Mastering Basic Pattern Printing: Ace Your Interviews knowing the most commonly asked questions

Pattern 1: Right-Angled Triangle

Let's start with the most basic pattern — a right-angled triangle of stars.

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

Intuition

The key insight here is that for row `i`, we need to print `i` stars. This is our first encounter with the relationship between row number and the number of elements to print.

Java Implementation

```
public class BasicPatterns {

    public static void printRightTriangle(int n) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println(); // Move to next line
        }
    }

    public static void main(String[] args) {
        printRightTriangle(5);
    }
}
```

Code Breakdown

- **Outer loop**: Controls the number of rows (1 to n)
- **Inner loop**: Prints stars for each row (1 to i)
- **Key relationship**: Row number = Number of stars

Pattern 2: Inverted Right-Angled Triangle

Now let's reverse the previous pattern:

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

Intuition

This is the mirror image of our first pattern. For row `i`, we need to print `(n - i + 1)` stars. This teaches us how to work with decreasing sequences.

Java Implementation

```
public static void printInvertedRightTriangle(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= (n - i + 1); j++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

Alternative Approach

```
public static void printInvertedRightTriangleAlt(int n) {
    for (int i = n; i >= 1; i--) {
        for (int j = 1; j <= i; j++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

Both approaches work, but the second one is more intuitive for many beginners.

Pattern 3: Pyramid (Centered Triangle)

Time to level up with a centered pyramid:

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

Intuition

This pattern introduces the concept of **spaces** before stars. The relationship becomes more complex:

- Row `i` has `(n - i)` spaces followed by `(2 * i - 1)` stars

- We need to think about both horizontal positioning and star count

Java Implementation

```
 public static void printPyramid(int n) {
    for (int i = 1; i <= n; i++) {
        // Print spaces
        for (int j = 1; j <= (n - i); j++) {
            System.out.print(" ");
        }

        // Print stars
        for (int j = 1; j <= (2 * i - 1); j++) {
            System.out.print("*");
        }

        System.out.println();
    }
}
```

Why (2 * i — 1) stars?

This is a common question. Let's trace through:

- Row 1: 1 star = 2(1) — 1 = 1
- Row 2: 3 stars = 2(2) — 1 = 3
- Row 3: 5 stars = 2(3) — 1 = 5

The pattern follows odd numbers: 1, 3, 5, 7, 9…

Pattern 4: Diamond Shape

Let's combine our knowledge to create a diamond:

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

Intuition

A diamond is essentially a pyramid followed by an inverted pyramid. We can break this into two parts:

1. Upper half (including middle): Regular pyramid

2. Lower half: Inverted pyramid

Java Implementation

```
 public static void printDiamond(int n) {
    // Upper half (including middle)
    for (int i = 1; i <= n; i++) {
        // Print spaces
        for (int j = 1; j <= (n - i); j++) {
            System.out.print(" ");
        }

        // Print stars
        for (int j = 1; j <= (2 * i - 1); j++) {
            System.out.print("*");
        }

        System.out.println();
    }

    // Lower half
    for (int i = n - 1; i >= 1; i--) {
        // Print spaces
        for (int j = 1; j <= (n - i); j++) {
            System.out.print(" ");
        }

        // Print stars
        for (int j = 1; j <= (2 * i - 1); j++) {
            System.out.print("*");
        }

        System.out.println();
    }
}
```

Pattern 5: Hollow Rectangle

Let's introduce the concept of hollow patterns:

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

Intuition

For hollow patterns, we need to identify:

- **Border positions**: Where to print the pattern character
- **Interior positions**: Where to print spaces

For a rectangle:

- First and last rows: All stars
- Middle rows: Stars only at first and last positions

Java Implementation

```
 public static void printHollowRectangle(int rows, int cols) {
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= cols; j++) {
            // Print star for border positions
            if (i == 1 || i == rows || j == 1 || j == cols) {
                System.out.print("*");
            } else {
                System.out.print(" ");
            }
        }
        System.out.println();
    }
}
```

Here's a complete program that demonstrates all patterns:

```
public class PatternPrinting {

    public static void printRightTriangle(int n) {
        System.out.println("Right Triangle:");
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void printInvertedRightTriangle(int n) {
        System.out.println("Inverted Right Triangle:");
        for (int i = n; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void printPyramid(int n) {
        System.out.println("Pyramid:");
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= (n - i); j++) {
                System.out.print(" ");
            }
            for (int j = 1; j <= (2 * i - 1); j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void printDiamond(int n) {
        System.out.println("Diamond:");
        // Upper half
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= (n - i); j++) {
                System.out.print(" ");
            }
            for (int j = 1; j <= (2 * i - 1); j++) {
                System.out.print("*");
            }
            System.out.println();
        }

        // Lower half
        for (int i = n - 1; i >= 1; i--) {
            for (int j = 1; j <= (n - i); j++) {
                System.out.print(" ");
            }
```

```
            for (int j = 1; j <= (2 * i - 1); j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void printHollowRectangle(int rows, int cols) {
        System.out.println("Hollow Rectangle:");
        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= cols; j++) {
                if (i == 1 || i == rows || j == 1 || j == cols) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void main(String[] args) {
        int n = 5;

        printRightTriangle(n);
        printInvertedRightTriangle(n);
        printPyramid(n);
        printDiamond(n);
        printHollowRectangle(n, 7);
    }
}
```