INTRODUCTION TO COMPUTER SCIENCE I
PROJECT 3B
Nested loops

Here is the second of a two-part project that will exercise your use of our new-found *conditional* and *iterative statements*. You will use them in various combinations to perform a few new types of calculations.

1 Pretty patterns

We are going to use loops to do something quite different now. In particular, you will write a new program by following these steps:

1. **Getting started:** Open a new and different source code file in *Emacs*:

   ```
   $ emacs Patterns.java &
   ```

   Once the *Emacs* window appears, write the usual stuff that surrounds the methods that you write:

   ```
   import java.util.Scanner;

   public class Patterns {

     public static Scanner keyboard = new Scanner(System.in);

     // YOUR METHODS WILL GO HERE.

   }
   ```

2. **Write the square pattern method:** Were going to make some really simple ASCII art.[1] Specifically, we're going to begin with a square. That is, with our printing patterns, a square of size 5 looks like this, with 5 stars on each side:

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

[1] *ASCII* is an old and commonly used standard set of characters used for English-based computers. You can look it up if you want to know more, but the key point is that you're going to make pictures by printing patterns of typical characters on the screen.

You must write a method named `printSquare` that takes as a parameter the size of the square to be printed, and then prints it. Your method should begin like this:

```
public static void printSquare (int size) {
```

That is, this method takes a `size` of the square to print, and returns nothing at all. It's only job is to print the asterisks in a square-like pattern of the requested `size`.

3. **Write the triangle pattern method:** Here's a slightly tricker pattern. A triangle of size 4 looks like this:

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

Write a method named `printTriangle` that takes a `size` parameter, prints a triangle pattern of that size, and returns nothing.

4. **Write the tree pattern:** Tricker still. A tree pattern of size 6:

```
     *
    ***
   *****
  *******
 *********
***********
     |
```

Write a method named `printTree` that takes a `size` parameter, prints a tree pattern of that size, and returns nothing. Don't forget the trunk of the tree, made using the *absolute value sign* (`|`).

As usual, feel free to write a temporary `main` method that calls each and all of these, passing known `size` values in order to test the function of your pattern-printing methods.

2 Allowing pattern selection

At long last, clear out any code you might have in your `main` method for this new `Pattern` program so that you may perform your last task. Specifically, you must write a main method that:

1. **Prints a menu of options:** Show the user their pattern choices, like so:

```
(1) square
(2) triangle
(3) tree
```

2. **Allows the user to select a pattern:** Prompt the user to select one of the three patterns, by number. Store their choice in a variable. Your code must ensure that the value entered **is a valid choice**.[2]

3. **Allows the user to enter a size:** Prompt the user to enter a size for the pattern. Store the requested size in another variable. Your code must ensure that the value entered **is a non-negative number**.[3]

4. **Print the pattern:** Call the correct method to print the requested pattern of the requested size.

3 How to submit your work

As usual, use the `cs111-submit` command:

```
cs111-submit project-3b Patterns.java
```

Part B is due on October 4/5, at the beginning of lab

[2]Hint: Use a loop!
[3]Hint: Same hint as above.