Iteration

Introduction

Iterations in Java are control structures that allow you to execute a block of code repeatedly once a specific condition is met to eliminate the need to rewrite code. Java provides several loops, each with its own use cases and syntax. Here are the main types of loops in Java:

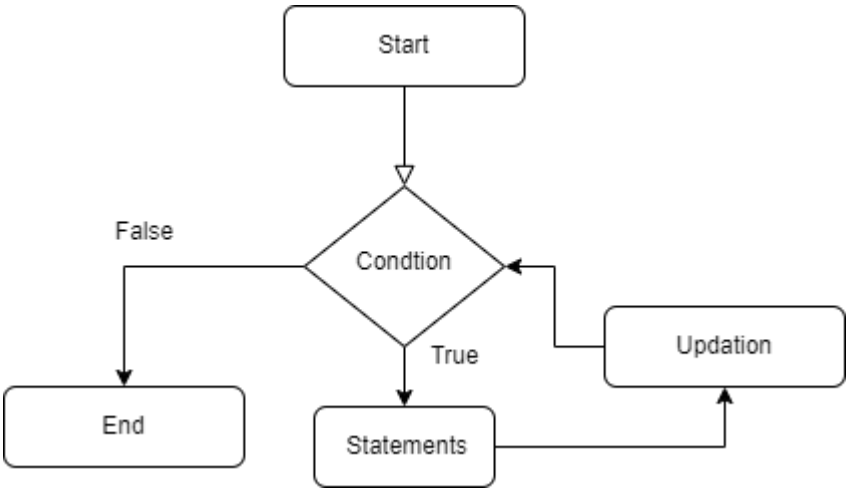
1. `for` Loop:

- The ‘for’ loop is the most commonly used loop in Java. It allows you to iterate over a sequence of values, typically numbers, for a specified number of times. The for loop consists of three steps: initialization, condition, and updation.
    1. Initialization: It's an expression used to initialise the loop variable.
    2. Condition: A boolean expression determining whether the loop should continue or terminate.
    3. Updation: It's an expression that updates the loop variable in each iteration.

The syntax is as follows:

```
for (initialization; condition; update) {
    // Code to be executed repeatedly
}
```

```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
//Will print numbers 1 to 5.
```



- **`break` statement:** The break statement is used to exit a loop prematurely, even if the loop's condition is still true. When encountered, it terminates the innermost loop (where the break statement is located) and continues with the code after the loop.
- Example:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break; // Exits the loop when i equals 5
    }
```

Note:- We can initialize variables outside the loop as well.

```
int i = 0;

for (; i < 5; i++) {
    System.out.println(i);
}
//Will print numbers 1 to 5.
```

2. `while` Loop

The `while` loop repeatedly executes a block of code as long as a specified condition is true. The syntax is as follows:

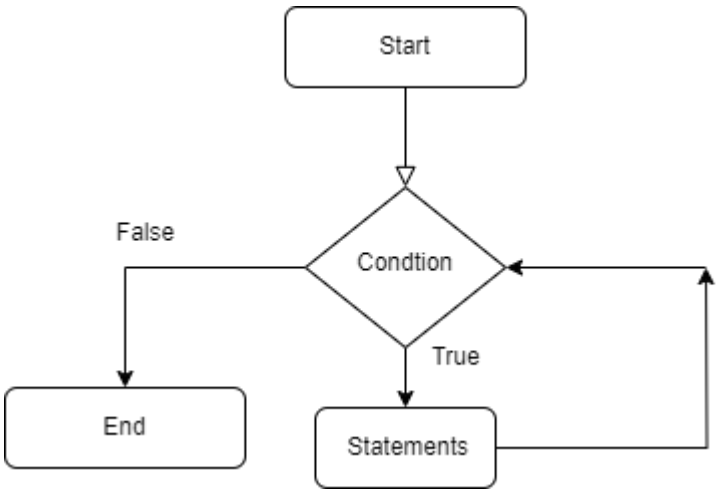
```
while (condition) {
    // Code to be executed repeatedly
}
```

Example:

```
int count = 0;
while (count < 5) {
    System.out.println(count);
    count++;
}

//will print numbers 0 to 4.
```

The above code can be visualized as below:



3. `do-while` Loop

The do-while loop is similar to the while loop, but it guarantees that the code block is executed at least once before checking the condition. The syntax is as follows:

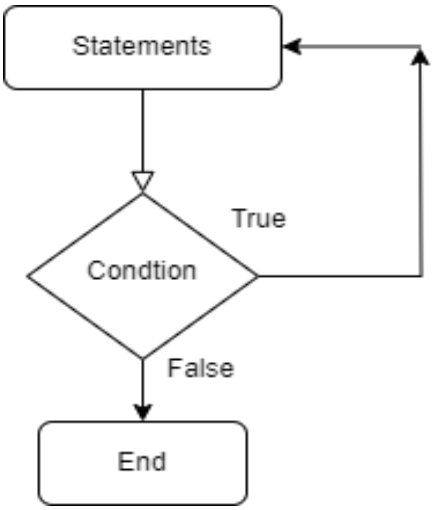
```
do {
    // Code to be executed repeatedly
} while (condition);
```

Example:

```
int number = 0;
do {
    System.out.println("Number: " + number);
    number++;
} while (number < 5);

//prints numbers 0 to 4.
```

The above code can be visualised as below:



4. Nested Loops

Nested loops in programming refer to placing one loop inside another loop. This technique is used to solve more complex problems involving iterating through multi-dimensional data structures, such as matrices, or performing repetitive tasks with multiple iteration levels. Here are some key points to understand about nested loops:

Nested loops are created by placing one loop inside another. Depending on your specific requirements, you can use any type of loop (e.g., for, while, while-while) as the outer loop or inner loop.

Example:

```
public class StarPattern {
    public static void main(String[] args) {
        int numRows = 5;

        // Outer loop for rows
        for (int i = 1; i <= numRows; i++) {
            // Inner loop for printing stars
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println(); // Move to the next line after each row
        }
    }
}

//
Will print stars in pattern:
*
* *
* * *
* * * *
* * * * *
//
```

Key Points to Remember:

- Ensure that you initialise loop variables correctly and that the termination condition is well-defined. Failing to do so can result in infinite loops or loops that never execute.
- If a loop variable needs to be used outside the loop, declare it outside the loop and initialise it appropriately. This ensures that the variable retains its value after the loop exits.

Conclusion

Loops in Java are fundamental control structures that enable the repetition of code execution based on specific conditions. They are pivotal in automating tasks, iterating through data structures, and solving various programming problems. Here are key takeaways about loops in Java:

- Java offers different loop constructs such as for, while, and do-while, each tailored to specific looping scenarios.
- The `for` loop is commonly used for iterating over sequences and is ideal when the number of iterations is known in advance.
- The `while` and `do-while` loops are helpful when the loop's termination depends on dynamic conditions or user input.