CMPE150.05

30.10.2018

CMPE150 ASSIGNMENT 1

Problem description: In this assignment we were asked to write a Java program to print out a sequence of ASCII cocktail glasses, “animated” to be a cocktail being drunk step by step with a straw, taking as input two parameters: glassSize as the size of the cocktail glass and strawPos as the position of the straw.

Problem solution: In order to solve this problem, I used 7 for-loops in total and 5 static methods (excluding main), 4 of them returning void as value and 1 of them returning an integer. These were, in relative order of increasing simplicity, as follows:

1) **printSpaces:** Since we need to print out a determined number of spaces many times during this problem, I created a method called printSpaces which takes as input an integer totalSpaces, and using a simple for-loop, prints out spaces totalSpaces times.
2) **returnNumberOfGlasses:** This method takes the integers glassSize and strawPos passed in by the user as arguments and using nested if/else loops, returns the number of glasses to be printed in total, an integer value.
3) **straw:** The purpose of this method is to print out the part of the straw sticking out from the top of the glass. It uses a for-loop to print out strawPos rows, and for each row it prints out (row – 1) spaces using the printSpaces method. Then it prints out the string “o”, which corresponds to the straw.
4) **glassBottom:** This method prints out the bottom part (the handle) of the glass. Again, it makes use of the printSpaces method, then prints out “- -“ for the first row. Then it uses a for loop to print out glassSize rows of “||”.
5) **glass:** It first saves the initial value of glassSize input by the user in another int variable called initialGlassSize, because we’ll have to make modifications on glassSize later and we still need to remember the initial value of the variable to later set glassSize back to its original value. It then gets the total number of glasses to be printed from the returnNumberOfGlasses method and stores it in another int variable called totalGlasses. Then it employs a for-loop, corresponding to each glass, using this value of totalGlasses. Inside the outermost loop, as the first line, we need to set glassSize back to its initial value. For the first iteration, this does not change anything but for later iterations, we will need to perform this otherwise glassSize will keep on shrinking until it eventually hits zero, and this causes unwanted outputs. It then calls the straw method to print out the straw sticking out from the top. Then there are declarations for various int variables, storing the values of the current row being printed out, current empty row being printed out and the number of spaces to be printed out to the right of the straw. It then uses another for-loop to print out “empty” rows, that is rows that have been drunk consisting of spaces and “o”s. An important thing to note here is that for each iteration, the number of spaces to the right of the straw is decreased by 2, to make the glass smaller as we move from top to bottom. The second for-loop inside the outermost for-loop is for “full” rows, consisting of asterisks. Another important thing to note about this loop is that we reduce the glassSize by 1 for each iteration, again, to make the glass smaller from top to bottom. After all the iterations for empty and full rows, when we break out of the outermost for-loop, our method calls the glassBottom method with the initialGlassSize

parameter to draw out the bottom of the glass. Note that the for-loops for empty and full rows are set in such a way that for the first glass, no empty rows are printed, then the number of empty rows increases by 1 for each glass and the number of full rows decrease by 1.

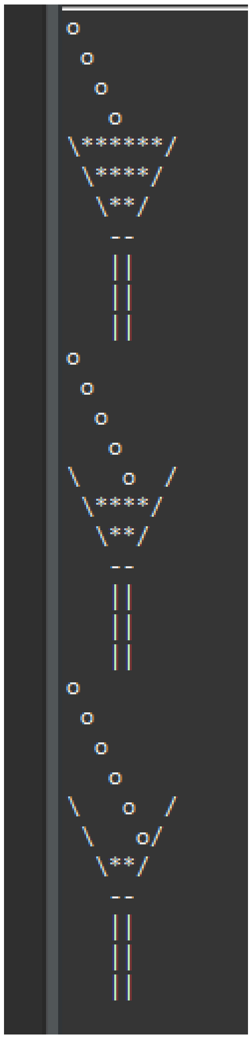
Implementation:

```
package ECK2018400018;

public class ECK2018400018
{
	public static void main(String[] args)
	{
		int glassSize = Integer.parseInt(args[0]); // Stores the first argument passed in by the user in an int variable called glassSize
		int strawPos  = Integer.parseInt(args[1]); // Stores the second argument passed in by the user in an int variable called strawPos
		
		// Prints out the expected output
		glass(glassSize, strawPos);
	}
	
	public static void glass(int glassSize, int strawPos)
	{
		// Save the initial value of glassSize inside another variable because we'll later make operations on glassSize
		int initialGlassSize = glassSize;
		
		// Get the total number of glasses to print from returnNumberOfGlasses method and assign it to an integer called totalGlasses
		int totalGlasses = returnNumberOfGlasses(glassSize, strawPos);
		
		// A for-loop that will print out NumberOfGlasses many glasses.
		for (int currentGlass = 1; currentGlass <= totalGlasses; currentGlass++) // currentGlass shows at any time the number of the glass currently being printed
		{
			// For each time the loop iterates, we need to set the glassSize variable back to its original value
			glassSize = initialGlassSize;
			
			// Call the straw method to print out part of the straw sticking out from the top
			straw(strawPos);
			
			int currentTotalRow = 0; // Declare a variable of type int which will store the number of the total (counting both empty and full rows) row currently being printed
			int currentEmptyRow = 0; // Declare a variable of type int which will store the number of the empty row currently being printed
			int totalRightSpaces = ((2 * glassSize) - strawPos); // Declare a variable of type int which shows the number of spaces to be printed to the right of the straw in an empty row
			
			// Another for-loop that will print out empty rows for glasses
			for ( ; currentEmptyRow < (currentGlass - 1); currentEmptyRow++, totalRightSpaces -= 2, currentTotalRow++)
```

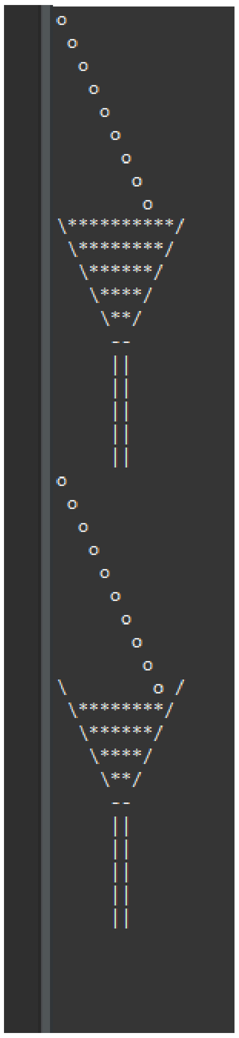
```
                    {
                            // Print spaces at the beginning of the row
                            printSpaces(currentEmptyRow);

                            System.out.print("\\");

                            // Print out (strawPos - 1) spaces to the left of the
straw
                            printSpaces(strawPos - 1);

                            System.out.print("o");

                            // Print out totalRightSpaces spaces to the right of the
straw
                            printSpaces(totalRightSpaces);

                            System.out.println("/");
                    }

                    // Another for-loop that will print out full rows for glasses
                    for (int currentFullRow = 0; currentFullRow < (initialGlassSize -
currentGlass + 1); currentFullRow++, glassSize--, currentTotalRow++) // Declare a
variable of type int which will store the number of the full row currently being
printed
                    {
                            printSpaces(currentTotalRow);

                            // Left edge of the glass
                            System.out.print("\\");

                            // For-loop for printing out (2 * (glassSize -
currentEmptyRow)) many asterisks
                            for (int asterisks = 0; asterisks < (2 * (glassSize -
currentEmptyRow)); asterisks++) // asterisks is a counter integer counting up to the
number of asterisks in a row (2 * (glassSize - currentEmptyRow)
                            {
                                    System.out.print("*");
                            }

                            // Right edge of the glass
                            System.out.println("/");
                    }

                    // Call the glassBottom method with the initial size of the glass
as input to print out the bottom part of the glass
                    glassBottom(initialGlassSize);
            }
    }

    // This method takes two inputs, the integers glassSize and strawPos and
performing certain arithmetic operations on them, calculates & returns the number of
glasses to be printed
    public static int returnNumberOfGlasses(int glassSize, int strawPos)
    {
```

```
		// If strawPos is less than or equal to half the glassSize, the number
of glasses to be printed is just (glassSize + 1)
		if ((2 * strawPos) <= glassSize)
		{
			return glassSize + 1;
		}

		// Otherwise, we need to go through a more complex process to find out
the number of glasses to be printed
		else
		{
			// We need to check if strawPos is an even number
			if (strawPos % 2 == 0)
			{
				// If strawPos happens to be an even number, the number of
glasses to be printed is expressed by ((((2 * glassSize) - strawPos) / 2) + 2) (I
found this relation by drawing a value-table on paper)
				return (((2 * glassSize) - strawPos) / 2) + 2;
			}
			else
			{
				// Otherwise if strawPos is an odd number, the number of
glasses to be printed is expressed by ((((2 * glassSize) - strawPos) / 2) + 1),
rounded UP to the nearest integer (I also found this relation by drawing a value-
table on paper)
				return (int) Math.ceil((((2.0 * glassSize) - strawPos) /
2) + 1);
			}
		}
	}

	// This method prints out the bottom part of the glass (the handle)
	public static void glassBottom(int glassSize)
	{
		// Print out glassSize spaces at the beginning of the row
		printSpaces(glassSize);

		System.out.println("--");

		for (int rows = 0; rows < glassSize; rows++) // rows is a counter
integer that counts up to the number of rows for the handle, which is equal to
glassSize
		{
			// Print out glassSize spaces at the beginning of each row
			printSpaces(glassSize);

			System.out.println("||");
		}
	}

	// This method prints out the part of the straw sticking out from the top
	public static void straw(int strawPos)
	{
		// For-loop for printing out the rows of the straw
		for (int rows = 1; rows <= strawPos; rows++) // rows is a counter
integer that counts up to the number of rows for the straw sticking out from the top,
which is equal to strawPos
		{
			printSpaces(rows - 1);
			
			System.out.println("o");
		}
	}
	
	// Since we need to print out spaces many times while printing out the
glasses, this method using a for-loop to print out a specified number of spaces will
be useful
	public static void printSpaces(int totalSpaces)
	{
		for (int spaceCount = 0; spaceCount < totalSpaces; spaceCount++)
//spaceCount is a counter that's used to count up to the wanted number of spaces (the
value input when calling the method)
		{
			System.out.print(" ");
		}
	}
}
```

<u>Output of the program:</u>



```
o
 o
  o
   o
\******/
 \****/
  \**/
   --
   ||
   ||
   ||
o
 o
  o
   o
\   o  /
 \****/
  \**/
   --
   ||
   ||
   ||
o
 o
  o
   o
\   o  /
 \   o/
  \**/
   --
   ||
   ||
   ||
```



```
o
 o
  o
   o
    o
     o
      o
       o
        o
\**********/
 \********/
  \******/
   \****/
    \**/
     --
     ||
     ||
     ||
     ||
     ||
o
 o
  o
   o
    o
     o
      o
       o
        o
\        o /
 \********/
  \******/
   \****/
    \**/
     --
     ||
     ||
     ||
     ||
     ||
```

<u>Conclusion:</u> I’ve correctly solved the problem as my program gives the correct, expected output for all valid inputs of glassSize and strawPos, as specified in the assignment description.