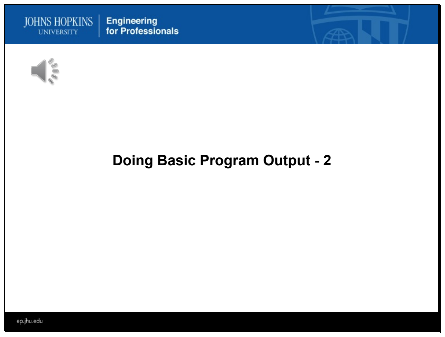
1



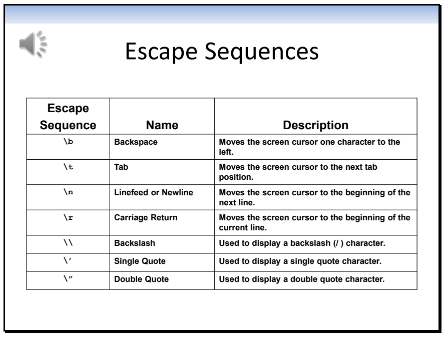
JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

Doing Basic Program Output - 2

ep.jhu.edu

In this lecture you will learn more about how to do basic program output.

2



Escape Sequences

Escape Sequence	Name	Description
`\b`	Backspace	Moves the screen cursor one character to the left.
`\t`	Tab	Moves the screen cursor to the next tab position.
`\n`	Linefeed or Newline	Moves the screen cursor to the beginning of the next line.
`\r`	Carriage Return	Moves the screen cursor to the beginning of the current line.
`\\`	Backslash	Used to display a backslash (/) character.
`\’`	Single Quote	Used to display a single quote character.
`\”`	Double Quote	Used to display a double quote character.

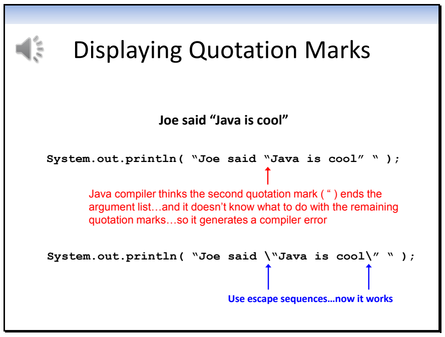
Java defines a set of special characters, called *escape sequences*, that allows it to deal with outputting certain specialized characters…like quotation marks and backslash characters…or specialized behaviors…like moving the screen cursor to the next tab position or to a new line.

An *escape sequence* consists of a backslash character followed by a character that has a special meaning to the Java compiler.

This table lists the most commonly used escape sequences.

Now let’s take a look at some examples of using escape sequences.

3



Displaying Quotation Marks

Joe said “Java is cool”

```
System.out.println( “Joe said “Java is cool” “ );
```

Java compiler thinks the second quotation mark (“) ends the argument list…and it doesn’t know what to do with the remaining quotation marks…so it generates a compiler error

```
System.out.println( “Joe said \“Java is cool\” “ );
```

Use escape sequences…now it works

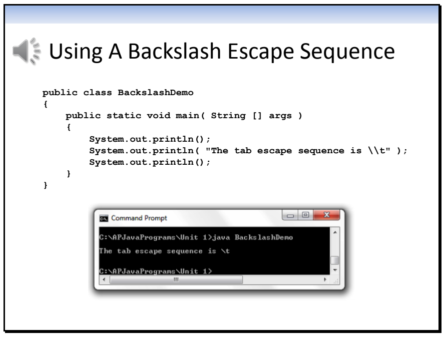
Suppose we wanted our program to output some text that contained quotation marks. How do you think we might do that?

A first thought would be to do something like this … just embed the quotes in the System.out.println() argument list.

But…this won’t work… because the Java compiler thinks the second quotation mark ends the argument list…and it doesn’t know what to do with the remaining quotation marks…so it will generate a compilation error.

What we need to do is to use the double quote escape sequence…like this .

4



Using A Backslash Escape Sequence

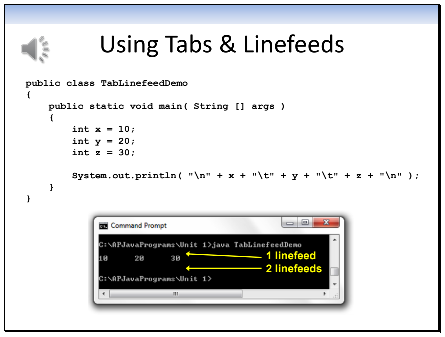
```
public class BackslashDemo
{
    public static void main( String [] args )
    {
        System.out.println();
        System.out.println( "The tab escape sequence is \\t" );
        System.out.println();
    }
}
```

```
C:\APJavaPrograms\Unit 1>java BackslashDemo
The tab escape sequence is \t

C:\APJavaPrograms\Unit 1>
```

Here’s a program that uses the backslash escape sequence so that the backslash character can be displayed as part of the program output.

The output of this program looks like this .

5



Using Tabs & Linefeeds

```
public class TabLinefeedDemo
{
    public static void main( String [] args )
    {
        int x = 10;
        int y = 20;
        int z = 30;

        System.out.println( "\n" + x + "\t" + y + "\t" + z + "\n" );
    }
}
```

```
C:\APJavaPrograms\Unit 1>java TabLinefeedDemo
10      20      30

C:\APJavaPrograms\Unit 1>
```

This program demonstrates how to incorporate tabs and linefeeds.

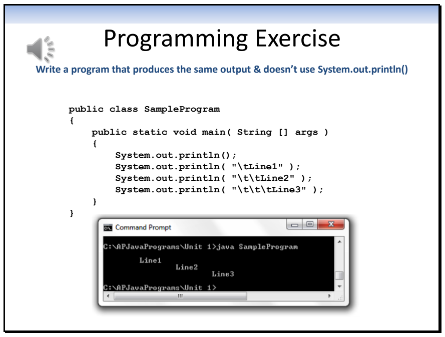
The program will print a linefeed, then the values of x, y, and z separated by a tab character, and then another linefeed.

The output will look like this

Note how the numeric values are nicely separated as a result of using the tab escape sequences.

Note also that the program generated a total of 3 linefeeds . Two linefeeds were generated as a result of using the linefeed escape sequence in the System.out.println() argument list…and another linefeed was generated automatically by the System.out.println() method itself.

6



Programming Exercise

Write a program that produces the same output & doesn't use System.out.println()

```
public class SampleProgram
{
    public static void main( String [] args )
    {
        System.out.println();
        System.out.println( "\tLine1" );
        System.out.println( "\t\tLine2" );
        System.out.println( "\t\t\tLine3" );
    }
}
```

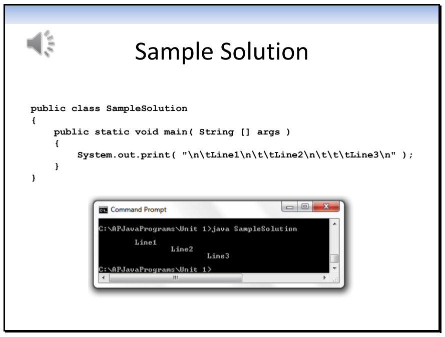
```
C:\APJavaPrograms\Unit 1>java SampleProgram

        Line1
                Line2
                        Line3

C:\APJavaPrograms\Unit 1>
```

At this point, I'd like you to do an exercise.

Consider the following program and its associated output. What I would like you to do is to write a program that produces the identical output, but doesn't use System.out.println().

Please pause this lecture now and continue it once you have completed your program.

7



Sample Solution

```
public class SampleSolution
{
    public static void main( String [] args )
    {
        System.out.print( "\n\tLine1\n\t\tLine2\n\t\t\tLine3\n" );
    }
}
```

```
C:\APJavaPrograms\Unit 1>java SampleSolution

        Line1
                Line2
                        Line3

C:\APJavaPrograms\Unit 1>
```

Here’s a sample solution.

In this example, I used a single call to the System.out.print() method. I didn’t have to put everything in a single method call…I could have used multiple method calls as well.