You are developing a software for an advanced math visualizationtool. One of the features is to generate complex patterns that combine mathematical concepts with visual representations. Specifically, you need to create a pattern that combines Pascal's Triangle and a half-diamond shape.

Task: Write a Java program that prints a half-diamond pattern where each row contains elements from Pascal's Triangle up to the middle row. For a given integer n, generate a pattern with 2n-1 rows. The first n rows should display the elements of Pascal's Triangle in increasing order, while the next n-1 rows should display them in decreasing order, forming a half-diamond.

Pascal’s Triangle is a triangular array of binomial coefficients. The value at position (i, j) in Pascal's Triangle is computed as C(i, j), where C(i, j) = i! / (j! * (i - j)!).

Example for n = 4:

Pattern Explanation:

- Row 1: C(0,0)
- Row 2: C(1,0) C(1,1)
- Row 3: C(2,0) C(2,1) C(2,2)
- Row 4: C(3,0) C(3,1) C(3,2) C(3,3)
- Row 5: Repeat Row 3
- Row 6: Repeat Row 2
- Row 7: Repeat Row 1

Test Cases:

Sample Input 1:

Input: n = 3

Sample Output 1:

Output:

1

1 1

1 2 1

1 1

1

Sample Input 2:

Input: n = 4

Sample Output 2:

Output:

1

1 1

1 2 1

1 3 3 1

1 2 1

1 1

1

Sample Input 3:

Input: n = 5

Sample Output 3:

Output:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 3 3 1

1 2 1

1 1

1

Explanation:

1. Pascal's Triangle Calculation:

o The triangle is built row by row, where each element is the binomial coefficient calculated using factorials.

o For example, C(3,2) is calculated as 3! / (2! * (3-2)!) = 3.

2. Pattern Construction:

o The first n rows display Pascal's Triangle in an expanding manner.

o The next n-1 rows reverse the pattern, forming a symmetric half-diamond.

Coding:

```
import java.io.*;

import java.util.*;


public class gfg2 {

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        int num = s.nextInt();

        int m = 1;
```

```
// First half Pascal's triangle in pyramid form
do {
    // Print leading spaces
    for (int space = 1; space <= num - m; space++) {
        System.out.print(" ");
    }

    int number = 1;
    for (int i = 1; i <= m; i++) {
        System.out.print(number + " ");
        number = number * (m - i) / i;
    }
    System.out.println();
} while (++m <= num);

// Reverse half Pascal's triangle in pyramid form
m = num - 1;
do {
    // Print leading spaces
    for (int space = 1; space <= num - m; space++) {
        System.out.print(" ");
    }

    int number = 1;
```

```
        for (int i = 1; i <= m; i++) {
            System.out.print(number + " ");
            number = number * (m - i) / i;
        }
        System.out.println();
    } while (--m > 0);
  }
}
```

Output:

```
E:\javacode.java\8-10-2024>javac gfg2.java

E:\javacode.java\8-10-2024>java gfg2
4
   1
  1 1
 1 2 1
1 3 3 1
 1 2 1
  1 1
   1
```