Patterns II

Introduction

In addition to basic patterns, advanced patterns in Java offer more complex and visually interesting designs that can be created using loops and control structures. In this guide, we'll explore several advanced patterns, provide step-by-step explanations, and offer code examples.

1. Character Patterns

- Printing characters as shown below:

```
aaaa
bbbb
cccc
```

Solution :

```java
import java.util.Scanner;

public class CharacterPatternNRows {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the value of N: ");
        int N = scanner.nextInt();

        for (int i = 0; i < N; i++) {
            char currentChar = (char) ('a' + i);
            for (int j = 0; j < N; j++) {
                System.out.print(currentChar);
            }
            System.out.println();
        }
    }
}
```

Algorithm to follow:

- Import the Scanner class for user input.
- Create a Scanner object to read input from the user.
- Prompt the user to enter the value of N.
- Read the user's input into the variable N.
- Use a nested loop with two variables, i and j, to control the rows and columns.

- In the outer loop (i), iterate from 0 to N-1. For each value of i, calculate the currentChar as the character 'a' plus i. This will give us 'a' for i = 0, 'b' for i = 1, and so on.
- In the inner loop (j), iterate from 0 to N-1. For each value of j, print the currentChar.
- After printing N characters, move to the next line using System.out.println() to create rows of characters.
- Repeat steps 6-8 for each value of i to create N rows of characters.

- Printing characters as shown below:

```
abcd
bcde
cdef
```

Solution:

```java
import java.util.Scanner;

public class CharacterPatternContinuous {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the value of N: ");
        int N = scanner.nextInt();

        for (int i = 0; i < N; i++) {
            char currentChar = (char) ('a' + i);
            for (int j = 0; j < N; j++) {
                System.out.print(currentChar);
                currentChar++;
            }
            System.out.println();
        }
    }
}
```

Algorithm to follow:

- Instead of printing the same character currentChar repeatedly, we increment currentChar inside the inner loop to print consecutive characters ('abcd', 'bcde', 'cdef', etc.).
- The outer loop (i) still controls the rows and the inner loop (j) controls the columns. Each row starts with the character corresponding to the current value of i.
- This code takes a user-defined value of N and prints character patterns 'abcd', 'bcde', 'cdef', and so on in rows.

2. Inverted Triangle Patterns

- Printing an inverted triangle pattern of asterisks(*), for example, if the input is ‘4’ the output should be

```
****
***
**
*
```

Solution:

```java
import java.util.Scanner;

public class InvertedTrianglePatternAsterisks {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows (N): ");
        int N = scanner.nextInt();

        for (int i = N; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Algorithm to follow:

- Prompt the user to enter the number of rows (N).
- Read the user's input into the variable N.
- Use a loop with the variable i to control the number of rows. The loop starts from N and goes down to 1.
- In each row, use another loop with the variable j to print asterisks ("). The inner loop iterates from 1 to i, printing " for each iteration.
- Use System.out.println() to move to the next line after completing each row.

- Printing an inverted triangle pattern of numbers. For example, if N = 5, the output should be

```
1 2 3 4 5
  1 2 3 4
    1 2 3
      1 2
        1
```

Solution:

```java
import java.util.Scanner;

public class InvertedTrianglePattern {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows (N): ");
        int N = scanner.nextInt();

        for (int i = 1; i <= N; i++) {
            // Print spaces for indentation
            for (int j = 1; j < i; j++) {
                System.out.print("  ");
            }

            // Print numbers in descending order
            for (int k = 1; k <= N - i + 1; k++) {
                System.out.print(k + " ");
            }
            System.out.println();
        }
    }
}
```

Algorithm to follow:

- Prompt the user to enter the number of rows (N).
- Read the user's input into the variable N.
- Use a nested loop with three variables, i, j, and k, to control the rows, spaces for indentation, and numbers.
- In the outer loop (i), iterate from 1 to N. This loop controls the number of rows in the inverted triangle.
- In the first inner loop (j), iterate from 1 to i - 1. This loop is responsible for printing spaces for indentation. The number of spaces increases with each row.

- After printing the spaces, enter another inner loop (k). This loop is responsible for printing the numbers in ascending order, starting from 1. The number of numbers in each row decreases from the top row to the bottom row.
- Use System.out.println() to move to the next line after completing each row.

3. Mirror Image Pattern

- Mirror image pattern with asterisks ('*'). Expected output:

```
        *
      * *
    * * *
  * * * *
* * * * *
```

Solution:

```java
import java.util.Scanner;

public class MirrorImagePattern {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows (N): ");
        int N = scanner.nextInt();

        for (int i = 1; i <= N; i++) {
            // Print spaces for indentation
            for (int j = 1; j <= N - i; j++) {
                System.out.print("  ");
            }

            // Print asterisks in ascending order
            for (int k = 1; k <= i; k++) {
                System.out.print("* ");
            }

            System.out.println(); // Move to the next line
        }
    }
}
```

Algorithm to follow:

- Prompt the user to enter the number of rows (N).
- Read the user's input into the variable N.

- Use a loop with the variable i to control the number of rows. The loop iterates from 1 to N.
- In each row, use another loop with the variable j to print spaces for indentation. The number of spaces decreases as we move to the right side of the pattern. We calculate the number of spaces as N - i.
- After printing the spaces, use a third loop with the variable k to print asterisks ('*') in ascending order. The number of asterisks increases from left to right, starting from 1 and going up to i.
- Use System.out.println() to move to the next line after completing each row.

4. Isosceles Triangle Pattern

- . For example, if N = 5, the output should be:

```
        1
      1 2
    1 2 3
  1 2 3 4
1 2 3 4 5
```

Solution:

```java
import java.util.Scanner;

public class MirrorImagePatternNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows (N): ");
        int N = scanner.nextInt();

        for (int i = 1; i <= N; i++) {
            // Print spaces for indentation
            for (int j = 1; j <= N - i; j++) {
                System.out.print("  ");
            }

            // Print numbers in ascending order
            for (int k = 1; k <= i; k++) {
                System.out.print(k + " ");
            }

            System.out.println(); // Move to the next line
        }
    }
}
```

Algorithm to follow:

- Prompt the user to enter the number of rows (N).
- Read the user's input into the variable N.
- Use a loop with the variable i to control the number of rows. The loop iterates from 1 to N.
- In each row, use another loop with the variable j to print spaces for indentation. The number of spaces decreases as we move to the right side of the pattern. We calculate the number of spaces as N - i.
- After printing the spaces, use a third loop with the variable k to print numbers in ascending order. The numbers start from 1 and go up to i.
- Use System.out.println() to move to the next line after completing each row.

Conclusion

In this exploration of advanced patterns in Java, we've covered a variety of fascinating patterns that enhance our understanding of programming logic and aesthetics. Let's summarize what we've learned from character patterns, inverted triangle patterns, mirror image patterns, and isosceles triangle patterns.

Mastering these advanced patterns not only enhances your programming skills but also develops your ability to visualize and create intricate designs in code. These patterns are valuable for both problem-solving and artistic expression in programming. As you continue your journey in programming, you'll find these pattern-building techniques applicable in various contexts, from creating engaging user interfaces to solving algorithmic challenges.