SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

COURSE CODE & NAME : 23CST205 - Object Oriented Programming Using Java

II YEAR/ III SEMESTER

UNIT – II INTRODUCTION TO JAVA

Topic: BASICS OF JAVA PROGRAMMING-LOOPING

Java Looping

Control Flow in Java

Java compiler executes the java code from top to bottom. The statements are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of java code. Such statements are called control flow statements.

Java provides three types of control flow statements.

1. Decision Making statements
2. Loop statements
3. Jump statements

Java for Loop

Java for Loop

In computer programming, loops are used to repeat a block of code. For example, if you want to show a message 100 times, then rather than typing the same code 100 times, you can use a loop.

In Java, there are three types of loops.

- for loop
- while loop
- do...while loop

Java for Loop

Java for Loop

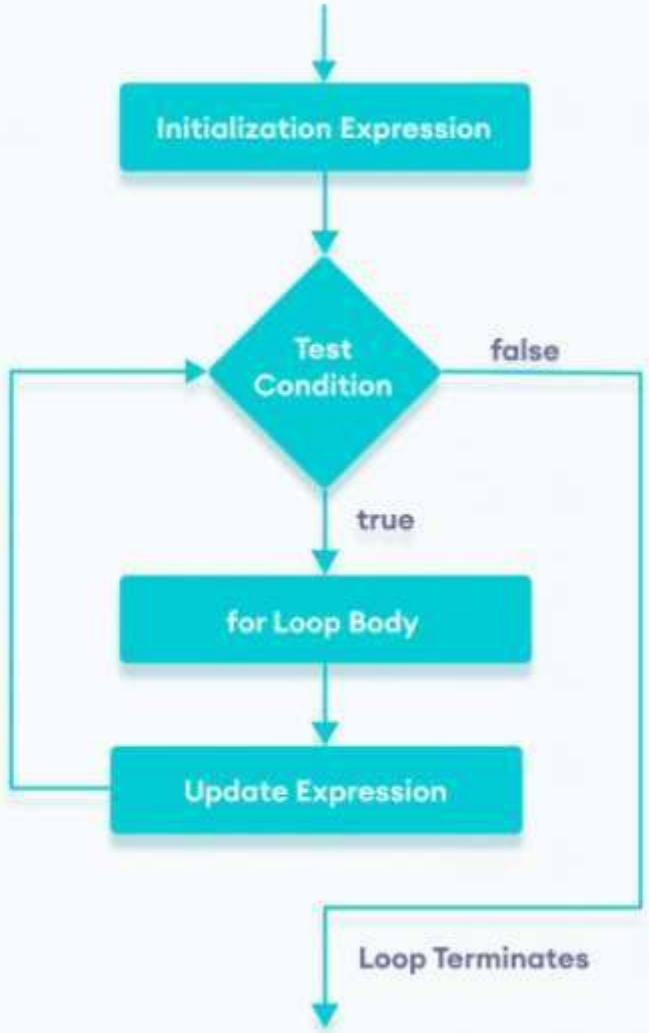
Java `for` loop is used to run a block of code for a certain number of times. The syntax of `for` loop is:

```
for (initialExpression; testExpression; updateExpression) {
    // body of the loop
}
```

Here,

1. The **initialExpression** initializes and/or declares variables and executes only once.
2. The **condition** is evaluated. If the **condition** is `true`, the body of the `for` loop is executed.
3. The **updateExpression** updates the value of **initialExpression**.
4. The **condition** is evaluated again. The process continues until the **condition** is `false`.

Java for Loop



Java for Loop

Example 1: Display a Text Five Times

```
// Program to print a text 5 times

class Main {
  public static void main(String[] args) {

    int n = 5;
    // for loop
    for (int i = 1; i <= n; ++i) {
      System.out.println("Java is fun");
    }
  }
}
```

Output

```
Java is fun
Java is fun
Java is fun
Java is fun
Java is fun
```

Java for Loop

Here is how this program works.

Iteration	Variable	Condition: i <= n	Action
1st	`i = 1` `n = 5`	`true`	`Java is fun` is printed. `i` is increased to **2**.
2nd	`i = 2` `n = 5`	`true`	`Java is fun` is printed. `i` is increased to **3**.
3rd	`i = 3` `n = 5`	`true`	`Java is fun` is printed. `i` is increased to **4**.
4th	`i = 4` `n = 5`	`true`	`Java is fun` is printed. `i` is increased to **5**.
5th	`i = 5` `n = 5`	`true`	`Java is fun` is printed. `i` is increased to **6**.
6th	`i = 6` `n = 5`	`false`	The loop is terminated.

Java for-each Loop

Java for-each Loop

In Java, the **for-each** loop is used to iterate through elements of arrays and collections (like ArrayList). It is also known as the enhanced for loop.

for-each Loop Sytnax

The syntax of the Java **for-each** loop is:

```
for(dataType item : array) {
    ...
}
```

Here,

- **array** - an array or a collection
- **item** - each item of array/collection is assigned to this variable
- **dataType** - the data type of the array/collection

Java for-each Loop

Example 1: Print Array Elements

```
// print array elements

class Main {
  public static void main(String[] args) {

    // create an array
    int[] numbers = {3, 7, 5, -5};

    // iterating through the array
    for (int number: numbers) {
       System.out.println(number);
    }
  }
}
```

Output

```
3
7
5
-5
```

Java Looping

for loop Vs for-each loop

Let's see how a `for-each` loop is different from a regular Java for loop.

1. Using for loop

```
class Main {
 public static void main(String[] args) {

   char[] vowels = {'a', 'e', 'i', 'o', 'u'};

   // iterating through an array using a for loop
   for (int i = 0; i < vowels.length; ++ i) {
     System.out.println(vowels[i]);
   }
 }
}
```

Output:

```
a
e
i
o
u
```

Java Looping

2. Using for-each Loop

```
class Main {
 public static void main(String[] args) {

   char[] vowels = {'a', 'e', 'i', 'o', 'u'};

   // iterating through an array using the for-each loop
   for (char item: vowels) {
     System.out.println(item);
   }
 }
}
```

Output:

```
a
e
i
o
u
```

Here, the output of both programs is the same. However, the **for-each** loop is easier to write and understand.

This is why the **for-each** loop is preferred over the **for** loop when working with arrays and collections.

Java Looping

Java Infinite for Loop

If we set the **test expression** in such a way that it never evaluates to `false`, the `for` loop will run forever. This is called infinite for loop. For example,

```
// Infinite for Loop

class Infinite {
    public static void main(String[] args) {

        int sum = 0;

        for (int i = 1; i <= 10; --i) {
            System.out.println("Hello");
        }
    }
}
```

Here, the test expression , `i <= 10` , is never `false` and `Hello` is printed repeatedly until the memory runs out.

Java while Loop

Java while loop

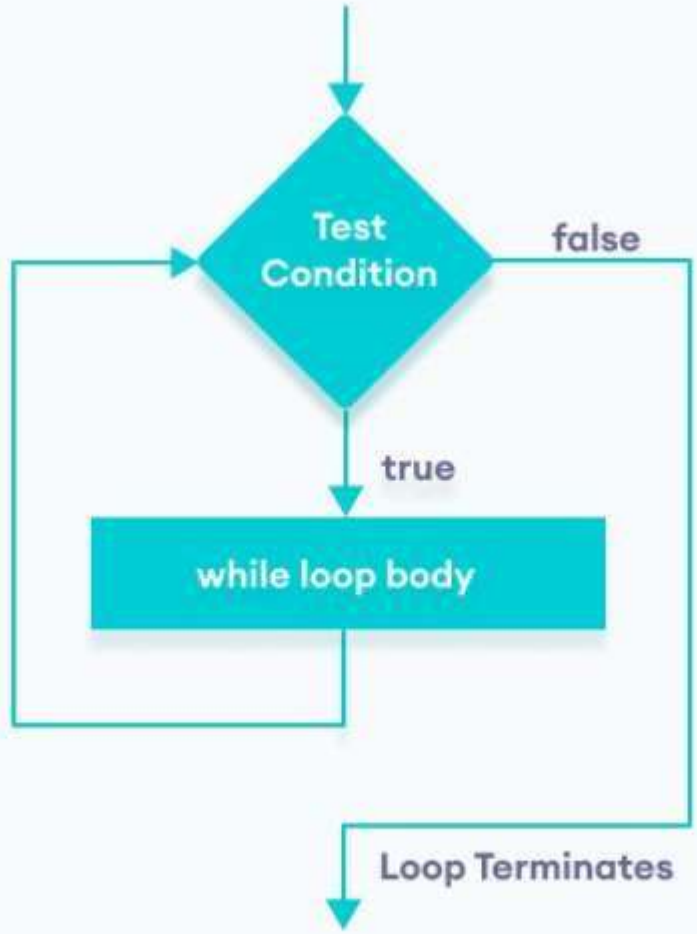
Java `while` loop is used to run a specific code until a certain condition is met. The syntax of the `while` loop is:

```
while (testExpression) {
    // body of loop
}
```

Here,

1. A `while` loop evaluates the **textExpression** inside the parenthesis `()`.
2. If the **textExpression** evaluates to `true`, the code inside the `while` loop is executed.
3. The **textExpression** is evaluated again.
4. This process continues until the **textExpression** is `false`.
5. When the **textExpression** evaluates to `false`, the loop stops.

Java while Loop



Flowchart of Java while loop

Java while Loop

Example 1: Display Numbers from 1 to 5

```
// Program to display numbers from 1 to 5

class Main {
  public static void main(String[] args) {

    // declare variables
    int i = 1, n = 5;

    // while loop from 1 to 5
    while(i <= n) {
      System.out.println(i);
      i++;
    }
  }
}
```

Output

```
1
2
3
4
5
```

Java while Loop

Here is how this program works.

Iteration	Variable	Condition: i <= n	Action
1st	`i = 1` `n = 5`	`true`	`1` is printed. `i` is increased to **2**.
2nd	`i = 2` `n = 5`	`true`	`2` is printed. `i` is increased to **3**.
3rd	`i = 3` `n = 5`	`true`	`3` is printed. `i` is increased to **4**.
4th	`i = 4` `n = 5`	`true`	`4` is printed. `i` is increased to **5**.
5th	`i = 5` `n = 5`	`true`	`5` is printed. `i` is increased to **6**.
6th	`i = 6` `n = 5`	`false`	The loop is terminated

Java do…while Loop

Java do...while loop

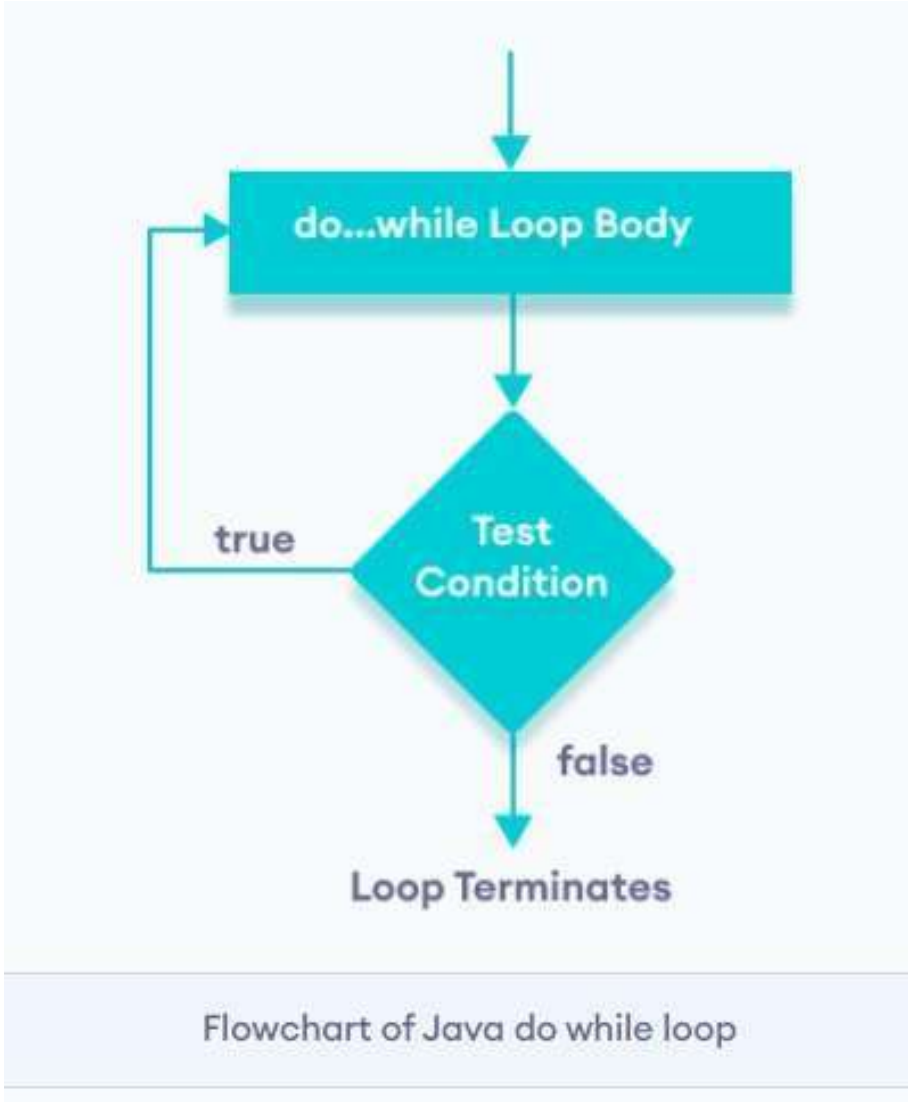
The `do...while` loop is similar to while loop. However, the body of `do...while` loop is executed once before the test expression is checked. For example,

```
do {
    // body of loop
} while(textExpression)
```

Here,

1. The body of the loop is executed at first. Then the **textExpression** is evaluated.
2. If the **textExpression** evaluates to `true`, the body of the loop inside the `do` statement is executed again.
3. The **textExpression** is evaluated once again.
4. If the **textExpression** evaluates to `true`, the body of the loop inside the `do` statement is executed again.
5. This process continues until the **textExpression** evaluates to `false`. Then the loop stops.

Java do…while Loop



Flowchart of Java do while loop

Java do…while Loop

Example 3: Display Numbers from 1 to 5

```
// Java Program to display numbers from 1 to 5

import java.util.Scanner;

// Program to find the sum of natural numbers from 1 to 100.

class Main {
  public static void main(String[] args) {

    int i = 1, n = 5;

    // do...while loop from 1 to 5
    do {
      System.out.println(i);
      i++;
    } while(i <= n);
  }
}
```

Output

```
1
2
3
4
5
```

Java do…while Loop

Here is how this program works.

Iteration	Variable	Condition: i <= n	Action
	`i = 1` `n = 5`	not checked	`1` is printed. `i` is increased to **2.**
1st	`i = 2` `n = 5`	`true`	`2` is printed. `i` is increased to **3.**
2nd	`i = 3` `n = 5`	`true`	`3` is printed. `i` is increased to **4.**
3rd	`i = 4` `n = 5`	`true`	`4` is printed. `i` is increased to **5.**
4th	`i = 5` `n = 5`	`true`	`6` is printed. `i` is increased to **6.**
5th	`i = 6` `n = 5`	`false`	The loop is terminated

Java do…while Loop

Infinite while loop

If **the condition** of a loop is always `true`, the loop runs for infinite times (until the memory is full). For example,

```
// infinite while loop
while(true){
    // body of loop
}
```

Here is an example of an infinite `do...while` loop.

```
// infinite do...while loop
int count = 1;
do {
   // body of loop
} while(count == 1)
```

In the above programs, the **textExpression** is always `true`. Hence, the loop body will run for infinite times.

Java Looping

for and while loops

The `for` loop is used when the number of iterations is known. For example,

```
for (let i = 1; i <=5; ++i) {
   // body of loop
}
```

And `while` and `do...while` loops are generally used when the number of iterations is unknown. For example,

```
while (condition) {
    // body of loop
}
```


THANK
YOU!