Java Document

By

Pujitha

Swaroopa

Prithviraj

Haritha

Introduction of Java:

Java is a Class-based, Object Oriented, General Purpose Computer programming language.

Created By: James Gosling and the team at Sun Microsystems.
First Released: May 23, 1995.
Current Owner: Oracle Corporation (acquired Sun Microsystems in 2010).

Features of Java:

1. Simple
2. Object-oriented
3. Distributed
4. Robust
5. Secure
6. Dynamic
7. Platform Independent
8. Portable
9. Interpreted
10. High Performance
11. Architecture Neutral
12. Multithreaded

Basic Syntax of Java:

```
public class HelloWorld
{
   public static void main(String[] args)
{
      System.out.println("Hello, World!");
   }
}
```

Components of the Java Platform:

- **Java Development Kit (JDK) :** It is for development purpose
- **Java Runtime Environment (JRE)** : It is for running the java programs
- **Java Virtual Machine (JVM):** JDK & JRE BOTH contains JVM so that we can run our java program

Java Program Compilation and Execution:

- **Compilation**: Java source code files (.java) are compiled by the Java compiler (javac) into bytecode files (.class)
  - javac HelloWorld.java
- **Execution**: The Java Virtual Machine (JVM) executes the bytecode using the Java interpreter (java)
  - java HelloWorld

Java Comments:

Comments are used to explain code and make it more readable. They are ignored by the compiler.

1. **Single-Line Comments:**

```
// This is a single-line comment
```

2. **Multi-Line Comments:**

```
/* This is a
  multi-line comment */
```

3. **Documentation Comments:**

```
/**
  This is a documentation comment
  used for generating Javadoc
 */
```

Literals, Keywords and Variables:

- **Literals:** Nothing but identifierS
  There are 3 types:
  1. Numeric Literals
  2. Character Literals
  3. Boolean Literals
- **Keywords:** These are special/reserved words

  There are 50 reserved words in java

- **Variables:** It is an identifier, behave like container.

  These are used to store input/output of our java program

Variable consists of declaration and initialization parts,

Declaration: To allocate sufficient memory to the variable

Ex: int age;

initialization: To store some value in that particular memory

Ex: int age;

age = 25;

Data Types:

Refer to the different sizes and values that can be stored in a variable

There are three types of datatypes,

1. **Primitive data types:** Can store only one value in a variable
2. **Derived data types:** Can store more than one value of similar type
3. **User defined data types:** Can store more than one value of similar/dissimilar type

Java Expressions:

An expression is a combination of variables, operators, and values that evaluates to a single value.

Example:
int sum = 5 + 3; // Expression evaluating to 8

Expression Evaluation:

Expressions is a combination of operands and operators is mainly depends on priority and associativity.

Example:
int result = 5 + 3 * 2;

Priority and Associativity:

- **Priority:** This represents the evaluation of expression starts from " what " operator
- **Associativity:** It represents which operator should be evaluated first if an expression is containing more than one operator with same priority

Types of Operators and Examples

It is a special symbol or character that tells the compiler to perform specific mathematical or logical Operation.

Operators help you manipulate data and control the logic of your program

These operators are the following types,

1. **Arithmetic operators:** used to perform common mathematical operations
2. **Assignment operators:** used to assign values to variables
3. **Comparision operators:** used to compare two values
4. **Logical operators**: used to determine the logic between values
5. **Bitwise operators:** that performs a specified operation on standalone bits

Java conditional / Control statements with example

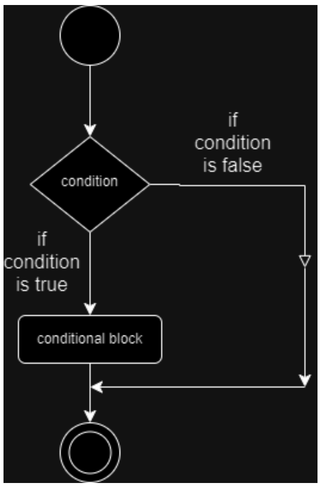
In java, conditional/control statements are used to control the flow of a program based on certain conditions

There are different types of conditional statements in java:

1. **If statement**
2. **Else statement**
3. **If-else statement**
4. **Switch statement**
5. **Nested if-else**
6. **Else-if ladder**

❖ **If statement:** This executes a block of code if its condition evaluates to true

Control-flow diagram:



Syntax:

```
if(condition/expression)

{

 statement-1;

statement-2;

. .

 statement-n;

}
```

Example:

```
int number = 10;

 if (number > 0)

{

 System.out.println("The number is positive.");

}
```

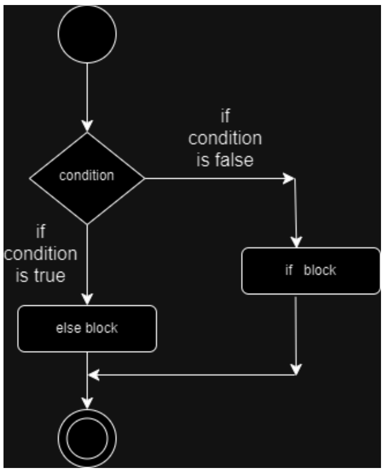
❖ **<u>Else statement</u>:** It is a keyword, by using this keyword we can create an alternative block for "if" part.

Using else is always optional i.e, it is recommended to use when we are having alternate block of condition (many conditions)

❖ **<u>If-else statement</u>:** The if-else statement is used to execute both the true part and the false part of a given condition.

If the condition is true, then if block code is executed and if the condition is false, then else block code is executed

Control-flow diagram:



Syntax:

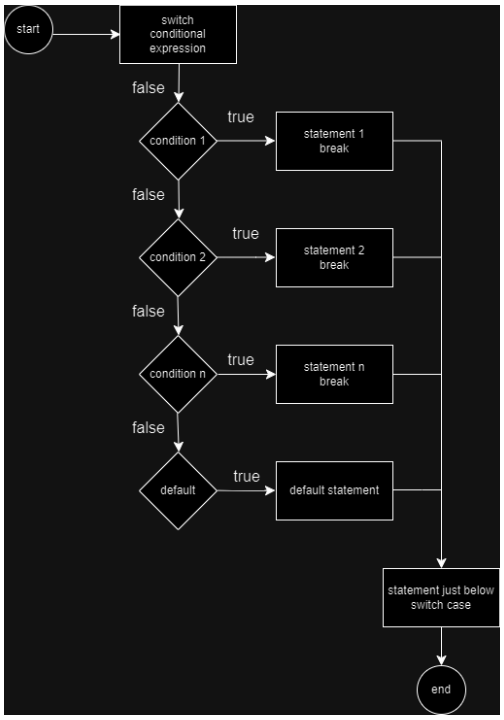
```
If(condition){
      Statements; }
Else{
Statements; }
```

Example:

```
int number = -10;
 if (number > 0) {
 System.out.println("The number is positive."); }
Else {
 System.out.println("The number is not positive."); }
```

❖ **<u>Switch statement:</u>** It is used to execute the code from multiple conditions or cases

Control-flow diagram:



Syntax:

```
Switch (exp)
{
Case1: val1:statement1;
          break ;

Case2:val2:statement2;
          break;

Case n: val n:statement n;
          break;

Default:default statement
}
```

Example:

```
public class SwitchCalculator {
    public static void main(String[] args) {
        double num1 = 10.5, num2 = 5.5;
        char operator = '+';
        switch (operator) {
            case '+':
                System.out.println(num1 + num2);
                break;
            case '-':
                System.out.println(num1 - num2);
                break;
            case '*':
                System.out.println (num1 * num2);
                break;
            case '/':
                if (num2 != 0) {
                    System.out.println(num1 / num2);
                } else {
                    System.out.println("Division by zero is not allowed.");
                }
                break;
            default:
                System.out.println("Invalid operator");
                break;
}}}
```

❖ **<u>Nested if-else</u>:** A nested if in java is an if-else statement that is the target of another if-statement means an if-else statement inside another if statement

Syntax:

```
if (condition1) {
        // Executes when condition1 is true
   if (condition2) {
        // Executes when condition2 is true
 }
 }
```

Example:

```
public class NestedIfExample {
    public static void main(String[] args) {
        int number = 10; // You can change this value to test different
cases

        if (number > 0) {
            System.out.println("The number is positive.");
            if (number % 2 == 0) {
                System.out.println("The number is even.");
            } else {
                System.out.println("The number is odd.");
            }
        }
    }
```

❖ Else-if ladder:

A user can decide among multiple options. The Java executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the Java else-if ladder is bypassed. If none of the conditions are true, then the final else statement will be executed.

Syntax:

```
if (condition)

statement;

 else if (condition)

 statement;

. . . .

 else statement;
```

Exmaple:

```
public class NestedIfExample {
    public static void main(String[] args) {
        int marks = 85; // You can change this value to test different cases
        boolean hasExtraCredit = true; // Change this to false to test different conditions

        if (marks >= 90) {
            System.out.println("Grade: A+");
        } else if (marks >= 80) {
            System.out.println("Grade: A");
            if (hasExtraCredit) {
                System.out.println("Congratulations! You've earned extra credit.");
            }
        } else if (marks >= 70) {
            System.out.println("Grade: B");
        } else if (marks >= 60) {
            System.out.println("Grade: C");
        } else if (marks >= 50) {
            System.out.println("Grade: D");
        } else {
            System.out.println("Grade: F");
        }
    }
}
```

<u>Java conditional / Control loops with example:</u>

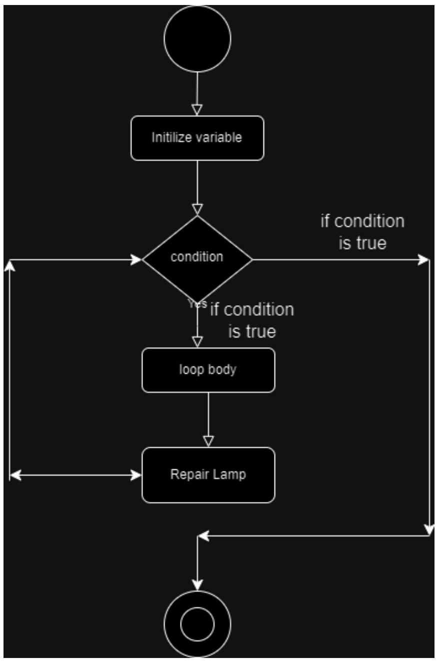
Looping statement are the statements execute one or more statements repeatedly several number of times

There are 3 types:

1. **For loop**
2. **While loop**
3. **Do-while loop**

❖ **<u>For loop:</u>** For loop is a statement which allows code to be repeatedly executed. Used to iterate the statements or a part of the program several times.

Control-flow diagram:



Syntax:

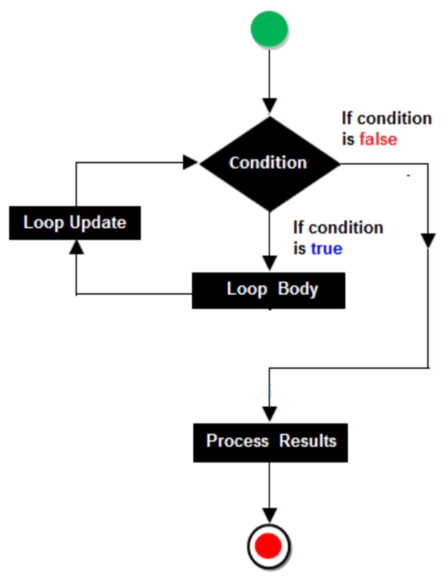
```
for(initialization; condition; increment/decrement) {

statements;}
```

Example:

```
For (int i = 1; i <= 5; i++)

{

System. out. println ("i “); }
```

❖ <u>**While loop:**</u> When we do not know about how many times loops are perform or iteration of loop is unknown.

Control-flow diagram:



Syntax:

```
initialization;

 while(condition) {

 print statement;

increment/decrement statement; }
```

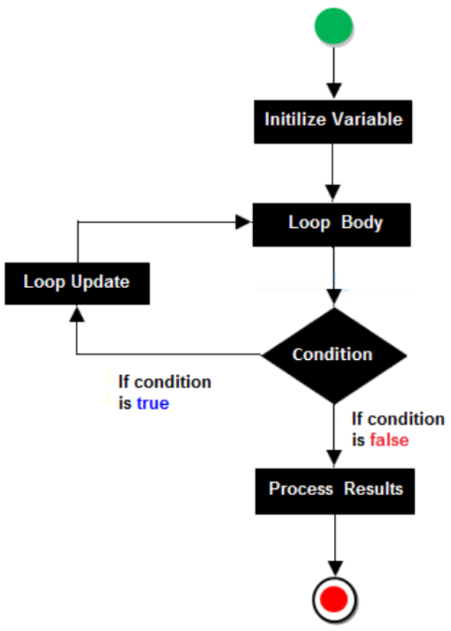
Example:

```
public class WhileLoopExample {

    public static void main(String[] args) {

        int i = 1; // Initialization

        while (i <= 10) { // Condition

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

            i++; // Increment

        }}}
```

❖ <u>Do-While loop:</u>

A do-while loop is similar to a while Loop in Java, except that a do-while loop is execute at least one time

Control-flow diagram:



Syntax:

```
initialization;

do {

print statement;

increment/decrement statement; }

while(condition);
```

Example:

```
public class DoWhileLoopExample {
    public static void main(String[] args) {
        int i = 1; // Initialization
 do {
            System.out.println("Number: " + i);
            i++; // Increment
        } while (i <= 10); // Condition
    }
}
```

<u>Jump statements and examples:</u>

There are 2 types:

1. Break statement
2. Continue statement

Break Statement:

- Break statement are used for terminates any type of loop e.g, while loop, do while loop or for loop.
- The break statement terminates the loop body immediately and passes control to the next statement after the loop.
- In case of inner loops, it terminates the control of inner loop only.
- **Use of break statement:** Break statement are mainly used with loop and switch statement. often use the break statement with the if statement

Continue Statement:

Continue statement are used to skip the rest of the current iteration in a loop and returns to the top of the loop. The continue statement works like a shortcut to the end of the loop body