JAVA TUTORIAL

## Variable:

**What is Variable?**

A Java variable is a piece of memory that can contain a data value. A variable thus has a data type. Data types are covered in more detail in the text on Java data types. Variables are typically used to store information which your Java program needs to do its job.

* **Local variable**
* **Instance variable**
* **Class/Static variable**

Let me discuss about each one of them:

**Local variable**: These are the variables which are declared within the method of a class. Let’s understand this with a programmatic example

**Instance variable**: Instance variable is declared in a class but outside a method, constructor or any block. Let’s understand this with a programmatic example.

**Class variable**: Class variables are also called as static variables. These variables have only one copy that is shared by all the different objects in a class. Let’s understand this with a programmatic example.

## Data Types:

· byte (number, 1 byte)

· short (number, 2 bytes)

· int (number, 4 bytes)

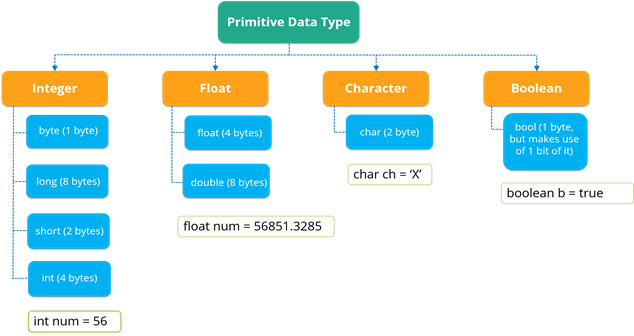
· long (number, 8 bytes)

· float (float number, 4 bytes)

· double (float number, 8 bytes)

· char (a character, 2 bytes)

· Boolean (true or false, 1 byte)



## Methods:

**What is Method in Java?**

A method is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the method's name. Think of a method as a subprogram that acts on data and often returns a value. Each method has its own name.

|  |  |
| --- | --- |
| **Method** | **Description** |
| void add(int index, Object element) | It is used to insert element into the invoking list at the index passed in the index. |
| Boolean addAll(int index, Collection c) | It is used to insert all elements of c into the invoking list at the index passed in the index. |
| object get(int index) | It is used to return the object stored at the specified index within the invoking collection. |
| object set(int index, Object element) | It is used to assign element to the location specified by index within the invoking list. |
| object remove(int index) | It is used to remove the element at position index from the invoking list and return the deleted element. |
| ListIterator listIterator() | It is used to return an iterator to the start of the invoking List. |
| ListIterator listIterator(int index) | It is used to return an iterator to the invoking list that begins at the specified index. |

## **Data Operators:**

There are mainly 4 different types of operators, which are listed below:

·**Arithmetic Operator:**Perform arithmetic operations such as addition, subtraction, multiplication, division and modulus.

·**Unary Operator:** Unary operators are used to increment or decrement a particular value. For example: ++ stands for increment, – – stands for decrement.

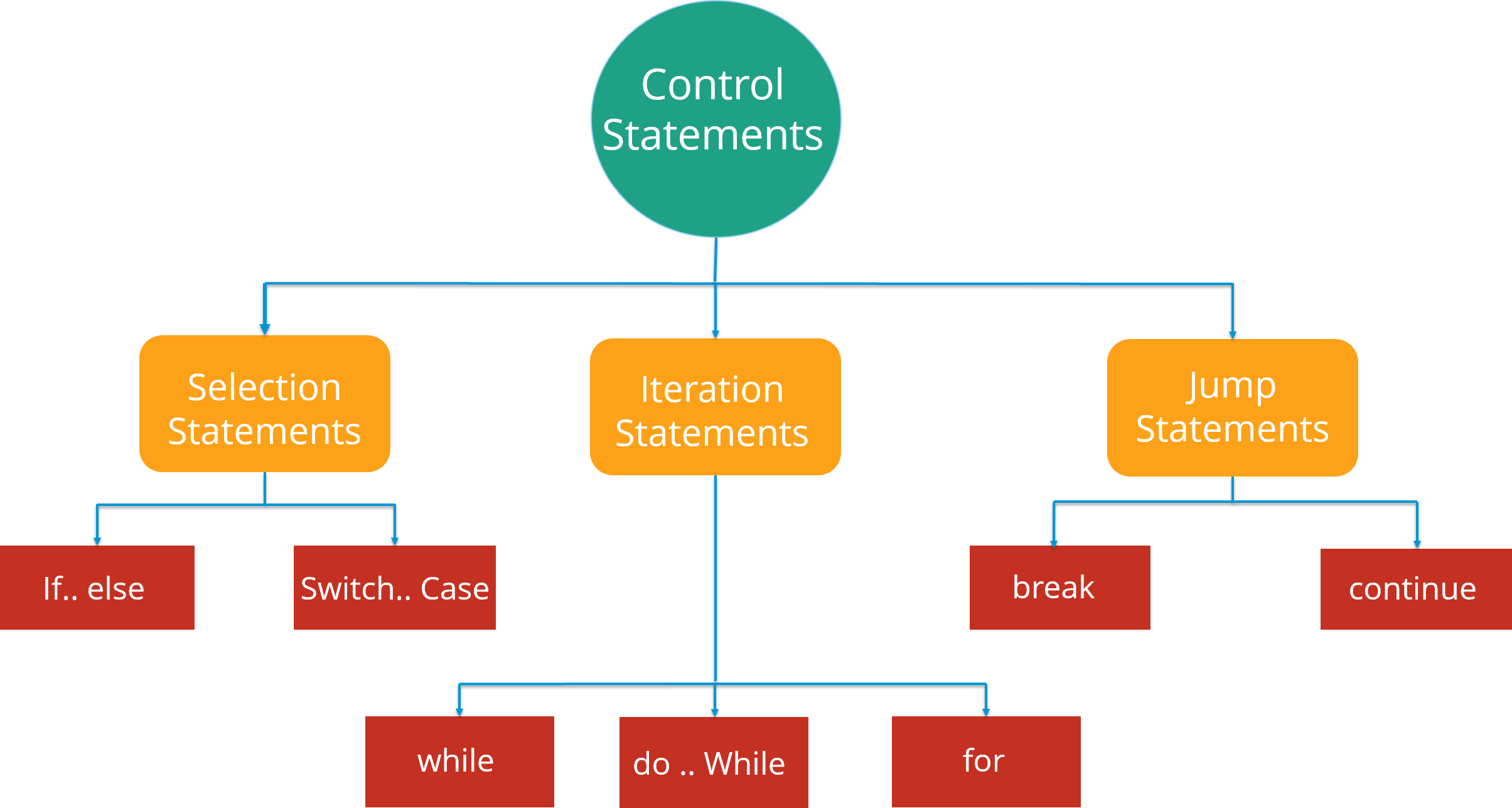
**Relational Operator:** It defines some kind of relation between two entities. For example: <, >, <=, >=, !=, ==.

· **Logical Operator:** Logical operators are typically used with Boolean (logical) values.

## **Control statements:**

Control statements are the statements that define the flow of your program.

* There are 3 types of control statements in Java: Selection, iteration and jump statements.



**Selection Statements:** Selection statements allow you to control the flow of the program during run time on the basis of the outcome of an expression or state of a variable.So here you are following a selection process from the various options available.

Now these statements can be further classified into the following:

* If-else Statements
* Switch Statements

**If-else Statement:**

**condition:**

1. **if**(condition){

2. //code if condition is true

3. }**else**{

4. //code if condition is false

5. }

**Example:**

public class Compare {

int a=10,

int b=5;

if(a>b)

{ // if condition

System.out.println(" A is greater than B");

}

else

{ // else condition

System.out.println(" B is greater");

}

}

**Switch statement:**

The switch statement defines multiple paths for execution of a set of statements. It is a better alternative than using a large set of if-else statements as it is a multi-way branch statement.

**Switch statement condition:**

1. **switch**(expression){

2. **case** value1:

3. //code to be executed;

4. **break**; //optional

5. **case** value2:

6. //code to be executed;

7. **break**; //optional

8. ......

9.

10. **default**:

11. code to be executed **if** all cases are not matched;

12. }

**Example:**

public class SwitchExample {

public static void main(String[] args) {

int number=2;

switch(number){

case 1: System.out.println("1");break;

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

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

default:System.out.println("Not in 1, 2 or 3");

}

}

}