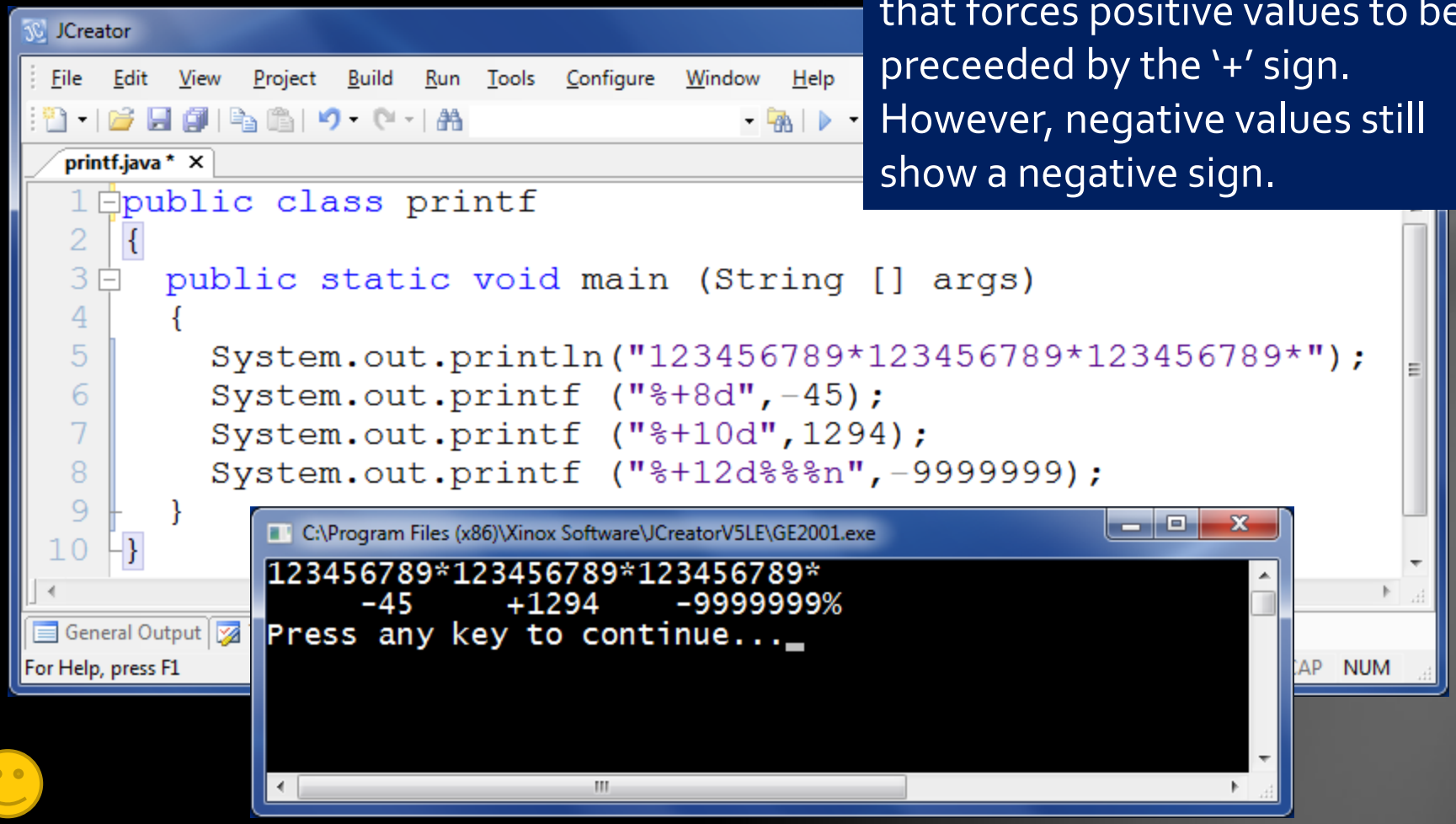
Example #7 – Negative values

In this example you can see that negative values also work as arguments, with the negative sign shown taking up one of the spaces in the field width.



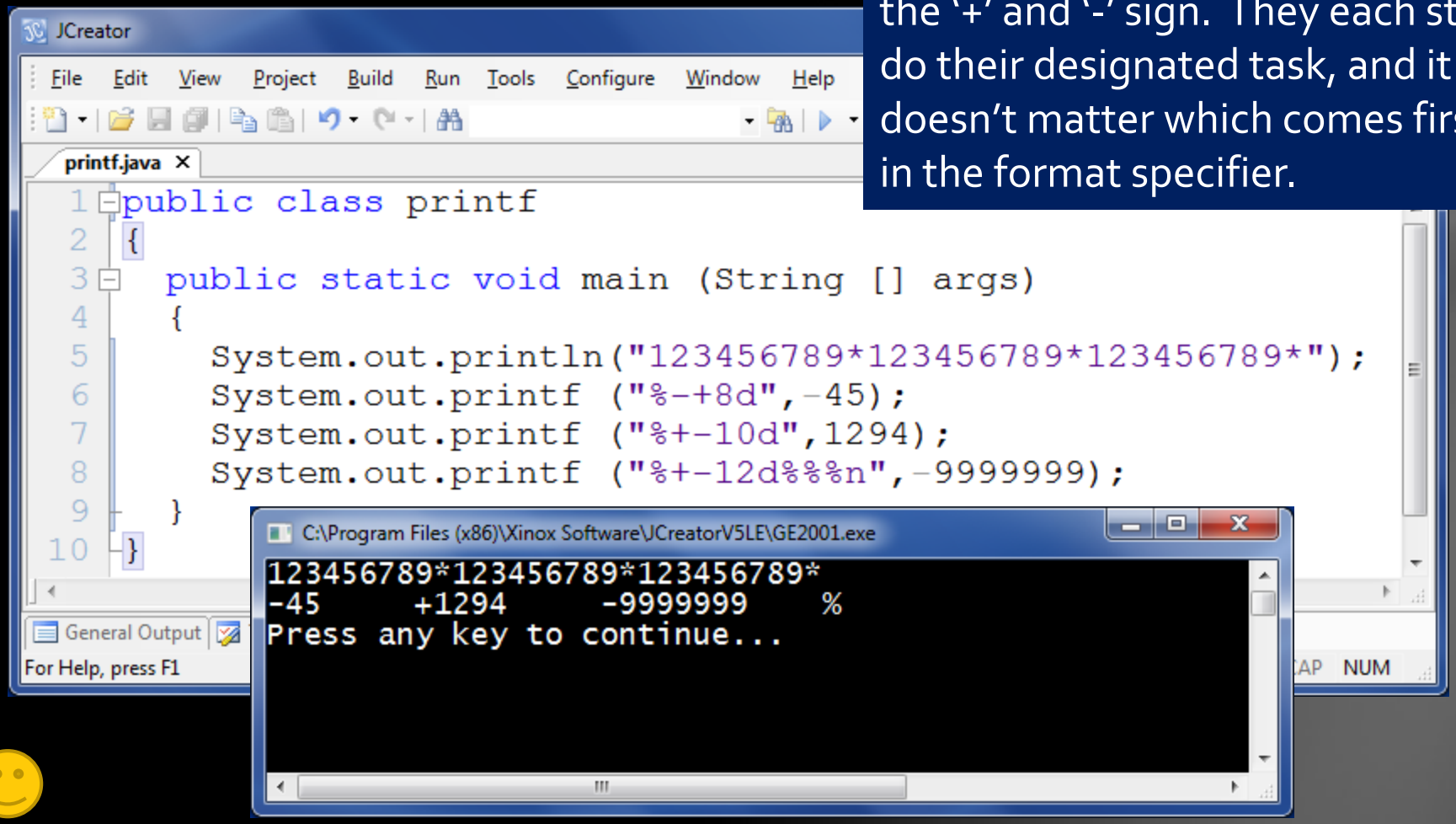
Example #8 – '+' flag shows positive

The '+' sign acts as a format flag that forces positive values to be preceeded by the '+' sign. However, negative values still show a negative sign.



Example #9 – "-+" flag combination

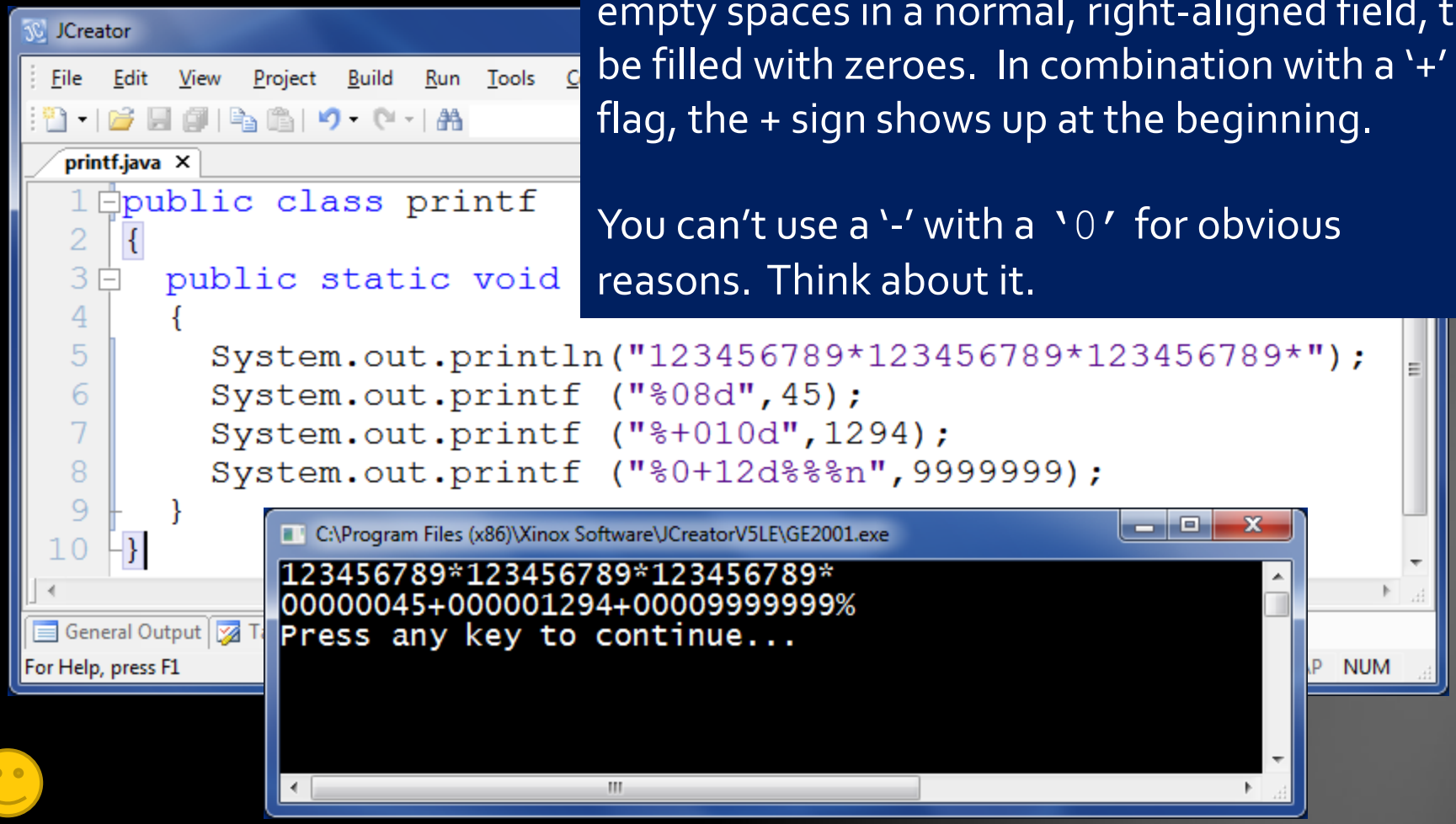
Here you see a combination of the '+' and '-' sign. They each still do their designated task, and it doesn't matter which comes first in the format specifier.



Example #10 – Zero flag '0'

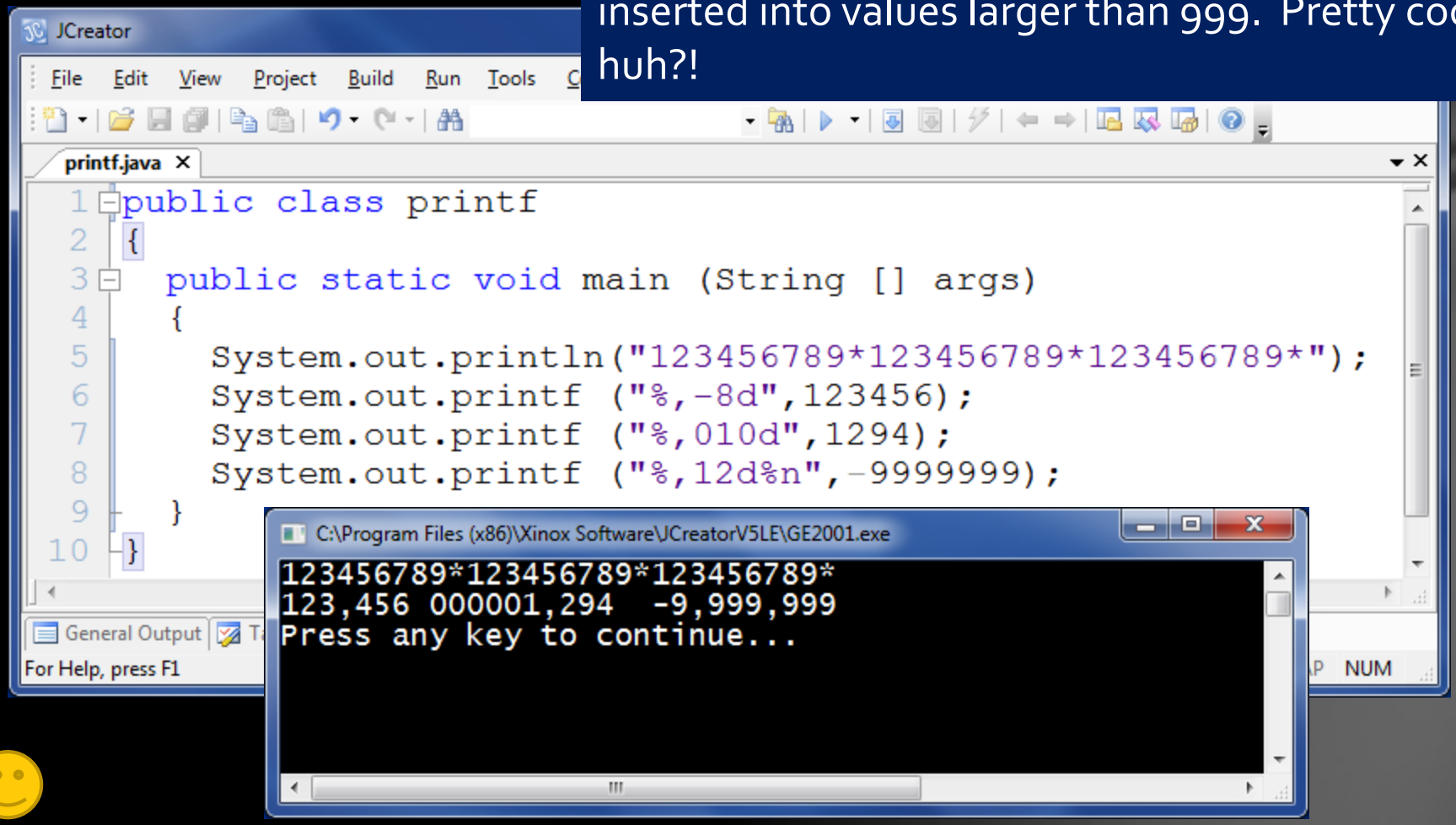
When a zero '0' is used as a flag, it causes any empty spaces in a normal, right-aligned field, to be filled with zeroes. In combination with a '+' flag, the + sign shows up at the beginning.

You can't use a '-' with a '0' for obvious reasons. Think about it.



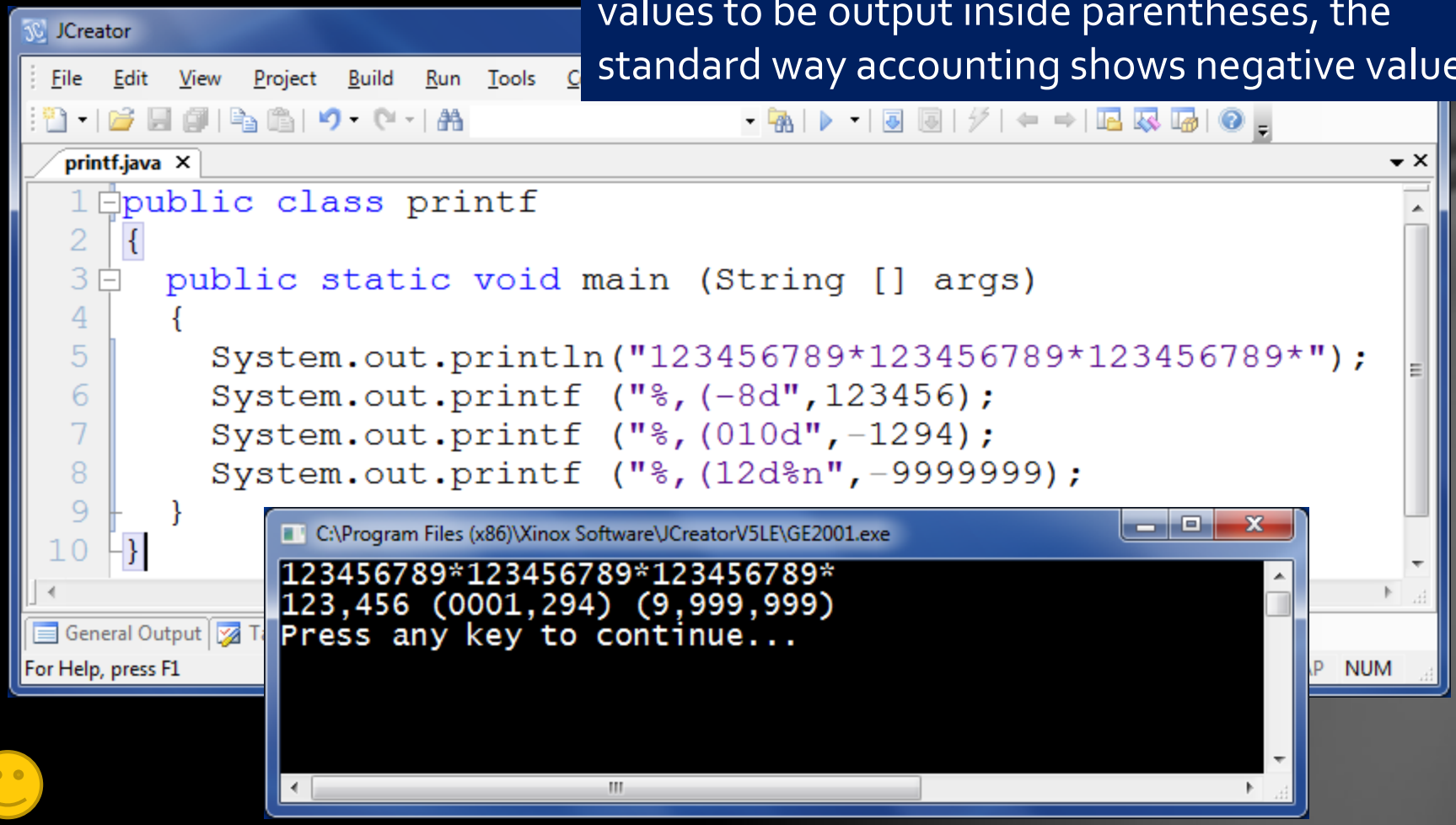
Example #11 – Comma flag ‘,’

The comma ‘,’ flag causes commas to be inserted into values larger than 999. Pretty cool, huh?!



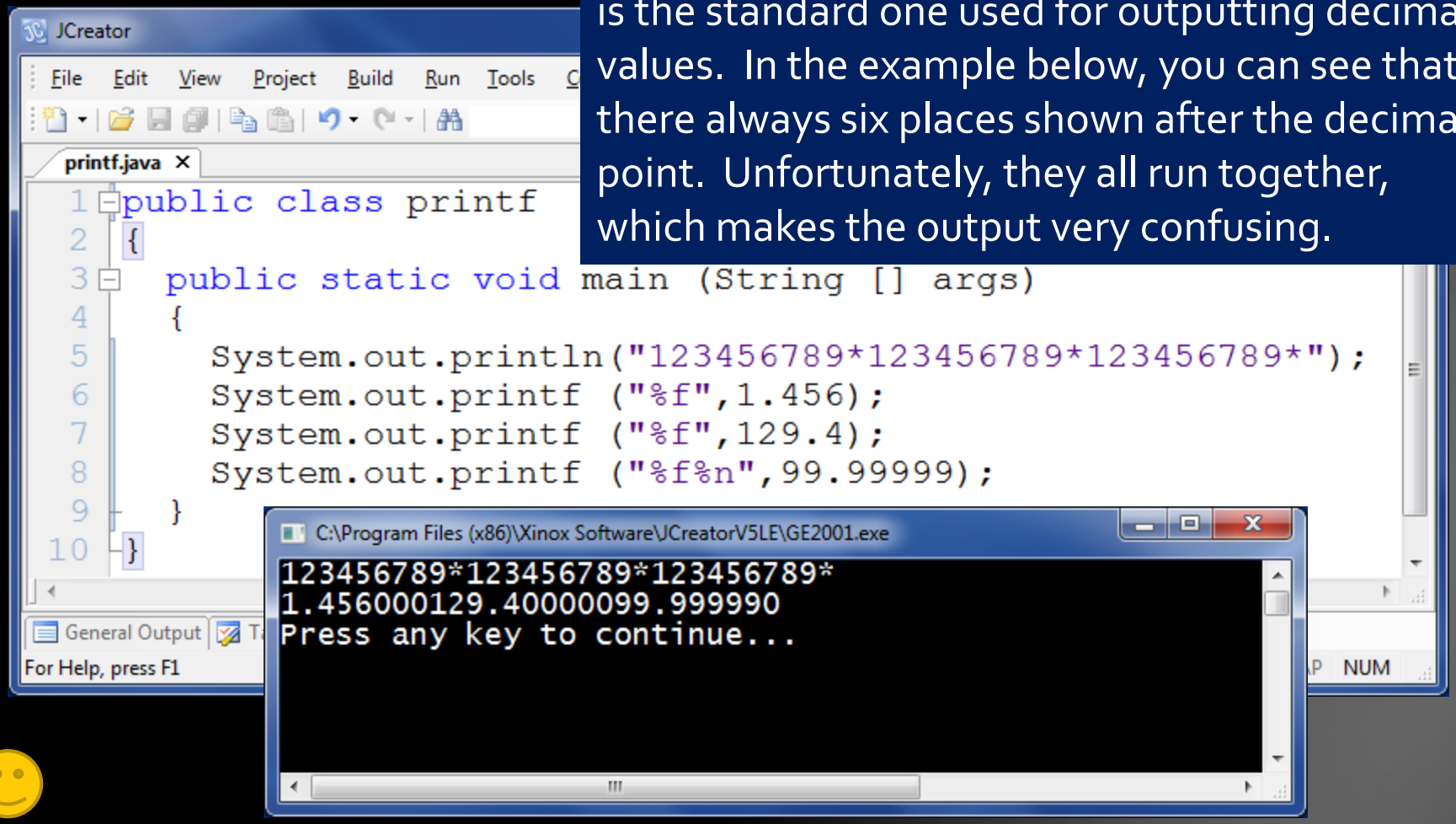
Example #12 – Parentheses flag '('

The left parentheses '(' flag causes any negative values to be output inside parentheses, the standard way accounting shows negative values.



Example #13 – Decimal values

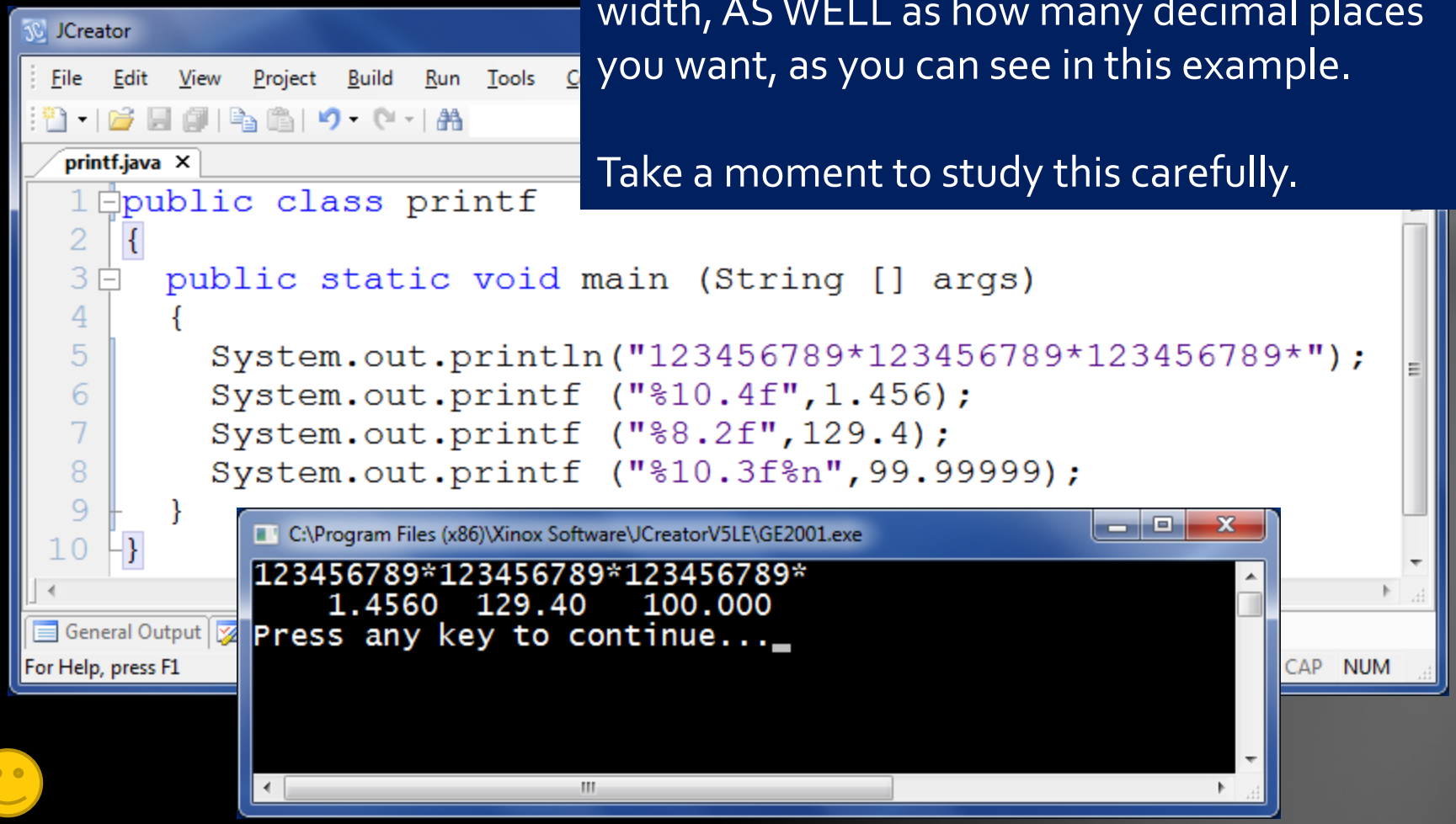
As you saw in lesson 1C, the ***%f*** format specifier is the standard one used for outputting decimal values. In the example below, you can see that there always six places shown after the decimal point. Unfortunately, they all run together, which makes the output very confusing.



Example #14 – Decimal values, formatted

Fortunately, it is also possible to control the field width, AS WELL as how many decimal places you want, as you can see in this example.

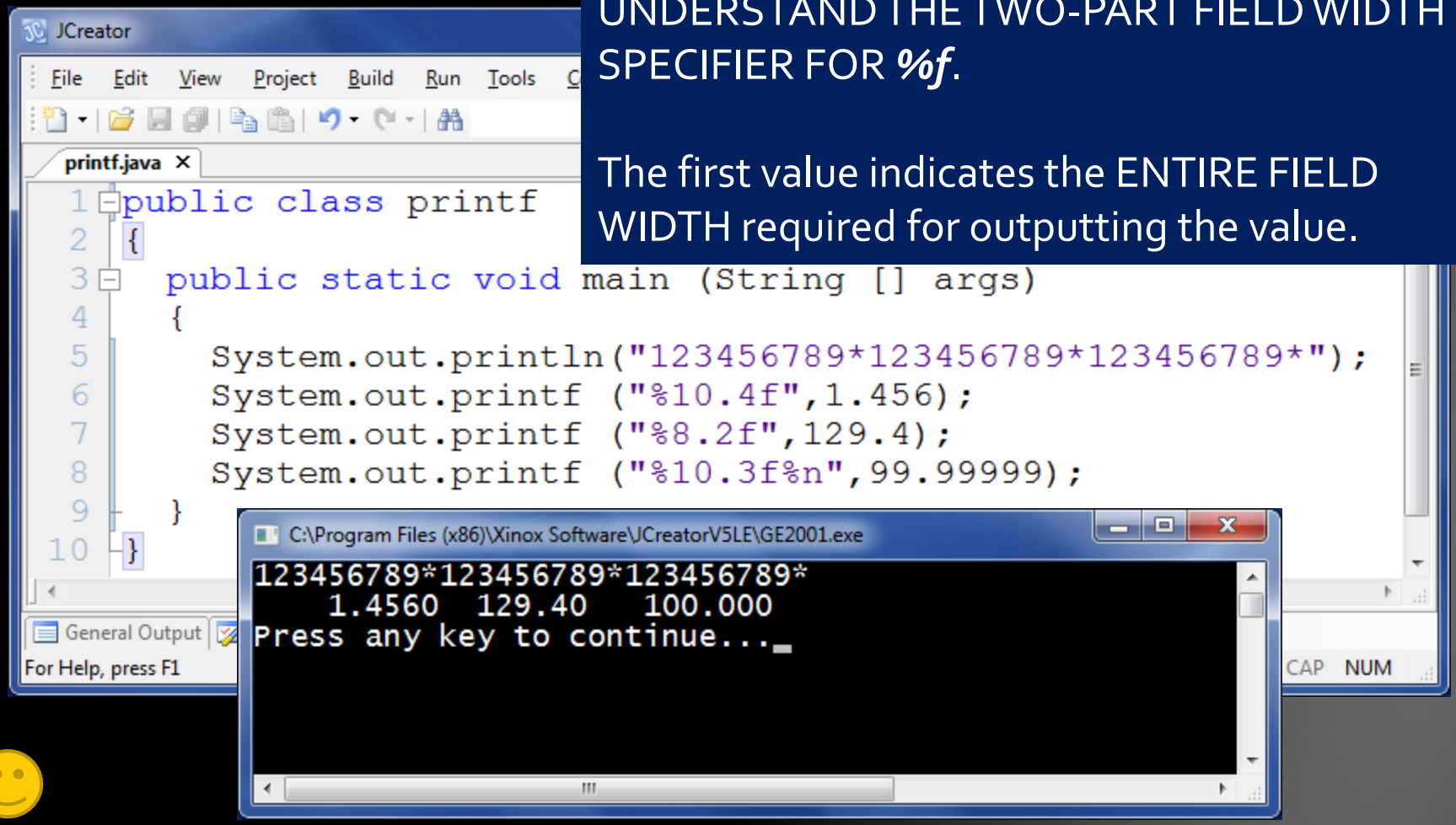
Take a moment to study this carefully.



Example #14 – Decimal values, formatted

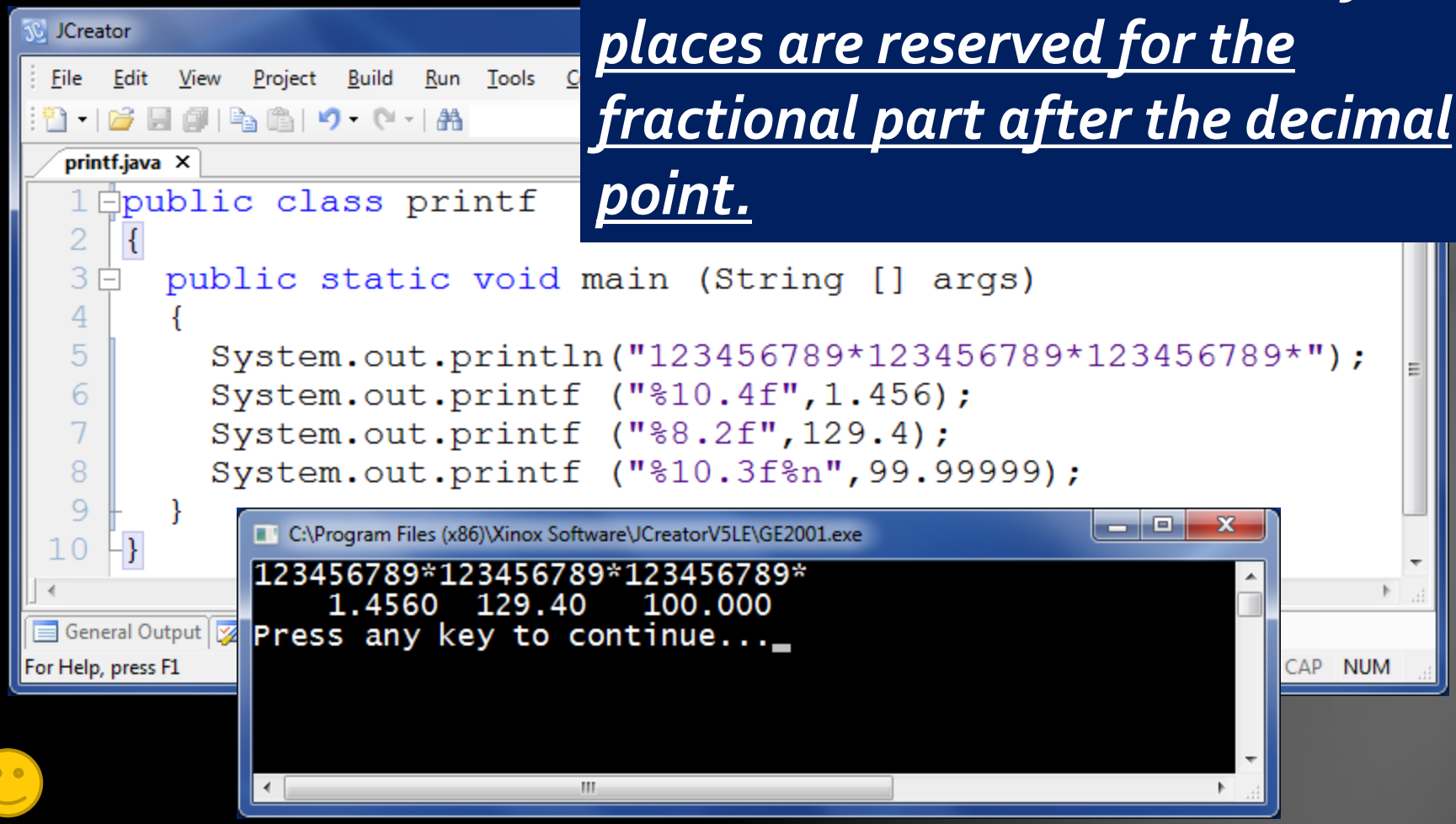
IT IS ABSOLUTELY CRUCIAL THAT YOU UNDERSTAND THE TWO-PART FIELD WIDTH SPECIFIER FOR ***%f***.

The first value indicates the ENTIRE FIELD WIDTH required for outputting the value.



Example #14 – Decimal values, formatted

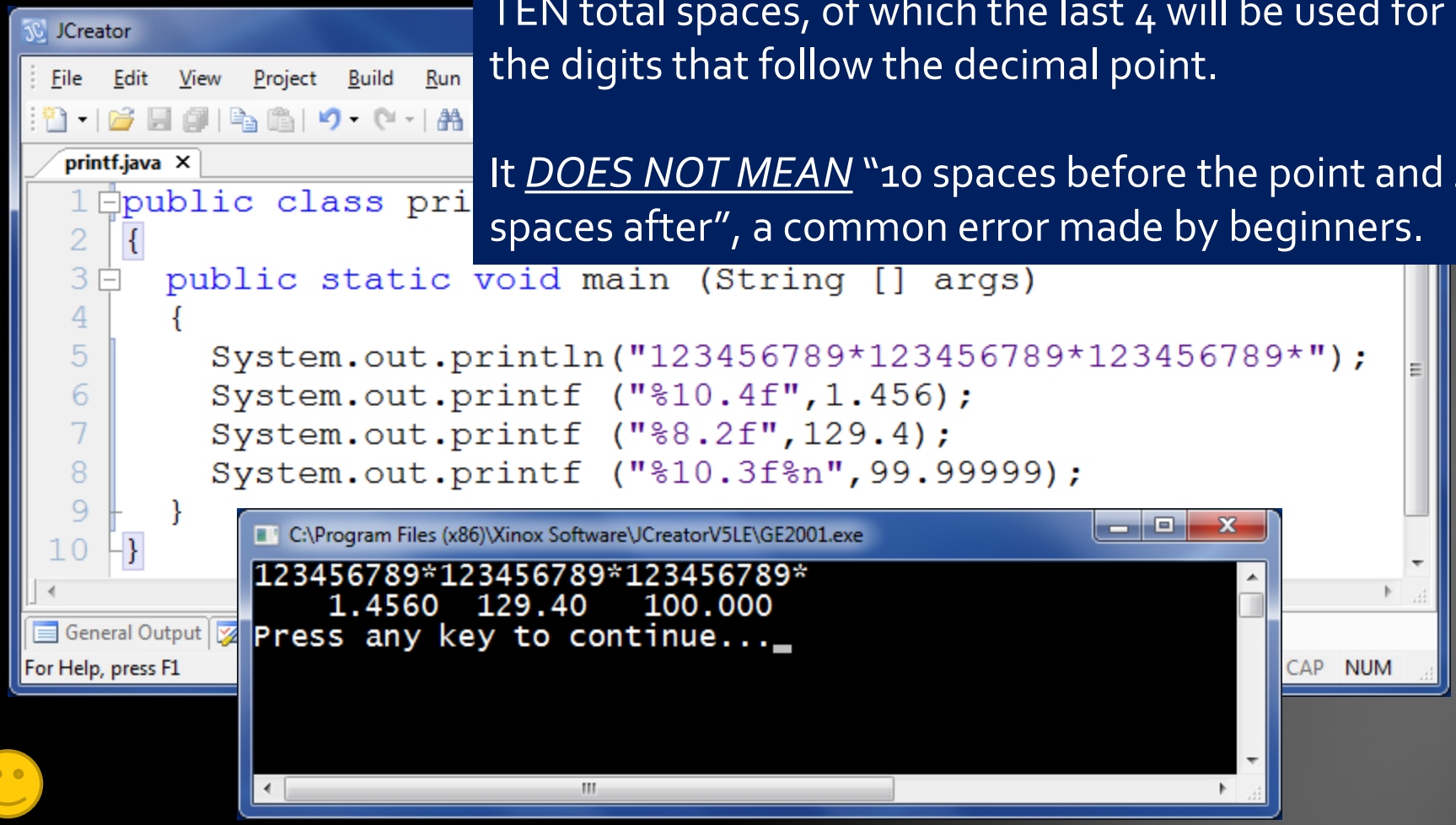
The second value indicates ***how many places are reserved for the fractional part after the decimal point.***



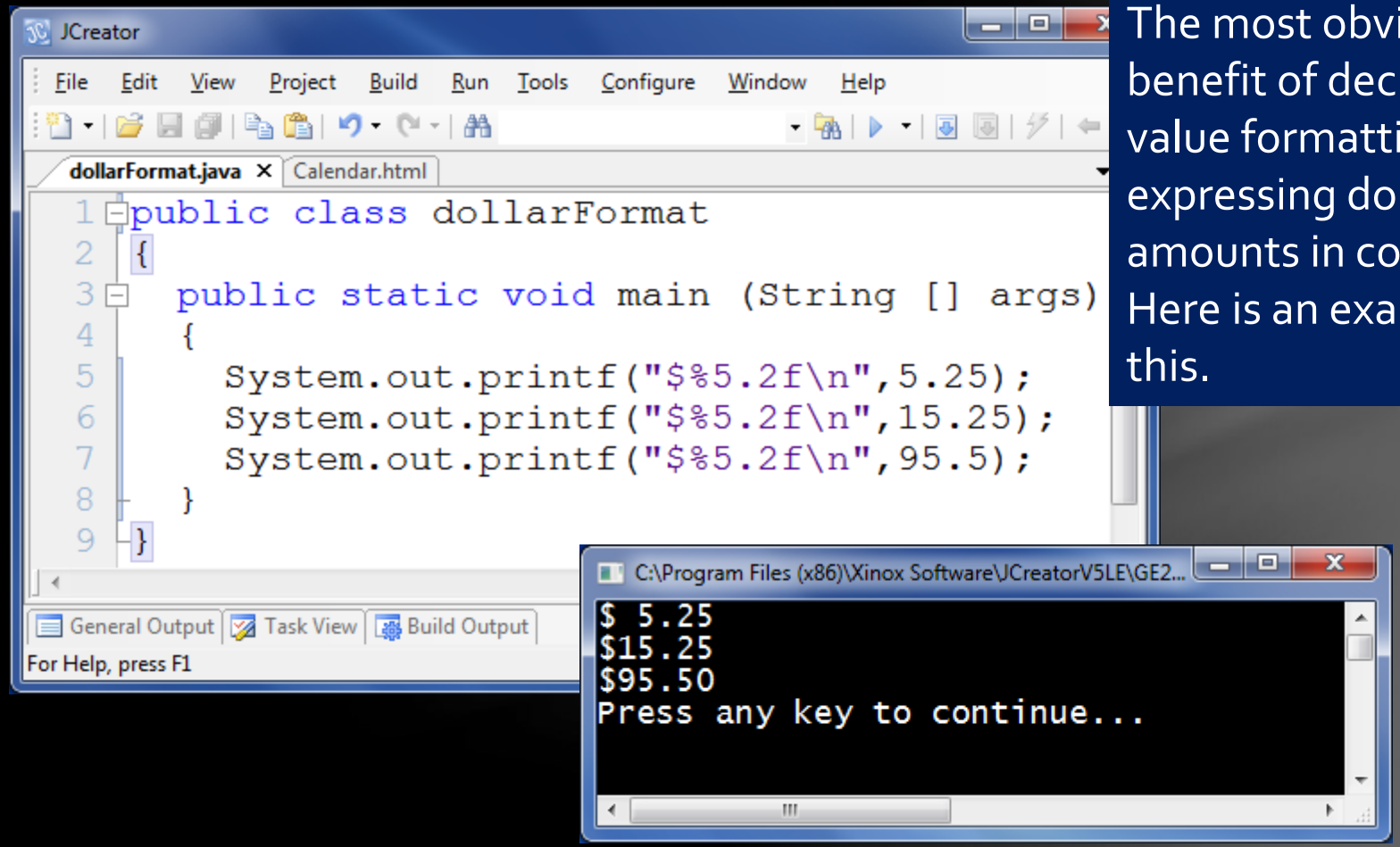
Example #14 – Decimal values, formatted

A format specifier of "%10.4f" means that there are TEN total spaces, of which the last 4 will be used for the digits that follow the decimal point.

It *DOES NOT MEAN* "10 spaces before the point and 4 spaces after", a common error made by beginners.

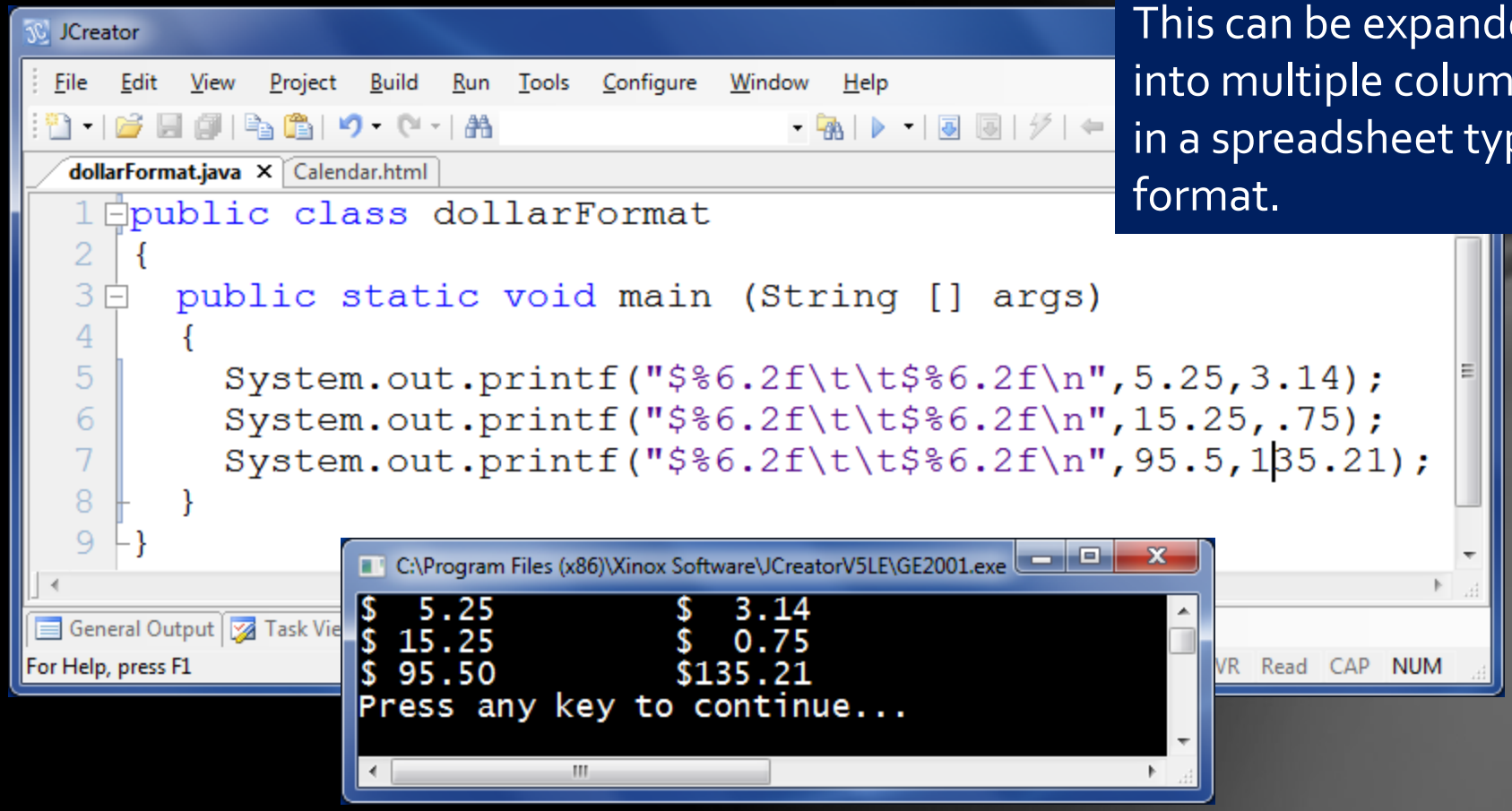


Example #15 – Dollar formatting



The most obvious benefit of decimal value formatting is expressing dollar amounts in columns. Here is an example of this.

Example #16 – Multiple columns



This can be expanded into multiple columns in a spreadsheet type format.

Calendar class suffixes

- Let's return to the Calendar class that was introduced in the last lesson.
- The format specifier for this class is %t, and is followed by a suffix character, like 'c' or 'a'
- As mentioned before, there are over 30 of these suffixes.

Calendar – more features

Here is a detailed list of more of the Calendar suffixes that work with the ***%t*** format specifier.

JCreator

File Edit View Project Build Run Tools Configure Window Help

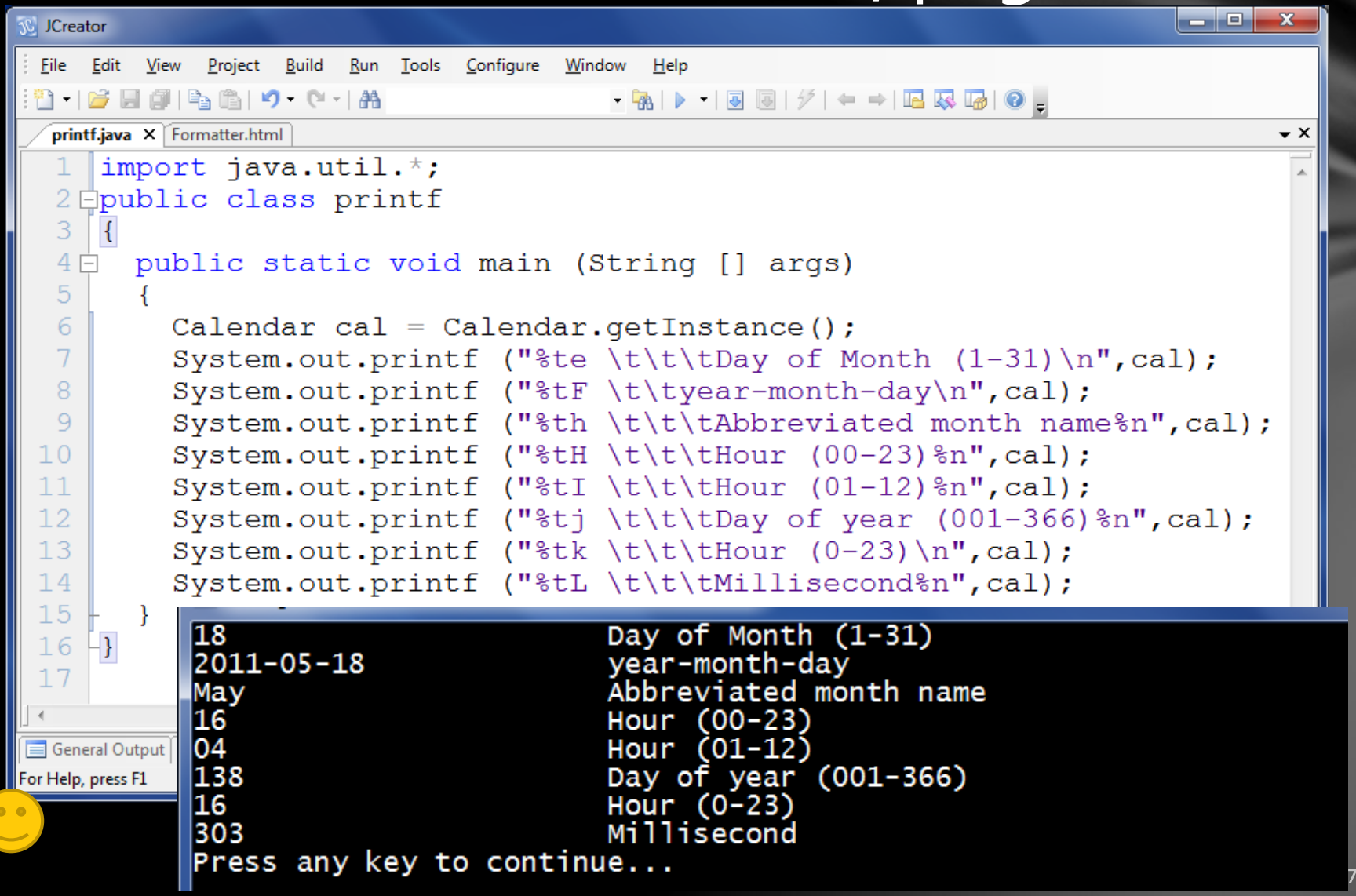
printf.java

```
1  import java.util.*;
2  public class printf
3  {
4    public static void main (String [] args)
5    {
6      Calendar cal = Calendar.getInstance();
7      System.out.printf ("%ta \t\t\t\tAbbreviated weekday name\n",cal);
8      System.out.printf ("%tA \t\t\tFull weekday name\n",cal);
9      System.out.printf ("%tb \t\t\t\tAbbreviated month name%n",cal);
10     System.out.printf ("%tB \t\t\t\tFull month name%n",cal);
11     System.out.printf ("%tc \tDate and time "
12                        +"hh:mm:ss timeZone year%n",cal);
13     System.out.printf ("%tC \t\t\t\tFirst two digits of the year%n",cal);
14     System.out.printf ("%td \t\t\t\tDay of month (01-31)\n",cal);
15     System.out.printf ("%tD \t\t\tmonth/day/year%n",cal);
```

C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe

```
Wed                             Abbreviated weekday name
Wednesday                       Full weekday name
May                             Abbreviated month name
May                             Full month name
Wed May 18 10:53:44 CDT 2011    Date and time hh:mm:ss timeZone year
20                              First two digits of the year
18                              Day of month (01-31)
05/18/11                        month/day/year
Press any key to continue...
```

Calendar – more features, page 2



Calendar – more features, page 3

JCreator

File Edit View Project Build Run Tools Configure Window Help

printf.java | Formatter.html

```
1  import java.util.*;
2  public class printf
3  {
4    public static void main (String [] args)
5    {
6      Calendar cal = Calendar.getInstance();
7      System.out.printf ("%tm \t\t\tMonth (01-13)\n",cal);
8      System.out.printf ("%tM \t\t\tMinute (00-59)\n",cal);
9      System.out.printf ("%tN \t\tNanosecond within the second"
10                        +"(000000000-999999999)%n",cal);
11     System.out.printf ("%tp \t\t\tam or pm%n",cal);
12     System.out.printf ("%Tp \t\t\tAM or PM%n",cal);
13     System.out.printf ("%tQ \t\tMilliseconds since 1/1/1970%n",cal
14     System.out.printf ("%tr \t\thh:mm:ss AM/PM (12 hr format)\n",c
15     System.out.printf ("%tR \t\t\thh:mm (24 hr format)%n",cal);
16   }
```

```
05                      Month (01-13)
11                      Minute (00-59)
624000000               Nanosecond within the second(000000000-999999999)
pm                      am or pm
PM                      AM or PM
1305756712624           Milliseconds since 1/1/1970
05:11:52 PM             hh:mm:ss AM/PM (12 hr format)
17:11                   hh:mm (24 hr format)
Press any key to continue...
```

Calendar – more features, page 4

JCreator

File Edit View Project Build Run Tools Configure Window Help

printf.java | Formatter.html

```
1  import java.util.*;
2  public class printf
3  {
4    public static void main (String [] args)
5    {
6      Calendar cal = Calendar.getInstance();
7      System.out.printf ("%ts \t\tSeconds since 1/1/1970\n",cal);
8      System.out.printf ("%tS \t\t\tSeconds (00-60)\n",cal);
9      System.out.printf ("%tT \t\thh:mm:ss (24 hour format)%n",cal);
10     System.out.printf ("%ty \t\t\tYear without century(00-99)%n",c
11     System.out.printf ("%tY \t\t\tYear with century (0001-9999)%n"
12     System.out.printf ("%ts \t\tOffset from UTC%n",cal);
13     System.out.printf ("%tZ \t\t\tTime zone name\n",cal);
14   }
15 }
```

```
1305757046              Seconds since 1/1/1970
26                      Seconds (00-60)
17:17:26                hh:mm:ss (24 hour format)
11                      Year without century(00-99)
2011                    Year with century (0001-9999)
1305757046              Offset from UTC
CDT                     Time zone name
```

Calendar class – more information

- For a more in-depth study of the Calendar class, look in the JAVA help files, or API, available from the java.sun.com website.

Lesson Summary

- This lesson introduced advanced features of the ***printf*** command specifically dealing with field width and various format flags.
- It also showed the many Calendar suffixes available with printf, producing various date and time output formats.

Calendar Lab

Write a program using the Calendar class to output a summary of your birthday, just like the one you see on the next slide. The output should match in format exactly, except for the information, of course. The first part of the program is shown to help get you started.

Calendar Lab

Below is the required output and a portion of the code that produced it. Use ***your information*** instead of mine. If you don't know the exact time you were born, just make up something.

```
import java.util.*;
public class printf
{
  public static void main (String [] ar
  {
    Calendar cal = Calendar.getInstance();
    System.out.printf("The current date and time is: %tc\n\n",cal);
    cal.set(1956,9,8,5,30,12);
    String name = "John B. Owen";
    System.out.printf ("My name is %s.  \nI was born at %Tr "
        +"on a %tA, \nthe %teth ",name,cal,cal,cal);
```

```
C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe
The current date and time is: Thu May 19 06:07:36 CDT 2011

My name is John B. Owen.
I was born at 05:30:12 AM on a Monday,
the 8th day of October in the year 1956.
This was the 282nd day of the year.

Press any key to continue...
```

Calendar Lab

Study the program carefully and **look back in the lesson** for the correct suffixes to produce the rest of the output. Have fun!

```
import java.util.*;
public class printf
{
  public static void main (String [] args)
  {
    Calendar cal = Calendar.getInstance();
    System.out.printf("The current date and time is: %tc\n\n",cal);
    cal.set(1956,9,8,5,30,12);
    String name = "John B. Owen";
    System.out.printf ("My name is %s.  \nI was born at %Tr "
        +"on a %tA, \nthe %teth ",name,cal,cal,cal);
```

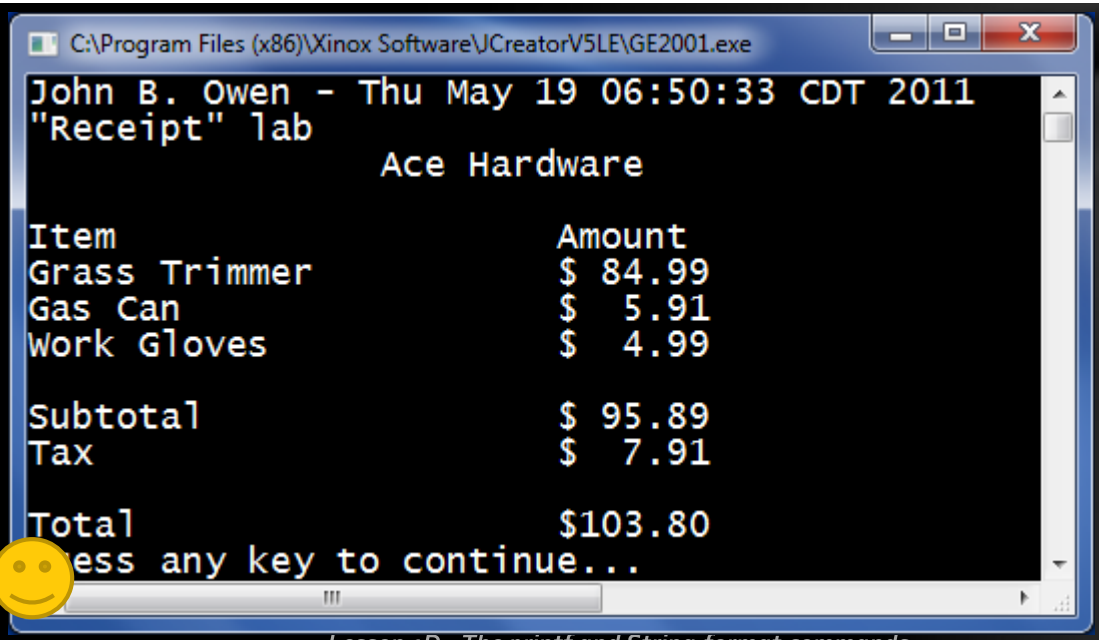
```
C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe
The current date and time is: Thu May 19 06:07:36 CDT 2011

My name is John B. Owen.
I was born at 05:30:12 AM on a Monday,
the 8th day of October in the year 1956.
This was the 282nd day of the year.

Press any key to continue...
```

Receipt Lab

Write a program that outputs a receipt from a store, with at least two but no more than three items. The output should resemble the receipt as closely as possible, with all dollar amounts preceded by the '$' sign and properly aligned at the decimal point. Include subtotal, tax, and grand total at the bottom.

Receipt Lab

```
import java.util.*;
public class receiptLab
{
  public static void main (String [] args)
  {
    Calendar cal = Calendar.getInstance();
    System.out.printf("John B. Owen - %tc\n",cal);
    System.out.println("\"Receipt\" lab");
    System.out.println("\t\tAce Hardware\n");
    System.out.println("Item\t\t\tAmount");
    System.out.printf("%s\t\t$%6.2f\n","Grass Trimmer",84.99);
    System.out.printf("%s\t\t\t$%6.2f\n","Gas Can",5.91);
    System.out.printf("%s\t\t$%6.2f\n\n","Work Gloves",4.99);
    System.out.printf("%s\t\t$%6.2f\n","Subtotal",
                      84.99+5.91+4.99);
```



Here is a portion of the program that produced my receipt lab. Study it carefully and adapt it to your own receipt. **Notice how you can "do math" inside the parameter list!** Be sure to include the "timestamp" beside your name.

CONGRATULATIONS!

- You can now control field width and decimal output, as well as use various Calendar formats in your output!
- *It is now time to move on to Lesson 2, which talks about data types and variables in much greater detail.*

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