Mahmut Alp Erkent

27.10.2019

Assignment #1

1) Problem description:

The problem asks for a Java program that takes two arguments as input, *stickmanHeight* and *stairHeight*, and use these arguments as parameters for the program to print ASCII images of a stickman of specified height that is climbing a stairway with the specified height frame by frame.

2) Problem solution:

To solve this problem, having two different methods for printing a stairway and a stickman separately is not possible using only for loops, because there are some lines that have parts from both objects, and internal structure of these lines change according to the arguments fed into the problem. Therefore, I decided to split the problem into two methods in a different way:

- one method called *stickmanAndStairs* that takes in parameters for *stickmanHeight*, *stairHeight* and *frame*, and prints out the stickman figure and parts of the stairway that line with it,
- another method called *onlyStairs* that takes in parameters for *stairHeight* and *frame*, and prints the part of the stairway that stays below the stickman figure.

I could have merged these two methods under one method, but I wanted to separate them because this separation emphasizes my approach for handling the problem and makes the code easier to read.

After I completed the program, I decided to modify it by adding another method to make it more modular:

- the method *space1* takes in a parameter for *frame* and prints the required number of spaces before the stickman figure at each line according to the number of frames the program is printing. Since I needed to use this for loop at each line that has the stickman figure in *stickmanAndStairs* method, I decided to turn it into a separate method instead of writing the same for loop over and over and execute it easily when needed.

The content of the code is explained in an orderly fashion as follows:

In the *main* method:

- I used a for loop that introduces a variable *frame* to be fed into these methods and prints out an appropriate frame in each iteration. Since the number of frames is always one more than the height of stairs, I started the loop from 0 and repeated it until *frame* equals to *stairHeight*.

Mahmut Alp Erkent

27.10.2019

In the *stickmanAndStairs* method:

- The *header* for loop prints the empty lines above each frame. Since the number of empty lines is equal to *stairHeight* at the first frame and decreases by one at each following frame, I structured it as such.
- The *space1* method is executed at the beginning of each line that *stickmanAndStairs* method prints out.
- The *torso1* for loop prints the part of stickman's trunk that stays above the stairway. Since the number of iterations needed to print this part is always equal to *stickmanHeight-stairHeight-3* at the first frame and decreases by one at each following frame, I structured it as such.
- The *level* nested for loop prints the part of each frame that includes both the stickman and the stairway. Since the number of iterations needed to print this part is always equal to *stairHeight* at the first frame and decreases by one at each following frame, I structured it as such. Each line printed with *level* for loop is structured this way:

  space1 + (either *torso2* or *foot*) + *space2* + ___| + *stars* + |

  The *level* nested for loop includes these for loops:
  - The *torso2* for loop prints the part of stickman's trunk that stays in line with the stairway. Since stickman's trunk is always printed in each line except the last (where the *foot* for loop is printed), I structured it as such. Then I set the update condition in a way that *torso2* is printed only once for each iteration of the *level* for loop.
  - The *foot* for loop prints the feet of the stickman. Since the feet are printed only when *level* is equal to *stairHeight-frame*, I structured it as such.
  - The *space2* for loop prints the spaces between the stickman and the stairway. Since the number of spaces needed is always equal to *stairHeight**3 at the first line of the first frame and decreases by three for each line and for each frame, I structured it as such.
  - The *stars* for loop prints the stars for each line of the stairway. Since the number of stars needed is always equal to *level**3, I structured it as such.

In the *onlyStairs* method:

- The *level* nested loop prints the part of the stairway that stays below the stickman for each frame. Since the number of iterations needed to print this part is always equal to zero at the first frame and increases by one at each following frame, I structured it as such. Each line printed with *level* for loop is structured this way:

  space + ___| + *stars* + |

  The *level* nested for loop includes these for loops:
  - The *space* for loop prints the spaces before the stairway. Since the number of spaces needed is always equal to *frame**3 at the first line of each frame and decreases by three for each following line, I structured it as such.
  - The *stars* for loop prints the stars for each line of the stairway. Since the number of stars needed at the bottom of the stairway is always equal to *stairHeight**3 and decreases by three as the stairway level goes up, I structured it as such.
- There is also a *println* statement at the end of this method that prints three empty lines each time this method is executed.

Mahmut Alp Erkent

27.10.2019

In the *spaces1* method:

- The *space1* for loop prints the spaces before the stickman figure at each line according to the frame number. Since the number of spaces is always equal to zero at the first frame and increases by three for each following frame, I structured it as such.

The code has 12 for loops.

3) Output of the program:

Arguments: 5, 2

```
 O
/|\
 |       ___||
 |    ___|***|
/ \___|******|



    O
   /|\
    |
    |    ___||
   / \___|***|
   ___|******|



       O
      /|\
       |
       |
      / \___||
      ___|***|
   ___|******|


|
```

Mahmut Alp Erkent

27.10.2019

Arguments 7, 3

```
 O
/|\
 |
 |          ___||
 |       ___|***|
 |    ___|******|
/ \___|*********|
```

```
    O
   /|\
    |
    |
    |       ___||
    |    ___|***|
   / \___|******|
   ___|*********|
```

```
       O
      /|\
       |
       |
       |
       |    ___||
      / \___|***|
      ___|******|
   ___|*********|
```

```
          O
         /|\
          |
          |
          |
          |
         / \___||
         ___|***|
      ___|******|
   ___|*********|
```

Mahmut Alp Erkent

27.10.2019

4) Conclusion:

With the code provided, I believe I have solved the problem correctly for each possible case. I tried to make my code as modular as possible, using the least amount of for loops.