Java Conditional Statements,Control Loops,Jump Statements and Examples

-K.L.Madhavi

Java Conditional/Control Statements

A programming language uses control statements to control the flow of execution of a program based on certain conditions.

Java's Conditional/Control Statements:

- **if**
- **if-else**
- **if-else-if**
- **Nested if**
- **Switch case**

1.if Statement

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

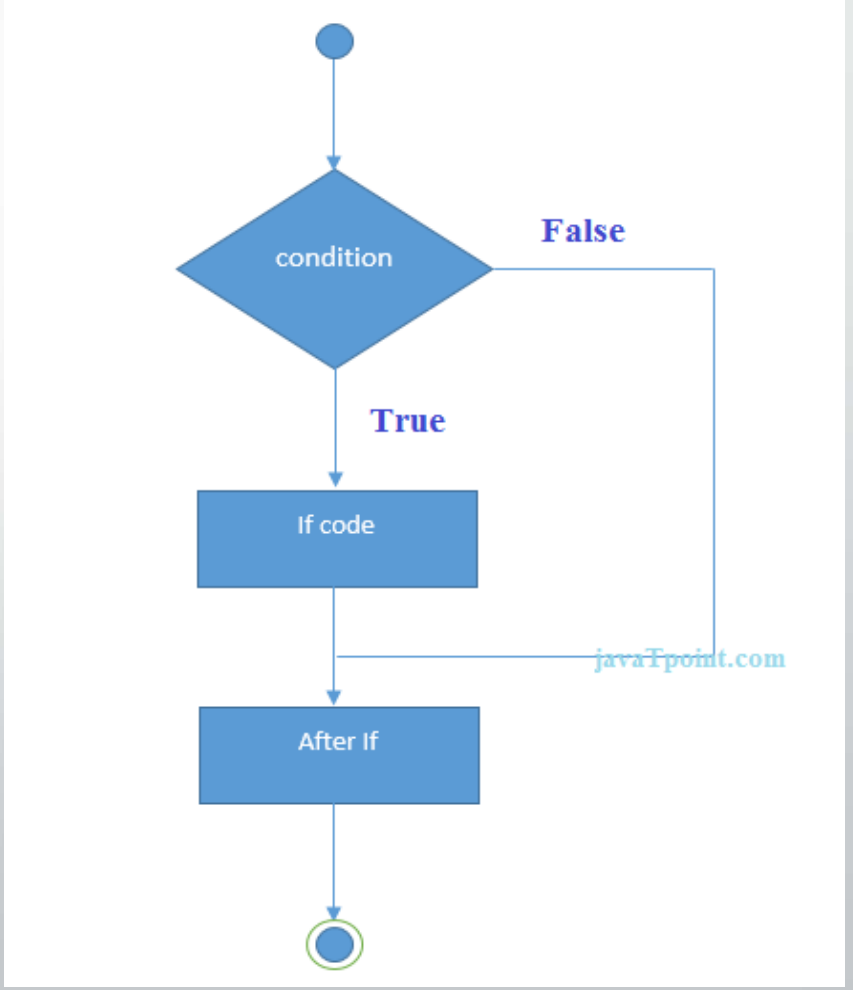
Syntax:if(condition){

//code to be executed

}

Example Program:

```
public class IfExample {
public static void main(String[] args) {
      int age=20;
      if(age>18){
       System.out.println("Age is greater than 18");
      }
      System.out.println(“out of if block");
   }
}
```



2.if-else Statement

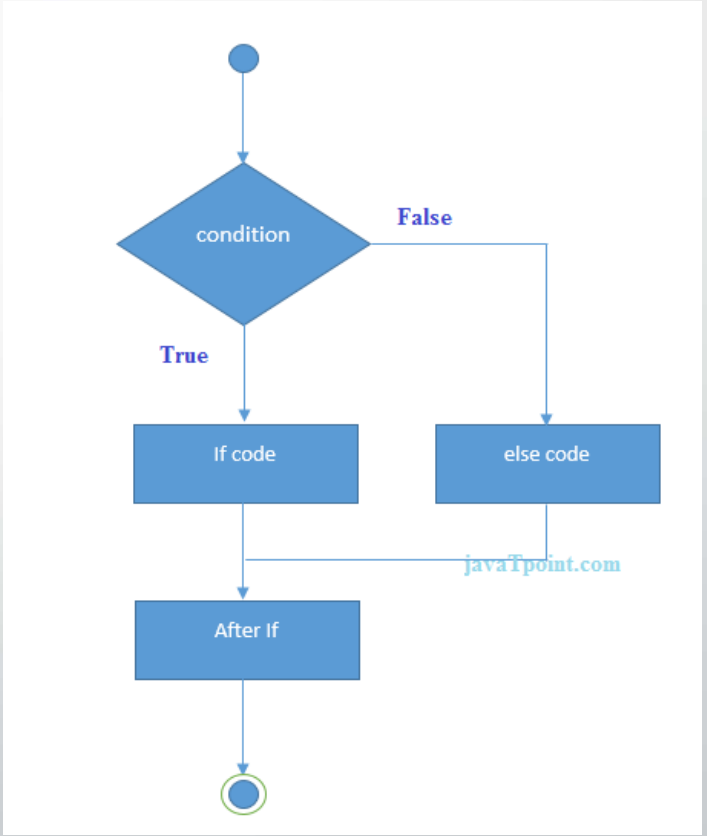
The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

Syntax:

```
if(condition){
//code if condition is true
}else{
//code if condition is false
}
```

Example Program:

```
public class IfExample {
public static void main(String[] args) {
        int age=13;
        if(age>18){
         System.out.println("Age is greater than 18");
        }
        else{
        System.out.println(“Age is not greater than 18"); }
    }
```

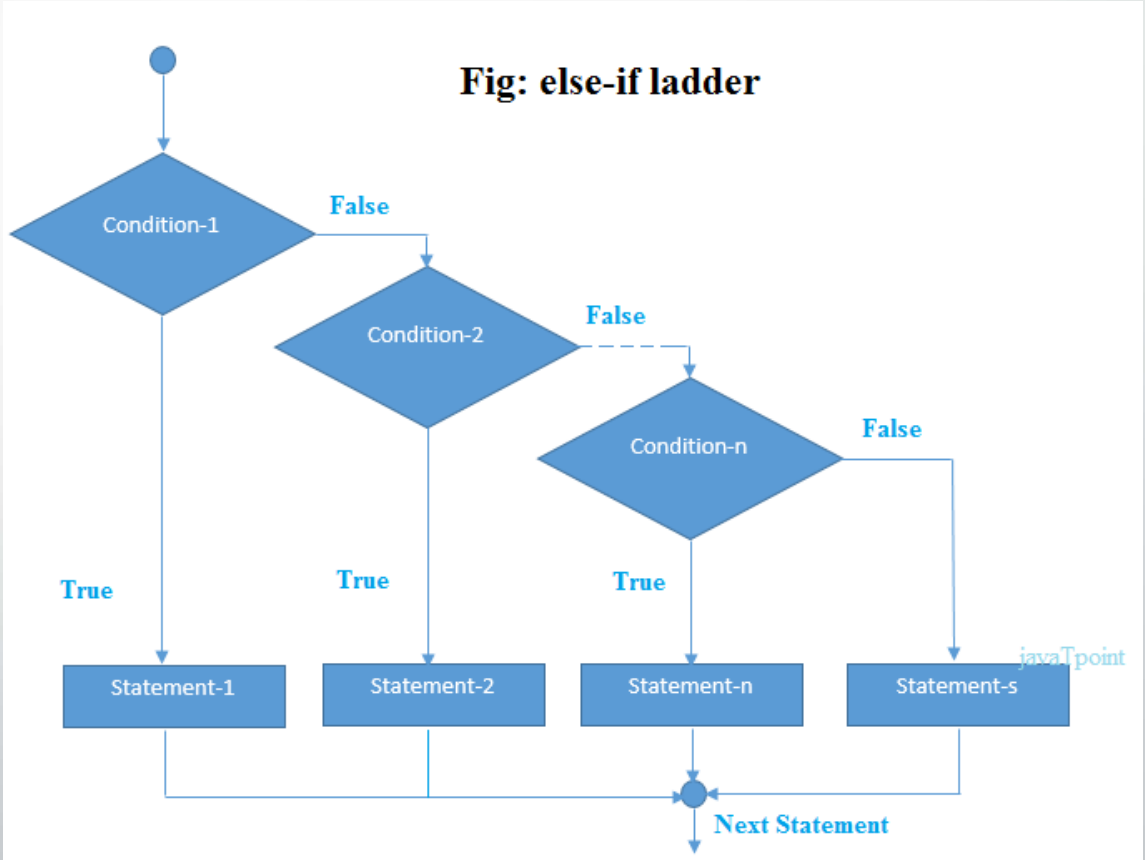


3.if-else-if Ladder

The if-else-if ladder statement executes one condition from multiple statements.

Syntax:

```
if(condition1){
//code to be executed if condition1 is true
}else if(condition2){
//code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
...
else{
//code to be executed if all the conditions are false
}
```



4.Nested if statement

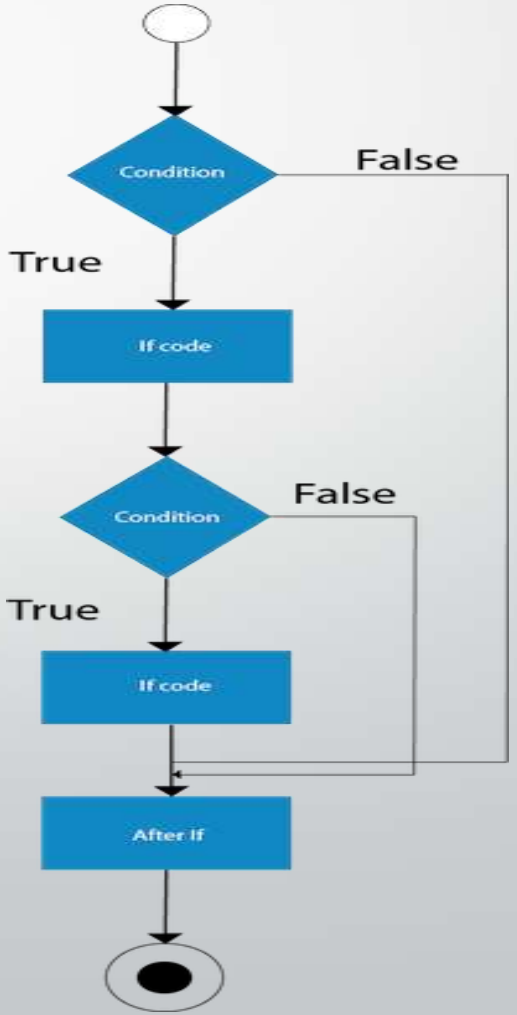
The nested if statement represents the *if block within another if block*. Here, the inner if block condition executes only when outer if block condition is true.

Syntax:

```
if(condition){
    //code to be executed
        if(condition){
            //code to be executed
    }
}
```

Example Program:

```
public class JavaNestedIfExample {
public static void main(String[] args) {
    int age=20;
    int weight=80;
     if(age>=18){
       if(weight>50){
           System.out.println("You are eligible to donate blood");
       }
    }
} }
```



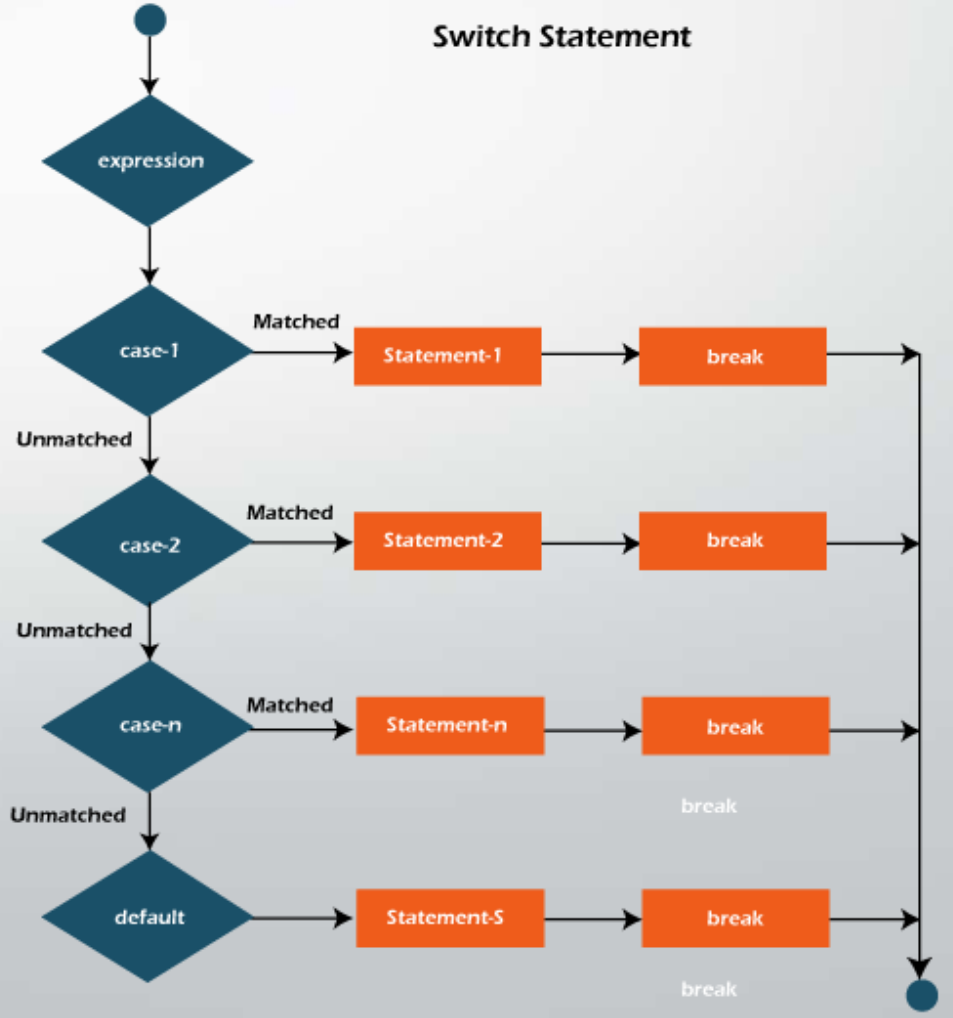
5.Switch-Case

The Java *switch statement* executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long and String types.

Syntax:

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
.....

default:
  code to be executed if all cases are not matched;
}
```



Example Program

```
public class SwitchExample {
public static void main(String[] args) {
    //Declaring a variable for switch expression
    int number=20;
    //Switch expression
    switch(number){
    //Case statements
    case 10: System.out.println("10");
    break;
    case 20: System.out.println("20");
    break;
    case 30: System.out.println("30");
    break;
    //Default case statement
    default:System.out.println("Not in 10, 20 or 30");
    }
}
}
```

Java Conditional/Control Loops

Loops in Java is a feature used to execute a particular part of the program repeatedly if a given condition evaluates to be true.

There are three types of loops in Java:

1.for Loop

2.While Loop

3.do-while Loop

1.for Loop

The Java *for loop* is used to iterate a part of the program several times. If the number of iteration is **fixed.**We can initialize the variable, check condition and increment/decrement value. It consists of four parts:

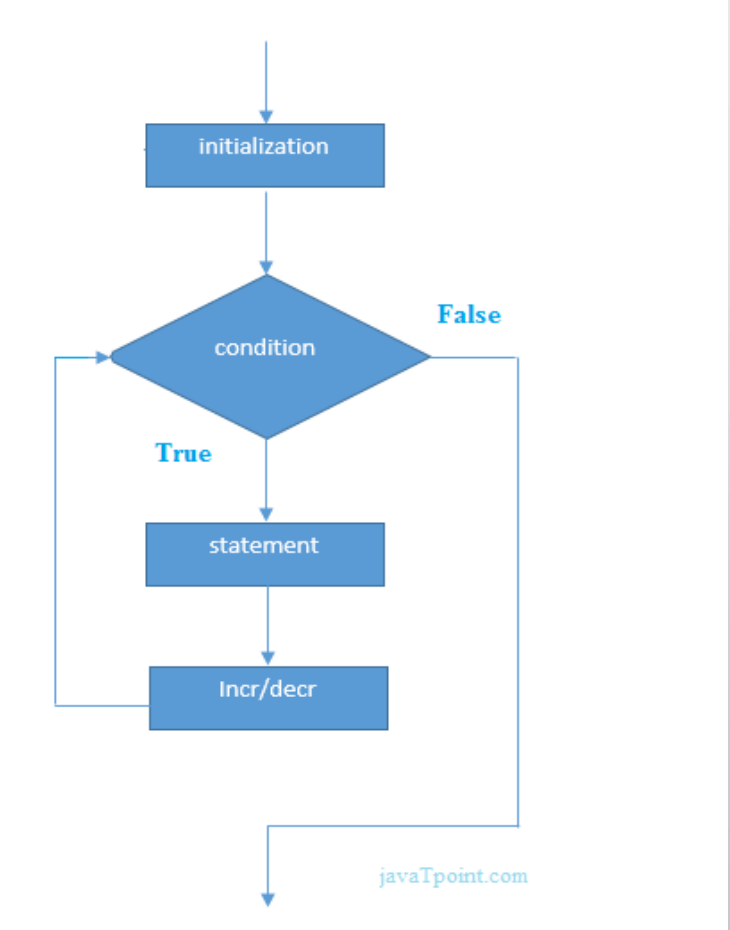
- **Initialization**
- **Condition**
- **Increment/Decrement**
- **Statement**

Syntax:

1. for(initialization; condition; increment/decrement){
2. //statement or code to be executed
3. }

Example Program:

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



2.While Loop

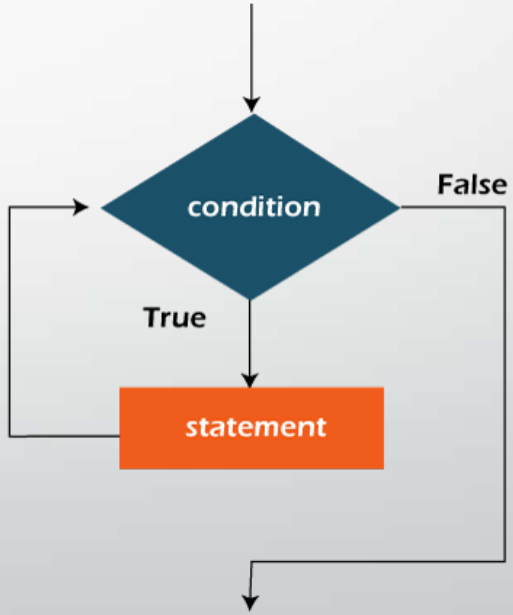
The Java *while loop* is used to iterate a part of the program repeatedly until the specified Boolean condition is true. As soon as the Boolean condition becomes false, the loop automatically stops.

Syntax:

```
while (condition){
//code to be executed
Increment / decrement statement
}
```

Example Program:

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



3.do-while Loop

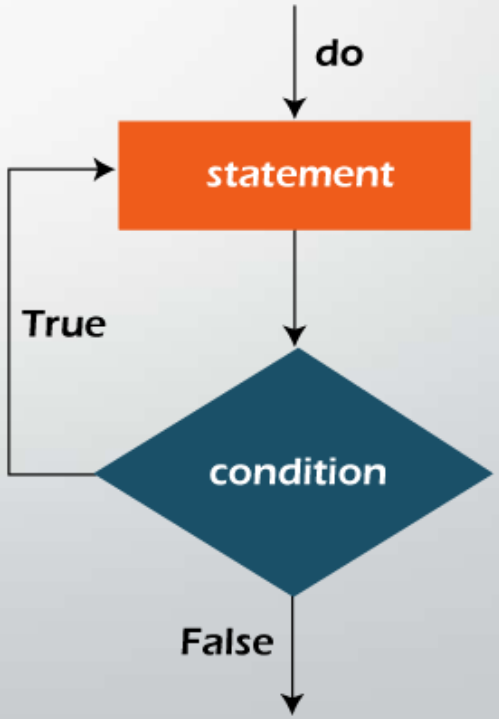
Java do-while loop is called an exit control loop. Therefore, unlike while loop and for loop, the do-while check the condition at the end of loop body. The Java *do-while loop* is executed at least once because condition is checked after loop body.

Syntax:

```
do{
//code to be executed / loop body
//update statement
}while (condition);
```

Example Program:

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



Java Jump Statements

1.Break Statement

When a break statement is encountered inside a loop, the loop is immediately terminated.

Syntax:

break;

Example Program:

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

2.Continue Statement

The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips code at the specified condition.

Syntax:

```
continue;
```

Example Program:

```
public class ContinueExample {
public static void main(String[] args) {
    for(int i=1;i<=10;i++){
        if(i==5){
            continue;//it will skip this statement
        }
        System.out.println(i);
    } } }
```

3.Return Statement

- Exits a method and returns control to the calling method.
- Can optionally return a value to the calling method.

Example Program:

```
public class ReturnStatement{
        public static void main(String args[]){
                int x=3,y=5;
                int sum=add(x,y);
                System.out.println(sum);
        }
        public static int add(int a,int b){
                    int sum=a+b;
                return sum;
         }
}
```