In another worksheet there was an example demonstrating how to write a nested for-loop to draw this figure.

```
*****
*****
*****
```

Nested for-loops can be confusing. It is often clearer to hide the inner loop by moving it into a method call. So, instead of pseudocode like this

```
for line = 1, 2, 3
    for i = 1, 2, 3, 4, 5
        print *
    newline
```

the process could be written with pseudocode like this

```
for line = 1, 2, 3
    printStars(5)
    newline
```

Because this second version disentangles the two loops and replaces the inner loop with a descriptive name, it is easier to understand. The method printStars can then be written separately.

1) Write a program where the main implements the second pseudocode above and which calls a method printStars that you should also write.

2) Follow the same process as above to write a program to draw the following figure.

```
*
**
***
****
*****
```

3) The figure

```
!!!!!!!!!!!!!!!!!!!!!!
\\!!!!!!!!!!!!!!!!!!//
\\\\!!!!!!!!!!!!!!////
\\\\\\!!!!!!!!!!//////
\\\\\\\\!!!!!!////////
\\\\\\\\\\!!//////////
```

can be thought of as six lines = 0, 1, 2, 3, 4, 5 where the number of \, ! and / characters in each can be calculated as a function of the variable line. In pseudocode

```
for line = 0, 1, 2, 3, 4, 5
    printBackslashes(expression using line)
    printExclamations(expression using line)
    printSlashes(expression using line)
    newline
```

Implement this idea to replicate the figure.

4) If time permits, write a method printChars with the header

```
    public static void printChars(char ch, int numTimes)
```

that prints the character ch numTimes times. Update your program from Problem 3 to use printChars instead printBackslashes, printExclamations, and printSlashes. This is an example of eliminating redundancy by noticing that the three old methods do the same thing but with a small variation that can be expressed as a parameter.