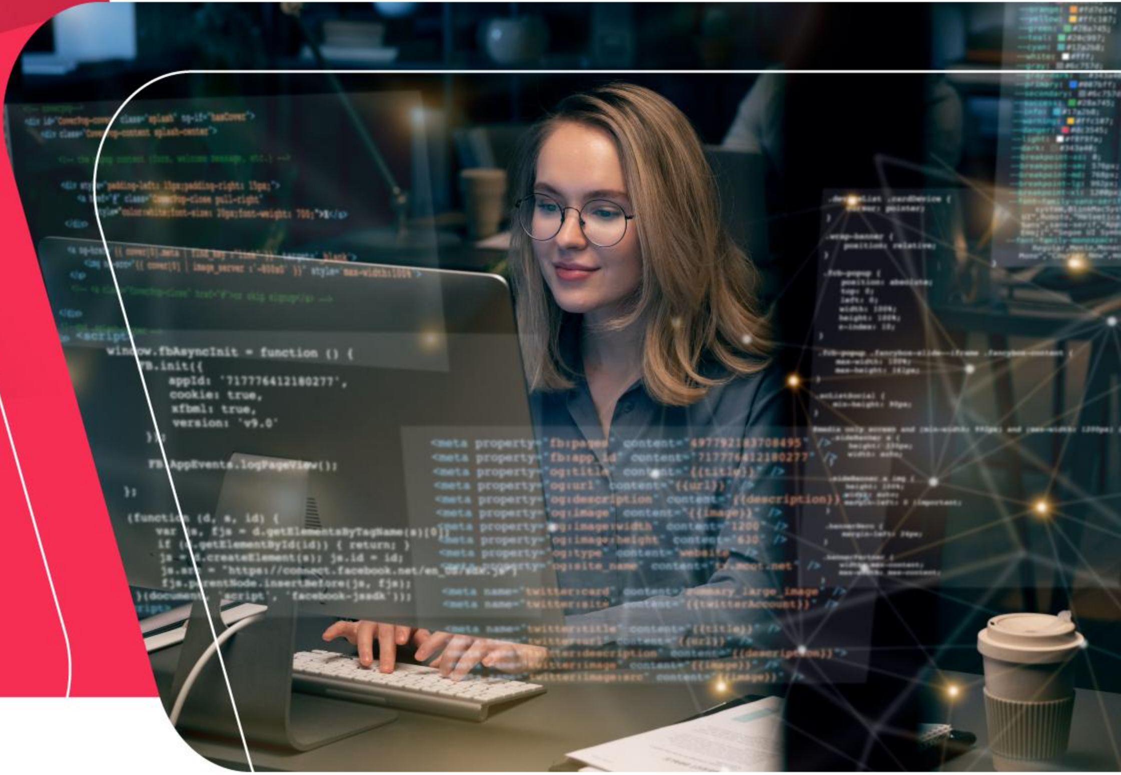
Java Bootcamp

Day 06



TIA-Academy

Technologies will be Use

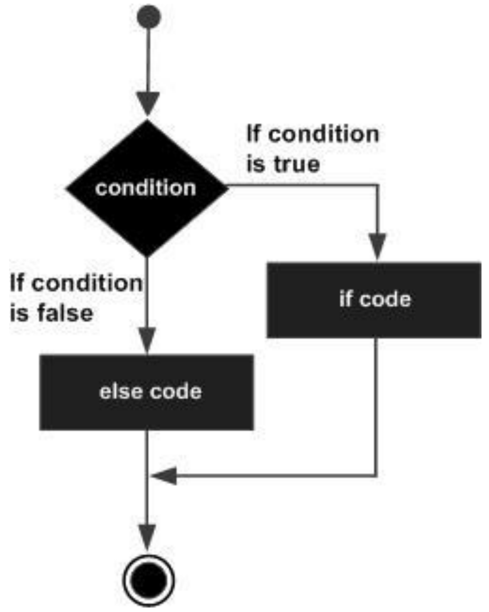
- JDK 8/**11**/15
- JRE 8/**11**/15
- Writing Code using Notepad++ (For Windows) or Vim (For Linux and Mac)
- Compiling with Command Prompt (For Windows) or Terminal (For Linux and Mac)
- ***Forbidden of using IDE Based Development Tools as of Intellij IDEA or Eclipse***

TIA-Academy

Java if...else Statement

Java if...else Statement

- In programming, we use the **if..else statement to run a block of code among more than one alternatives.**
- For example, assigning grades (A, B, C) based on the percentage obtained by a student.
  - if the percentage is above **90**, assign grade **A**
  - if the percentage is above **75**, assign grade **B**
  - if the percentage is above **65**, assign grade **C**



TIA-Academy

Java if (if-then) Statement

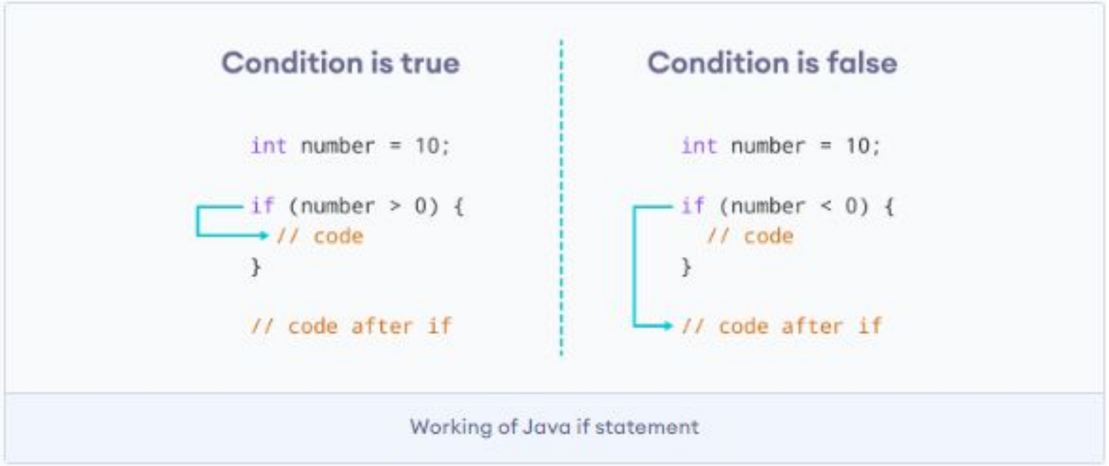
- The syntax of an if-then statement is:

```
if (condition) {
  // statements
}
```

- Here, condition is a boolean expression such as **age >= 18**.
  - if condition evaluates to **true, statements are executed**
  - if condition evaluates to **false, statements are skipped**

Java if (if-then) Statement

- Working of if Statement



```
int number = 10;

if (number > 0) {
  // code
}

// code after if
```

```
int number = 10;

if (number < 0) {
  // code
}

// code after if
```

Working of Java if statement

TIA-Academy

Java if (if-then) Statement

- Example 1: Java if Statement
- Output

```
Statement outside if block
```

- In the program, number < 0 is false.

  Hence, the code inside the parenthesis is

  skipped.

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

    int number = 10;

    // checks if number is less than 0
    if (number < 0) {
      System.out.println("The number is negative.");
    }

    System.out.println("Statement outside if block");
  }
}
```

Java if (if-then) Statement

- Example 2: Java if with String
- Output

```
Best Programming Language
```

- In the above example, we are comparing two strings in the if block.

```
class Main {
  public static void main(String[] args) {
    // create a string variable
    String language = "Java";

    // if statement
    if (language == "Java") {
      System.out.println("Best Programming Language");
    }
  }
}
```

TIA-Academy

Java if...else (if-then-else) Statement

- The if statement **executes a certain section of code if the test expression is evaluated to true**. However, if the test expression is **evaluated to false, it does nothing**.
- In this case, we can use an optional else block. Statements inside the body of else block are executed if the test expression is evaluated to false. This is known as the if-...else statement in Java.

TIA-Academy

Java if...else (if-then-else) Statement

- The syntax of the if...else statement is:

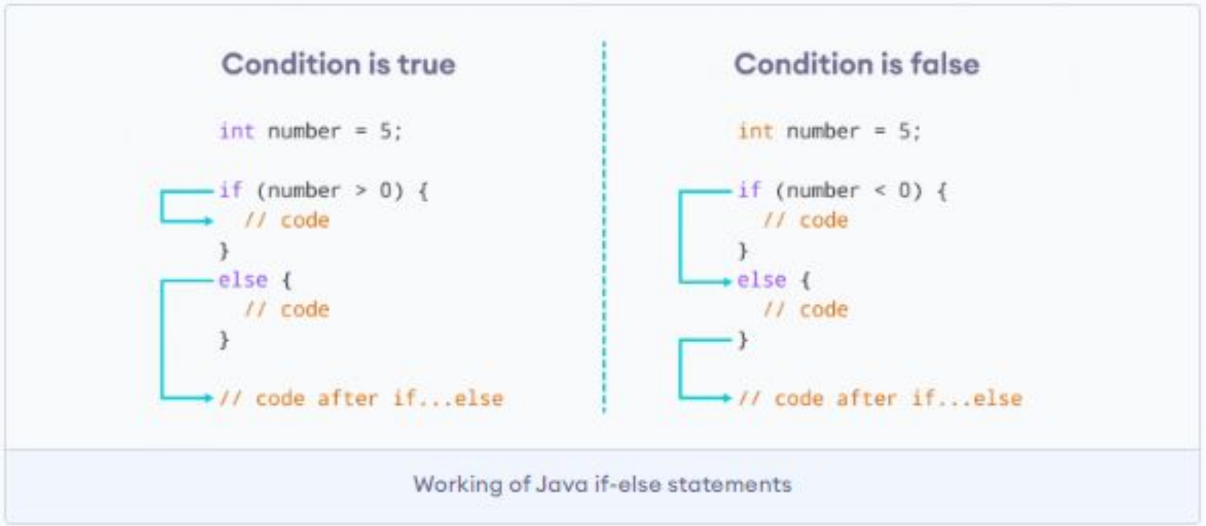
```
if (condition) {
   // codes in if block
}
else {
   // codes in else block
}
```

- Here, the program will do one task (codes inside if block) if the condition is true and another task (codes inside else block) if the condition is false.

TIA-Academy

Java if...else (if-then-else) Statement

- How the if...else statement works?



```
int number = 5;

if (number > 0) {
  // code
}
else {
  // code
}

// code after if...else
```

```
int number = 5;

if (number < 0) {
  // code
}
else {
  // code
}

// code after if...else
```

Working of Java if-else statements

Java if...else (if-then-else) Statement

- Example 3: Java if...else Statement
- Output :

```
The number is positive.
Statement outside if...else block
```

```
class Main {
  public static void main(String[] args) {
    int number = 10;

    // checks if number is greater than 0
    if (number > 0) {
      System.out.println("The number is positive.");
    }

    // execute this block
    // if number is not greater than 0
    else {
      System.out.println("The number is not positive.");
    }

    System.out.println("Statement outside if...else block");
  }
}
```

Java if...else (if-then-else) Statement

- In the above example, we have a variable named number. Here, the test expression number > 0 checks if number is greater than 0.
- Since the value of the number is 10, the test expression evaluates to true. Hence code inside the body of if is executed.

TIA-Academy

Java if...else (if-then-else) Statement

- Now, change the value of the number to a negative integer. Let's say -5.

```
int number = -5;
```

- If we run the program with the new value of number, the output will be:

```
The number is not positive.
Statement outside if...else block
```

- Here, the value of number is -5. So the test expression evaluates to false. Hence code inside the body of else is executed.

Java if...else...if Statement

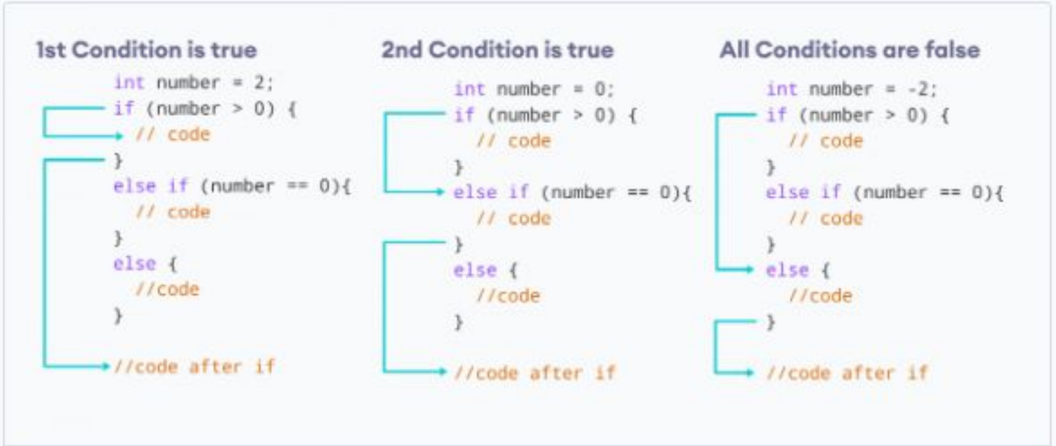
- In Java, we have an if...else...if ladder, that can be used to execute one block of code among multiple other blocks.
- Here, if statements are executed **from the top towards the bottom**. When the test condition is true, codes inside the body of that if block is executed. And, program control jumps outside the if...else...if ladder.
- If all test expressions are false, codes inside the body of else are executed.

```
if (condition1) {
   // codes
}
else if(condition2) {
   // codes
}
else if (condition3) {
   // codes
}
.
.
else {
   // codes
}
```

TIA-Academy

Java if...else...if Statement

- How the if...else...if ladder works?



1st Condition is true

```
int number = 2;
if (number > 0) {
  // code
}
else if (number == 0){
  // code
}
else {
  //code
}

//code after if
```

2nd Condition is true

```
int number = 0;
if (number > 0) {
  // code
}
else if (number == 0){
  // code
}
else {
  //code
}

//code after if
```

All Conditions are false

```
int number = -2;
if (number > 0) {
  // code
}
else if (number == 0){
  // code
}
else {
  //code
}

//code after if
```

Working of if...else...if ladder

TIA-Academy

Java if...else...if Statement

- Example 4: Java if...else...if Statement
- Output :

```
The number is 0.
```

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

    int number = 0;

    // checks if number is greater than 0
    if (number > 0) {
      System.out.println("The number is positive.");
    }

    // checks if number is less than 0
    else if (number < 0) {
      System.out.println("The number is negative.");
    }

    // if both condition is false
    else {
      System.out.println("The number is 0.");
    }
  }
}
```

Java if...else...if Statement

- In the above example, we are checking whether number is positive, negative, or zero.

  Here, we have two condition expressions:

  - number > 0 - checks if number is greater than 0
  - number < 0 - checks if number is less than 0

- Here, the value of number is 0. So both the conditions evaluate to false. Hence the statement inside the body of else is executed.

TIA-Academy

Java Nested if..else Statement

- In Java, it is also possible to use if..else statements inside an if...else statement. It's called the nested if...else statement.
- Here's a program to find the largest of 3 numbers using the nested if...else statement.
- Example 5: Nested if...else Statement

TIA-Academy

Java Nested if..else Statement

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

    // declaring double type variables
    Double n1 = -1.0, n2 = 4.5, n3 = -5.3, largest;

    // checks if n1 is greater than or equal to n2
    if (n1 >= n2) {

      // if...else statement inside the if block
      // checks if n1 is greater than or equal to n3
      if (n1 >= n3) {
        largest = n1;
      }

      else {
        largest = n3;
      }
    } else {

      // if..else statement inside else block
      // checks if n2 is greater than or equal to n3
      if (n2 >= n3) {
        largest = n2;
      }

      else {
        largest = n3;
      }
    }

    System.out.println("Largest Number: " + largest);
  }
}
```

Java Nested if..else Statement

- Output:

```
Largest Number: 4.5
```

- In the above programs, we have assigned the value of variables ourselves to make this easier.
- However, **in real-world applications, these values may come from user input data, log files, form submission, etc.**

TIA-Academy

ASSIGNMENT 01

Java switch Statement

Java switch Statement

- The switch statement allows us to execute a block of code among many alternatives.
- The syntax of the switch statement in Java is:

```
switch (expression) {

  case value1:
    // code
    break;

  case value2:
    // code
    break;

  ...
  ...

  default:
    // default statements
  }
```

TIA-Academy

Java switch Statement

How does the switch-case statement work?

- The expression is evaluated once and compared with the values of each case.
  - If expression matches with value1, the code of case value1 are executed. Similarly, the code of case value2 is executed if expression matches with value2.
  - If there is no match, the code of the default case is executed.
- **Note:** The working of the switch-case statement is similar to the Java if...else...if ladder. However, **the syntax of the switch statement is cleaner and much easier to read and write.**

Java switch Statement

- Example: Java switch Statement
- Output :

```
Size: Large
```

```
// Java Program to check the size
// using the switch...case statement

class Main {
  public static void main(String[] args) {
    int number = 44;
    String size;

    // switch statement to check size
    switch (number) {

      case 29:
        size = "Small";
        break;
      case 42:
        size = "Medium";
        break;
      // match the value of week
      case 44:
        size = "Large";
        break;
      case 48:
        size = "Extra Large";
        break;
      default:
        size = "Unknown";
        break;
    }
    System.out.println("Size: " + size);
  }
}
```

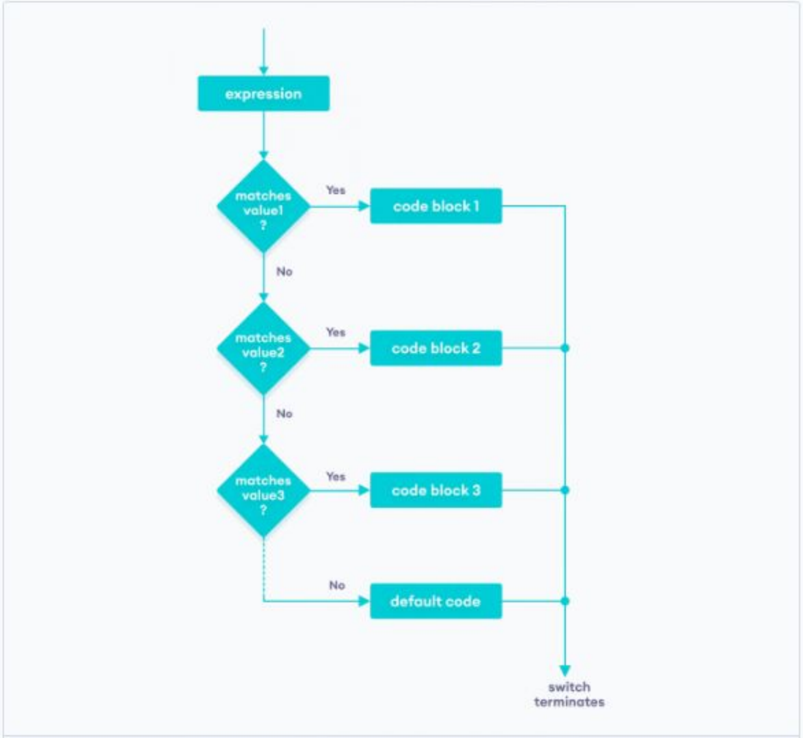
Java switch Statement

- In the above example, we have used the switch statement to find the size. Here, we have a variable **number**. The **variable is compared with the value of each case statement**.
- Since the value matches with 44, the code of case 44 is executed.

```
 size = "Large";
break;
```

- Here, the size variable is assigned with the value Large.

Java switch Statement



Flow chart of the Java switch statement

- Flowchart of switch Statement

break statement in Java switch...case

- Notice that we have been using break in each case block.

```
 ...
case 29:
  size = "Small";
  break;
...
```

TIA-Academy

break statement in Java switch...case

- The break statement is used to terminate the switch-case statement.
- If break is not used, all the cases after the matching case are also executed.
- For example :

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

    int expression = 2;

    // switch statement to check size
    switch (expression) {
      case 1:
        System.out.println("Case 1");
        // matching case
      case 2:
        System.out.println("Case 2");
      case 3:
        System.out.println("Case 3");
      default:
        System.out.println("Default case");
    }
  }
}
```

- Output :

```
Case 2
Case 3
Default case
```

break statement in Java switch...case

- In the above example, **expression matches with case 2.** Here, we haven't used the break statement after each case.
- Hence, all the cases after case 2 are also executed.
- This is why the **break statement is needed to terminate** the switch-case statement after the matching case. To learn more, visit Java break Statement.

TIA-Academy

default case in Java switch-case

- The switch statement also includes an optional default case.
- It is executed when the expression doesn't match any of the cases. For example :

- Output :

```
Unknown Size
```

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

    int expression = 9;

    switch(expression) {

      case 2:
        System.out.println("Small Size");
        break;

      case 3:
        System.out.println("Large Size");
        break;

      // default case
      default:
        System.out.println("Unknown Size");
    }
  }
}
```

TIA-Academy

default case in Java switch-case

- In the above example, we have created a switch-case statement. Here**, the value of expression doesn't match with any of the cases.**
- Hence, the code inside the default case is executed.

```
default:
  System.out.println("Unknown Size);
```

default case in Java switch-case

Note: The Java switch statement only works with:

- **Primitive data types: byte, short, char, and int**
- *Enumerated types*
- **String Class**
- *Wrapper Classes: Character, Byte, Short, and Integer*

TIA-Academy

ASSIGNMENT 02

TIA-Academy

ASSIGNMENT 03

TIA-Academy

Java for Loop

TIA-Academy

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
- This tutorial focuses on the for loop. You will learn about the other type of loops in the upcoming tutorials.

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:

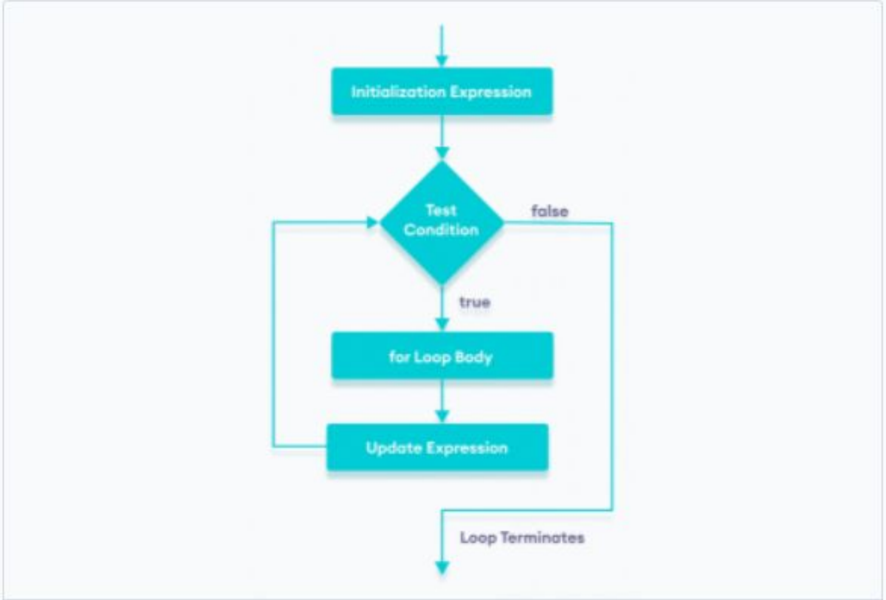
```
for (initialExpression; testExpression; updateExpression) {
    // body of the loop
}
```

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.

TIA-Academy

Java for Loop



Flowchart of Java for loop

Java for Loop

- Example 1: Display a Text Five Times
- Output

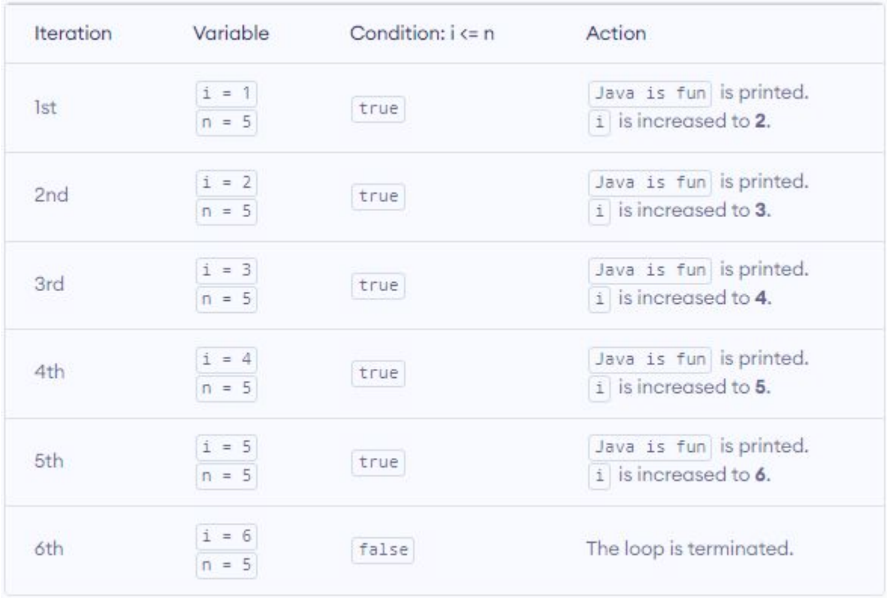
```
Java is fun
Java is fun
Java is fun
Java is fun
Java is fun
```

```
// 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");
    }
  }
}
```

TIA-Academy

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.

TIA-Academy

Java for Loop

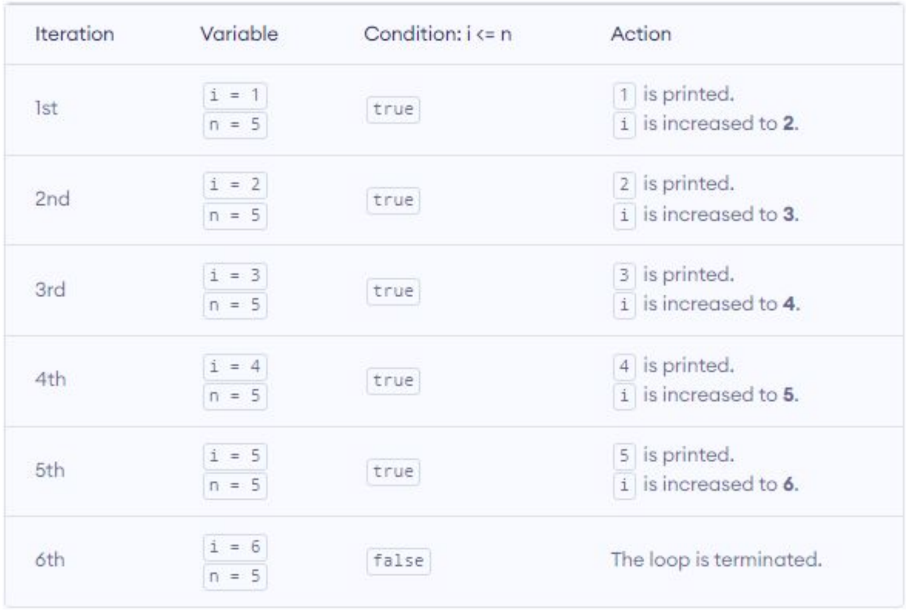
- Example 2: Display numbers from 1 to 5
- Output

```
1
2
3
4
5
```

```
// Program to print numbers from 1 to 5
class Main {
  public static void main(String[] args) {

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

Java for Loop

- Here is how the 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.

TIA-Academy

Java for Loop

- Example 3: Display Sum of n Natural Numbers
- Output

```
Sum = 500500
```

```
// Program to find the sum of natural numbers
// from 1 to 1000.
class Main {
  public static void main(String[] args) {

    int sum = 0;
    int n = 1000;

    // for loop
    for (int i = 1; i <= n; i++) {
      // body inside for loop
      sum += i;     // sum = sum + i
    }
    System.out.println("Sum = " + sum);
  }
}
```

Java for Loop

- Here, the value of sum is 0 initially. Then, the for loop is iterated from **i = 1 to 1000**. In each iteration, iis added to sum and its value is increased by 1.
- When i becomes 1001, the test condition is false and sum will be equal to **0 + 1 + 2 + ... + 1000.**

Java for Loop

- The above program to add the sum of natural numbers can also be written as
- The output of this program is the same as the Example 3.

```
// Program to find the sum of natural numbers
// from 1 to 1000.
class Main {
  public static void main(String[] args) {

    int sum = 0;
    int n = 1000;

    // for loop
    for (int i = n; i >= 1; --i) {
      // body inside for loop
      sum += i;     // sum = sum + i
    }

    System.out.println("Sum = " + sum);
  }
}
```

TIA-Academy

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

```
// 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");
        }
    }
}
```

ASSIGNMENT 03

Java for-each Loop

TIA-Academy

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.

- 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

TIA-Academy

Java for-each Loop

- Example 1: Print Array Elements
- Output

```
3
9
5
-5
```

```java
// print array elements
class Main {
  public static void main(String[] args) {

    // create an array
    int[] numbers = {3, 9, 5, -5};

    // for each loop
    for (int number: numbers) {
      System.out.println(number);
    }
  }
}
```

TIA-Academy

Java for-each Loop

Here, we have used the for-each loop to print each element of the numbers array one by one.

- In the **first iteration**, item will be **3**.
- In the **second iteration**, item will be **9**.
- In the **third iteration**, item will be **5**.
- In the **fourth iteration**, item will be **-5**.

TIA-Academy

Java for-each Loop

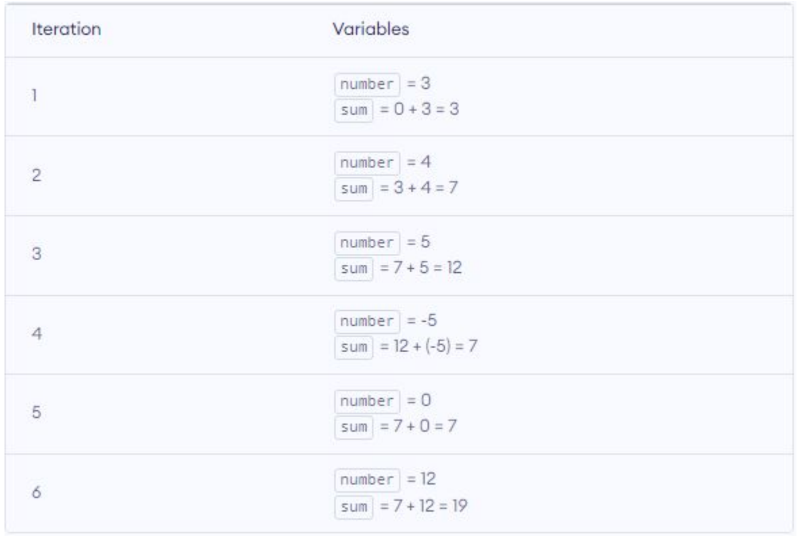
- Example 2: Sum of Array Elements
- Output

```
Sum = 19
```

```
// Calculate the sum of all elements of an array
class Main {
 public static void main(String[] args) {

   // an array of numbers
   int[] numbers = {3, 4, 5, -5, 0, 12};
   int sum = 0;

   // iterating through each element of the array
   for (int number: numbers) {
     sum += number;
   }

   System.out.println("Sum = " + sum);
 }
}
```

Java for-each Loop

- In the above program, the execution of the for each loop looks as:



Iteration	Variables
1	`number` = 3 `sum` = 0 + 3 = 3
2	`number` = 4 `sum` = 3 + 4 = 7
3	`number` = 5 `sum` = 7 + 5 = 12
4	`number` = -5 `sum` = 12 + (-5) = 7
5	`number` = 0 `sum` = 7 + 0 = 7
6	`number` = 12 `sum` = 7 + 12 = 19

- As we can see, we have added each element of the numbers array to the sum variable in each iteration of the loop.

for loop Vs for-each loop

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

```
a
e
i
o
u
```

```
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]);
   }
 }
}
```

TIA-Academy

for loop Vs for-each loop

- Using for-each Loop
- Output:

```
a
e
i
o
u
```

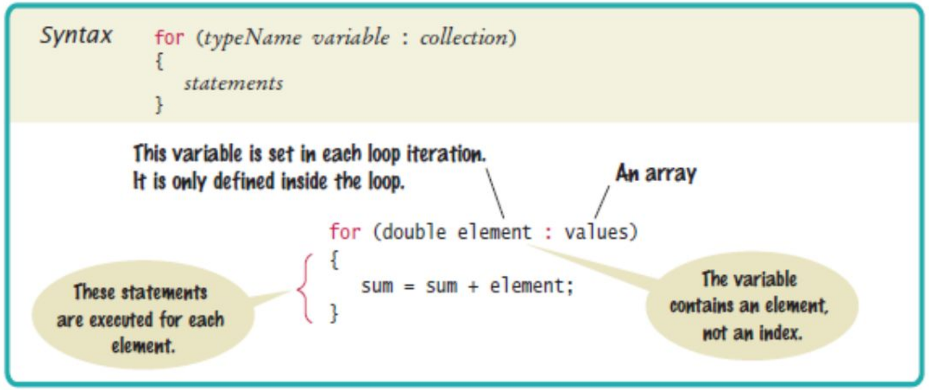
```
class Main {
 public static void main(String[] args) {

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

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

TIA-Academy

for loop Vs for-each loop

- 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.

The Enhanced "for" Loop



```
Syntax   for (typeName variable : collection)
         {
            statements
         }
```

```
for (double element : values)
{
   sum = sum + element;
}
```

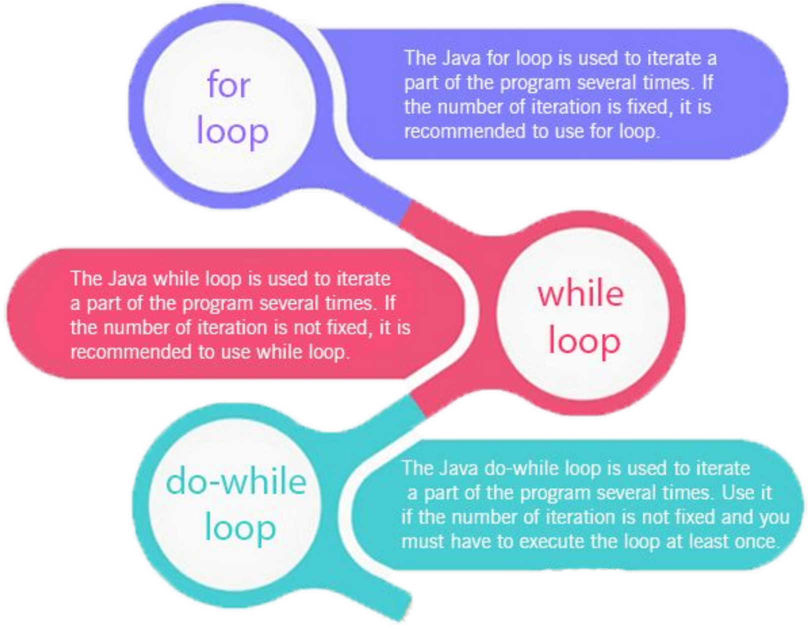
This variable is set in each loop iteration. It is only defined inside the loop.
An array
These statements are executed for each element.
The variable contains an element, not an index.

TIA-Academy

Java while and do...while Loop

TIA-Academy

Java while and do...while Loop


for loop
The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.
while loop
The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.
do-while loop
The Java do-while loop is used to iterate a part of the program several times. Use it if the number of iteration is not fixed and you must have to execute the loop at least once.

- In computer programming, loops are used to repeat a block of code.
- For example, **if you want to show a message 100 times, then you can use a loop.**
- It's just a simple example; you can achieve much more with loops.
- In the previous tutorial, you learned about Java for loop. Here, you are going to learn about **while** and **do...while** loops.

TIA-Academy

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
}
```

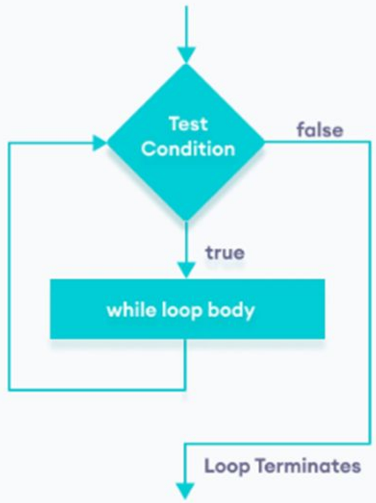
- Here,

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.

TIA-Academy

Java while loop

- Flowchart of while loop



Flowchart of Java while loop

TIA-Academy

Java while loop

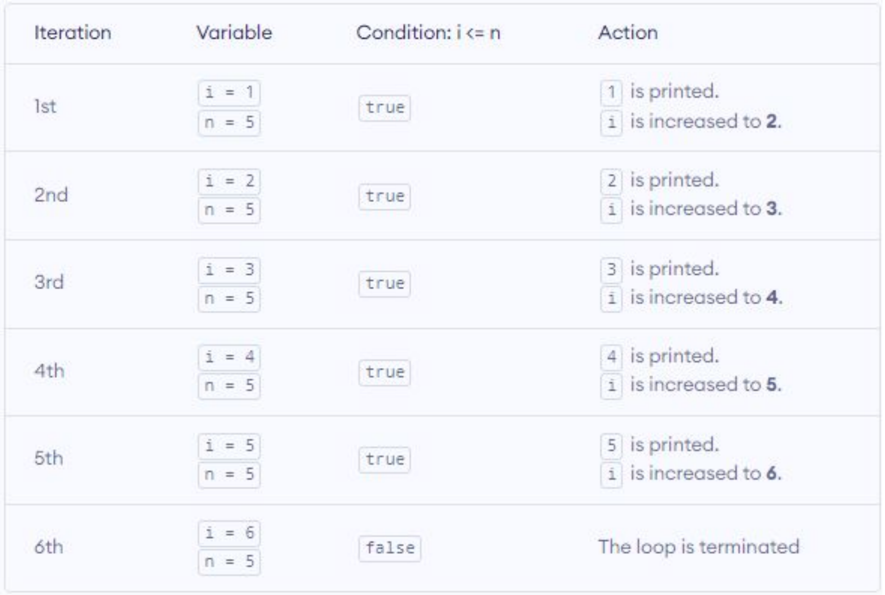
- Example 1: Display Numbers from 1 to 5
- Output

```
1
2
3
4
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++;
    }
  }
}
```

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

TIA-Academy

Java while loop

- Example 2: Sum of Positive Numbers Only
- Output

```
Enter a number
25
Enter a number
9
Enter a number
5
Enter a number
-3
Sum = 39
```

```java
// Java program to find the sum of positive numbers
import java.util.Scanner;

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

    int sum = 0;

    // create an object of Scanner class
    Scanner input = new Scanner(System.in);

    // take integer input from the user
    System.out.println("Enter a number");
    int number = input.nextInt();

    // while loop continues
    // until entered number is positive
    while (number >= 0) {
      // add only positive numbers
      sum += number;

      System.out.println("Enter a number");
      number = input.nextInt();
    }

    System.out.println("Sum = " + sum);
    input.close();
  }
}
```

Java while loop

- In the above program, we have used the Scanner class to take input from the user. Here, **nextInt() takes integer input from the user**.
- **The while loop continues until the user enters a negative number**. During each iteration, **the number entered by the user is added to the sum variable.**
- When the **user enters a negative number, the loop terminates**. Finally, the total sum is displayed.

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,

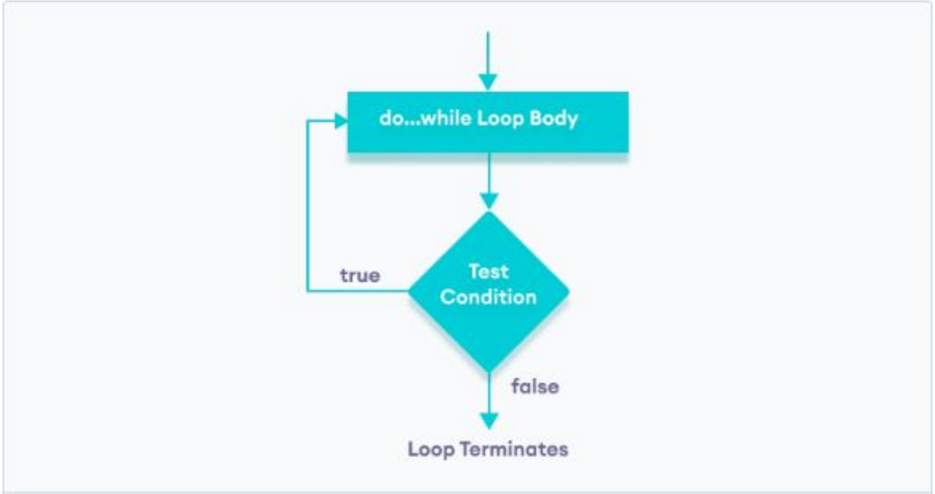
```
do {
    // body of loop
} while(textExpression);
```

- 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 do...while loop



Flowchart of Java do while loop

Java do...while loop

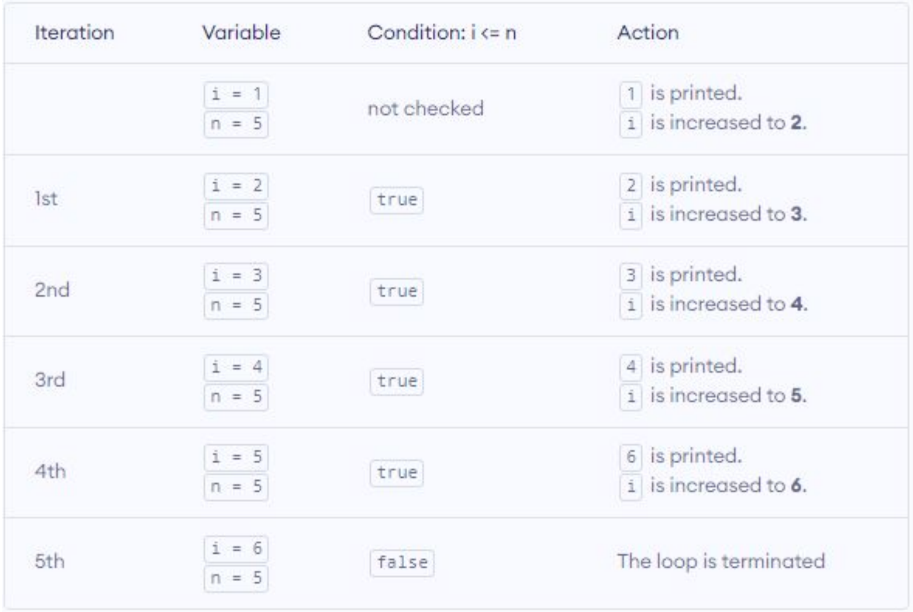
- Example 3: Display Numbers from 1 to 5
- Output

```
1
2
3
4
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);
  }
}
```

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

TIA-Academy

Java do...while loop

- Example 4: Sum of Positive Numbers
- Output

```
Enter a number
25
Enter a number
9
Enter a number
5
Enter a number
-3
Sum = 39
```

```
// Java program to find the sum of positive numbers
import java.util.Scanner;

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

    int sum = 0;
    int number = 0;

    // create an object of Scanner class
    Scanner input = new Scanner(System.in);

    // do...while loop continues
    // until entered number is positive
    do {
      // add only positive numbers
      sum += number;
      System.out.println("Enter a number");
      number = input.nextInt();
    } while(number >= 0);

    System.out.println("Sum = " + sum);
    input.close();
  }
}
```

TIA-Academy

Java do...while loop

- Here, the user enters a positive number, that number is added to the sum variable. And this process continues until the number is negative.
- When the number is negative, the loop terminates and displays the sum without adding the negative number.

TIA-Academy

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.

TIA-Academy

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
}
```

TIA-Academy

ASSIGNMENT 04

TIA-Academy

ASSIGNMENT 05 (HOME ASSIGNMENT)

TIA-Academy

Thank You