A **Java virtual machine** (**JVM**) is an abstract computing machine that enables a computer to run a Javaprogram. There are three notions of the JVM: specification, implementation, and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into Java bytecode.

Java Runtime Environment (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library. The Oracle Corporation, which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called HotSpot.

Java Development Kit (**JDK**) is a superset of a JRE and contains tools for Java programmers, e.g. a javaccompiler. The Java Development Kit is provided free of charge either by Oracle Corporation directly, or by theOpenJDK open source project, which is governed by Oracle.

A **Java virtual machine** (**JVM**) is an abstract computing machine that enables a computer to run a Javaprogram. There are three notions of the JVM: specification, implementation, and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into Java bytecode.

Java Runtime Environment (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library. The Oracle Corporation, which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called HotSpot.

Java Development Kit (**JDK**) is a superset of a JRE and contains tools for Java programmers, e.g. a javaccompiler. The Java Development Kit is provided free of charge either by Oracle Corporation directly, or by theOpenJDK open source project, which is governed by Oracle.

JVM specification

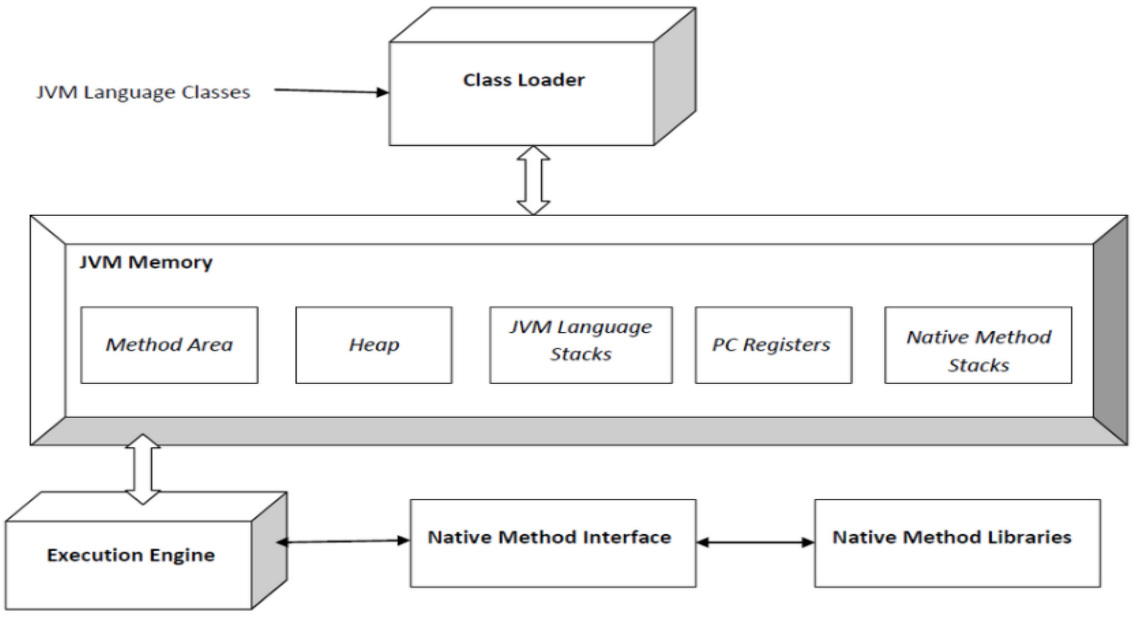
The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability: the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine.

Starting with Java Platform, Standard Edition (J2SE) 5.0, changes to the JVM specification have been developed under the Java Community Process as JSR 924.[2] As of 2006, changes to specification to support changes proposed to the class file format (JSR 202) are being done as a maintenance release of JSR 924. The specification for the JVM was published as the *blue book*, The preface states:

We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Oracle provides tests that verify the proper operation of implementations of the Java Virtual Machine.

VM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms



java

Java is a programming language and a platform.

Java is a high level, robust, secured and object-oriented programming language.

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

Java Example

```
class Simple{
   public static void main(String args[]){
    System.out.println("Hello Java");
   }
}
```

Applications of java

There are many devices where java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System

6. Smart Card
7. Robotics
8. Games etc.

Types of Java Applications

There are mainly 4 type of applications that can be created using java programming:

1) Standalone Application

It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

4) Mobile Application

An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

Features of Java

- Simple
- Object-Oriented
- Platform Independent
- secured
- Robust
- Architecture Neutral
- Portable
- High Performance
- Distributed
- Multi-threaded

There is given many features of java. They are also known **as java buzzwords**.

Simple

According to Sun, Java language is simple because:

Syntax is based on C++ (so easier for programmers to learn it after C++).java removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc. No need to remove unreferenced objects because there is Automatic Garbage Collection in java.

Object-oriented

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour.Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules.

Basic concepts of OOPs are:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

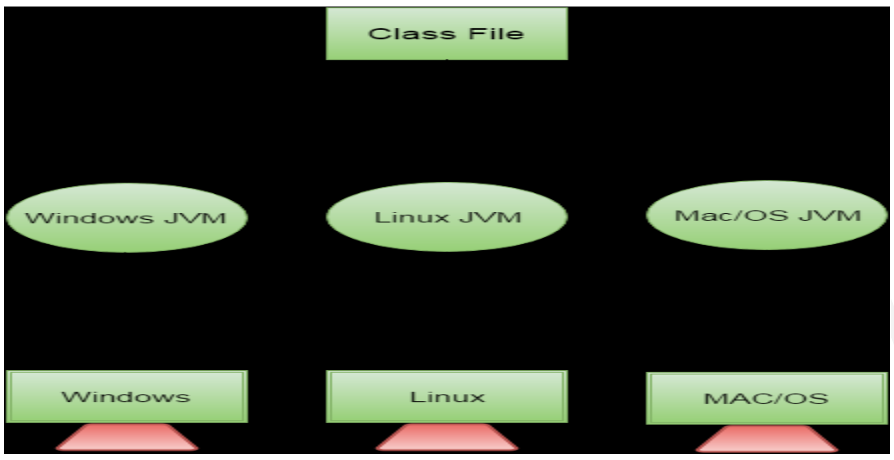
Platform Independent

java is platform independent.A platform is the hardware or software environment in which a program runs.There are two types of platforms software-based and hardware-based. Java provides software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

- Runtime Environment
- API(Application Programming Interface)

Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).



Secured

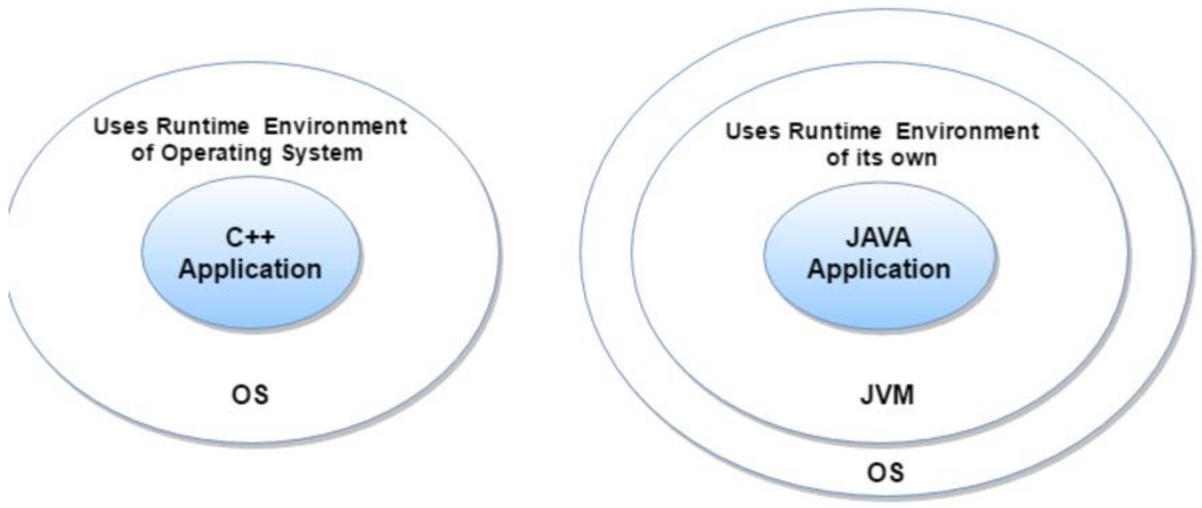
Java is secured because:

- No explicit pointer
- Java Programs run inside virtual machine sandbox

how java is secured

Classloader: adds security by separating the package for the classes of the local file system from those that are imported from network sources.Bytecode Verifier: checks the code fragments for illegal code that can violate access right to objects.

Security Manager: determines what resources a class can access such as reading and writing to the local disk.These security are provided by java language. Some security can also be provided by application developer through SSL, JAAS, Cryptography etc.


Uses Runtime Environment of Operating System
C++ Application
OS
Uses Runtime Environment of its own
JAVA Application
JVM
OS

Robust

Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust.

Architecture-neutral

There is no implementation dependent features e.g. size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.

Portable

We may carry the java bytecode to any platform.

High-performance

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)

Distributed

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc.

Creating hello java example

```
class Simple{
    public static void main(String args[]){
     System.out.println("Hello Java");
    }
}
```

Save this file as Simple.java

```
To compile:    javac Simple.java
To execute:    java Simple
```

Output**: Hello Java**

Understanding first java program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument..
- **System.out.println()** is used print statement.

JVM (Java Virtual Machine)

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

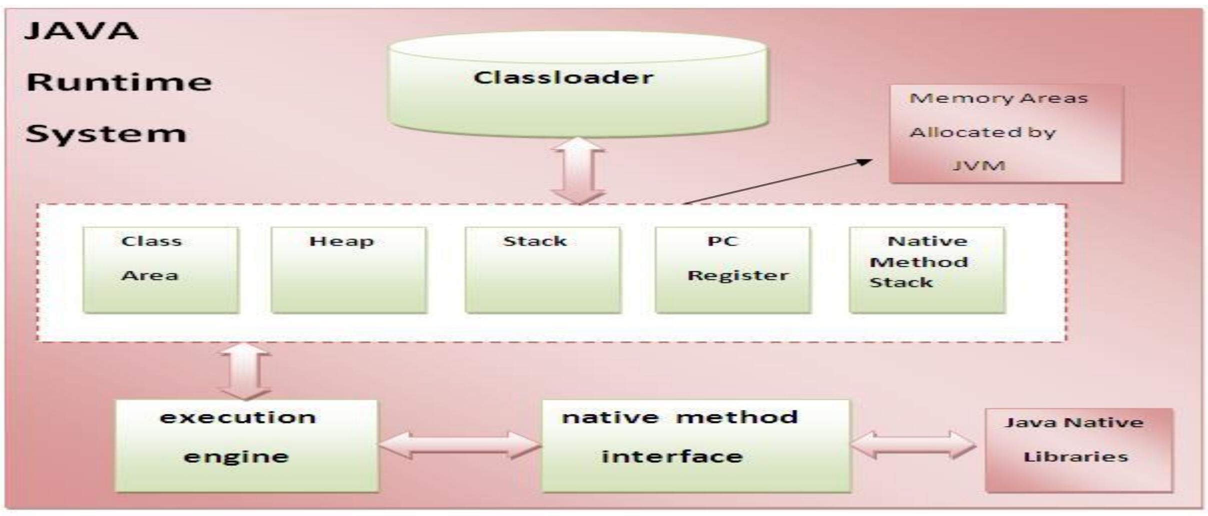
JVM provides definitions for the:

- Memory area
- Class file format

- Register set
- Garbage-collected heap
- Fatal error reporting etc.

Internal Architecture of JVM

Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



Class loader

Is a sub system of JVM that is used to load class files.

Class Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods

Heap

It is the runtime data area in which objects are allocated.

Stack

Java Stack stores frames.It holds local variables and partial results, and plays a part in method invocation and return.Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

Program Counter Register

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

Native Method Stack

It contains all the native methods used in the application.

Execution Engine

It contains:

1) A virtual processor

2) Interpreter: Read bytecode stream then execute the instructions.

3) Just-In-Time(JIT) compiler: It is used to improve the performance.JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

Variables and Data Types in Java

Variable is a name of memory location. There are three types of variables in java: local, instance and static.

There are two types of data types in java: primitive and non-primitive.

Variable

Variable is name of reserved area allocated in memory. In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be changed.

variables in java

int data=50;//Here data is variable

Types of Variable

There are three types of variables in java:

- local variable
- instance variable
- static variable

types of variables in java

1) **Local Variable**

A variable which is declared inside the method is called local variable.

2) **Instance Variable**

A variable which is declared inside the class but outside the method, is called instance variable . It is not declared as static**.**

3) Static variable

A variable that is declared as static is called static variable. It cannot be local.

Example to understand the types of variables in java

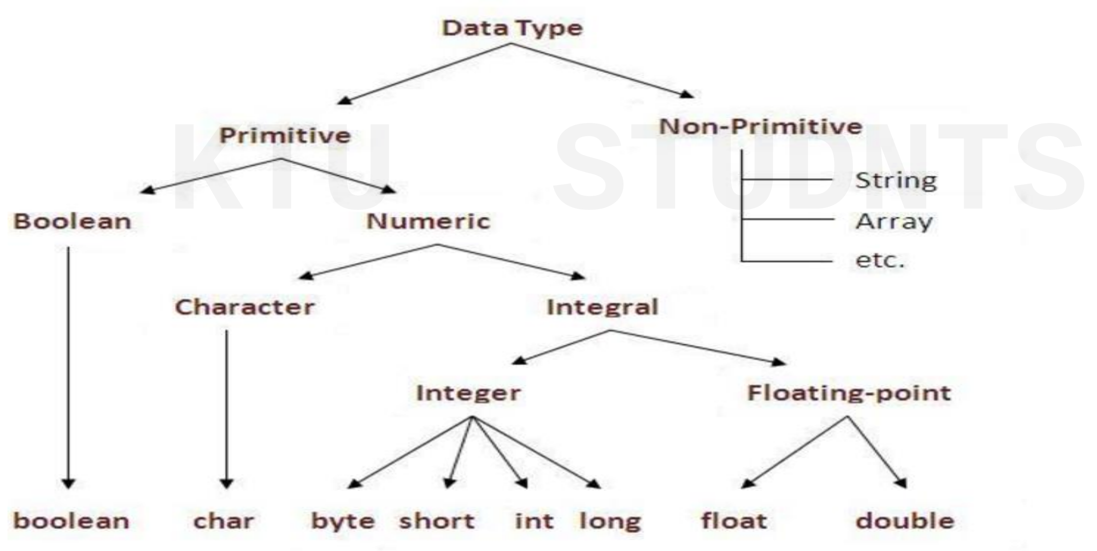
```
class A{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```

Data Types in Java

Data types represent the different values to be stored in the variable. In java, there are two types of data types:

- Primitive data types
- Non-primitive data types
- datatype in java

Data Type	**Default Value**	**Default size**
Boolean	**false**	**1 bit**
char	**'\u0000'**	**2 byte**
byte	**0**	**1 byte**
short	**0**	**2 byte**
int	**0**	**4 byte**
long	**0L**	**8 byte**
float	**0.0f**	**4 byte**
double	**0.0d**	**8 byte**



Operators in java

Operator in java is a symbol that is used to perform operations. There are many types of operators in java such as unary operator, arithmetic operator, relational operator, shift operator, bitwise operator, ternary operator and assignment operator.

Operators	**Precedence**
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %

additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

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.

- if statement
- if-else statement
- nested if statement
- if-else-if ladder

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
}
```

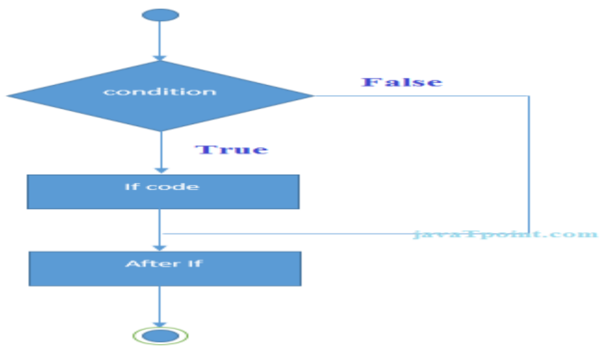
if statement in java

Example:

```
public class IfExample {
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



Java 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
}
```

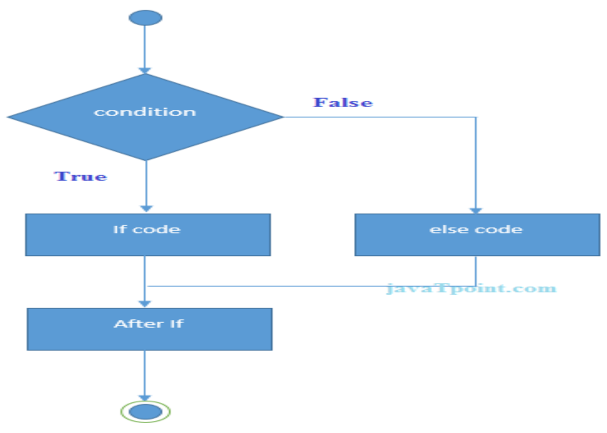
if-else statement in java

Example:

```
public class IfElseExample {
public static void main(String[] args) {
    int number=13;
    if(number%2==0){
        System.out.println("even number");
    }else{
        System.out.println("odd number");
    }
}
}
```

Output:

odd number



Java IF-else-if ladder Statement

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
}
```

if-else-if ladder statement in java

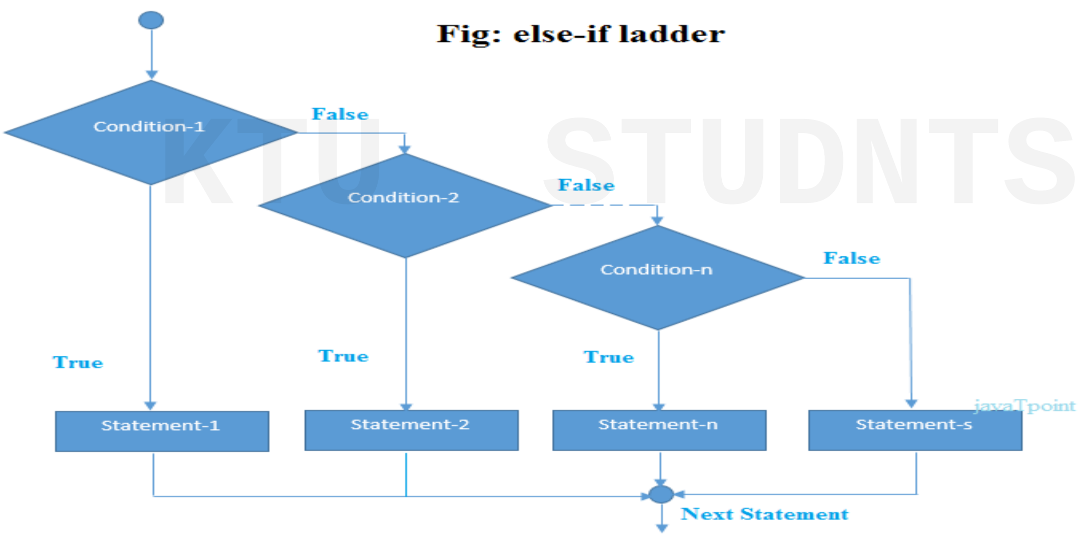
Example:

```
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!");
    }
}
}
```

Output:

```
C grade
```

Fig: else-if ladder



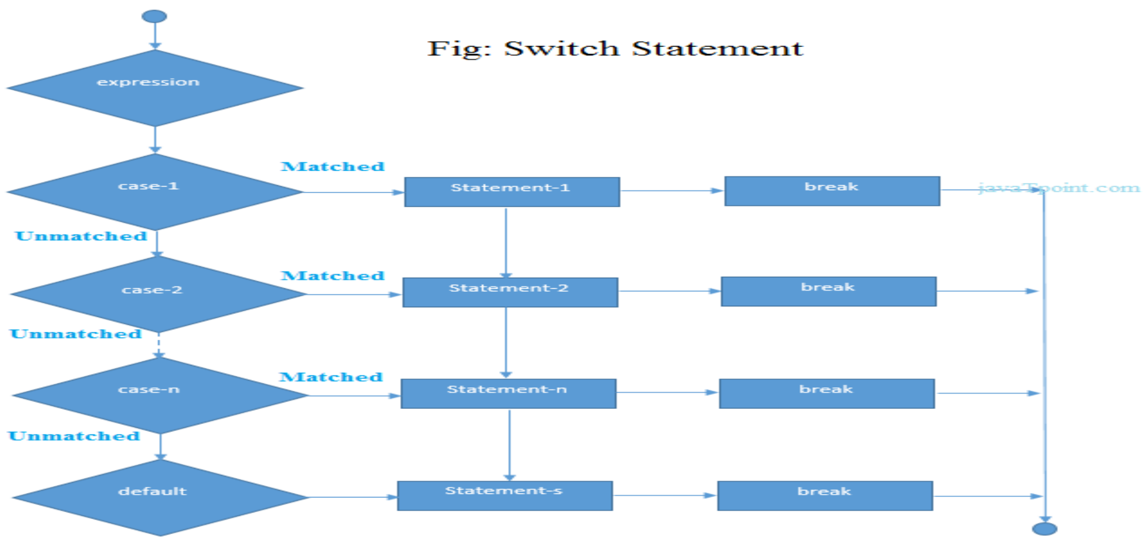
Java Switch Statement

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

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) {
    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
```

Java Switch Statement is fall-through

The java switch statement is fall-through. It means it executes all statement after first match if break statement is not used with switch cases.

Example:

```
public class SwitchExample2 {
public static void main(String[] args) {
    int number=20;
    switch(number){
    case 10: System.out.println("10");
    case 20: System.out.println("20");
    case 30: System.out.println("30");
    default:System.out.println("Not in 10, 20 or 30");
    }
}
}
```

Output:

```
20
30
Not in 10, 20 or 30
```

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 loop in java.

- Simple For Loop
- For-each or Enhanced For Loop
- Labeled For Loop

Java Simple For Loop

The simple for loop is same as C/C++. We can initialize variable, check condition and increment/decrement value.

Syntax:

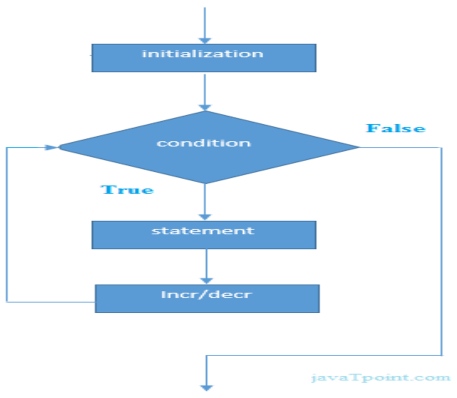
```
for(initialization;condition;incr/decr){
//code to be executed
}
```

Example:

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

Output:

```
1 2 3 4 5 6  7  8 9  10
```

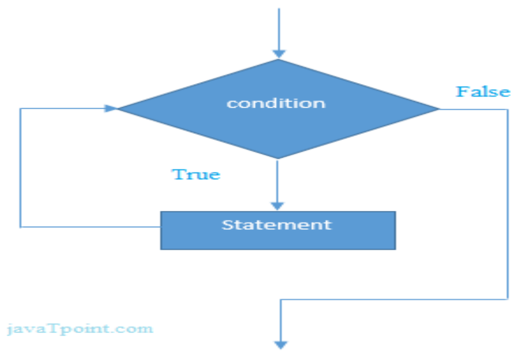


Java While Loop

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++;
    }
}
}
```

Output:

```
1  2  3  4      5        6        7        8        9        10
```

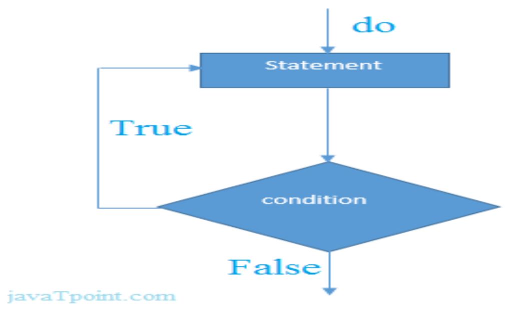
Java do-while Loop

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);
}
```

www.ktustudents.in

```
}
```

Output:

```
1          2           3         4          5          6          7          8          9          10
```

Java Break Statement

The Java *break* 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.

Syntax:

```
jump-statement;
break;
```

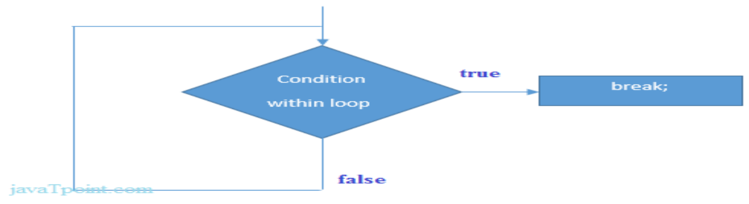


Figure: Flowchart of break statement

Java Break Statement with Loop

Example:

```
public class BreakExample {
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
```

Java Break Statement with Inner Loop

It breaks inner loop only if you use break statement inside the inner loop.

Example:

```
public class BreakExample2 {

public static void main(String[] args) {
         for(int i=1;i<=3;i++){
               for(int j=1;j<=3;j++){
                  if(i==2&&j==2){
                     break;
                  }
                  System.out.println(i+" "+j);
               }
         }
}
}
```

Output:

```
1 1    1 2    1 3    2 1    3 1    3 2    3 3
```

java Continue Statement

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

Syntax:

jump-statement;

continue;

Java Continue Statement Example

Example:

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

Output:

```
1        2        3        4        6        7        8        9        10
```

Java Continue Statement with Inner Loop

It continues inner loop only if you use continue statement inside the inner loop.

Example:

```
public class ContinueExample2 {
public static void main(String[] args) {
            for(int i=1;i<=3;i++){
                    for(int j=1;j<=3;j++){
                        if(i==2&&j==2){
                            continue;
                        }
                        System.out.println(i+" "+j);
                    }
            }
}
}
```

Output:

1 1   1 2   1 3   2 1   2 3   3 1   3 2   3 3

Java Comments

The java comments are 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 for specific time.

Types of Java Comments

There are 3 types of comments in java.

- Single Line Comment
- Multi Line Comment
- Documentation Comment

1) Java Single Line Comment

The single line comment is used to comment only one line.

Syntax:

//This is single line comment

Example:

```
public class CommentExample1 {
public static void main(String[] args) {
    int i=10;//Here, i is a variable
    System.out.println(i);
}
}
```

Output:

```
10
```

2**) Java Multi Line Comment**

The multi line comment is used to comment multiple lines of code.

Syntax:

/* This is multi line comment */

Example:

```
public class CommentExample2 {
public static void main(String[] args) {
/* Let's declare and
 print variable in java. */
    int i=10;
    System.out.println(i);
}
}
```

Output:

```
10
```

3) **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.