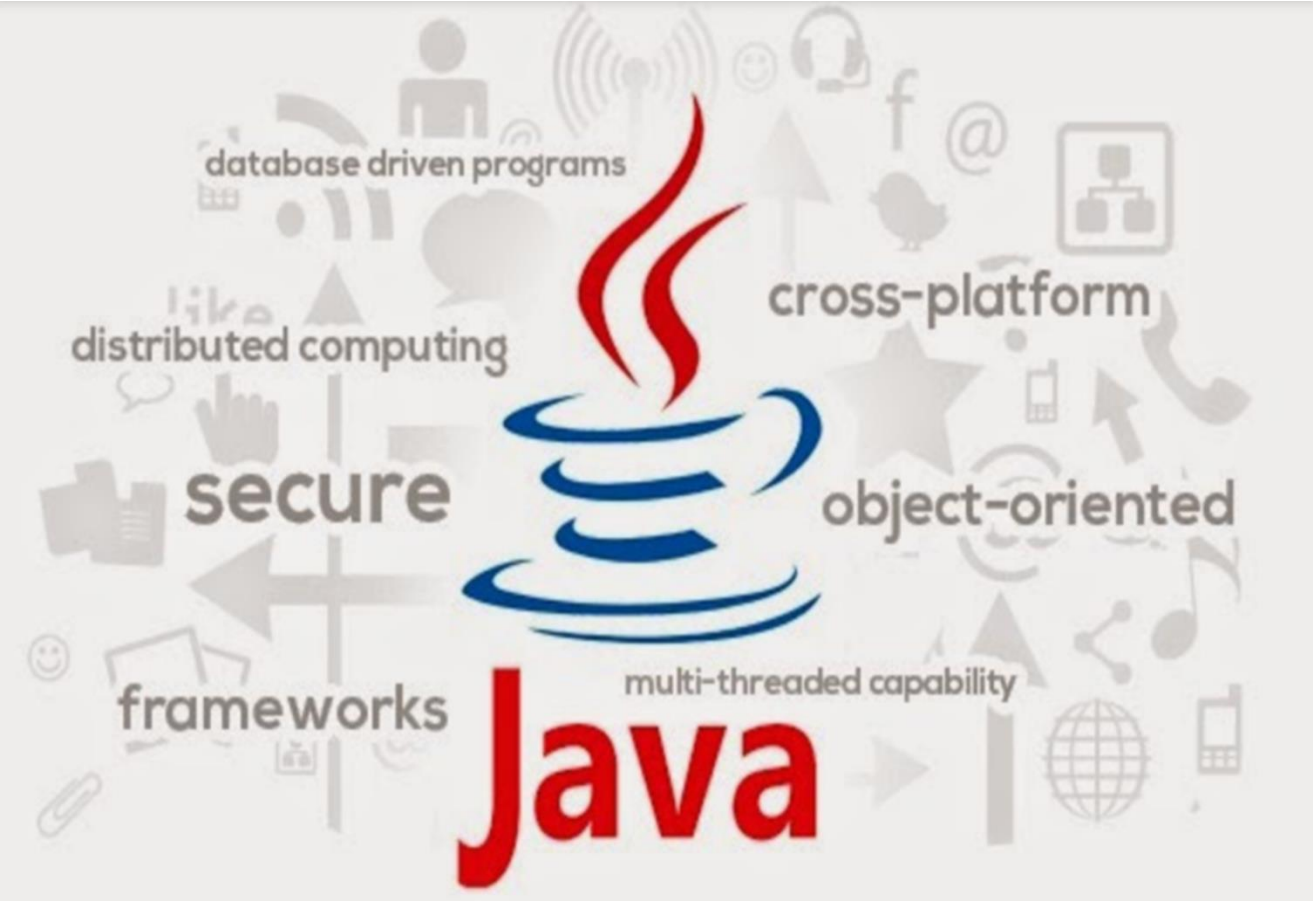
JAVA (CORE AND ADVANCED)



History of Java

- **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
- Java team members (also known as **Green Team**), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc.

Core JavaTopics

1. Basics about java.

2. Control Statements.

3. OOPS concepts (Data Abstraction, Encapsulation, Inheritance, Polymorphism)

4. String handling.

5. Multithreading.

6. Exception handling.

7. Java array.

8. Java conversion.

9. Java Date

Advanced Java Topics

1. Collection framework.

2. Java AWT.

3. Java swing.

4. Java applet.

5. Java JDBC.

6. Java FX(2D, 3D Layouts).

Basics about Java

1. Java is a programming language and a platform.
2. Java is a high level, robust, secured and object-oriented programming language.
3. According to Sun, 3 billion devices run java. There are many devices where
4.Desktop Applications such as acrobat reader, media player, antivirus etc.
5. Web Applications such as irctc.co.in, javatpoint.com etc.
6. Enterprise Applications such as banking applications. General application like Mobile, Embedded System, Smart Card, Robotics, Games etc.

Features

1.Simple
2.Object-Oriented
3.Portable
4.Platform independent
5.Secured
6.Robust
7. Multithreaded
8.Dynamic
9.Interpreted
10.High Performance

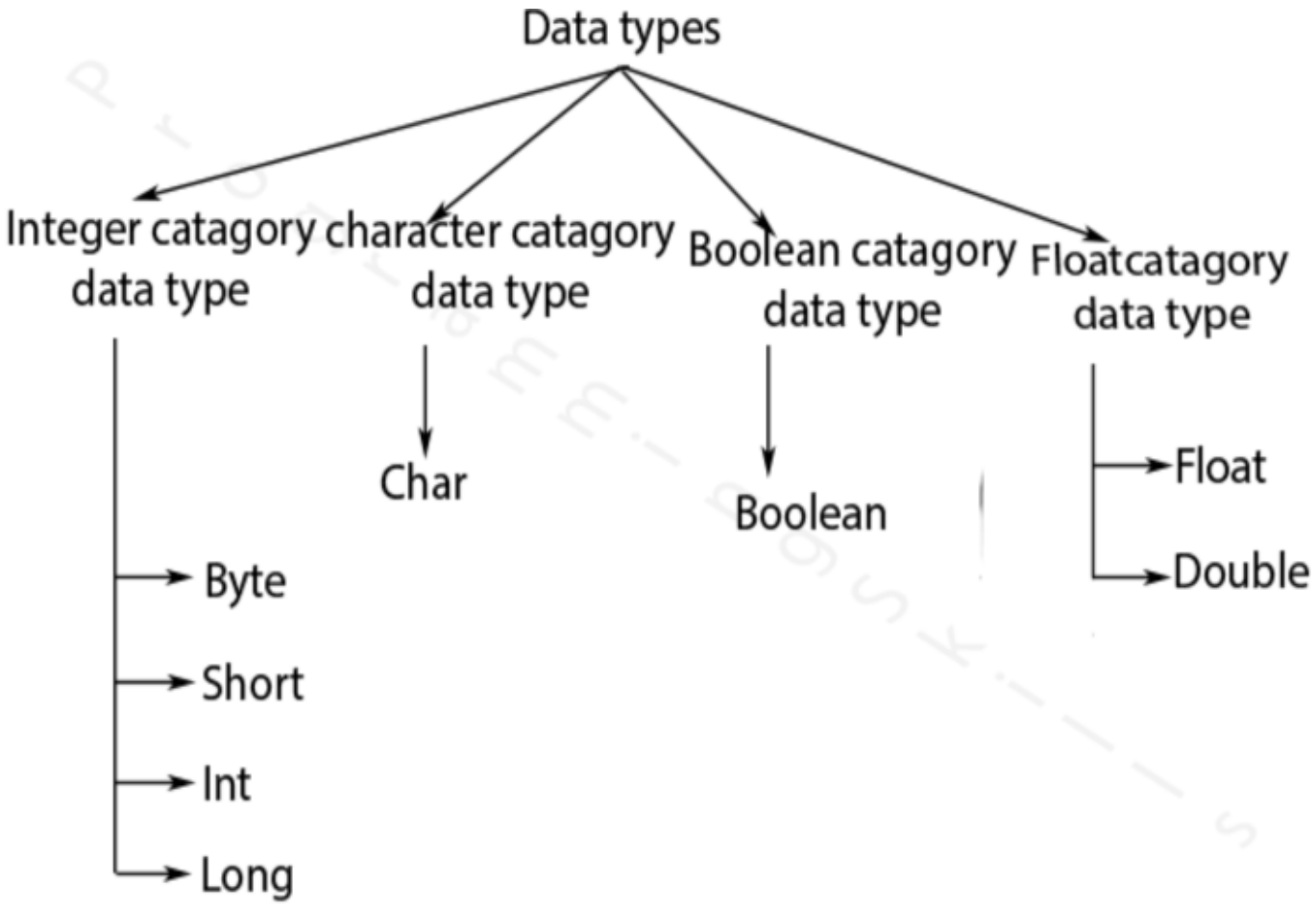
Basic Data types

There are two data types available in Java

1. Primitive Data Types
2. Reference/Object Data Types

Primitive Data Types

- There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a keyword.



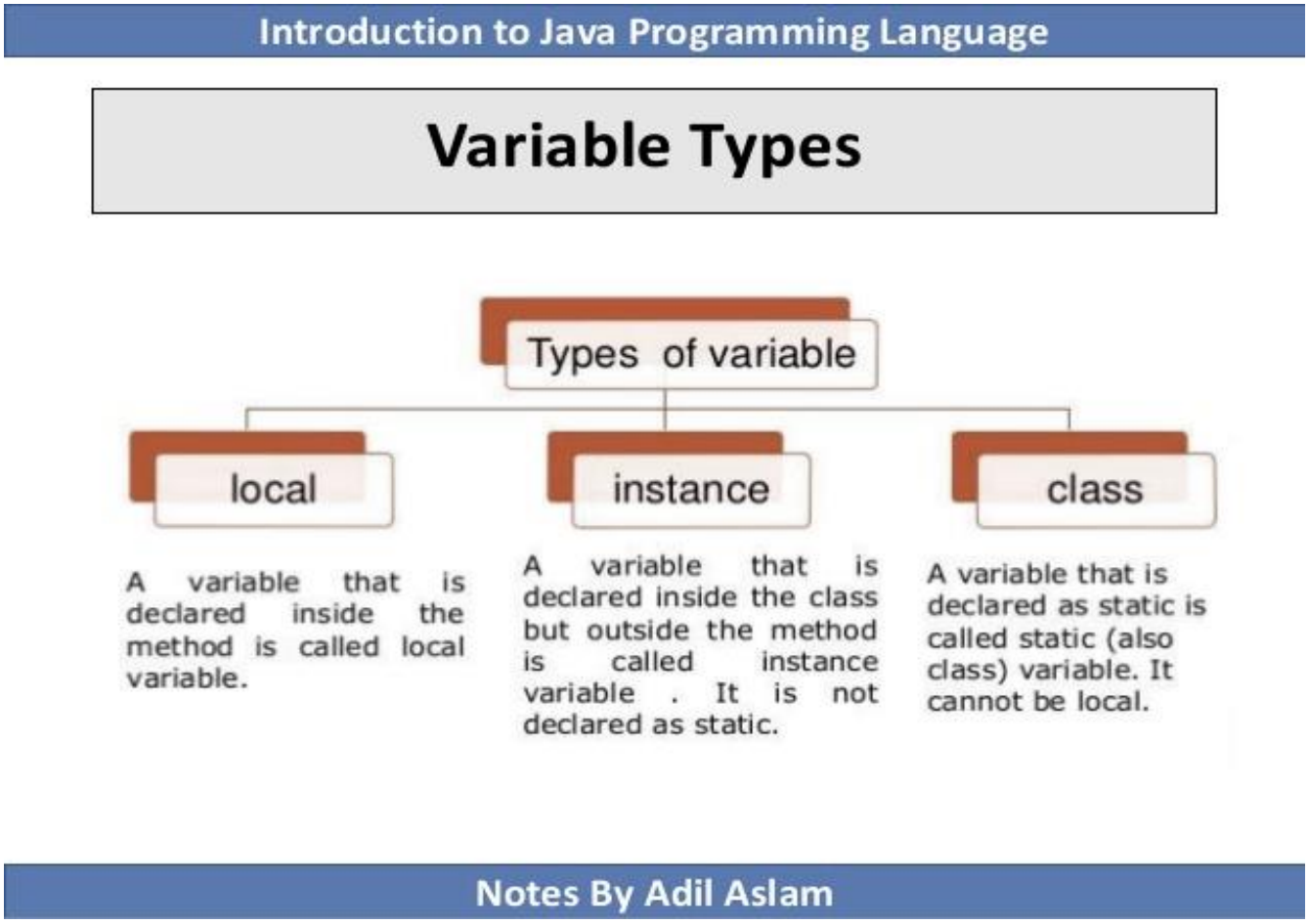
Reference Data types

- Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed.
- For example, Employee, Puppy, etc.

Variable Types

There are three kinds of variables in Java,

- Local variables (Local variables are declared in methods, constructors, or blocks).
- Instance variables (Instance variables are declared in a class, but outside a method, constructor or any block).
- Class/Static variables (Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block).



Modifier Types

- Java language has a wide variety of modifiers, including the following
  1. Java Access Modifiers
  2. Non Access Modifiers

Access Control Modifiers

- Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are

1. Visible to the package, the default. No modifiers are needed.
2. Visible to the class only (private).
3. Visible to the world (public).
4. Visible to the package and all subclasses (protected).

Most Restrictive ⟷ Least Restrictive

Access Modifiers ->	private	Default/no-access	protected	public
Inside class	Y	Y	Y	Y
Same Package Class	N	Y	Y	Y
Same Package Sub-Class	N	Y	Y	Y
Other Package Class	N	N	N	Y
Other Package Sub-Class	N	N	Y	Y

Same rules apply for inner classes too, they are also treated as outer class properties

Basic Operators

We can divide all the Java operators into the following groups ,

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Misc Operators

Arithmetic Operators

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	A + B will give 30
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	A - B will give -10
* (Multiplication)	Multiplies values on either side of the operator.	A * B will give 200
/ (Division)	Divides left-hand operand by right-hand operand.	B / A will give 2
% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.	B % A will give 0
++ (Increment)	Increases the value of operand by 1.	B++ gives 21
-- (Decrement)	Decreases the value of operand by 1.	B-- gives 19

Relational Operators

Operator	Description	Example
== (equal to)	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!= (not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<= (less than or equal to)	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

Operator	Description	Example
&& (logical and)	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false
|| (logical or)	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true
! (logical not)	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C – A
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2

Miscellaneous Operators

- Conditional operator is also known as the ternary operator. The goal of the operator is to decide, which value should be assigned to the variable.

variable x = (expression) ? value if true : value if false

Example

```
public class Test {

  public static void main(String args[]) {
    int a, b;
    a = 10;
    b = (a == 1) ? 20: 30;
    System.out.println( "Value of b is : " +  b );

    b = (a == 10) ? 20: 30;
    System.out.println( "Value of b is : " + b );
  }
}
```

This will produce the following result

Output

```
Value of b is : 30
Value of b is : 20
```

Control Statements

If - else Statement

The Java if statement tests the condition.
It executes the if block if condition is true.

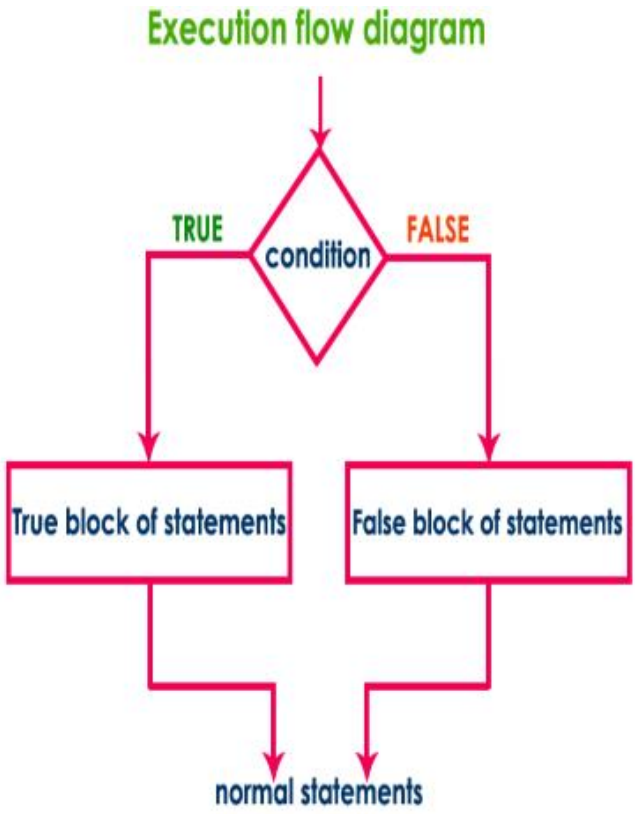
```
public class Ifjava {
public static void main(String[] args) {
   int age=20;
```

```
    if(age>18){
        System.out.print("Age is greater than 18");
    }
}
}
```

Output:
Age is greater than 18

Syntax

```
if ( condition )
{
   ....
   True block of statements;
   ....
}
else
{
   ....
   False block of statements;
   ....
}
```



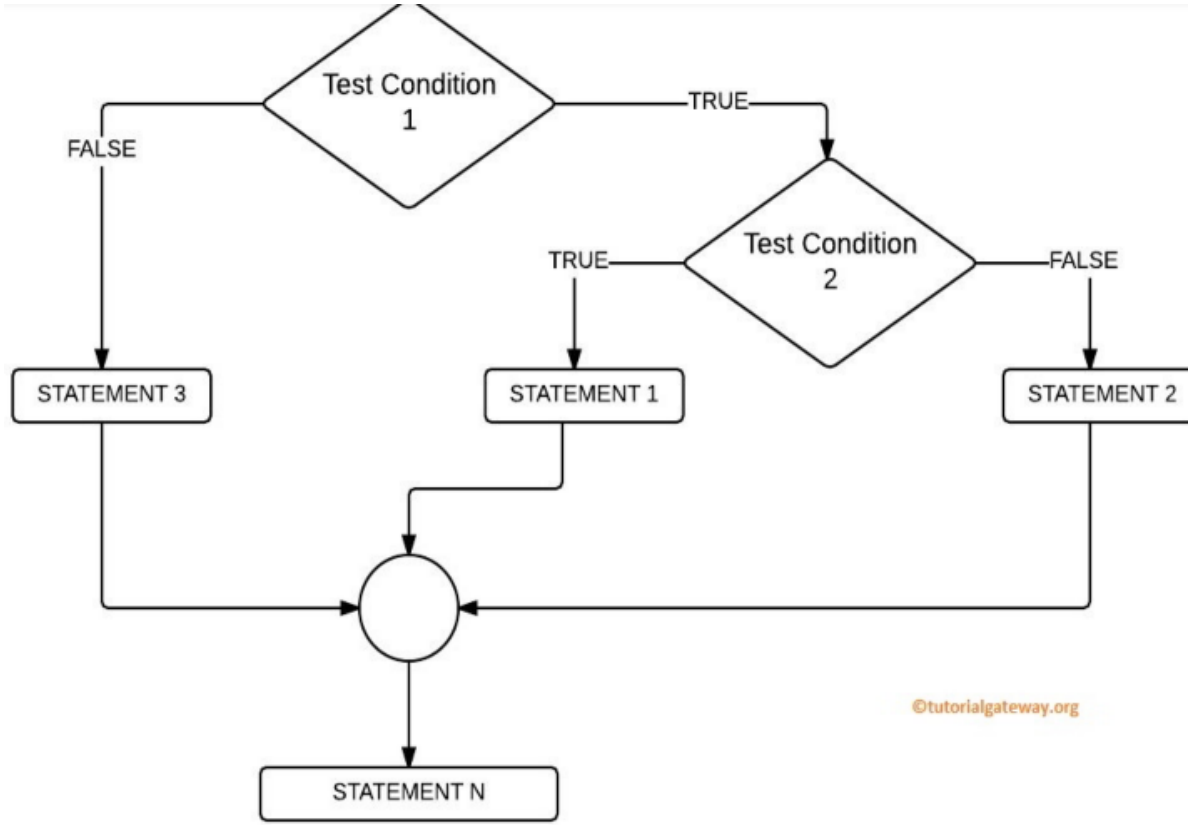
Nested if

It is always legal to nest if-else statements which means you can use one if or else if statement inside another if or else if statement.

```
public class Test {
   public static void main(String args[]) {
      int x = 30;
      int y = 10;
      if( x == 30 ) {
         if( y == 10 ) {
            System.out.print("X = 30 and Y = 10");
         }
      }
   }
}
```

Output:

X = 30 and Y = 10

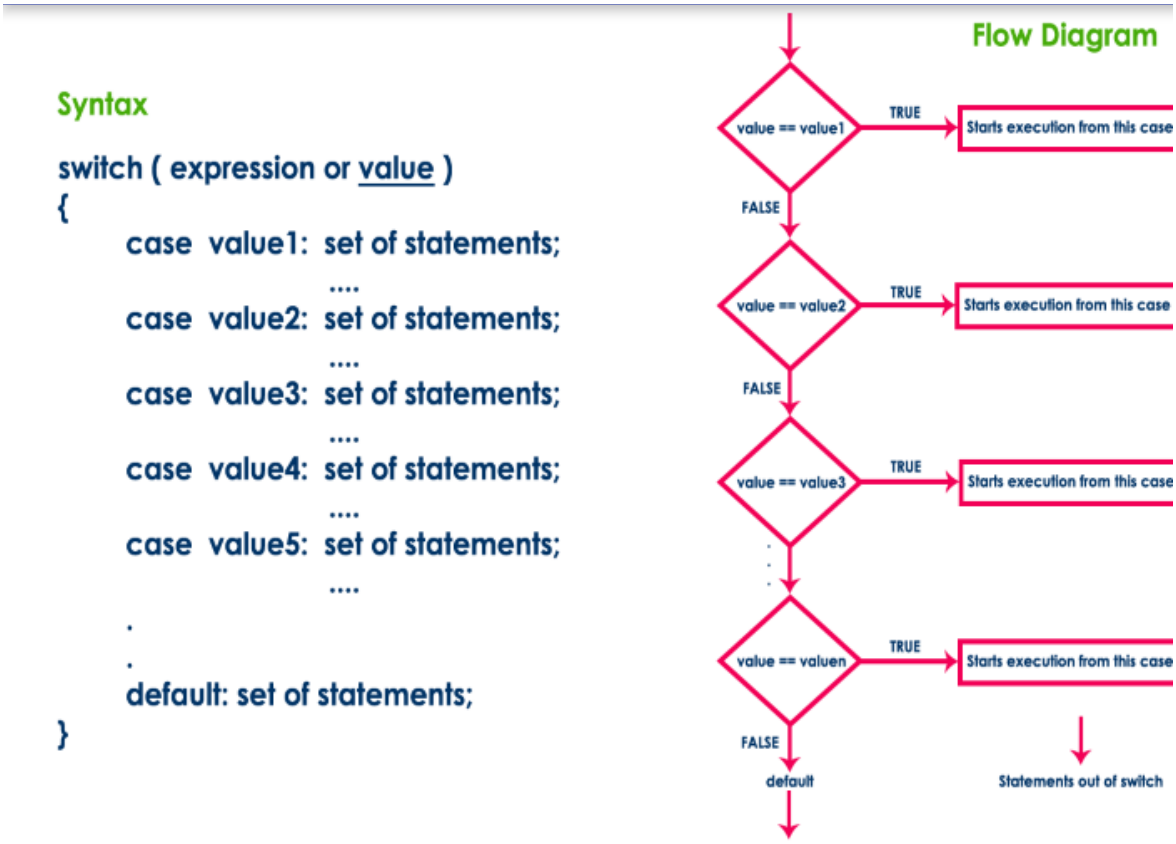


Switch Statement

The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement.

```
public class Switchjava {
public static void main(String[] args) {
    int number=20;
    switch(number){
    case 10: System.out.println("10");break;
    case 20: System.out.println("20");break;
    case 30: System.out.println("30");break;
    default:System.out.println("Not in 10, 20 or 30");
    }
}
}
```

Output:
20



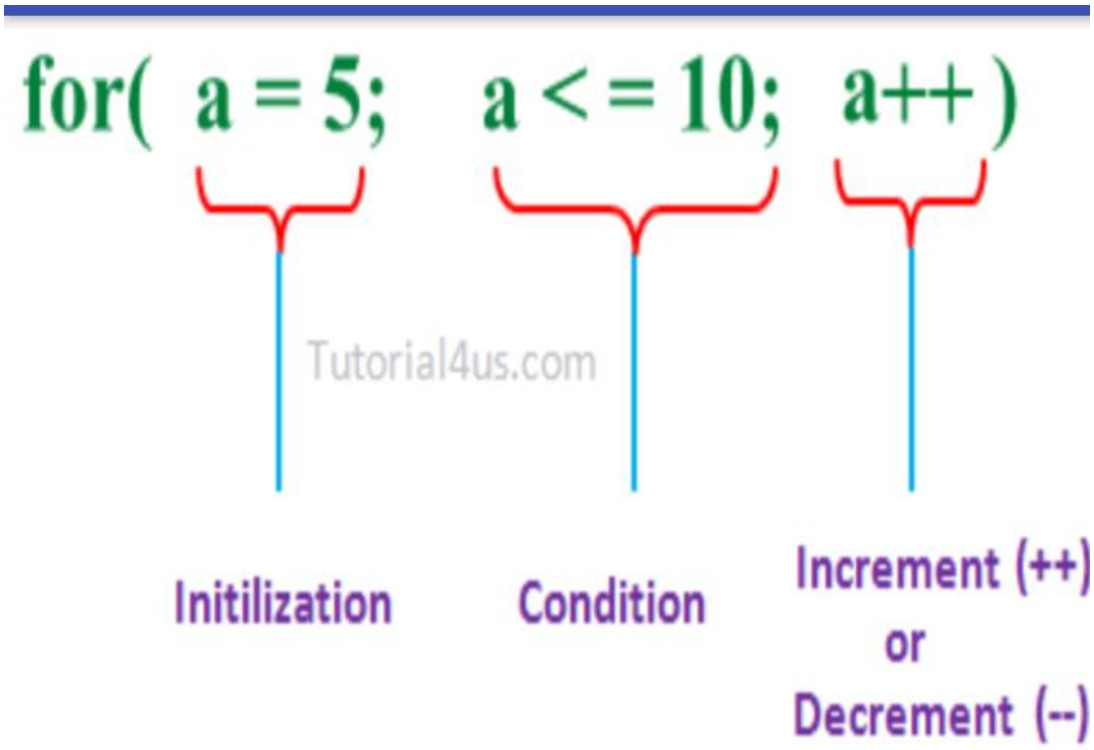
For Loop

In computer science, a for-loop (or simply for loop) is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

```
public class Forjava {
public static void main(String[] args) {
    for(int i=1;i<=10;i++){
        System.out.println(i);
    }
}
}
```

Output:

1 2 … 10



While Loop

```
public class Whilejava {
public static void main(String[] args) {
    int i=1;
    while(i<=10){
        System.out.println(i);
    i++;
    }
}
}
```

Output:
1 to 10

Do While Loop

```
public class DoWhilejava {
public static void main(String[] args) {
    int i=1;
    do{
        System.out.println(i);
    i++;
    }while(i<=10);
}
}
```

Output:
1 to 10

Continue Statement

```
public class Continuejava {
public static void main(String[] args) {
   for(int i=1;i<=10;i++){
      if(i==5){
         continue;
      }
      System.out.println(i);
   }
}
}
```

Output:
1 to 10

Break Statement

```
public class Breakjava {
public static void main(String[] args) {
   for(int i=1;i<=10;i++){
      if(i==5){
         break;
      }
      System.out.println(i);
   }
}
}
```

Output:
1
2
3
4