🔁 Loops in Java

Loops allow a programmer to execute a block of code multiple times under specific conditions.
They form the foundation of iteration and repetitive logic in programming.

Types of Loops

1. **for Loop** – Used when the number of iterations is known.

   ```
   for (initialization; condition; increment/decrement) {
      // body
   }
   ```

2. **while Loop** – A **pre-check loop**, executes only if the condition is true.

   ```
   while (condition) {
      // body
      // update loop variables
   }
   ```

3. **do-while Loop** – A **post-check loop**, executes the body at least once before checking the condition.

   ```
   do {
      // body
   } while (condition);
   ```

⚙️ Fundamental Examples

🌟 Taking User Input

```java
import java.util.*;

public class UserInput {
      public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter a number : ");
            int n = sc.nextInt();
            System.out.println("You entered : " + n);
            sc.close();
      }

}
```

💻 Output (Example):

```
Enter a number :
7
You entered : 7
```

🧩 Pattern Printing with While Loops

Pattern programs rely on nested loops:

- Outer loop → rows
- Inner loops → spaces and stars

🌟 Printing Five Stars in a Row

```java
public class Pattern0{
      public static void main(String[] args) {
            int star = 5;
            int i = 1;
            while (i <= star) {
                  System.out.print("* ");
                  i++;
            }
      }

}
```

🧠 **Explanation:**
A simple `while` loop printing five `*` symbols on one line.
No newline used within the loop.

💻 **Output:**

```
* * * * *
```

🌟 Print a solid 5×5 block of stars

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

}
```

🧠 **Explanation:**
Each row prints `n` stars – creating a square pattern.

💻 **Output:**

```
* * * * *

* * * * *

* * * * *

* * * * *

* * * * *
```

🌟 Right Triangle

```
public class Pattern2 {
      public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int n = sc.nextInt();
            int row = 1;
            int star = 1;
            while (row <= n) {
                  int j = 1;
                  while (j <= star) {
                        System.out.print("* ");
                        j++;
                  }
                  System.out.println();
                  row++;
                  star++;
            }
            sc.close();
      }

}
```

💻 Example (n = 5):

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

🌟 Left Triangle

```
public class Pattern3 {
      public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int n = sc.nextInt();
            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;
                  while (j <= star) {
                        System.out.print("* ");
                        j++;
```

```
                    }
                    System.out.println();
                    row++;
                    star++;
                    space--;
            }
            sc.close();
      }

}
```

💻 **Example (n = 5):**

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

🌟 Center-Aligned Pyramid

```
public class Pattern4 {
    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;
            while (j <= star) {
                System.out.print("*");
                j++;
            }

            System.out.println();
            row++;
            star += 2;
            space--;
        }
    }

}
```

🧠 **Explanation:**
Spaces decrease and stars increase by 2 each row to form a pyramid.

💻 **Output:**

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

🌟 Mirrored Double Pyramid

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

}
```

🧠 **Explanation:**
Creates two mirrored pyramids separated by increasing spaces.

💻 **Output:**

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

🌟 Alternate Star 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;
                  while (j <= star) {
                        if (j % 2 != 0) System.out.print("* ");
                        else System.out.print("  ");
                        j++;
                  }
                  System.out.println();
                  row++;
                  star += 2;
                  space--;
            }
      }

}
```

🧠 **Explanation:**
Prints stars at odd positions for a hollow-style pattern.

💻 **Output:**

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

🌟 Hollow Frame

```java
import java.util.Scanner;

public class Pattern1 {
      public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();

      int row = 1;
      while(row <= n){
            int i = 1;
            if(row == 1 || row == n){
                  while(i <= n){
                  System.out.print("* ");
                  i++;
                  }
            }
            else{
            while(i <= n){
                  if(i==1 || i==n){
                        System.out.print("* ");
                  }
                  else{
                      System.out.print("  ");
                  }
                  i++;
                  }
            }

            System.out.println();
            row++;
      }

      sc.close();
      }
}
```

💻 **Output:**

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

💡 Practical Applications

- Repetition in UI and graphics rendering.
- Generating shapes, number series, or structured data output.
- Logic foundation for matrix, recursion, and algorithmic loops.