**CONTROLLED STATEMENTS**

**JAVA IF-ELSE STATEMENT**

The Java if statement is used to test the condition. It checks boolean condition: **true** or **false**. There are various types of if statement in Java.

1. if statement
2. if-else statement
3. if-else-if ladder
4. nested if statement

**1.Java 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:**

public class IfExample {

public static void main(String[] args) {

//defining an 'age' variable

int age=20;

//checking the age

if(age>18){

System.out.print("Age is greater than 18");

}

}

}

**2.Java if-else Statement**

if(condition){

//code if condition is true

}

else{

//code if condition is false

}

**Example:**

public class IfElseExample {

public static void main(String[] args) {

//defining a variable

int number=13;

//Check if the number is divisible by 2 or not

if(number%2==0){

System.out.println("even number");

}else{

System.out.println("odd number");

}

}

}

**Leap Year Example:**

A year is leap, if it is divisible by 4 and 400. But, not by 100.

public class LeapYearExample {

public static void main(String[] args) {

int year=2020;

if(((year % 4 ==0) && (year % 100 !=0)) || (year % 400==0)){

System.out.println("LEAP YEAR");

}

else{

System.out.println("COMMON YEAR");

}

}

}

**Using Ternary Operator**

We can also use ternary operator (? :) to perform the task of if...else statement. It is a shorthand way to check the condition. If the condition is true, the result of ? is returned. But, if the condition is false, the result of : is returned.

**Example:**

public class IfElseTernaryExample {

public static void main(String[] args) {

int number=13;

//Using ternary operator

String output=(number%2==0)?"even number":"odd number";

System.out.println(output);

}

}

**3.Java if-else-if ladder Statement**

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

}

**Example:**

public class IfElseIfExample {

public static void main(String[] args) {

int marks=65;

if(marks<50){

System.out.println("fail");

}

else if(marks>=50 && marks<60){

System.out.println("D grade"); }

else if(marks>=60 && marks<70){

System.out.println("C grade");

}

else if(marks>=70 && marks<80){

System.out.println("B grade");

}

else if(marks>=80 && marks<90){

System.out.println("A grade");

}else if(marks>=90 && marks<100){

System.out.println("A+ grade");

}else{

System.out.println("Invalid!");

}

}

}

**JAVA SWITCH STATEMENT**

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 types, String . Since Java 7, you can use strings in the switch statement.

In other words, the switch statement tests the equality of a variable against multiple values.

**Points to Remember**

There can be one or N number of case values for a switch expression.

The case value must be of switch expression type only. The case value must be literal or constant. It doesn't allow variables.

The case values must be unique. In case of duplicate value, it renders compile-time error.

Each case statement can have a break statement which is optional. When control reaches to the break statement, it jumps the control after the switch expression. If a break statement is not found, it executes the next case.

The case value can have a default label which is optional.

**Syntax:**

switch(expression){

case value1:

//code to be executed;

break; //optional

case value2:

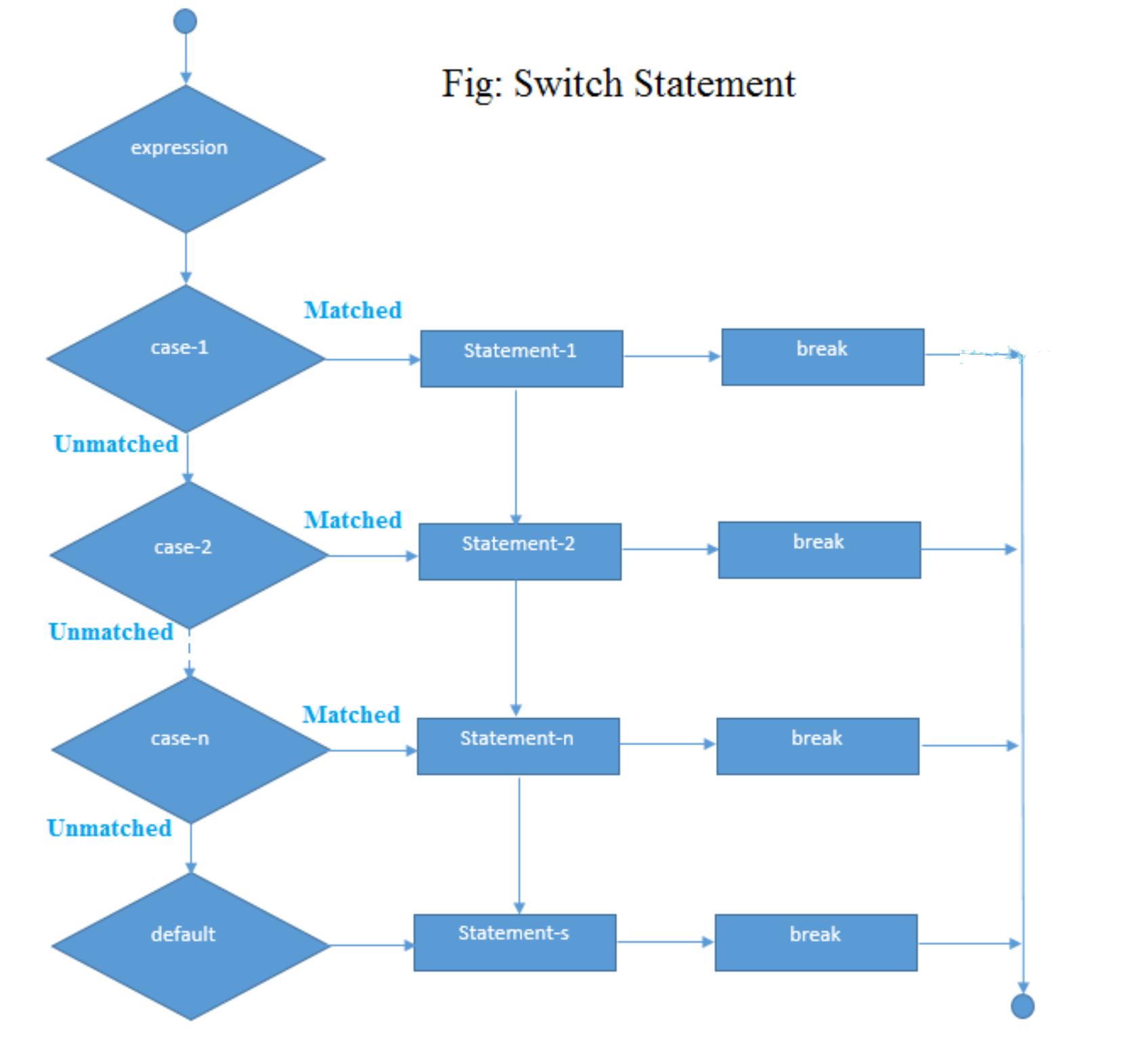
//code to be executed;

break; //optional

default:

code to be executed if all cases are not matched;

}



**Example:**

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");

}

}

}

**Program to check Vowel or Consonant:**

public class SwitchVowelExample {

public static void main(String[] args) {

char ch='O';

switch(ch)

{

case 'a':

System.out.println("Vowel");

break;

case 'e':

System.out.println("Vowel");

break;

case 'i':

System.out.println("Vowel");

break;

case 'o':

System.out.println("Vowel");

break;

case 'u':

System.out.println("Vowel");

break;

case 'A':

System.out.println("Vowel");

break;

case 'E':

System.out.println("Vowel");

break;

case 'I':

System.out.println("Vowel");

break;

case 'O':

System.out.println("Vowel");

break;

case 'U':

System.out.println("Vowel");

break;

default:

System.out.println("Consonant");

}

}

}

**LOOPS IN JAVA**

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true.

There are three types of loops in Java.

1. For loop
2. while loop
3. do-while loop

**1.Java 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.

There are three types of for loops in java.

1. Simple For Loop
2. For-each or Enhanced For Loop – Will be covered in Arrays/Collection
3. Labeled For Loop – Will be covered in later sessions

**Java Simple For Loop**

A simple for loop is the same as C/C++. We can initialize the variable, check condition and increment/decrement value. It consists of four parts:

**Syntax:**

for(initialization;condition;incr/decr){

//statement or code to be executed

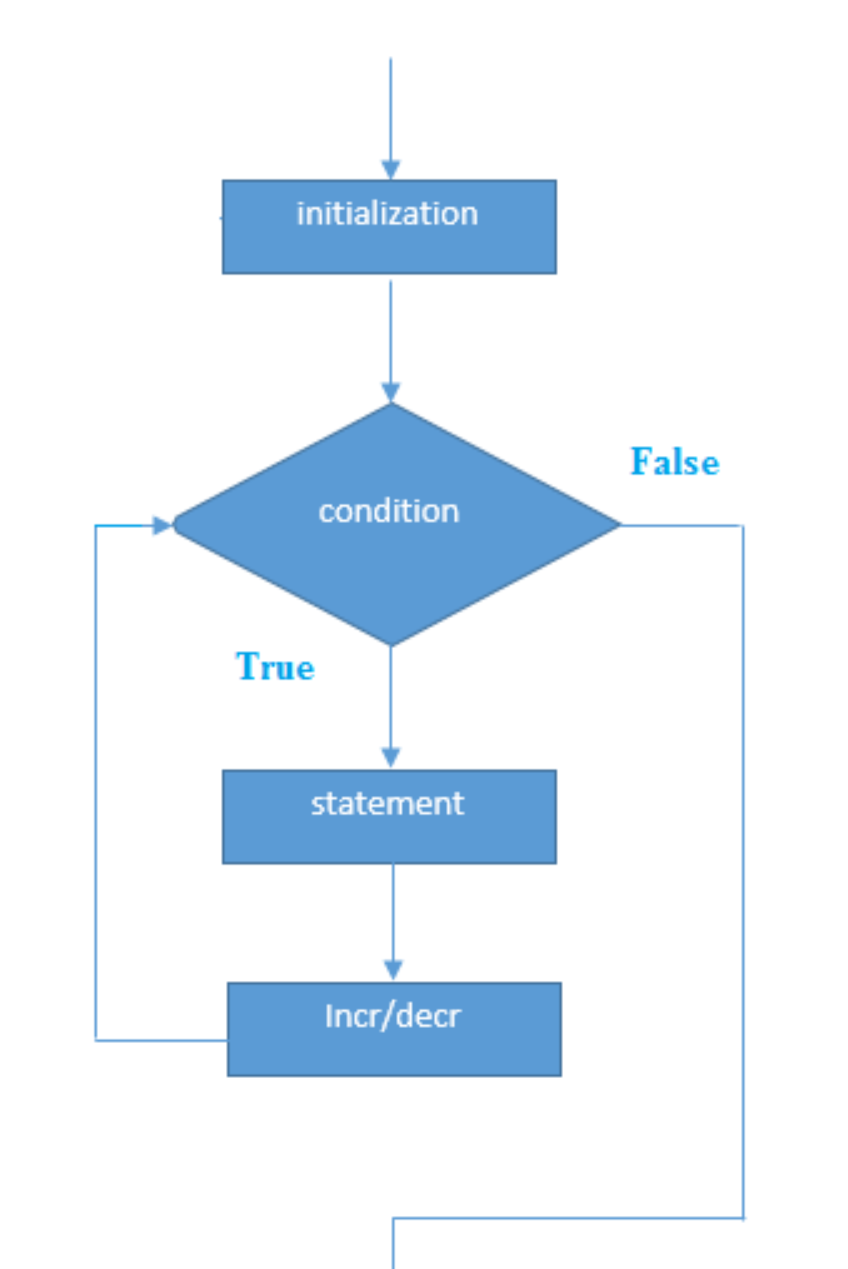
}

**Initialization**: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.

**Condition**: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.

**Statement**: The statement of the loop is executed each time until the second condition is false.

**Increment/Decrement:** It increments or decrements the variable value. It is an optional condition.



**Example:**

public class ForExample {

public static void main(String[] args) {

//Code of Java for loop

for(int i=1;i<=10;i++){

System.out.println(i);

}

}

}

**Java 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.

Syntax:

while(condition){

//code to be executed

}

**Example:**

public class WhileExample {

public static void main(String[] args) {

int i=1;

while(i<=10){

System.out.println(i);

i++;

}

}

}

**Java Infinitive While Loop**

If you pass true in the while loop, it will be infinitive while loop.

**Example:**

public class WhileExample2 {

public static void main(String[] args) {

while(true){

System.out.println("infinitive while loop");

}

}

}

**Java do-while Loop**

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

The Java do-while loop is executed at least once because condition is checked after loop body.

Syntax:

do{

//code to be executed

}while(condition);

**Example:**

public class DoWhileExample {

public static void main(String[] args) {

int i=1;

do{

System.out.println(i);

i++;

}while(i<=10);

}

}

**Java Break Statement**

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

The Java break statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

**Example:**

public class BreakExample {

public static void main(String[] args) {

//using for loop

for(int i=1;i<=10;i++){

if(i==5){

//breaking the loop

break;

}

System.out.println(i);

}

}

}

**Java Continue Statement**

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop.

The Java continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

Java Continue Statement Example

Example:

//Java Program to demonstrate the use of continue statement

//inside the for loop.

public class ContinueExample {

public static void main(String[] args) {

//for loop

for(int i=1;i<=10;i++){

if(i==5){

//using continue statement

continue;//it will skip the rest statement

}

System.out.println(i);

}

}

}

**Java Comments**

The Java comments are the statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code.

Types of Java Comments

There are three types of comments in Kava.

1. Single Line Comment
2. Multi Line Comment
3. Documentation Comment

//This is single line comment

/\*

This

is

multi line

comment

\*/

**Java Documentation Comment**

The documentation comment is used to create documentation API. To create documentation API, you need to use javadoc tool.

Syntax:

/\*\*

This

is

documentation

comment

\*/

Example:

/\*\* The Calculator class provides methods to get addition and subtraction of given 2 numbers.\*/

public class Calculator {

/\*\* The add() method returns addition of given numbers.\*/

public static int add(int a, int b){return a+b;}

/\*\* The sub() method returns subtraction of given numbers.\*/

public static int sub(int a, int b){return a-b;}

}

Compile it by javac tool:

javac Calculator.java

Create Documentation API by javadoc tool:

javadoc Calculator.java

Now, there will be HTML files created for your Calculator class in the current directory. Open the HTML files and see the explanation of Calculator class provided through documentation comment.