**Patterns Using Loops in Java**

**Overview**

In this class, we explored various ways to generate patterns using loops in Java. The focus was primarily on understanding how to utilize nested loops to produce different types of patterns. Below, we detail each pattern discussed, along with the corresponding logic and code snippets.

**Concepts Covered**

1. **Loops**: Understanding and using for and while loops.
2. **Nested Loops**: Using loops within loops to handle rows and columns in patterns.
3. **Pattern Observation**: Recognizing and extracting patterns from given visual outputs.
4. **Variables**: Utilizing variables efficiently within loops.

**Patterns**

**Pattern 1: Simple Asterisks**

Given a number n, print n asterisks on a single line.

**Example**:

* Input: 5
* Output: \*\*\*\*\*

**Code**:

int n = 5; // Input

for (int i = 1; i <= n; i++) {

System.out.print("\*");

}

System.out.println();

This pattern is simple and requires a single for loop that runs n times to print the asterisks.

**Pattern 2: Square of Asterisks**

Given two numbers n and m, print a rectangle of asterisks with n rows and m columns.

**Example**:

* Input: 3, 5 (3 rows, 5 columns)
* Output:
* \*\*\*\*\*
* \*\*\*\*\*
* \*\*\*\*\*

**Code**:

int n = 3; // Number of rows

int m = 5; // Number of columns

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= m; j++) {

System.out.print("\*");

}

System.out.println();

}

This pattern involves a nested loop: the outer loop runs for n rows and the inner loop runs for m columns.

**Pattern 3: Right-Angled Triangle of Asterisks**

Given a number n, print a right-angled triangle of asterisks.

**Example**:

* Input: 3
* Output:
* \*
* \*\*
* \*\*\*

**Code**:

int n = 3; // Input

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= i; j++) {

System.out.print("\*");

}

System.out.println();

}

The inner loop runs from 1 to i (current row number), progressively increasing the number of asterisks printed per row.

**Pattern 4: Inverted Right-Angled Triangle of Asterisks**

Given a number n, print an inverted right-angled triangle of asterisks.

**Example**:

* Input: 3
* Output:
* \*\*\*
* \*\*
* \*

**Code**:

int n = 3; // Input

for (int i = n; i >= 1; i--) {

for (int j = 1; j <= i; j++) {

System.out.print("\*");

}

System.out.println();

}

The outer loop starts from n and decrements down to 1, while the inner loop prints i asterisks in each row.

**Pattern 5: Pyramid Pattern**

Given a number n, print a pyramid pattern where each row contains increasing number of asterisks centered by spaces.

**Example**:

* Input: 3
* Output:
* \*
* \*\*\*
* \*\*\*\*\*

**Code**:

int n = 3; // Input

for (int i = 1; i <= n; i++) {

for (int j = n - i; j > 0; j--) {

System.out.print(" ");

}

for (int k = 1; k <= 2 \* i - 1; k++) {

System.out.print("\*");

}

System.out.println();

}

This involves calculating spaces before the stars, using n - i spaces for left padding and 2 \* i - 1 asterisks.

**Pattern 6: Diamond Shape**

Given a number n, print a diamond shape.

**Example**:

* Input: 3
* Output:
* \*
* \*\*\*
* \*\*\*\*\*
* \*\*\*
* \*

**Code**:

int n = 3; // Input

// Top part

for (int i = 1; i <= n; i++) {

for (int j = n - i; j > 0; j--) {

System.out.print(" ");

}

for (int k = 1; k <= 2 \* i - 1; k++) {

System.out.print("\*");

}

System.out.println();

}

// Bottom part

for (int i = n - 1; i > 0; i--) {

for (int j = n - i; j > 0; j--) {

System.out.print(" ");

}

for (int k = 1; k <= 2 \* i - 1; k++) {

System.out.print("\*");

}

System.out.println();

}

This code is a combination of two pyramid patterns: one inverted and one normal.

**Conclusion**

Practicing different patterns not only helps in understanding loops but also improves problem-solving skills. These problems teach how to break down a problem into smaller parts, recognize patterns, and implement nested loops efficiently.

Note: This document captures the essence of patterns discussed in the class based on the given transcript and instructor’s notes. This can be used to revise the concepts and solutions effectively.

<https://github.com/KingsGambitLab/Lecture_Notes/blob/non-dsa/Academy%20DSA%20Typed%20Notes/Java%20Refresher/Refresher%20Patterns.md>

PATTERNS:

Output problem where the goal is to generate a specific visual pattern using loops.

FOR LOOP:

A control flow statement for specifying iteration, which allows code to be executed repeatedly.

WHILE LOOP:

A control flow statement that allows code to be executed repeatedly based on a given Boolean condition.

NESTED LOOP:

A loop inside another loop, often used for multi-dimensional data structures or patterns.

RECTANGLE PATTERN:

A pattern where each row contains a fixed number of columns of the same character, typically asterisks.

SQUARE PATTEREN:

A type of rectangle pattern where the number of rows is equal to the number of columns.

STAIRCASE PATTERN:

A pattern where each row contains an increasing number of characters, forming a stair-like structure.

REVERSE STAIRCASE PATTERN:

A pattern where each row contains a decreasing number of characters, reversing the staircase structure.

ROW NUMBER:

The current iteration index in the outer loop when dealing with 2D patterns.

COLUMN NUMBER:

The current iteration index in the inner loop when dealing with 2D patterns.

N AS INPUT:

A generic variable representing the size or number of rows/columns in the pattern.

PRINT STATEMENT:

A command that outputs data to the consol or screen, used here to display characters in pattern.

**Revision Notes: Pattern Printing with Loops**

In this class, we focused on becoming comfortable with using loops and nested loops through various pattern printing exercises. Below is a detailed breakdown of the concepts and patterns discussed in the class:

**Introduction to Pattern Printing**

* **Loops and Nested Loops**: The class emphasized the foundational concepts of using loops, particularly how nested loops work when generating patterns. This includes understanding the logic behind running loops a specific number of times and how one loop can be used inside another to manage rows and columns for patterns.

**Pattern Examples**

**1. Printing N Stars in a Line**

**Problem**: Given an integer n, print \* (star) n times in a single row.

**Solution**:

* Use a for loop to iterate n times and print \* in each iteration.

for(int i = 1; i <= n; i++) {

System.out.print("\*");

}

**2. Printing an N x N Square**

**Problem**: Print an n x n square of stars.

**Solution**:

* Use one loop to handle the rows and an inner loop to manage columns, both running n times.

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= n; j++) {

System.out.print("\*");

}

System.out.println();

}

**3. Printing a Rectangle with Given n Rows and m Columns**

**Problem**: Print a rectangle of stars with n rows and m columns.

**Solution**:

* Outer loop iterates n times, inner loop iterates m times.

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= m; j++) {

System.out.print("\*");

}

System.out.println();

}

**4. Staircase Pattern**

**Problem**: Print a staircase pattern with decreasing number of stars in each row.

**Solution**:

* The outer loop controls the rows, while the inner loop prints i stars for the i-th row.

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= i; j++) {

System.out.print("\*");

}

System.out.println();

}

**5. Pyramid Pattern**

**Problem**: Print a centered pyramid of stars.

**Observations**:

* Number of spaces in a row decreases as the row number increases.
* Number of stars increase as the row number increases.

**Solution**:

for(int i = 1; i <= n; i++) {

// Print leading spaces

for(int j = i; j < n; j++) {

System.out.print(" ");

}

// Print stars

for(int j = 1; j <= (2\*i-1); j++) {

System.out.print("\*");

}

System.out.println();

}

**6. Checkerboard Pattern with Stars and Column Numbers**

**Problem**: In a pattern, even columns print numbers while odd columns print stars.

**Solution**:

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= n; j++) {

if(j % 2 == 0) {

System.out.print(j);

} else {

System.out.print("\*");

}

}

System.out.println();

}

**7. Custom Patterns with Increasing Complications**

As the class progressed, various custom patterns and their solutions were discussed, demonstrating how to adjust the loop logic to fit different visual requirements. These include altering the sequence of stars and spaces, and utilizing conditions within loops to modify the output based on the current iteration indices.

**Key Learning Points**

* **Conditions within Loops**: It's crucial to grasp how conditions can be applied within loops to customize the output, as illustrated by the checkerboard example.
* **Nested Loop Dependencies**: Understanding that in nested loops, the execution of an inner loop is often dependent on the outer loop, as seen in staircase and pyramid patterns.
* **Standard Practices**: Using standard index variables like i, j, k is a common practice which, while flexible with variable naming, helps maintain clear and recognizable code.

By addressing these patterns and challenges, learners can build a robust understanding of loops and nested loops, foundational for solving more complex algorithmic problems.

PYRAMID PATTERN:

A tringle like pattern that centers characters and typically has a symmetric shape.

PATTERN WITH EVEN/ODD COLUMNS:

A pattern where column indices affect the printed character, often using modulo operations.

VARIABLE SCOPE:

The context within which a variable is defined and accessible.

ROW AND COLUMN NUMBER CALCULATION:

Using arithmetic expressions to determine the number and position of symbols in a pattern.

INITIALIZATION IN LOOPS:

Setting initial value in iteration constructs, typically defining loop counters.

public class Main {

  public static void main(String[] args) {

      Scanner ip=new Scanner(System.in);

      //print star

      int num=ip.nextInt();

      for(int i=1;i<=num;i++) System.out.print("\*");

      System.out.println();

      System.out.println("----------------------");

      //print matrix star

      int num2=ip.nextInt();

      for(int i=1;i<=num2;i++) {

        for(int j=1;j<=num2;j++){

        System.out.print("#");

      }

      System.out.println();

      }

      System.out.println("----------------------");

      //Given 2 numbers print rectangle

      int N=ip.nextInt(),M=ip.nextInt();

      for(int i=1;i<=N;i++) {

        for(int j=1;j<=M;j++){

        System.out.print("\*");

      }

      System.out.println();

      }

      System.out.println("----------------------");

      //Given N input print staircase pattern

      int num3=ip.nextInt();

      for(int i=1;i<=num3;i++){

        for(int j=1;j<=i;j++){

          System.out.print("\*");

        }

        System.out.println();

      }

      System.out.println("----------------------");

      /\*

        print pattern like below

        N=4

        \*

        \* 2

        \* 2 \*

        \* 2 \* 4

        N=3

        \*

        \* 2

        \* 2 \*

      \*/

      int num4=ip.nextInt();

      for(int i=1;i<=num4;i++) {

        for(int j=1;j<=i;j++){

        System.out.print(j%2==0?j:"\*");

      }

      System.out.println();

      }

      System.out.println("----------------------");

      /\*

        print pattern like below

        N=4

        \*  \*

        \*  \*

        \*  \*

        \*  \*

        N=3

        \* \*

        \* \*

        \* \*

      \*/

      int num5=ip.nextInt();

      for(int i=1;i<=num5;i++) {

        for(int j=1;j<=num5;j++){

        System.out.print(j==1 || j==num5?"\*":" ");

      }

      System.out.println();

      }

      //2nd approach

      System.out.println("----");

      int num6=ip.nextInt();

      for(int i=1;i<=num6;i++) {

        System.out.print("\*");

        for(int j=1;j<=num6-2;j++){

        System.out.print(" ");

        }

       System.out.print("\*");

       System.out.println();

      }

      System.out.println("----------------------");

      // print reverse staircase pattern

      int num7=ip.nextInt();

      for(int i=num7;i>=1;i--){

        for(int j=1;j<=i;j++){

          System.out.print("\*");

        }

        System.out.println();

      }

      System.out.println("----------------------");

  }

}

CUSTOM INPUT:

5

5

2

3

5

5

4

4

5

OUTPUT:

\*\*\*\*\*

----------------------

#####

#####

#####

#####

#####

----------------------

\*\*\*

\*\*\*

----------------------

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

----------------------

\*

\*2

\*2\*

\*2\*4

\*2\*4\*

----------------------

\* \*

\* \*

\* \*

\* \*

----

\* \*

\* \*

\* \*

\* \*

----------------------

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

----------------------

Reverse Staircase pattern logic Idea:

N=4

\*\*\*\*

\*\*\*

\*\*

\*

Rows Star

1 4 N+1

2 3 N+1

3 2 N+1

4 1 N+1

Rows+Stars=N+1

Stars=N+1-Rows

CODE:

import java.util.\*;

public class Main {

  public static void main(String[] args) {

      int num=new Scanner(System.in).nextInt();

      for(int i=1;i<=num;i++){

        for(int j=1;j<=num+1-i;j++){

          System.out.print("\*");

        }

        System.out.println();

      }

  }

}

CUSTOM INPUT:

5

OUTPUT:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

N=4

\* \*

\* \*

\* \*

\*\*

Row Space Star

1 3 2

2 2 2

3 1 2

4 0 2

Space=N-Row

Star=2

import java.util.\*;

public class Main {

  public static void main(String[] args) {

      int num=new Scanner(System.in).nextInt();

      for(int i=1;i<=num;i++){

        System.out.print("\*");

        for(int j=1;j<=num-i;j++){

          System.out.print(" ");

        }

        System.out.print("\*");

        System.out.println();

      }

  }

}

CUSTOM INPUT:

5

OUTPUT:

\* \*

\* \*

\* \*

\* \*

\*\*

Print below pattern

N=3

\*

\*\*

\*\*\*

N=4

\*

\*\*

\*\*\*

\*\*\*\*

Row Space Star

1 3 1

2 2 2

3 1 3

4 0 4

Space=N-row

Star=row

public class Main {

  public static void main(String[] args) {

      int num=new Scanner(System.in).nextInt();

      for(int i=1;i<=num;i++){

        for(int j=1;j<=num-i;j++){

          System.out.print(" ");

        }

        for(int j=1;j<=i;j++){

          System.out.print("\*");

        }

        System.out.println();

      }

  }

}

CUSTOM INPUT:

5

OUTPUT:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

Print below pattern

N=4



Row LeftStar LeftSpace RightSpace RightStar

4 4 0 0 4

3 3 1 1 3

2 2 2 2 2

1 1 3 3 1

LeftStar=RightStar=Row

LeftSpace=RightSpace=N-row

import java.util.\*;

public class Main {

  public static void main(String[] args) {

      int num=new Scanner(System.in).nextInt();

      for(int i=num;i>=1;i--){

        for(int j=1;j<=i;j++){

          System.out.print("\*");

        }

        for(int j=1;j<=num-i;j++){

          System.out.print(" ");

        }

        for(int j=1;j<=num-i;j++){

          System.out.print(" ");

        }

        for(int j=1;j<=i;j++){

          System.out.print("\*");

        }

        System.out.println();

      }

  }

}

CUSTOM INPUT

4

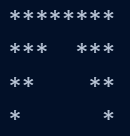
OUTPUT:

\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\* \*\*

\* \*



ANOTHER WAY

ROW Star1 Space Star2

1 4 0 4

2 3 2 3

3 2 4 2

4 1 6 1

Star1=Star2=n+1-row

Space=2\*row-2

import java.util.\*;

public class Main {

  public static void main(String[] args) {

      int num=new Scanner(System.in).nextInt();

      for(int i=1;i<=num;i++){

        for(int j=1;j<=num+1-i;j++){

          System.out.print("\*");

        }

        for(int j=1;j<=2\*i-2;j++){

          System.out.print(" ");

        }

        for(int j=1;j<=num+1-i;j++){

          System.out.print("\*");

        }

        System.out.println();

      }

  }

}



Print below pattern

N=4



Row Space1 Star Space2

1 3 1 3

2 2 3 2

3 1 5 1

4 0 7 0

Space1=Space2=n-row

Star=row\*2-1

public class Main {

  public static void main(String[] args) {

      int num=new Scanner(System.in).nextInt();

      for(int i=1;i<=num;i++){

        for(int j=1;j<=num-i;j++){

          System.out.print(" ");

        }

        for(int j=1;j<=2\*i-1;j++){

          System.out.print("\*");

        }

        for(int j=1;j<=num-i;j++){

          System.out.print(" ");

        }

        System.out.println();

      }

  }

}



**Q1. Skip Even Numbers Half Pyramid**

**Problem Description**

Take an integer **N** as input, print the corresponding pattern for **N**.

For example if **N = 5** then pattern will be like:

1

1\_

1\_3

1\_3\_

1\_3\_5

**Note** : Here '\_' represents space for explanation purpose only. You have to print space in your code.  
**Problem Constraints**

1 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

1

1

Output 2:

1

1

1 3  
  
**Example Explanation**

Print the pattern as described.

CODE:

public class Main {

    public static void main(String[] args) {

        short num=new Scanner(System.in).nextShort();

        for(short i=1;i<=num;i++){

            for(short j=1;j<=i;j++){

                System.out.print(j%2==0?" ":j);

            }

            System.out.println();

        }

    }

}

**Q2. Leading spaces pyramid**

**Problem Description**

Take an integer **N** as input, print the corresponding pattern for **N**.

For example if **N = 5** then pattern will be like:

\_\_\_\_\*

\_\_\_\*\*

\_\_\*\*\*

\_\*\*\*\*

\*\*\*\*\*

**Note** : Here '\_' represents space for explanation purpose only. You have to print space in your code.  
**Problem Constraints**

1 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

\*

\*\*

Output 2:

\*

\*\*

\*\*\*  
  
**Example Explanation**

Print the pattern as described.

ROUGH WORK:

N=5

Row Space Star

1 4 1

2 3 2

3 2 3

4 1 4

5 0 5

Space=N-Row

Star=Row

CODE:

public class Main {

    public static void main(String[] args) {

        int row=new Scanner(System.in).nextInt();

        for(int i=1;i<=row;i++){

            for(int j=1;j<=row-i;j++){

                System.out.print(" ");

            }

            for(int j=1;j<=i;j++){

                System.out.print("\*");

            }

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=N;i++){

            for(byte j=1;j<=N;j++)System.out.print((j<=N-i)?" ":"\*");

            System.out.println();

        }

    }

}

**Q3. Inverted Numeric Pyramid**

**Problem Description**

Take an integer **N** as input, print the corresponding **Numeric Inverted Half Pyramid** pattern for **N**.

For example if **N = 4** then pattern will be like:

1 2 3 4

1 2 3

1 2

1  
**Problem Constraints**

1 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Numeric Inverted Half Pyramid pattern corresponding to the given **N**.

**NOTE:** There should be no extra spaces after last integer and before first integer in any . All integers in any row in the pattern are separated by a single space.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

1 2

1

Output 2:

1 2 3

1 2

1  
**Example Explanation**

Example Input 1  
 We have the input of integer as 2. So there will be 2 rows.   
 The first row will have 2 integers separated by a single space (1 2)  
 The next row will have 2-1 = 1 integer (1)  
 As we do not have any more integers left we stop printing the pattern.  
 Example Input 2  
 We have the input of integer as 3. So there will be 3 rows.   
 The first row will have 3 integers separated by a single space (1 2 3)  
 The next row will have 3-1 = 2 integers separated by a single space (1 2)  
 The next row will have 2-1 = 1 integer.(1)  
 As we do not have any more integers left we stop printing the pattern.

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=num;i>=1;i--){

            for(int j=1;j<=i;j++){

                System.out.print(j==i?j:j+" ");

            }

            System.out.println();

        }

    }

}

**Q4. Hollow inverted pyramid pattern**

**Problem Description**

Take an integer **N** as input, print the corresponding pattern for **N**.

For example if **N = 5** then pattern will be like:

\*\_\_\_\_\_\_\_\_\* // 8 spaces

\*\*\_\_\_\_\_\_\*\* // 6 spaces

\*\*\*\_\_\_\_\*\*\* // 4 spaces

\*\*\*\*\_\_\*\*\*\* // 2 spaces

\*\*\*\*\*\*\*\*\*\* // 0 spaces

**NOTE:** Here '\_' is used to represent spaces. You have to print spaces in your code.  
**Problem Constraints**

2 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Full Pyramid pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

\*\_\_\* // 2 spaces

\*\*\*\* // 0 spaces

Output 2:

\*\_\_\_\_\* // 4 spaces

\*\*\_\_\*\* // 2 spaces

\*\*\*\*\*\* // 0 spaces

**NOTE:** Here '\_' is used to represent spaces. You have to print spaces in your code.  
**Example Explanation**

Print the pattern as described.

ROUGH WORK:

N=5

Row LeftStar Space RightStar

1 1 8 1

2 2 6 2

3 3 4 3

4 4 2 4

5 5 0 5

LeftStar=RightStar=Row

Space=2N-2Row

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num;i++){

            for(int j=1;j<=i;j++){

                System.out.print("\*");

            }

            for(int j=1;j<=2\*num-2\*i;j++){

                System.out.print(" ");

            }

            for(int j=1;j<=i;j++){

                System.out.print("\*");

            }

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=N;i++){

            for(byte j=1;j<=i;j++)System.out.print("\*");

            for(short j=1;j<=(N-i)\*2;j++)System.out.print(" ");

            for(byte j=1;j<=i;j++)System.out.print("\*");

            System.out.println();

        }

    }

}

**Q5. Hollow pyramid pattern**

**Problem Description**

Take an integer **N** as input, print the corresponding pattern for **N**.

For example if **N = 5** then pattern will be like:

\*\*\*\*\*\*\*\*\*\* // 0 spaces

\*\*\*\*\_\_\*\*\*\* // 2 spaces

\*\*\*\_\_\_\_\*\*\* // 4 spaces

\*\*\_\_\_\_\_\_\*\* // 6 spaces

\*\_\_\_\_\_\_\_\_\* // 8 spaces

**NOTE:** Here '\_' is used to represent spaces. You have to print spaces in your code.  
**Problem Constraints**

2 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Full Pyramid pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

\*\*\*\* // 0 spaces

\*\_\_\* // 2 spaces

Output 2:

\*\*\*\*\*\* // 0 spaces

\*\*\_\_\*\* // 2 spaces

\*\_\_\_\_\* // 4 spaces

**NOTE:** Here '\_' is used to represent spaces. You have to print spaces in your code.  
**Example Explanation**

Print the pattern as described.

ROUGH WORK:

N=5

Row LeftStar Space RightStar

1 5 0 5

2 4 2 4

3 3 4 3

4 2 6 2

5 1 8 1

LeftStar=RightStar=N+1-Row

Space=Row\*2-2

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num;i++){

            for(int j=1;j<=num+1-i;j++){

                System.out.print("\*");

            }

            for(int j=1;j<=i\*2-2;j++){

                System.out.print(" ");

            }

            for(int j=1;j<=num+1-i;j++){

                System.out.print("\*");

            }

            System.out.println();

        }

    }

}

ROUGH WORK2:

N=5

Row LeftStar Space RightStar

5 5 0 5

4 4 2 4

3 3 4 3

2 2 6 2

1 1 8 1

LeftStar=RightStar=Row

Space=N\*2-Row\*2

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=N;i>=1;i--){

            for(byte j=1;j<=i;j++) System.out.print("\*");

            for(short j=1;j<=N\*2-i\*2;j++) System.out.print(" ");

            for(byte j=1;j<=i;j++) System.out.print("\*");

            System.out.println();

        }

    }

}

**Q6. Full pyramid**

**Problem Description**

Take an integer **N**, print the corresponding **Full Pyramid** pattern for **N**.

For example if **N = 5** then pattern will be like:

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**NOTE:** There should be exactly one extra space after each \* for each row.  
**Problem Constraints**

2 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Full Pyramid pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3

Input 3:

4  
**Example Output**

Output 1:

\*

\* \*

Output 2:

\*

\* \*

\* \* \*

Output 3:

\*

\* \*

\* \* \*

\* \* \* \*

**Example Explanation**

Print the pattern as described.

ROUGH WORK:

N=5

Row Space StarSpace

1 4 1

2 3 2

3 2 3

4 1 4

5 0 5

Space=N-Row

StarSpace=Row

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num;i++){

            for(int j=1;j<=num-i;j++){

               System.out.print(" ");

            }

            for(int j=1;j<=i;j++){

                System.out.print("\* ");

            }

            System.out.println();

        }

    }

}

**Q1. Leading spaces inverted pyramid**

**Problem Description**

Take an integer **N** as input, print the corresponding pattern for **N**.

For example if **N = 5** then pattern will be like:

\*\*\*\*\*

\_\*\*\*\*

\_\_\*\*\*

\_\_\_\*\*

\_\_\_\_\*

**Note** : Here '\_' represents space for explanation purpose only. You have to print space in your code.  
**Problem Constraints**

1 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

\*\*

\_\*

Output 2:

\*\*\*

\_\*\*

\_\_\*

**Note** : Here '\_' represents space for explanation purpose only. You have to print space in your code.  
**Example Explanation**

Print the pattern as described.

ROUGHT WORK:

N=5

Row Space Star

1 0 5

2 1 4

3 2 3

4 3 2

5 4 1

Space=Row-1

Star=N+1-Row

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num;i++){

            for(int j=1;j<=i-1;j++){

                System.out.print(" ");

            }

            for(int j=1;j<=num+1-i;j++){

                System.out.print("\*");

            }

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=N;i++){

            for(byte j=1;j<=N;j++)System.out.print((j<=i-1)?" ":"\*");

            System.out.println();

        }

    }

}

**Q2. Photo Frame Pattern**

**Problem Description**

Take an integer **N** as input, print the corresponding pattern for **N**.

For example if **N = 5** then pattern will be like:

\*\*\*\*\*

\* \*

\* \*

\* \*

\*\*\*\*\*

**Note** : Except for the first line and the last line, every line will have N-2 spaces between the two stars.  
**Problem Constraints**

3 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the pattern corresponding to the given **N**.  
**Example Input**

Input 1:

3

Input 2:

4  
**Example Output**

Output 1:

\*\*\*

\* \*

\*\*\*

Output 2:

\*\*\*\*

\* \*

\* \*

\*\*\*\*  
**Example Explanation**

Print the pattern as described.

CODE1:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=N;i++){

            for(byte j=1;j<=N;j++)System.out.print((i==1 || j==1 || i==N || j==N)?"\*":" ");

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num;i++){

            if(i==1 || i==num){

                for(int j=1;j<=num;j++){

                    System.out.print("\*");

                }

            }else{

                for(int j=1;j<=num;j++){

                    System.out.print(j==1||j==num?"\*":" ");

                }

            }

            System.out.println();

        }

    }

}

**Q3. Different Number Triangle**

**Problem Description**

Take an integer **N** as input, print the corresponding **Full Numeric Pyramid** pattern for **N**.

For example if **N = 5** then pattern will be like:

0 0 0 0 5 0 0 0 0

0 0 0 4 8 12 0 0 0

0 0 3 6 9 12 15 0 0

0 2 4 6 8 10 12 14 0

1 2 3 4 5 6 7 8 9

**NOTE:** There should be exactly one extra space after each number for each row.  
**Problem Constraints**

2 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Full Pyramid pattern corresponding to the given **N**.

**NOTE:**

 There is no extra space before the first integer of any row.

 There is an extra space after the last integer of any row.

 All the integers in any row are space separated.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

0 2 0

1 2 3

Output 2:

0 0 3 0 0

0 2 4 6 0

1 2 3 4 5  
  
**Example Explanation**

Print the pattern as described.

ROUGH WORK:

N=5

Row LeftZero Number RightZero

1 4 1 4

2 3 3 3

3 2 5 2

4 1 7 1

5 0 9 0

LeftZero= RightZero=N-Row

Number=Row\*2-1

CODE:

public class Main {

    public static void main(String[] args) {

        int num = new Scanner(System.in).nextInt(),count=num;

        for(int i=1;i<=num;i++){

            for(int j=1;j<=num-i;j++){

                System.out.print("0 ");

            }

            for(int j=1;j<=i\*2-1;j++){

                System.out.print(count\*j+" ");

            }

            count--;

            for(int j=1;j<=num-i;j++){

                System.out.print("0 ");

            }

            System.out.println();

        }

    }

}

ROUGH WORK 2:

N=5

Row LeftZero Number RightZero

5 4 1 4

4 3 3 3

3 2 5 2

2 1 7 1

1 0 9 0

LeftZero= RightZero=Row-1

Number=N\*2-Row\*2+1

CODE2:

public class Main {

    public static void main(String[] args) {

        int num = new Scanner(System.in).nextInt();

        for(int i=num;i>=1;i--){

            for(int j=1;j<=i-1;j++){

                System.out.print("0 ");

            }

            for(int j=1;j<=num\*2-i\*2+1;j++){

                System.out.print(i\*j+" ");

            }

            for(int j=1;j<=i-1;j++){

                System.out.print("0 ");

            }

            System.out.println();

        }

    }

}

CODE3:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=N;i++){

            for(short j=1;j<=N-i;j++) System.out.print("0 ");

            for(short j=1;j<=(i\*2)-1;j++) System.out.print((N+1-i)\*j+" ");

            for(short j=1;j<=N-i;j++) System.out.print("0 ");

            System.out.println();

        }

    }

}

**Q4. Half Diamond**

**Problem Description**

Take an integer **N** as input, print the corresponding **Half Diamond** pattern with **2\*N - 1 rows**.

For example if **N = 5** then pattern will be like:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**NOTE:** There should be no spaces after any \* .  
**Problem Constraints**

2 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Half Diamond pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3  
**Example Output**

Output 1:

\*

\*\*

\*

Output 2:

\*

\*\*

\*\*\*

\*\*

\*  
**Example Explanation**

Print the pattern as described.

ROUGH WORK

N=5

Row Star

1 1

2 2

3 3

4 4

5 5

6 4

7 3

8 2

9 1

Star=Row<=N?Row:N\*2-Row

CODE1:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num\*2-1;i++){

            for(int j=1;j<=(i<=num?i:num\*2-i);j++){

                System.out.print("\*");

            }

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=(N\*2)-1;i++){

            if(i<=N) for(byte j=1;j<=i;j++) System.out.print("\*");

            else for(int j=(N\*2)-i;j>=1;j--) System.out.print("\*");

            System.out.println();

        }

}

}

CODE3:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=1;i<=N;i++){

            for(byte j=1;j<=i;j++)System.out.print("\*");

            System.out.println();

        }

        for(int i=N-1;i>=1;i--){

            for(int j=i;j>=1;j--) System.out.print("\*");

            System.out.println();

        }

    }

}

**Q5. Inverted Full Pyramid**

**Problem Description**

Take an integer **N** as input, and print the corresponding **Inverted Full Pyramid** pattern for **N**.

For example if **N = 5** then pattern will be like:

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**NOTE:** There should be exactly one extra space after each \* for each row.  
**Problem Constraints**

2 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Inverted Full Pyramid pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3

Input 3:

4  
**Example Output**

Output 1:

\* \*

\*

Output 2:

\* \* \*

\* \*

\*

Output 3:

\* \* \* \*

\* \* \*

\* \*

\*  
**Example Explanation**

Print the pattern as described.

ROUGH WORK:

N=5

Row Space StarSpace

1 0 5

2 1 4

3 2 3

4 3 2

5 4 1

Space=Row-1

StarSpace=N+1-Row

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=1;i<=num;i++){

            for(int j=1;j<=i-1;j++){

                System.out.print(" ");

            }

            for(int j=1;j<=num+1-i;j++){

                System.out.print("\* ");

            }

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=N;i>=1;i--){

            for(byte j=1;j<=N;j++)System.out.print((j<=N-i)?" ":"\* ");

            System.out.println();

        }

    }

}

**Q6. Hollow Inverted Half Pyramid**

**Problem Description**

Given an integer **N** as input, print the corresponding **Hollow Inverted Half Pyramid** pattern for **N**.

For example if **N = 6** then pattern will be like:

\*\*\*\*\*\*

\* \*

\* \*

\* \*

\*\*

\*  
**Problem Constraints**

1 <= N <= 100  
**Input Format**

First and only line of input contains a single integer **N**.  
**Output Format**

Output the Hollow Inverted Half Pyramid pattern corresponding to the given **N**.  
**Example Input**

Input 1:

2

Input 2:

3

Input 3:

4

**Example Output**

Output 1:

\*\*

\*

Output 2:

\*\*\*

\*\*

\*

Output 3:

\*\*\*\*

\* \*

\*\*

\*  
**Example Explanation**

Print the pattern as described.

CODE:

public class Main {

    public static void main(String[] args) {

        int num=new Scanner(System.in).nextInt();

        for(int i=num;i>=1;i--){

            if(i==1 || i==num){

                for(int j=1;j<=i;j++){

                    System.out.print("\*");

                }

            }else{

                for(int j=1;j<=i;j++){

                    System.out.print(j==1||j==i?"\*":" ");

                }

            }

            System.out.println();

        }

    }

}

CODE2:

public class Main {

    public static void main(String[] args) {

        byte N=new Scanner(System.in).nextByte();

        for(byte i=N;i>=1;i--){

            for(byte j=1;j<=i;j++)System.out.print((j==1 || i==1 || j==i || i==N)?"\*":" ");

            System.out.println();

        }

    }

}