Assignment 5 - Patterns

CS1400 Summer 2011

Description:

Write a Java application that includes 2 files: Patterns.java and PatternsTest.java
The class Patterns includes 4 methods that print one pattern each. PatternsTest lets the user choose which pattern should be displayd.

Here is how the patterns look like:

pattern1	patterns2	pattern3	pattern4
@	*	@@@@@@@@@@@@@@@@@	@@@@@@@@*@@@@@@@@
@@@	@*@	@@@@@@@@@@@@@@@	@@@@@@@*@@@@@@@
@@@@@	@@*@@	@@@@@@@@@@@@@	@@@@@@*@@@@@@
@@@@@@@	@@@*@@@	@@@@@@@@@@@	@@@@@*@@@@@
@@@@@@@@@	@@@@*@@@@	@@@@@@@@@	@@@@*@@@@
@@@@@@@@@@@	@@@@@*@@@@@	@@@@@@@	@@@*@@@
@@@@@@@@@@@@@	@@@@@@*@@@@@@	@@@@@	@@*@@
@@@@@@@@@@@@@@@	@@@@@@@*@@@@@@@	@@@	@*@
@@@@@@@@@@@@@@@@@	@@@@@@@@*@@@@@@@@	@	*

Ad PatternsTest:

1. Create and instance (object) of type Patterns
2. Read in a number between 1 and 4 from the use
   you may assume that the user enters a valid input
3. Use a switch statement to display the corresponding pattern

Ad Patterns:

Class Patterns declares no fields and no constructor (the Java compiler will provide a no-argument default constructor)
It has 4 methods called **pattern1 pattern2 pattern3** and **pattern4**. None of the methods takes any arguments and all of the methods have a return value void.

Requirements:

1. Each pattern looks exactly like displayed above (same number and type of characters)
2. The only output statements allowed are:
   - `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)

You'll find some help on how to print patterns at the end of the assignment

Sample Output:

```
Please choose a pattern (1 - 4): 2
         *
        @*@
       @@*@@
      @@@*@@@
     @@@@*@@@@
    @@@@@*@@@@@
   @@@@@@*@@@@@@
  @@@@@@@*@@@@@@@
 @@@@@@@@*@@@@@@@@
```

Turning in:

Zip up your project (the directory containing the source code and the project files) and name the file **A5_*YourName*.zip** where you substitute your first name and last name for *YourName* . Turn it in via Virtual Campus.

Maximum Points: **30 pts**

Help on how to draw patterns:

The difficult part about drawing the patterns is finding the right algorithm – knowing what instructions should be executed in what order.

Here I demonstrate top-down stepwise refinement for pattern2:

```
        *
       @*@
      @@*@@
     @@@*@@@
    @@@@*@@@@
   @@@@@*@@@@@
  @@@@@@*@@@@@@
 @@@@@@@*@@@@@@@
@@@@@@@@*@@@@@@@@
```

Step1: (original problem)
draw pattern2

Step2:

```
for each of the rows do the following:
    draw the current row
```

Step3:

```
for each of the rows do the following:
    draw the right amount (n1) of ' '
    draw the right amount (n2) of '@'
    draw the right amount (n3) of '*'
    draw the right amount (n4) of '@'
    print a new line
```

We still need to figure out:

- The values for `n1` `n2` `n3` `n4` and
- how to draw `n` characters in a row

In order to figure out the values for n1 – n4 lets have another look at the pattern:

pattern2	**n1** (number of `' '`)	**n2** (number of `'@'`)	**n3** (number of `'*'`)	**n4** (number of `'@'`)
*	8	0	1	0
@*@	7	1	1	1
@@*@@	6	2	1	2
@@@*@@@	5	3	1	3
@@@@*@@@@	4	4	1	4
@@@@@*@@@@@	3	5	1	5
@@@@@@*@@@@@@	2	6	1	6
@@@@@@@*@@@@@@@	1	7	1	7
@@@@@@@@*@@@@@@@@	0	8	1	8

The table above shows us the values for n1 n2 n3 and n4 for each of the rows.
Now we need to express that number in terms of the control variable (`i`) of the outer loop. How easy or how hard that will be depends on the value of `i`. At this point `i` is not defined yet. We define `i` by defining the outer loop.
There are two options:

- I can count up or
- I can count down

To decide which one to use I look at the 4 number series I am given:
8 7 6 5 4 3 2 1 0 for n1, 0 1 2 3 4 5 6 7 8 for n2, 1 1 1 1 1 1 1 1 1 for n3, and 0 1 2 3 4 5 6 7 8 for n4
I see that only 3 out of the 4 number series depend on `i` (the third one is always 1, regardless of `i`)
Two out of three number series go up, so I like my `i` in the outer loop to count up, too. In this particular case two of the number series are 0 1 2 3 4 5 6 7 8. This is very convenient, because starting to count by 0 is a common way to count for programmers. If I choose the following outer loop

```
for (int i = 0; i < 9; i++)
{
    // TODO: draw the current rows
}
```

I am already handed the correct values for two of my number lines:

pattern2	`i` (control var)	**n1** (number of `' '`)	**n2** (number of `'@'`)	**n3** (number of `'*'`)	**n4** (number of `'@'`)
*	0	8	0	1	0
@*@	1	7	1	1	1
@@*@@	2	6	2	1	2
@@@*@@@	3	5	3	1	3
@@@@*@@@@	4	4	4	1	4
@@@@@*@@@@@	5	3	5	1	5
@@@@@@*@@@@@@	6	2	6	1	6
@@@@@@@*@@@@@@@	7	1	7	1	7
@@@@@@@@*@@@@@@@@	8	0	8	1	8

Every time we draw a row given an outer loop as described above we do know the current control variable `i`.

For each of the rows we know now the following:

- `n2` is the same as `i`
- `n3` is 1 (it is always 1 regardless of `i`)
- `n4` is the same as `i`

What we still need to find out is the value of n1 in terms of i. A good start is to look for an arithmetic expression. The simpler an expression I can find here, the easier it will be to implement the code.
If I look at the second and third column in the table above (they show me the values of `i` and n1) I notice that one goes up and one goes down. I also notice that together the sum is always 8. Thus I came up with the expression

- `n1` is the same as `( 8 - i)`

At this point I can express n1, n2, n3 and n4 in terms of the control variable (`i`) of the outer loop.
Lets refine Step 3 of the algorithm like this:

Step4:

```
for each of the rows do the following:
   draw (8 – i) ' '
   draw i '@'
   draw one '*'
   draw i '@'
   print a new line
```

The question that remains is the following:
How do I draw a given number of characters with the 4 print statements I am allowed to use

Let's consider how we would draw 3 circles - or n circles:

```
for (int i = 0; i < 3; i++)
{
    System.out.print("o");
}
```

```
for (int i = 0; i < n; i++)
{
      System.out.print("o");
}
```

The only thing that changes when I want to draw `i` circles Is the fact, that I no longer can use the name `i` for my control variable (`i` has already been defined in the outer loop and is still in scope when I declare my inner loop)
So I resort to the next letter j.
Here is how I can draw `i` circles or (`8 - i`) circles:

```
for (int j = 0; j < i; j++)
{
    System.out.print("o");
}
```

```
for (int j = 0; j < (8 - i); j++)
{
      System.out.print("o");
}
```

When I write one loop nested inside another loop the control variable of the outer loop (e.g. `i`) is still in scope and I need to use a different name for the control variable of the inner loop. However, if write one loop after another I can keep using the same name for the control variable over and over again.

Good luck!