Class 06

In Java, a **for** loop is a control structure that allows you to execute a block of code repeatedly based on a specified condition. It's commonly used when you know in advance how many times you want to repeat a certain task**. Let's break down the parts of a for loop:**

java code

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

Here's what each part does:

1. **Initialization:** This is where you initialize a control variable (usually an integer) to a starting value. It's executed only once when the loop begins.
2. **Condition:** This is a boolean expression that's checked before each iteration of the loop. If the condition is **true**, the loop continues to execute. If it's **false**, the loop terminates.
3. **Update:** This part is responsible for changing the value of the control variable after each iteration. It's executed at the end of each iteration, just before re-checking the condition.

Now, let's look at an example:

java code

```
for (int i = 1; i <= 5; i++)
 {
 System.out.println("Iteration " + i);
}
```

In this example:

- Initialization: **int i = 1** initializes **i** to 1.
- Condition: **i <= 5** checks if **i** is less than or equal to 5.
- Update: **i++** increments **i** by 1 after each iteration.

The output of this loop will be:

java code

Iteration 1 Iteration 2 Iteration 3 Iteration 4 Iteration 5

The loop starts with **i** equal to 1. It prints "Iteration 1," increments **i** to 2, and so on until **i** becomes 6, at which point the condition becomes **false**, and the loop terminates.

You can use **for** loops for a wide range of tasks, such as iterating over arrays, processing data, or performing calculations. They provide precise control over the number of iterations, making them a valuable tool in Java programming.

Class 06

While and Do While Loops in Java

Let's Learn about the **while** and **do-while** loops in Java, providing detailed explanations and example programs. Both of these loops are essential for java developers.

while Loop

The **while** loop in Java is used to repeatedly execute a block of code as long as a specified condition is true. Here's the basic structure of a **while** loop:

java code

```
while (condition) {

// Code to be executed repeatedly

}
```

- **Condition:** The loop continues executing as long as the condition evaluates to **true**. If it becomes **false**, the loop terminates.

Example Program (while loop):

Let's write a simple program that uses a **while** loop to print numbers from 1 to 5:

java code

```
public class WhileLoopExample {

 public static void main(String[] args) {

 int i = 1; // Initialization

while (i <= 5) { // Condition

System.out.println("Number: " + i);

i++; // Update

}//while block

 }// main method block

 } // class block
```

In this program:

- **int i = 1** initializes **i** to 1.

- **i <= 5** checks if **i** is less than or equal to 5.
- **System.out.println("Number: " + i)** prints the value of **i**.
- **i++** increments **i** by 1 after each iteration.

This program will output

Number:1

Number:2

Number:3

Number:4

Number:5

do-while Loop

The **do-while** loop is similar to the **while** loop but with a crucial difference: it guarantees that the block of code is executed at least once, even if the condition is initially **false**. Here's the basic structure:

java code

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

- **Condition:** The condition is checked after the block of code executes. If it's **true**, the loop continues; if it's **false**, the loop terminates.

Example Program (do-while loop):

Let's write a program that uses a **do-while** loop to take input from the user until they enter a positive number:

java code

```
import java.util.Scanner;
public class DoWhileLoopExample {
public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 int number;
do {
System.out.print("Enter a positive number: ");
number = scanner.nextInt();
```

```
} while (number <= 0);

System.out.println("You entered a positive number: " + number);

scanner.close();

}

}
```

In this program:

- We use a **do-while** loop to repeatedly prompt the user for input.
- The loop continues until the user enters a positive number (greater than 0).

The **do-while** loop ensures that the input is collected at least once, making it suitable for situations where you want a guaranteed initial execution.

Both **while** and **do-while** loops are fundamental for controlling program flow and iteration in Java. The choice between them depends on your specific requirements.

for each loop in java

Lets learn about the "enhanced for loop" or "for-each loop" in Java. This loop is used for iterating over collections like arrays and lists without the need for indices. It's quite beginner-friendly and simplifies iteration.

Here's how it works:

java code

```
for (datatype element : collection) { // Code to be executed for each element }
```

- **datatype**: The type of elements in the collection.
- **element**: A variable to represent each element in the collection.
- **collection**: The array or collection you want to iterate over.

Example (for-each loop):

Let's say you have an array of integers and you want to print each element:

java code

```
public class ForEachLoopExample {

public static void main(String[] args) {

 int[] numbers = {1, 2, 3, 4, 5};

for (int num : numbers) {

System.out.println("Number: " + num);

} //for each block
```

} //main method block

} //class block

In this example:

- **int num** is used to represent each element in the **numbers** array.
- **for (int num : numbers)** iterates over each element in the array.
- **System.out.println("Number: " + num)** prints each element.

The for-each loop is efficient and simplifies the code when you need to iterate over elements in collections, making it a great choice for beginners.