Char Generator

CSE 174 – Fall 2021

Lab8: 14 points – Due Saturday, Oct 16, 2021 by 11:59pm

Part 1.1 - Nested Loop:

Using the nested loops, write a program that gets an int value from the user, and generates an output as shown below:

```
Enter an int number: 8
OXXXXXXX
OOXXXXXX
OXOXXXXX
OOXOXXXX
OXXXOXXX
OOOXXOXX
OXXXXXOX
OOXOXXXO
```

A square of X and Os with 8 rows and 8 columns, since the user entered 8.

- 'O' is printed where the number of rows are divisible by the number of columns.
- 'X' is printed where the number of rows are **not** divisible by the number of columns.

Examples:

- Looking at the first row, only the first column is 'O' because column is 1, row is 1, which means the row number (1) is divisible by the column number (1).
- Looking at the third row, only the first and third columns are 'O' because row 3 is divisible by those column numbers (which are 1 and 3).

More sample runs:

```
Enter an int number: 12
OXXXXXXXXXXX
OOXXXXXXXXXX
OXOXXXXXXXXX
OOXOXXXXXXXX
OXXXOXXXXXXX
OOOXXOXXXXXX
OXXXXXOXXXXX
OOXOXXXOXXXX
OXOXXXXXOXXX
OOXXOXXXXOXX
OXXXXXXXXXOX
OOOOXOXXXXXO
```

Continue on the Next Page!

```
Enter an int number: 4
OXXX
OOXX
OXOX
OOXO
```

Part 1.2 - Repeat the Program:

After printing the output, your program should ask the user whether or not the program should be repeated.

- If the user answers ‘y’ or ‘Y’: the program should be repeated again.
- If the user answers ‘n’ or ‘N’: the program should print the “End” message and stop.
- If the user answers anything else: the program should repeat the question again.
- Use a **do-while loop** for repeating the program. Can you come up with the reason why using do-while loop is a better option here compared to for-loop or while-loop? If you can’t think of any reason, please ask your instructor about that.

Sample Runs:

```
----jGRASP exec: java Asterisk
Enter an int number: 4
OXXX
OOXX
OXOX
OOXO
Do you want to repeat (y/n)? yeS
Do you want to repeat (y/n)? YEs
Do you want to repeat (y/n)? y
Enter an int number: 5
OXXXX
OOXXX
OXOXX
OOXOX
OXXXO
Do you want to repeat (y/n)? Y
Enter an int number: 6
OXXXXX
OOXXXX
OXOXXX
OOXOXX
OXXXOX
OOOXXO
Do you want to repeat (y/n)? No
Do you want to repeat (y/n)? N
End

----jGRASP: operation complete.
```

Continue on the Next Page!

```
   ----jGRASP exec: java Lab6P1
Enter an int number: 0
Do you want to repeat (y/n)? n
End

   ----jGRASP: operation complete.
```

Submission:

After testing your code thoroughly, when you believe that you have correctly solved the problem, and that your code follows the style guidelines, submit it on canvas. If the autograder rejects your solution, fix it and resubmit it again.

Scoring Rubric (Full Rubric on Canvas):

- **Successful Submission via CODE (2 points):** You will receive full credit for this if your code passes all the tests. Otherwise you will get zero or partial credit.
- **Hidden Test (2 points):** You will receive full credit for this if your code passes all the additional tests. Otherwise you will get zero or partial credit.
- **Correct Style (2 points):** You will receive full credit if your code has 0 style errors. Also your code has comments at the top at least with your name and description, and some comments inside the code to explain the code.
- **Proper Programming Practices (2 Points):** If your program is written applying all of the programming practices **covered in class:**
  - Variable naming
  - Proper logic
  - Using nested loops properly
  - Using proper conditions
  - **Using do-while loop for repeating the program**
  - Proper use of String methods
  - Not using things that aren't discussed in the class

Part2 on the Next Page!

Part 2 - Exception Handling:

Before doing this part, make sure that you already submitted your code for part1 and already passed all the tests. For part2, you should complete your code from Part 1 by adding new codes.

A. Catch the exception where the user enters anything but an int value e.g. if the user enters “hi” or a double value such as 12.56 the following should happen:
   - The message “Invalid Input!” should be printed.
   - Ask the user to re-enter an int number.

   Examples:

```
----jGRASP exec: java Lab6P2
Enter an int number: hi
Invalid Input!
Enter an int number: 12.4
Invalid Input!
Enter an int number: 8test
Invalid Input!
Enter an int number: 3
OXX
OOX
OXO
Do you want to repeat (y/n)?
```

B. If the user enters a negative number or zero the following should happen:
   - **The program should throw an IllegalArgumentException**.
   - Before generating the output, this exception **needs to be catched** and the message “No result with an input less than or equal to zero!” should be printed.
   - Keep that in mind the program should not ask the user to re-enter an int number like part A.

   Examples:

```
----jGRASP exec: java Lab6P2
Enter an int number: 0
No result with an input less than or equal to zero!
Do you want to repeat (y/n)? Y
Enter an int number: -10
No result with an input less than or equal to zero!
Do you want to repeat (y/n)? y
Enter an int number: 3
OXX
OOX
OXO
Do you want to repeat (y/n)?
```

Instructions Continue on Next Page!

Sample Runs:

```
 ----jGRASP exec: java Lab6P2
Enter an int number: 3
OXX
OOX
OXO
Do you want to repeat (y/n)? y
Enter an int number: 0
No result with an input less than or equal to zero!
Do you want to repeat (y/n)? y
Enter an int number: -2
No result with an input less than or equal to zero!
Do you want to repeat (y/n)? Y
Enter an int number: test
Invalid Input!
Enter an int number: 2
OX
OO
Do you want to repeat (y/n)? Yes
Do you want to repeat (y/n)? No
Do you want to repeat (y/n)? y
Enter an int number: 5
```

```
OXXXX
OOXXX
OXOXX
OOXOX
OXXXO
Do you want to repeat (y/n)? N
End

 ----jGRASP: operation complete.
```

Test your code comprehensively:

As a programmer, you always need to test your program using different inputs to make sure everything in your code works as expected. Create new text files and test your code with them. Check the result to make sure that your code is generating the exact result.

Instructions Continue on Next Page!

Submission:

After testing your code thoroughly, when you believe that you have correctly solved the problem, and that your code follows the style guidelines, submit it on canvas. If the autograder rejects your solution, fix it and resubmit it again.

Scoring Rubric:

- **Successful Submission via CODE (2 points):** You will receive full credit for this if your code passes all the tests. Otherwise you will get zero or partial credit.
- **Hidden Test (2 points):** You will receive full credit for this if your code passes all the additional tests. Otherwise you will get zero or partial credit.
- **Correct Style (2 points):** You will receive full credit if your code has 0 style errors. Also your code has comments at the top at least with your name and description, and some comments inside the code to explain the code.
- **Proper Programming Practices (2 Points):** If your program is written applying all of the programming practices **covered in class:**
  - Throwing an exception properly
  - Catching exceptions properly
  - Proper logic
  - Using nested loops properly
  - Using proper conditions
  - Not using things that aren't discussed in the class