🧩 Core Concepts

- **Loop Constructs**
  - `while`, `for`, and `do-while` manage repeated execution.
  - They follow a structure: **Initialization** → **Condition** → **Execution** → **Update**.
- **Nested Loops**
  - A loop inside another loop, used for 2D structures like patterns.
  - Outer loop → rows; Inner loop → columns.
- **Control Variables**
  - `row` → number of iterations (lines)
  - `star` → number of printed characters
  - `space` → indentation before each row
- **Mirror Logic**
  - Patterns often reflect about a center axis (increasing and decreasing phases).

🌟 Increasing Star Triangle

```java
public class Pattern1 {
    public static void main(String[] args) {
        int n = 5;
        int row = 1;
        int star = 1;
        while (row <= 2 * n - 1) {
            int i = 1;
            while (i <= star) {
                System.out.print("* ");
                i++;
            }
            if (row < n) {
                star++;
            } else {
                star--;
            }
            System.out.println();
            row++;
        }
    }

}
```

🧠 **Explanation:**
Prints a symmetric triangle pattern of stars.
Stars increase line-by-line until the middle, then decrease.

💻 **Output:**

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

🌟 Centered Star Diamond

```java
public class Pattern2 {
    public static void main(String[] args) {
        int n = 5;
        int row = 1;
        int space = n - 1;
        int star = 1;
        while (row <= 2 * n - 1) {
            int i = 1;
            while (i <= space) {
                System.out.print("\t");
                i++;
            }
            int j = 1;
            while (j <= star) {
                System.out.print("*\t");
                j++;
            }
            if (row < n) {
                star += 2;
                space--;
            } else {
                star -= 2;
                space++;
            }
            System.out.println();
            row++;
        }
    }

}
```

🧠 **Explanation:**
This code prints a **diamond-shaped star pattern** centered with tabs.

💻 **Output:**

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

🌟 Number Diamond

```java
public class Pattern3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row = 1;
        int space = n - 1;
        int star = 1;
        int val = 1;
        while (row <= 2 * n - 1) {
            int i = 1;
            while (i <= space) {
                System.out.print("\t");
                i++;
            }
            int j = 1;
            while (j <= star) {
                System.out.print(val + "\t");
                j++;
            }
            if (row < n) {
                star += 2;
                space--;
                val++;
            } else {
                star -= 2;
                space++;
                val--;
            }
            System.out.println();
            row++;
        }
    }

}
```

🧠 **Explanation:**
This number-based diamond increases `val` in the upper half and decreases in the lower half, forming a mirrored numeric pattern.

💻 **Output (for n = 5):**

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

🌟 Hollow Butterfly

```java
public class Pattern4 {
    public static void main(String[] args) {
        int n = 7;
        int row = 1;
        int star = n / 2 + 1;
        int space = -1;
        while (row <= n) {
            int i = 1;
            while (i <= star) {
                System.out.print("* ");
                i++;
            }
            int j = 1;
            while (j <= space) {
                System.out.print("  ");
                j++;
            }
            int k = 1;
            if (row == 1 || row == n) {
                k = 2;
            }
            while (k <= star) {
                System.out.print("* ");
                k++;
            }
            if (row <= n / 2) {
                star--;
                space += 2;
            } else {
                star++;
                space -= 2;
            }
            System.out.println();
            row++;
        }
    }

}
```

🧠 **Explanation:**
Generates a hollow butterfly pattern by manipulating space between star blocks.
The first and last rows are fully filled.

💻 **Output:**

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

🌟 Number Pyramid

```java
public class Pattern5 {
    public static void main(String[] args) {
        int n = 5;
        int row = 1;
        int star = 1;
        int space = n - 1;
        int val = 1;
        while (row <= n) {
            int i = 1;
            while (i <= space) {
                System.out.print(" ");
                i++;
            }
            int j = 1;
            while (j <= star) {
                System.out.print(val + " ");
                val++;
                j++;
            }
            System.out.println();
            row++;
            star += 2;
            space--;
        }
    }

}
```

🧠 **Explanation:**
Prints a continuous increasing sequence of numbers in pyramid shape.

💻 **Output:**

```
  1
  2 3 4
 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25
```

🌟 Symmetric Number Pyramid

```java
public class Pattern6 {
    public static void main(String[] args) {
        int n = 5;
        int row = 1;
        int star = 1;
        int space = n - 1;
        while (row <= n) {
            int i = 1;
            while (i <= space) {
                System.out.print(" ");
                i++;
            }
            int j = 1;
            int val = 1;
            while (j <= star) {
                System.out.print(val + " ");
                if (j <= star / 2) {
                    val++;
                } else {
                    val--;
                }
                j++;
            }
            System.out.println();
            row++;
            star += 2;
            space--;
        }
    }

}
```

🧠 **Explanation:**
Numbers increase up to the middle of each row, then decrease, forming a symmetric numeric pyramid.

💻 **Output:**

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

🌟 Diamond Using Tabs

```
public class Pattern7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row = 1;
        int space = n - 1;
        int star = 1;
        while (row <= 2 * n - 1) {
            int i = 1;
            while (i <= space) {
                System.out.print("\t");
                i++;
            }
            int j = 1;
            while (j <= star) {
                System.out.print("*\t");
                j++;
            }
            if (row < n) {
                star += 2;
```

```java
                space--;
            } else {
                star -= 2;
                space++;
            }
            System.out.println();
            row++;
        }
    }

}
```

🧠 **Explanation:**
Similar to Pattern16, this uses tab spacing instead of normal spaces to ensure visual symmetry for console alignment.

💻 **Output (for n = 5):**

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

🌟 Solid Rhombus

```java
public class PatternRhombus {
    public static void main(String[] args) {
        int n = 5;
        int row = 1;
        int space = n - 1;
        int star = n;
        while (row <= n) {
            int i = 1;
            while (i <= space) {
                System.out.print(" ");
                i++;
            }
            int j = 1;
            while (j <= star) {
                System.out.print("* ");
                j++;
            }
            System.out.println();
            space--;
            row++;
        }
    }
}
```

🧠 **Explanation:**
Creates a **solid rhombus** — a shifted square made of stars.

💻 **Output:**

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