Patterns

Introduction

Patterns are a handy application of loops and will provide you with better clarity and understanding of the implementation of loops.

Before printing any pattern, you must consider the following three things:

- The first step in printing any pattern is to figure out the number of rows that the pattern requires.
- Next, you should know how many columns are there in the ith row.
- Once, you have figured out the number of rows and columns, then focus on the pattern to print.

For eg. We want to print the following pattern for N rows: **(Pattern 1.1)**

```
// For N=4:
****
****
****
****
```

Approach:

From the above pattern, we can observe:

- ➔ **Number of Rows:** The pattern has 4 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** All the rows have 4 columns. Thus, in a pattern of N rows, all the rows will have N columns.
- ➔ **What to print:** We have to print `*` 4 times in all the 4 rows. Thus, in a pattern of N rows, we will have to print `*` N times in all the rows.

Now, let us discuss how to implement such patterns using Java.

Java Implementation for Patterns

We generally need two loops to print patterns. The outer loop iterates over the rows, while the inner nested loop is responsible for traversing the columns. The **algorithm** to print any pattern can be described as follows:

- Accept the number of rows or size of the pattern from a user using the `.nextInt()` function.
- Iterate the rows using the outer loop.
- Use the nested inner loop to handle the column contents. The internal loop iteration depends on the values of the outer loop.
- Print the required pattern contents using the `print` function.
- Add a new line after each row.

The implementation of **Pattern 1.1** in Java will be:

Step 1: Let us first use a loop to traverse the rows. This loop will start at the first row and go on till the Nth row. Below is the implementation of this loop:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N row
        // <Here goes the Nested Loop>
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new line after each row
    }
}
```

Printing a New Line: Since we need to print the pattern in multiple lines, we will have to add a new line after each row. Thus for this purpose, we use an empty `print` statement. The `print` function in Java, can be written as `System.out.println()`. '`ln`' after print indicates a new line.

Step 2: Now, we need another loop to traverse the row during each iteration and print the pattern; this can be done as follows:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print(“*”) // printing (*) in each column
            col = col+1 //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

There are two popular types of patterns-related questions that are usually posed:

- Square Pattern - **Pattern 1.1** is square.
- Triangular Pattern

Let us now look at the implementation of some common patterns.

Square Patterns

Pattern 1.2

```
// N = 5
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

Approach:

From the above pattern **we can observe:**

- **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.
- **Number of Columns:** All the rows have 5 columns. Thus, in a pattern of N rows, all the rows will have N columns.
- **What to print:** All the entries in any row are the same as the corresponding row numbers. Thus in a pattern of N rows, all the entries of the `i`th row are `i` (1st row has all 1's, 2nd row has all 2's, and so on).

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print(row); // printing (*) in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Pattern 1.3

```
// N = 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Approach:

From the above pattern **we can observe:**

➔ **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.

➔ **Number of Columns:** All the rows have 5 columns. Thus, in a pattern of N rows, all the rows will have N columns.

➔ **What to print:** All the entries in any row are the same as the corresponding column numbers. Thus in a pattern of N rows, all the entries of the `i`th column are `i` (1st column has all 1's, 2nd column has all 2's, and so on).

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                        //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print(col); // printing (*) in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Pattern 1.4

```
// N = 5
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
```

Approach:

From the above pattern **we can observe:**

- ➔ **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** All the rows have 5 columns. Thus, in a pattern of N rows, all the rows will have N columns.
- ➔ **What to print:** All the entries in any row are `N-columnNumber+1`. Thus in a pattern of N rows, all the entries of the `i`th column are `N-i+1` (1st column has all 5's (`5-1+1`), 2nd column has all 4's (`5-2+1`), and so on).

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                        //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print(N-col+1); // printing (*) in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

This way there can be several other square patterns and you can easily print them using this approach- **By finding the number of Rows, Columns and What to print**.

Pattern 1.5

```
// N = 5
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

Approach:

From the above pattern **we can observe:**

➔ **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.

➔ **Number of Columns:** All the rows have 5 columns. Thus, in a pattern of N rows, all the rows will have N columns.

➔ **What to print:** The first entry in the 1st row is 1, the first entry in the 2nd row is 2, and so on. Further, these values are incremented continuously by 1 in the remaining entries of any particular row. Thus in a pattern of N rows, the first entry of the `i`th row is `i`. The remaining entries in the `i`th row are `i+1,i+2, and so on`. It can be observed that any entry in this pattern can be written as `row+col-1`.

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print(col+row-1); // printing in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Triangular Patterns

Pattern 1.6

```
// N = 5
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Approach:

From the above pattern **we can observe:**

- ➔ **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** The number of columns in any row is the same as the corresponding row number.1st row has 1 column, 2nd row has 2 columns, and so on. Thus, in a pattern of N rows, the `i`th row will have `i` columns.
- ➔ **What to print:** All the entries in any row are the same as the corresponding row numbers. Thus in a pattern of N rows, all the entries of the `i`th row are `i` (1st row has all 1's, 2nd row has all 2's, and so on).

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= row) { //Loop will on for row times
            System.out.print(row); // printing in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Pattern 1.7

```
// N = 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Approach:

From the above pattern **we can observe:**

➔ **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.

- ➔ **Number of Columns:** The number of columns in any row is the same as the corresponding row number.1st row has 1 column, 2nd row has 2 columns, and so on. Thus, in a pattern of N rows, the `i`th row will have `i` columns.
- ➔ **What to print:** All the entries in any row are the same as the corresponding column numbers. Thus in a pattern of N rows, all the entries of the `i`th column are `i` (1st column has all 1's, 2nd column has all 2's, and so on).

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= row) { //Loop will on for row times
            System.out.print(col); // printing in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Pattern 1.8

```
// N = 5
1
2 3
4 5 6
7 8 9 10
```

```
11 12 13 14 15
```

Approach:

From the above pattern **we can observe:**

- ➔ **Number of Rows:** The pattern has 5 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** The number of columns in any row is the same as the corresponding row number.1st row has 1 column, 2nd row has 2 columns, and so on. Thus, in a pattern of N rows, the `i`th row will have `i` columns.
- ➔ **What to print:** The pattern starts with 1 and then each column entry is incremented by 1. Thus, we will initialize a variable `temp=1`. We will keep printing the value of `temp` in the successive columns and upon printing, we will increment the value of temp by 1.

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    int temp = 1;
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= row) { //Loop will on for row times
            System.out.print(temp); // printing in each column
            temp = temp+1;
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Character Patterns

Pattern 1.9

```
// N = 4
ABCD
ABCD
ABCD
ABCD
```

Approach:

From the above pattern **we can observe:**

- ➔ **Number of Rows:** The pattern has 4 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** All the rows have 4 columns. Thus, in a pattern of N rows, all the rows will have N columns.
- ➔ **What to print:** The 1st column has all A's, 2nd column has all B's, and so on. The **ASCII** value of A is `65`. In the 1st column, the character corresponds to the **ASCII** value 65 `(64+1).` In the 2nd column, the character corresponds to the **ASCII** value 66 `(64+2)`. Thus, all the entries in the `i`th column are equal to the character corresponding to the **ASCII** value **`64+i`**. The `char()` function gives the character associated with the integral ASCII value within the parentheses.

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
```

```
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print((char)(64+col)); //print in each column
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Pattern 1.10

```
// N = 4
ABCD
BCDE
CDEF
DEFG
```

Approach:

From the above pattern **we can observe:**

➔ **Number of Rows:** The pattern has 4 rows. We have to print the pattern for N rows.

➔ **Number of Columns:** All the rows have 4 columns. Thus, in a pattern of N rows, all the rows will have N columns.

➔ **What to print:** This pattern is very similar to **Pattern 1.5.** We can implement this using a similar code with a minor change. Instead of integers, we need capital letters of the same order. Instead of 1, we need A, instead of 2, we need B and so on. **ASCII** value of A is 65. Thus if we add 64 to all the entries in **Pattern 1.5** and find their **ASCII** values, we will get our result. The `char()` function gives the character associated with the integral **ASCII** value within the parentheses.

Java Implementation:

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt(); // Take user input, N= Number of Rows
    int row = 1; // The loop starts with the 1st row
    while (row <= N) { // Loop will on for N rows
        int col = 1; // loop starts with the first column in the
                     //current row
        while (col <= N) { //Loop will on for N columns
            System.out.print((char)(64+col+row-1)); //print character
            col = col+1; //Increment the current column (Inner Loop)
        }
        row = row+1; // Increment the current row (Outer Loop)
        System.out.println(); // Add a new Line after each row
    }
}
```

Practice Problems

Here are a few similar patterns problems for your practice. All the patterns have been drawn for N=4.

```
A
AB
ABC
ABCD
```

```
12344321
123**321
12****21
```

```
1******1
```

```
ABCD
ABC
AB
A
```

```
4555
3455
2345
1234
```

```
1
11
202
3003
```

```
A
BB
CCC
DDDD
```