Unit-1 Part-1 Object Oriented Programming with Java

Content-: Introduction: Why Java, History of Java, JVM, JRE, Java Environment, Java Source File Structure, and Compilation. Fundamental, Programming Structures in Java: Access Specifies, Static Members, Final Members, Comments, Data types, Variables, Operators, Control Flow.

Outcome of this topic –: Overall, completing these topics will equip learners with essential skills and knowledge in Java programming, enabling them to build a strong foundation for further exploration of advanced Java concepts and software development practices.

What is Java?

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

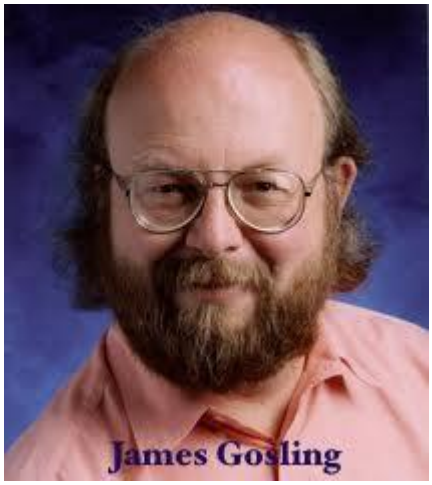
History of Java

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the 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. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

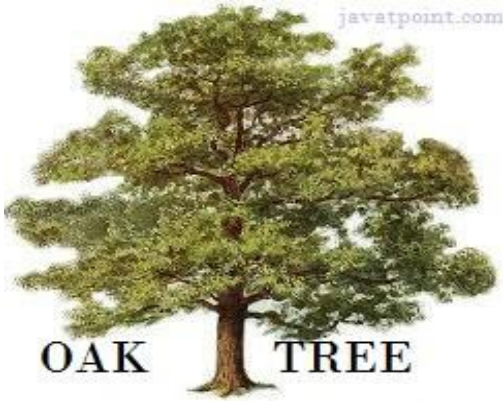
Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. Following are given significant points that describe the history of Java.

Why Java was named as "Oak"?



5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

6) In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Why Java Programming named "Java"?

7) Why had they chose the name Java for Java language? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell, and fun to say.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

According to James Gosling, "Java was one of the top choices along with **Silk**". Since Java was so unique, most of the team members preferred Java than other names.

8) Java is an island in Indonesia where the first coffee was produced (called Java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having a cup of coffee nearby his office.

9) Notice that Java is just a name, not an acronym.

10) Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 was released on January 23, 1996. After the first release of Java, there have been many additional features added to the language. Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications, cards, etc. Each new version adds new features in Java.

Java Example

Simple.java

```
1. class Simple{
2.     public static void main(String args[]){
3.      System.out.println("Hello Java");
4.     }
5. }
```

Java Comments

There are two types of comments in Java.

1. Single Line Comment

2. Multi Line Comment

1) Java Single Line Comment

The single-line comment is used to comment only one line of the code. It is the widely used and easiest way of commenting the statements.

Syntax:

```
1. //This is single line comment
```

CommentExample1.java

```
1. public class CommentExample1 {
2. public static void main(String[] args) {
3. int i=10; // i is a variable with value 10
4. System.out.println(i);  //printing the variable i
5. }
6. }
```

Output:

```
10
```

2) Java Multi Line Comment

The multi-line comment is used to comment multiple lines of code. It can be used to explain a complex code snippet or to comment multiple lines of code at a time (as it will be difficult to use single-line comments there).

Syntax:

```
1. /*
2. This
3. is
4. multi line
5. comment
6. */
```

Unit-1 Part-1 Object Oriented Programming with Java

Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are 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 a dynamic page is called a web application. Currently, Servlet, JSP, Struts, Spring, Hibernate, 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. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

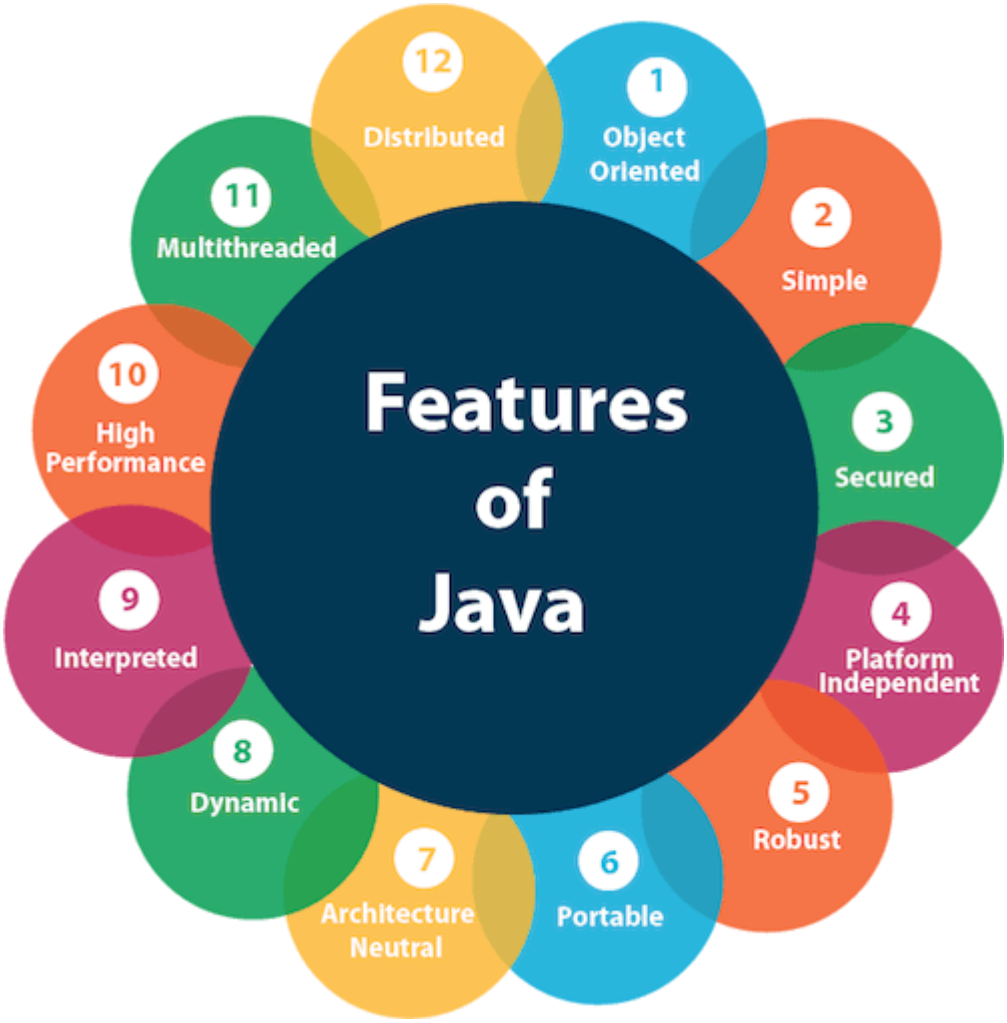
Features of Java

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java buzzwords.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

A list of the most important features of the Java language is given below.



1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

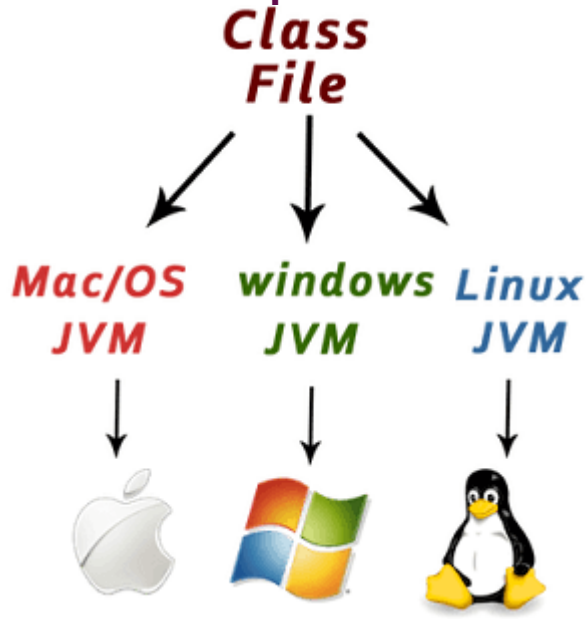
Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

Platform Independent



Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

Java code can be executed on multiple platforms, for example, 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).

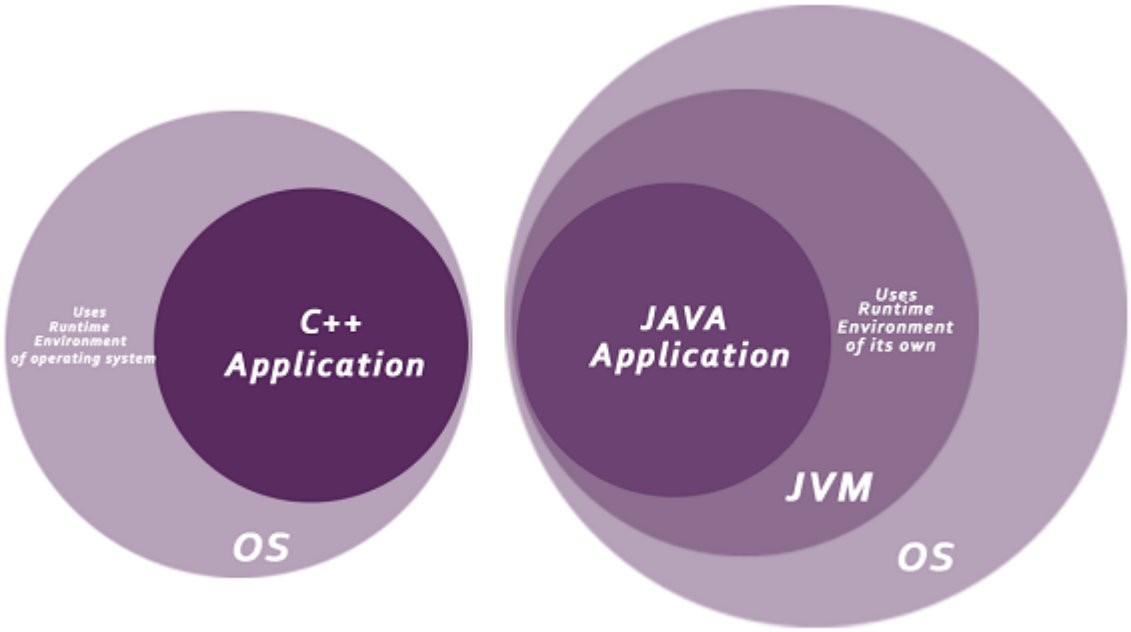
Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

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

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



- **Classloader:** Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
- **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access rights to objects.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Robust

The English mining of Robust is strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

First Java Program | Hello World Example

Creating Hello World Example

Let's create the hello java program:

```
1. class Simple{
2.     public static void main(String args[]){
3.      System.out.println("Hello Java");
4.     }
5. }
```

Output:

```
Hello Java
```

Difference between JDK, JRE, and JVM

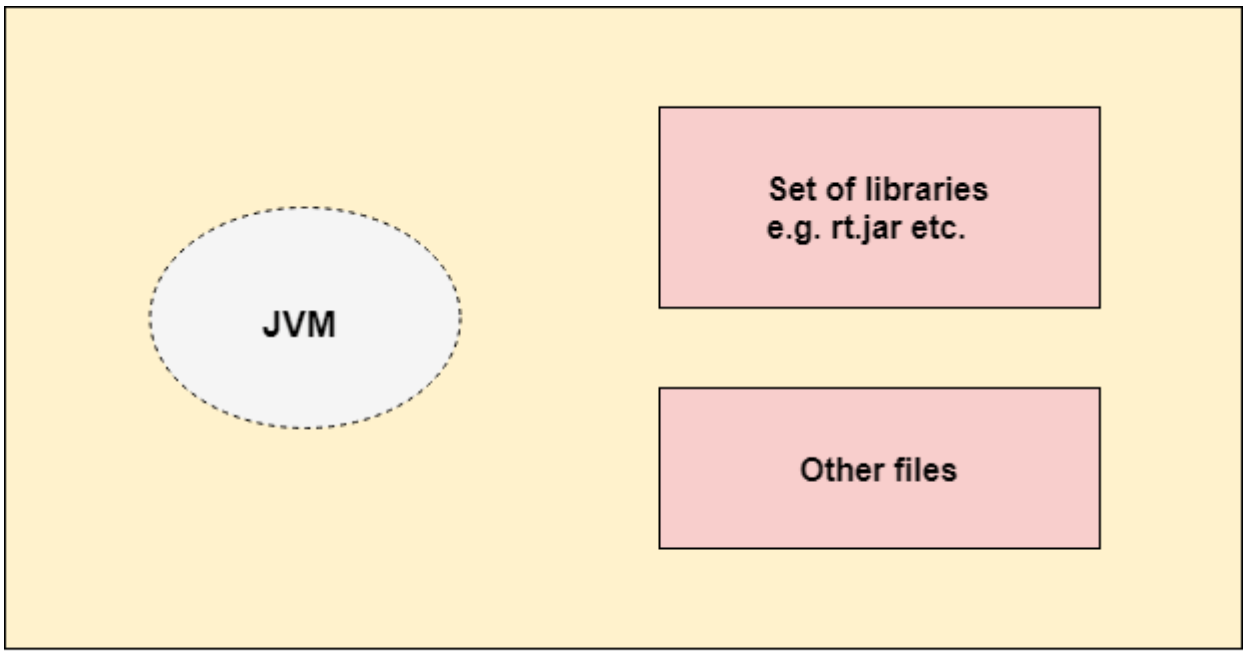
JVM

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JRE

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

Unit-1 Part-1 Object Oriented Programming with Java



JRE

JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

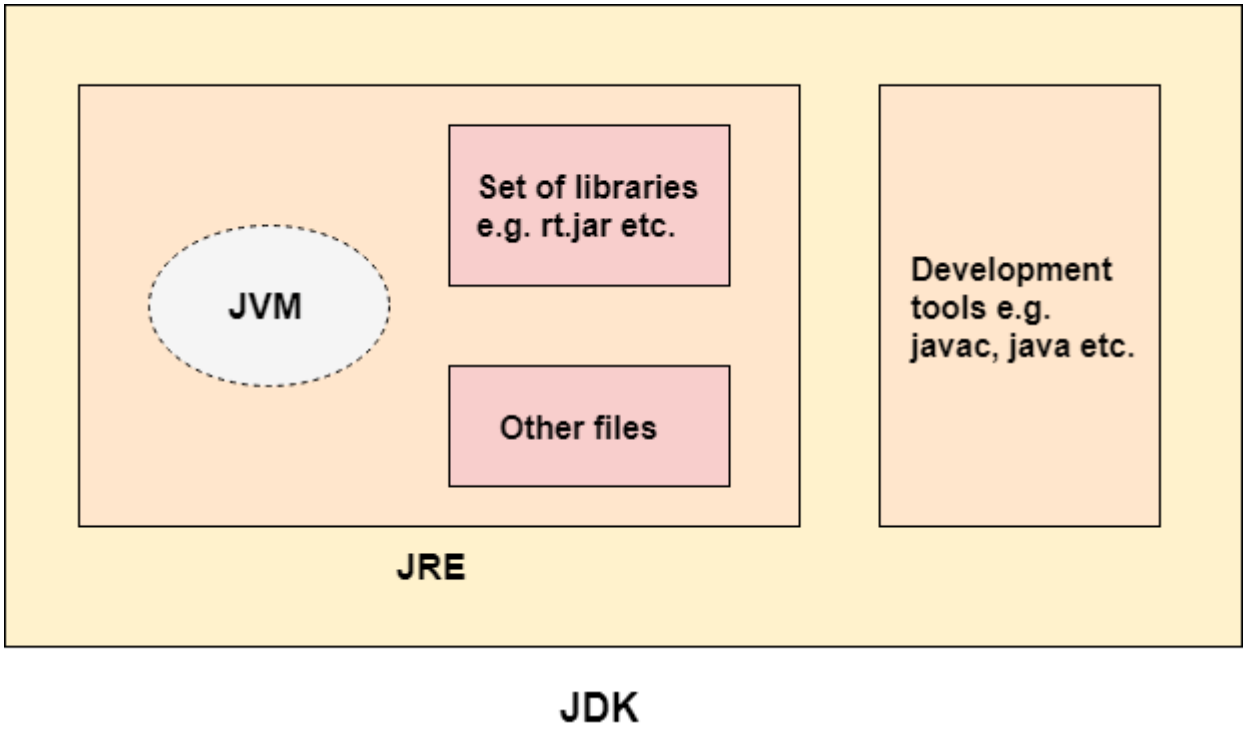
JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



Variable

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

```
1. int data=50;//Here data is variable
```

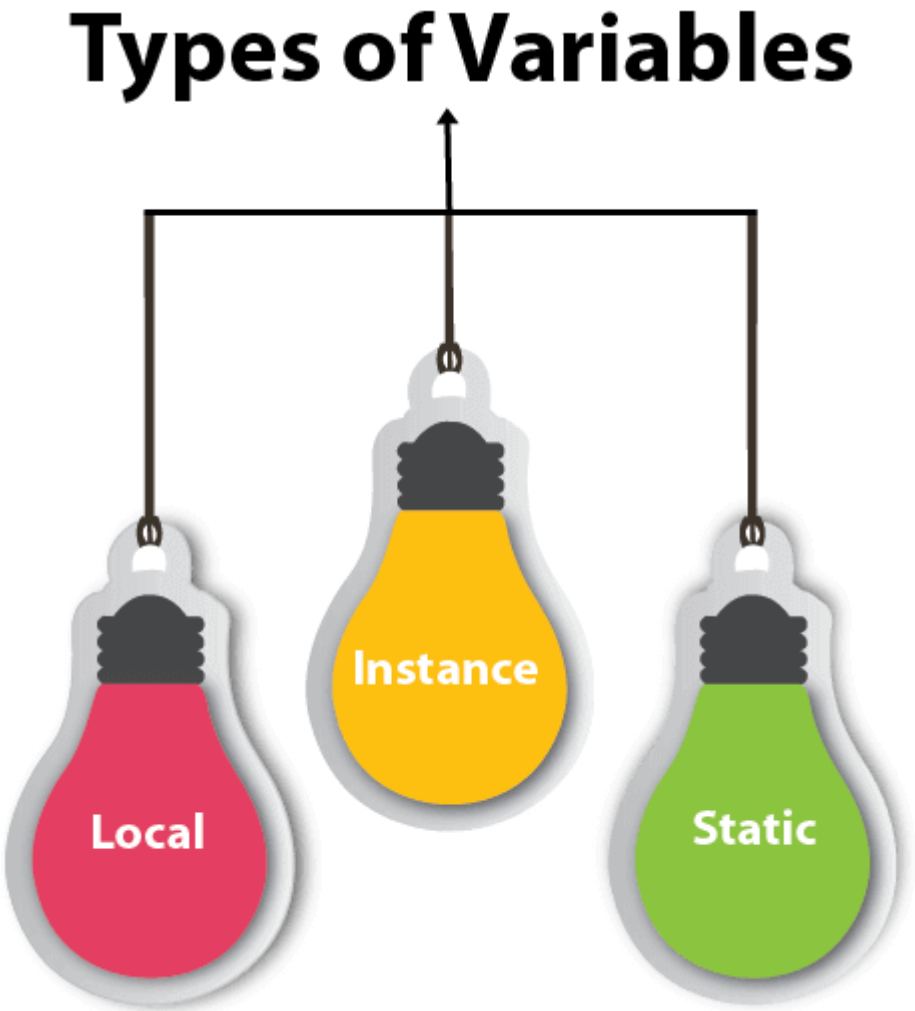
Types of Variables

There are three types of variables in Java:

- local variable
- instance variable
- static variable

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

3) Static variable

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Example to understand the types of variables in java

```
1. public class A
2. {
3.     static int m=100;//static variable
4.     void method()
5.     {
6.         int n=90;//local variable
7.     }
8.     public static void main(String args[])
9.     {
10.        int data=50;//instance variable
11.    }
12. }//end of class
```

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

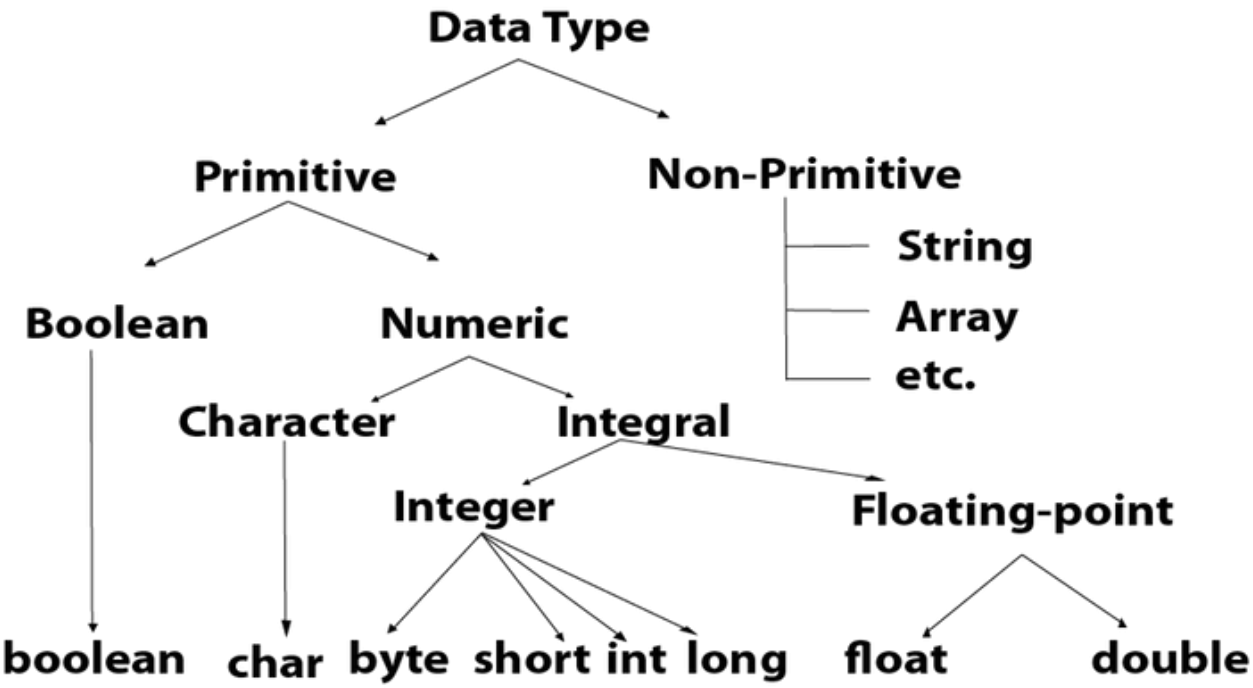
Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

Java is a statically-typed programming language. It means, all variables must be declared before its use. That is why we need to declare variable's type and name.

There are 8 types of primitive data types:

- boolean data type
- byte data type
- char data type
- short data type
- int data type
- long data type
- float data type
- double data type



Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.

The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Example:

```
1. Boolean one = false
```

Byte Data Type

The byte data type is an example of primitive data type. It isan 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0.

The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.

Example:

```
1. byte a = 10, byte b = -20
```

Short Data Type

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example:

```
1. short s = 10000, short r = -5000
```

Int Data Type

The int data type is a 32-bit signed two's complement integer. Its value-range lies between - 2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive). Its minimum value is - 2,147,483,648and maximum value is 2,147,483,647. Its default value is 0.

The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

Example:

```
1. int a = 100000, int b = -200000
```

Unit-1 Part-1 Object Oriented Programming with Java

Long Data Type

The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63 -1)(inclusive). Its minimum value is - 9,223,372,036,854,775,808and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Example:

```
1. long a = 100000L, long b = -200000L
```

Float Data Type

The float data type is a single-precision 32-bit IEEE 754 floating point.Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

Example:

```
1. float f1 = 234.5f
```

Double Data Type

The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

Example:

```
1. double d1 = 12.3
```

Char Data Type

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.

Example:

Unit-1 Part-1 Object Oriented Programming with Java

1. **char** letterA = 'A'

Operators in Java

Operator in Java is a symbol that is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Java Operator Precedence

Operator Type	Category	Precedence
Unary	postfix	*expr++ expr--*
	prefix	*++expr --expr +expr -expr ~ !*
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Relational	comparison	`< > <= >= instanceof`
	equality	`== !=`
Bitwise	bitwise AND	`&`
	bitwise exclusive OR	`^`
	bitwise inclusive OR	`|`
Logical	logical AND	`&&`
	logical OR	`||`
Ternary	ternary	`? :`
Assignment	assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Java Unary Operator

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

Java Unary Operator Example: ++ and --

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. int x=10;
4. System.out.println(x++);//10 (11)
5. System.out.println(++x);//12
```

6. System.out.println(x--);//12 (11)
7. System.out.println(--x);//10
8. }}

Output:

```
10
12
12
10
```

Java Unary Operator Example 2: ++ and --

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=10;
5. System.out.println(a++ + ++a);//10+12=22
6. System.out.println(b++ + b++);//10+11=21
7.
8. }}
```

Output:

```
22
21
```

Java Arithmetic Operators

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

Java Arithmetic Operator Example

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. System.out.println(a+b);//15
6. System.out.println(a-b);//5
7. System.out.println(a*b);//50
8. System.out.println(a/b);//2
```

```
9. System.out.println(a%b);//0
10. }}
```

Output:

```
15
5
50
2
0
```

Java Arithmetic Operator Example: Expression

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. System.out.println(10*10/5+3-1*4/2);
4. }}
```

Output:

```
21
```

Java AND Operator Example: Logical && vs Bitwise &

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. int c=20;
6. System.out.println(a<b&&a++<c);//false && true = false
7. System.out.println(a);//10 because second condition is not checked
8. System.out.println(a<b&a++<c);//false && true = false
9. System.out.println(a);//11 because second condition is checked
10. }}
```

Output:

```
false
10
false
11
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

Java Ternary Operator

Java Ternary operator is used as one line replacement for if-then-else statement and used a lot in Java programming. It is the only conditional operator which takes three operands.

Java Ternary Operator Example

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. int a=2;
4. int b=5;
5. int min=(a<b)?a:b;
6. System.out.println(min);
7. }}
```

Output:

```
2
```

Another Example:

```
1. public class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. int min=(a<b)?a:b;
6. System.out.println(min);
7. }}
```

Output:

```
5
```

Java Assignment Operator

Java assignment operator is one of the most common operators. It is used to assign the value on its right to the operand on its left.

Java Assignment Operator Example

```
1. public class OperatorExample{
```

```
2. public static void main(String args[]){
3. int a=10;
4. int b=20;
5. a+=4;//a=a+4 (a=10+4)
6. b-=4;//b=b-4 (b=20-4)
7. System.out.println(a);
8. System.out.println(b);
9. }}
```

Output:

```
14
16
```

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
- if-else-if ladder
- nested if statement

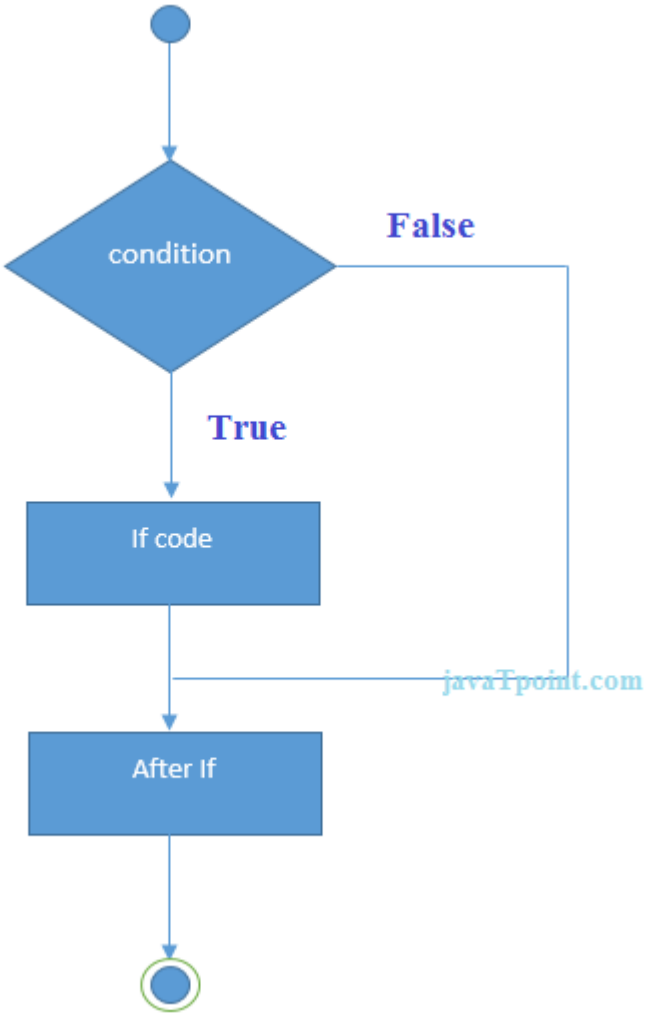
Java if Statement

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

Syntax:

```
1. if(condition){
2. //code to be executed
3. }
```

Unit-1 Part-1 Object Oriented Programming with Java



Example:

```
1. //Java Program to demonstate the use of if statement.
2. public class IfExample {
3. public static void main(String[] args) {
4.     //defining an 'age' variable
5.     int age=20;
6.     //checking the age
7.     if(age>18){
8.         System.out.print("Age is greater than 18");
9.     }
10. }
11. }
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

Test it Now

Output:

```
Age is greater than 18
```

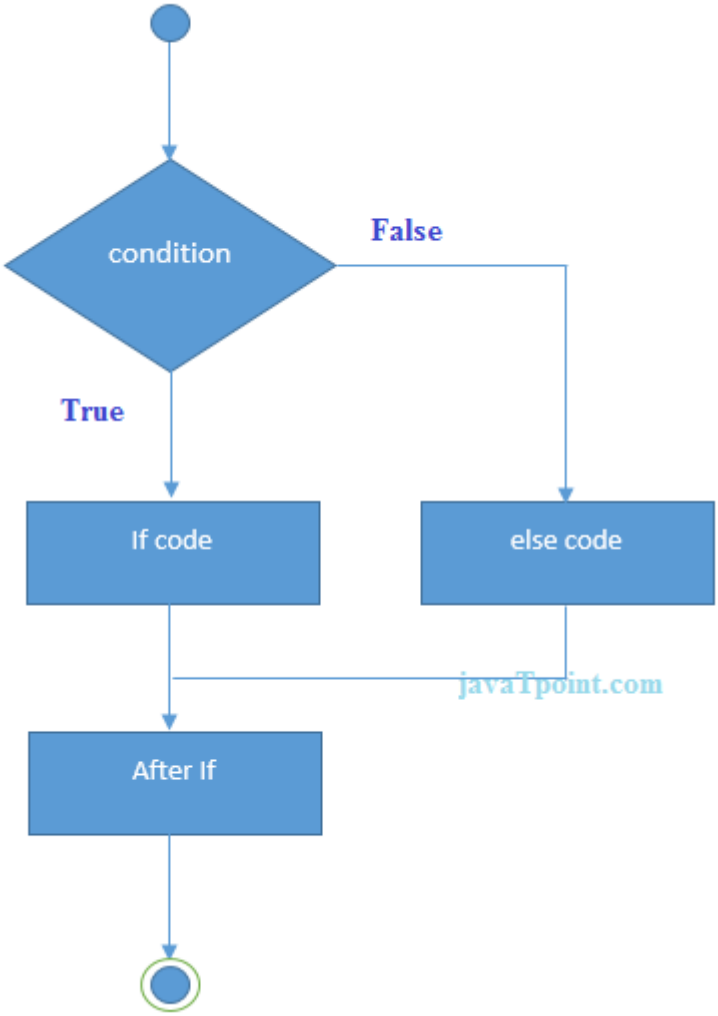
Java if-else Statement

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

Syntax:

```
1. if(condition){
2. //code if condition is true
3. }else{
4. //code if condition is false
5. }
```

Unit-1 Part-1 Object Oriented Programming with Java



Example:

```
1. //A Java Program to demonstrate the use of if-else statement.
2. //It is a program of odd and even number.
3. public class IfElseExample {
4. public static void main(String[] args) {
5.     //defining a variable
6.     int number=13;
7.     //Check if the number is divisible by 2 or not
8.     if(number%2==0){
9.         System.out.println("even number");
10.    }else{
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

```
11.         System.out.println("odd number");
12.     }
13. }
14. }
```

Test it Now

Output:

```
odd number
```

Leap Year Example:

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

```
1.  public class LeapYearExample {
2.  public static void main(String[] args) {
3.      int year=2020;
4.      if(((year % 4 ==0) && (year % 100 !=0)) || (year % 400==0)){
5.          System.out.println("LEAP YEAR");
6.      }
7.      else{
8.          System.out.println("COMMON YEAR");
9.      }
10. }
11. }
```

Output:

```
LEAP 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:

```
1.  public class IfElseTernaryExample {
2.  public static void main(String[] args) {
```

```
3.      int number=13;
4.      //Using ternary operator
5.      String output=(number%2==0)?"even number":"odd number";
6.      System.out.println(output);
7.  }
8.  }
```

Output:

```
odd number
```

Java if-else-if ladder Statement

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

Syntax:

```
if(condition1){
1.  //code to be executed if condition1 is true
2.  }else if(condition2){
3.  //code to be executed if condition2 is true
4.  }
5.  else if(condition3){
6.  //code to be executed if condition3 is true
7.  }
8.  ...
9.  else{
10. //code to be executed if all the conditions are false
11. }
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

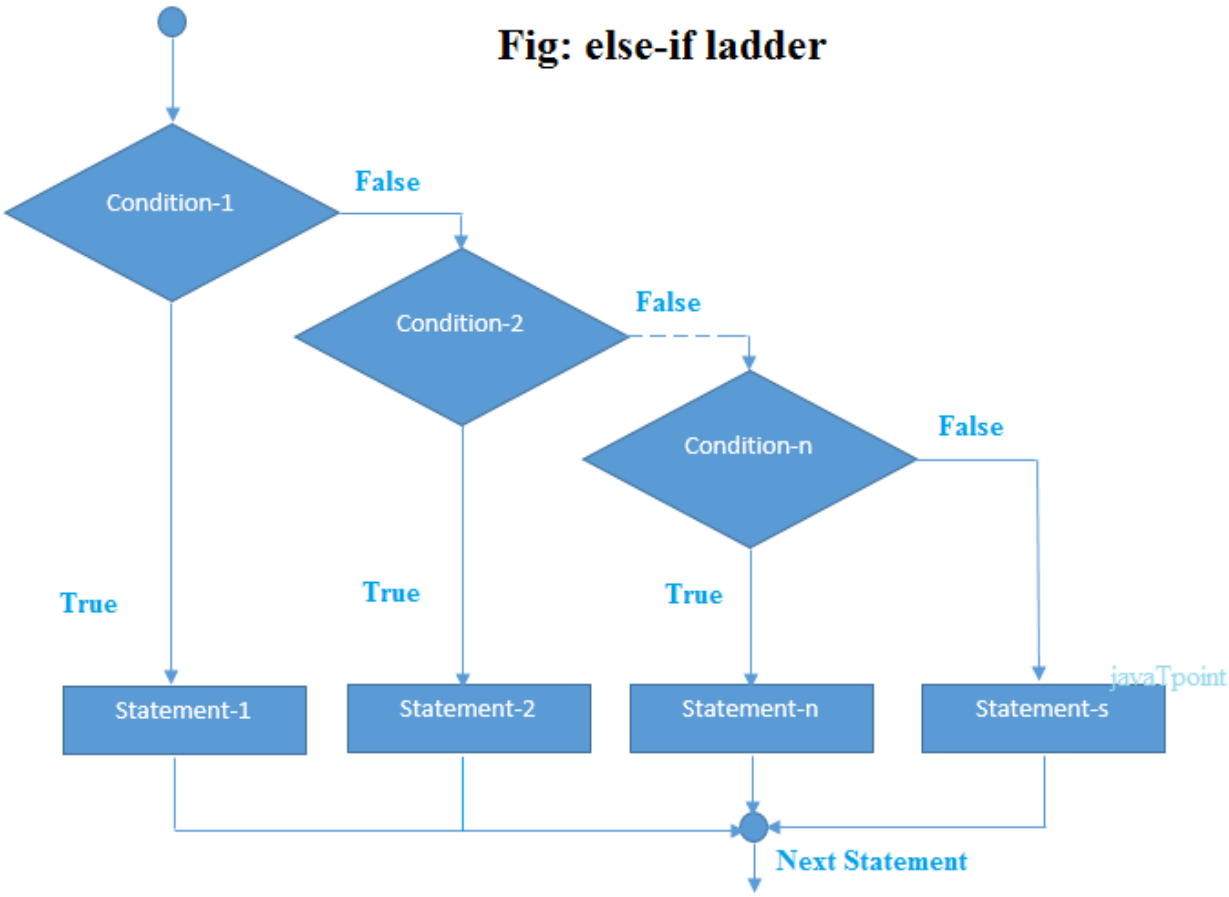


Fig: else-if ladder

Example:

```
1. //Java Program to demonstrate the use of If else-if ladder.
2. //It is a program of grading system for fail, D grade, C grade, B grade, A grade and A
   +.
3. public class IfElseIfExample {
4. public static void main(String[] args) {
5.     int marks=65;
6.
7.     if(marks<50){
8.         System.out.println("fail");
9.     }
10.    else if(marks>=50 && marks<60){
11.        System.out.println("D grade");
12.    }
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

```
13.    else if(marks>=60 && marks<70){
14.        System.out.println("C grade");
15.    }
16.    else if(marks>=70 && marks<80){
17.         System.out.println("B grade");
18.    }
19.    else if(marks>=80 && marks<90){
20.        System.out.println("A grade");
21.    }else if(marks>=90 && marks<100){
22.        System.out.println("A+ grade");
23.    }else{
24.        System.out.println("Invalid!");
25.    }
26. }
27. }
```

Output:

```
C grade
```

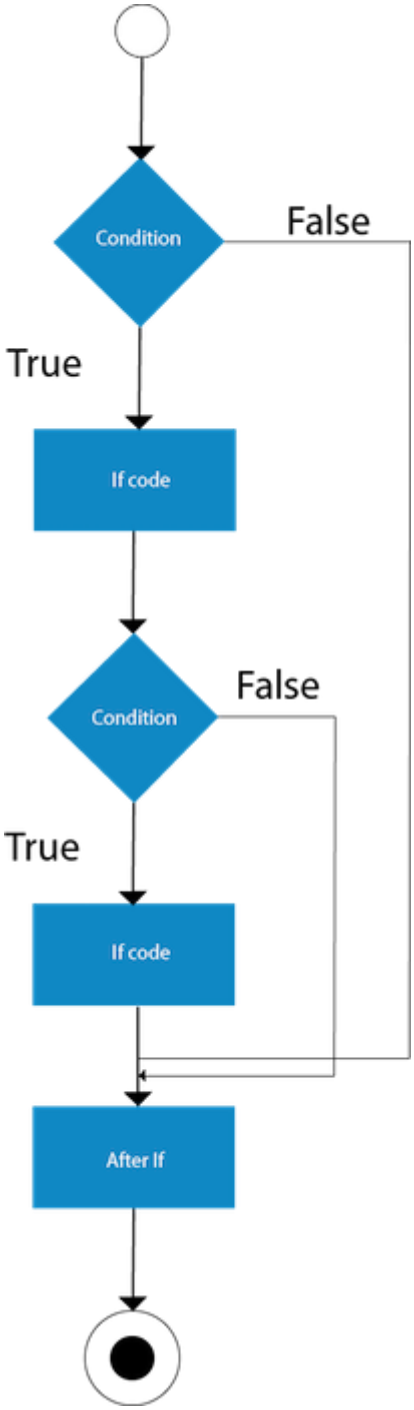
Java Nested if statement

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:

```
1.  if(condition){
2.       //code to be executed
3.            if(condition){
4.              //code to be executed
5.       }
6.  }
```

Unit-1 Part-1 Object Oriented Programming with Java



Example:

```
1. //Java Program to demonstrate the use of Nested If Statement.
2. public class JavaNestedIfExample {
3. public static void main(String[] args) {
4.     //Creating two variables for age and weight
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

```
5.      int age=20;
6.      int weight=80;
7.      //applying condition on age and weight
8.      if(age>=18){
9.          if(weight>50){
10.             System.out.println("You are eligible to donate blood");
11.         }
12.     }
13. }}
```

Test it Now

Output:

```
You are eligible to donate blood
```

Loops in Java

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.

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

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:

Unit-1 Part-1 Object Oriented Programming with Java

1. **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.
2. **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.
3. **Increment/Decrement**: It increments or decrements the variable value. It is an optional condition.
4. **Statement**: The statement of the loop is executed each time until the second condition is false.

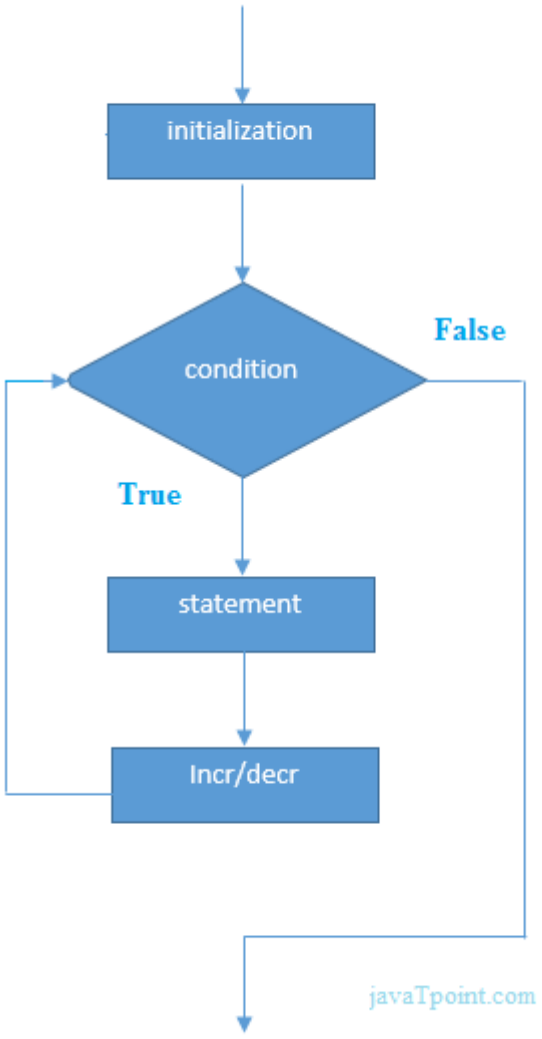
Syntax:

```
for(initialization; condition; increment/decrement){
//statement or code to be executed
}
```

Flowchart:

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



Example:

ForExample.java

```
1. //Java Program to demonstrate the example of for loop
2. //which prints table of 1
3. public class ForExample {
4. public static void main(String[] args) {
5.     //Code of Java for loop
6.     for(int i=1;i<=10;i++){
7.         System.out.println(i);
8.     }
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

```
9.  }
10. }
```

Test it Now

Output:

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

Java Nested for Loop

If we have a for loop inside the another loop, it is known as nested for loop. The inner loop executes completely whenever outer loop executes.

Example:

NestedForExample.java

```
1.  public class NestedForExample {
2.  public static void main(String[] args) {
3.  //loop of i
4.  for(int i=1;i<=3;i++){
5.  //loop of j
6.  for(int j=1;j<=3;j++){
7.          System.out.println(i+" "+j);
8.  }//end of i
9.  }//end of j
10. }
11. }
```

Output:

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

Unit-1 Part-1 Object Oriented Programming with Java

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

Pyramid Example 1:

PyramidExample.java

```
1. public class PyramidExample {
2. public static void main(String[] args) {
3. for(int i=1;i<=5;i++){
4. for(int j=1;j<=i;j++){
5.         System.out.print("* ");
6. }
7. System.out.println();//new line
8. }
9. }
10. }
```

Output:

```
*
* *
* * *
* * * *
* * * * *
```

Pyramid Example 2:

PyramidExample2.java

```
1. public class PyramidExample2 {
2. public static void main(String[] args) {
3. int term=6;
4. for(int i=1;i<=term;i++){
5. for(int j=term;j>=i;j--){
6.         System.out.print("* ");
7. }
8. System.out.println();//new line
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

9. }
10. }
11. }

Output:

```
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

Java 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.

The while loop is considered as a repeating if statement. If the number of iteration is not fixed, it is recommended to use the while loop.

Syntax:

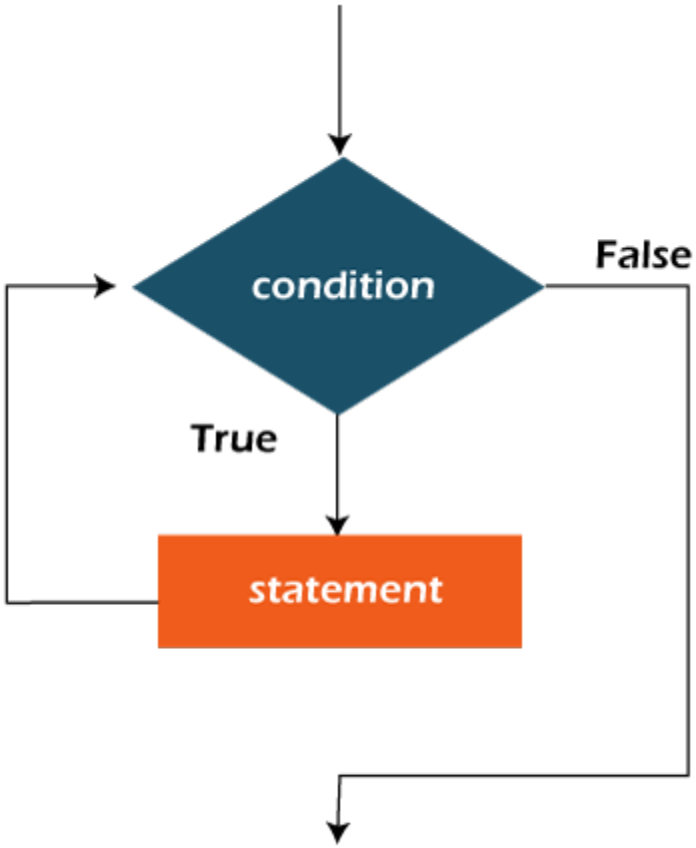
1. **while** (condition){
2. //code to be executed
3. I ncrement / decrement statement
4. }

Flowchart of Java While Loop

Here, the important thing about while loop is that, sometimes it may not even execute. If the condition to be tested results into false, the loop body is skipped and first statement after the while loop will be executed.

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



WhileExample.java

```
1. public class WhileExample {
2. public static void main(String[] args) {
3.     int i=1;
4.     while(i<=10){
5.         System.out.println(i);
6.     i++;
7.     }
8. }
9. }
```

Test it Now

Output:

```
1
2
3
```


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

```
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 repeatedly, until the specified condition is true. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use a 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.

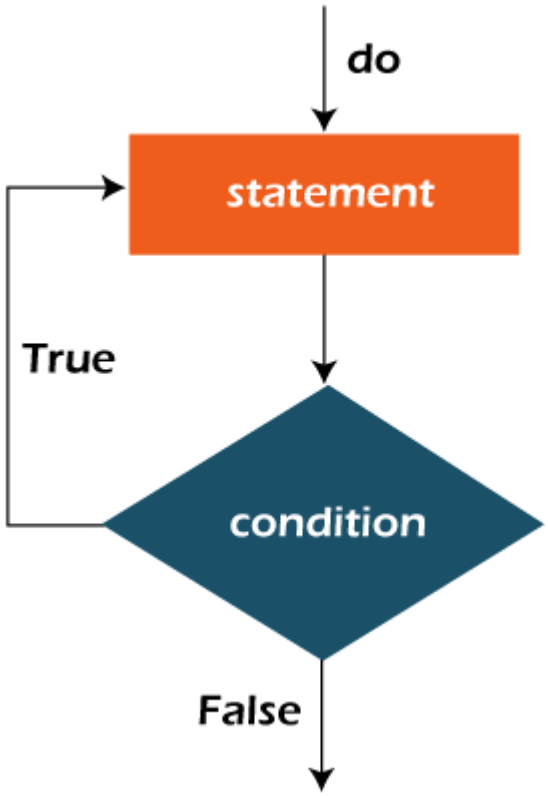
Syntax:

```
1. do{
2. //code to be executed / loop body
3. //update statement
4. }while (condition);
```

Flowchart of do-while loop:

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java



DoWhileExample.java

```
1. public class DoWhileExample {
2. public static void main(String[] args) {
3.     int i=1;
4.     do{
5.         System.out.println(i);
6.     i++;
7.     }while(i<=10);
8. }
9. }
```

Test it Now

Output:

```
1
2
3
4
5
```

Unit-1 Part-1 Object Oriented Programming with Java

```
6
7
8
9
10
```

Java for Loop vs while Loop vs do-while Loop

Comparison	for loop	while loop	do-while loop
Introduction	The Java for loop is a control flow statement that iterates a part of the programs multiple times.	The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.	The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.
When to use	If the number of iteration is fixed, it is recommended to use for loop.	If the number of iteration is not fixed, it is recommended to use while loop.	If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop.

Syntax	for(init;condition;incr/decr){
	// code to be executed
	}
		while(condition){
		//code to be executed
		}
			do{
			//code to be executed
			}while(condition);

Example	//for loop
	for(int i=1;i<=10;i++){
	System.out.println(i);
	}
		//while loop
		int i=1;
		while(i<=10){
		System.out.println(i);
		i++;
		}
			//do-while loop
			int i=1;
			do{
			System.out.println(i);
			i++;
			}while(i<=10);

Syntax for infinitive loop	for(;;){
	//code to be executed
	}
		while(true){
		//code to be executed
		}
			do{
			//code to be executed
			}while(true);

Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

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.

.

Syntax:

```
1. jump-statement;
2. break;
```

Java Break Statement with Loop

BreakExample.java

```
1. //Java Program to demonstrate the use of break statement
2. //inside the for loop.
3. public class BreakExample {
4. public static void main(String[] args) {
5.     //using for loop
6.     for(int i=1;i<=10;i++){
7.         if(i==5){
8.           //breaking the loop
9.            break;
10.       }
11.       System.out.println(i);
12.    }
13. }
14. }
```

Output:

```
1
2
3
```

4

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.

Syntax:

```
1. jump-statement;
2. continue;
```

Java Continue Statement Example

ContinueExample.java

```
1. //Java Program to demonstrate the use of continue statement
2. //inside the for loop.
3. public class ContinueExample {
4. public static void main(String[] args) {
5.     //for loop
6.     for(int i=1;i<=10;i++){
7.         if(i==5){
8.           //using continue statement
9.            continue;//it will skip the rest statement
10.       }
11.       System.out.println(i);
12.    }
13. }
14. }
```

Test it Now

Output:

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


Faculty: SHAHRUKH KAMAL
Shahrukhkamal7@gmail.com

Unit-1 Part-1 Object Oriented Programming with Java

```
10
```