Learning Objectives:

- Practice the use of nested for loops
- Use pseudo-code with a top-down refinement strategy to build algorithms
- Implement a sentinel controlled loop to gather user input
- Use a switch statement to respond to user choices
- Declare and use static methods

pattern1	pattern2	pattern3	pattern4
::@... ::@@... ::@@@... ::@@@@... ::@@@@@... ::@@@@@@... ::@@@@@@@...	::@@@@@@@... ::@@@@@@... ::@@@@@... ::@@@@... ::@@@... ::@@... ::@...	...@:: ...@@:: ...@@@:: ...@@@@:: ...@@@@@:: ...@@@@@@:: ...@@@@@@@::	...@@@@@@@:: ...@@@@@@:: ...@@@@@:: ...@@@@:: ...@@@:: ...@@:: ...@:::

Output:

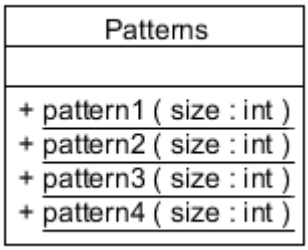
```
Choose a pattern (1-4) or 0 to quit: 1
      ::@...
     ::@@...
    ::@@@...
   ::@@@@...
  ::@@@@@...
 ::@@@@@@...
::@@@@@@@...

Choose a pattern (1-4) or 0 to quit: 2
::@@@@@@@...
 ::@@@@@@...
  ::@@@@@...
   ::@@@@...
    ::@@@...
     ::@@...
      ::@...

Choose a pattern (1-4) or 0 to quit: 3
...@::
...@@::
...@@@::
...@@@@::
...@@@@@::
...@@@@@@::
...@@@@@@@::

Choose a pattern (1-4) or 0 to quit: 4
...@@@@@@@::
...@@@@@@::
...@@@@@::
...@@@@::
...@@@::
...@@::
...@::

Choose a pattern (1-4) or 0 to quit: 0
good bye
```

Description:

Create a new project APatterns that includes 2 files: Patterns.java and PatternsTest.java

The class Patterns includes 4 methods that print one pattern each. Each pattern includes a parameter called size. This parameter determines the size of the triangles in rows. The triangles below are of size 7.

Here is the UML class diagram of class Patterns:



Notice that the operations are underlined. That indicates that the methods are static. Make sure to call the methods on the type. No instance of class Patterns should be created.

Turning in:

Ensure that you included a comment with your name and the assignment number on top of each code file.

Zip up your project including all java files and turn it in via Canvas.

Ad PatternsTest:

1. Use a do-while loop to read in use input until the user chooses to quit
   - Read in a number between 1 and 4 to choose a pattern or 0 to quit
     No input validation is required. You may assume that the user enters a valid input
   - If the user chose a pattern (1 – 4) read in a number indicating the size of the pattern
     (i.e. size corresponds to the number of rows)
2. Use a switch statement to display the appropriate pattern depending on the user selection
3. Create an output that looks like the example provided.
4. **Do NOT create an instance** (object) of type Patterns. The methods are static and should be called on the type
   e.g. `Patterns.pattern1(size);`

Ad Patterns:

Class Patterns declares no fields and no constructor (the Java compiler will provide a no-argument default constructor)
It has four methods. None of the methods depends on instance specific data (instance fields). That's why we declare these methods as static. They belong to the type and not to a specific instance.
e.g.

```
public static void pattern1(int size)
{
    // . . .
}
```

Requirements:

1. Each pattern looks exactly like displayed above (same number and type of characters) when you call it with an argument value 9
2. The only output statements allowed are:
   - `System.out.print("@");`
   - `System.out.print(":");`
   - `System.out.print(".");`
   - `System.out.print(" ");`
   - `System.out.println();`

   NO multi-character Strings like `"@@@*@@@"`
3. Use **nested loops** to print the patterns (no switch statements or if statements)

Recommendation:
Develop the algorithms by following a step-wise refinement approach like it was shown in the video – analogous to what we practiced it in class.
Make sure to draw the tables and to write down the number of spaces, dots, colons, and '@' for each of the lines.