Exercise 5.3 A Diamond is FORever

We can also place loops inside a loop. This requires us to find patterns within patterns. Exercise 5.4 use similar techniques. So, there is no solution design for Exercise 5.4

Problem Description:

Write a Java method void diamond(int n) that takes an ***odd*** integer argument n and prints an n-by-n pattern like the test cases below, with an *asterisk* (*) for each element which makes up a diamond, and a *dot* (.) for each entry that is not, and *one space* between each * or .

Key words:

Odd integer input

Print patterns

Solution Design

We start with simulations. To make discussion convenient, we first ignore the space between symbols. And focus on the count of each symbol. Obviously, we need to print rows within a loop except that the contents differ between loops

Simulation 1:

diamond(5)

Loop	Dot count	Star count	Dot count
0	2	1	2
1	1	3	1
2	0	5	0

3	1	3	1
4	2	1	2

Simulation 2:

diamond(9)

Row index	Dot count	Star count	Dot count
0	4	1	4
1	3	3	3
2	2	5	2
3	1	7	1
4	0	9	0
5	1	7	1
6	2	5	2
7	3	3	3
8	4	1	4

Patterns

From above simulations and problem descriptions, we know one fact. Printed rows are symmetric (both vertically and horizontally).

The property of vertical symmetry gives us that the sum of symbol counts in each row is equal to input argument n.

That is

$$2 \times Num(.) + Num(*) = n$$

Thus, if we can calculate any term in above equation. We know the pattern of current row.

The property of horizontal symmetry gives us that the pattern is also related to the index of row.

Since the reveal of either term in equation is enough for us, let us study $Num(*)$. (I am too lazy to study $Num(.)$.)

$Num(*)$

Start from row 0 until Row index reach the middle one, the $Num(*)$ increases with stride 2. That is

$$Num(*) = 2 \times i + 1 \ \ \forall 0 \leq i \ \leq \frac{n-1}{2}$$

After row 0 exceeds middle one, the Num(*) decreases with stride 2.

Row index	n-1-Row index	Star count
4	4	9
5	3	7
6	2	5
7	1	3
8	0	1

That is

$$Num(*) = 2 \times (\text{n-1-}i) + 1 = 2(\text{n-i})\text{-}1 \ \ \forall \frac{n-1}{2} \leq i \ < \text{n}$$

Let merge two cases together

$$\text{X} = \text{Num}(*, \text{i}, \text{n}) = \begin{cases} 2 \times i + 1 \ \ \forall 0 \leq i \ < \dfrac{n-1}{2} \\ 2(\text{n-i})\text{-}1 \ \ \forall \dfrac{n-1}{2} \leq i \ < \text{n} \end{cases}$$

Abstract pattern:

For any odd number n and row index $i \ s.t. \ 0 \leq i < n$,

Row index	Dot count	Star count	Dot count
i	(n-X)/2	X	(n-X)/2

Now we know the pattern of each row and don't forget the space between symbols. So, we print a symbol along with a space after it. Don't worry about the space at the

end of each row. Examiner ignores that.

Check implementations.

Optimization:

Notice we have a symmetric property and an "if" structure is used. Why not use a symmetric math function to replace "if" structure. This requires us to see the change of stars in a different way. Let take the middle row as a base row. The further row has less stars. This pattern applies both as row index increase from middle or decrease from middle.

Row index (middle relative)	Distance	Star count
(n-1)/2-2	2	X-4
(n-1)/2-1	1	X-2
(n-1)/2	0	X
(n-1)/2-1	1	X-2
(n-1)/2-2	2	X-4

Thus

$$Num(*, \mathrm{i}) = n - 2 \times \left| \frac{n-1}{2} - i \right| \quad \forall 0 \leq i \ < n$$

Now one equation is enough to give us the number of stars.

Also, we have another item to optimize. Which is "print". In implementation, we use three separate loops to print a sequence of symbol. The only difference between these loops are the number of symbol and symbol. Define a new method to do print job! (Hint: void method and take a String and a int variable as arguments)