JAVA PROGRAMMING

Unit – I

Introduction to JAVA

Topic : **Internet origin and development**

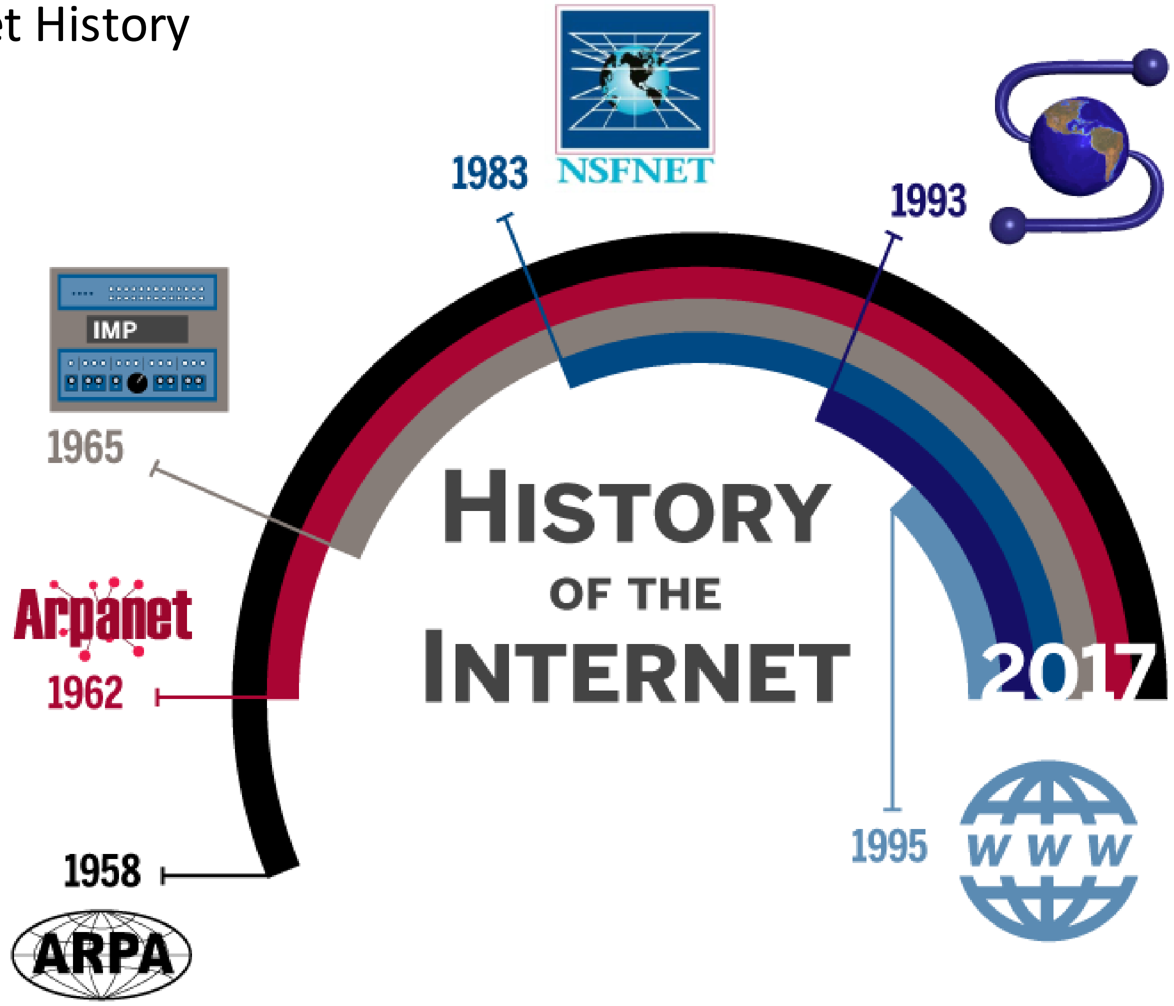
- **What is Internet?**

  A means of connecting a computer to any other computer anywhere in the world via dedicated routers and servers

- **Is internet and WWW are same?**

  The internet is a huge network of computers all connected together. The world wide web (‘www’ or ‘web’ for short) is a collection of webpages found on this network of computers. Your web browser uses the internet to access the web.

- Internet History



JAVA PROGRAMMING

- The US military was concerned about the Soviet Union attacking from space and destroying the US long-distance communications network.
- February 7, 1958 US Defense launched the Advanced Research Projects Agency (ARPA), now known as the Defense Advanced Research Projects Agency (DARPA).
- The existing national defense network relied on telephone lines and wires that were susceptible to damage. In 1962, J.C.R. Licklider, a scientist from ARPA and MIT, suggested connecting computers to keep a communications network active in the US in the event of a nuclear attack. This network came to be known as the ARPA Network, or ARPAnet.
- Packet switching made data transmission possible in 1965, and by 1969, military contractor developed an early form of routing devices known as Interface Message Processors (Imps), which revolutionized data transmission.

- ARPAnet adopted the transmission control protocol (TCP)in1983 and separated out the military network (MILnet), assigning a subset for public research. Launched formally as the National Science Foundation Network (NSFNET) in 1985, engineers designed it to connect university computer science departments across the US.

- 1989 saw a major step forward in internet communications. Tim Berners-Lee of the European Organization for Nuclear Research (CERN) created the hypertext transfer protocol (http), a standardization that gave diverse computer platforms the ability to access the same internet sites. For this reason, Berners-Lee is widely regarded as the father of the world wide web (www).

- The Mosaic web browser, created in 1993 at the National Center for Supercomputing Applications (NCSA).

Evolution Of Java

- The development of each programming language is based on a fact: there is a need to solve a problem that was not resolved by previous programming languages.
- Programming languages such as Cobol, Fortran do not have structural principles. They use the Goto statement to control the flow of the program.
- Therefore, C was invented in 1970, to replace the assembly language and to create a structured, effective and high-level language. When we write a program in C, it has a limit, such as a maximum of 25000 lines of code.
- C++ came with object-oriented programming features. C++ is the extension of C language which has been used extensively.
- C ++ with OOP became quite famous but then a new problem arose, to control the software on different machines, a separate compiler is required for that CPU. But building a C++ compiler was quite expensive. Therefore, an efficient and easy solution was needed, and this requirement became the reason for the creation of Java, which is a portable and platform-independent language.

History of Java

- James Gosling, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
- Originally designed for small, embedded systems in electronic appliances like set-top boxes.
- Firstly, it was called **"Greentalk"** by James Gosling, and file extension was .gt. After that, it was called **Oak** and was developed as a part of the Green project.

Why Java named "Oak"?

- **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania, etc.
- In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Why Java Programming named "Java"?

- Java is an island of Indonesia where first coffee was produced .
- Java is slang word of coffee(in America) and team members love drinking coffee.
- Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

- Features of Java

  The features of Java are also known as java *buzzwords*.

  - ➢Simple
  - ➢Object-Oriented
  - ➢Portable
  - ➢Platform independent
  - ➢Secured
  - ➢Robust
  - ➢Interpreted
  - ➢Multithreaded
  - ➢Distributed

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand

- 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

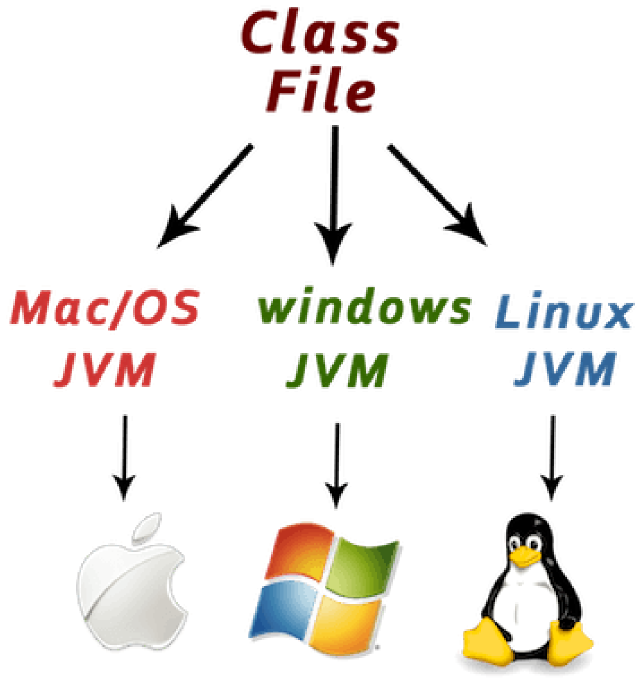
Java is an Object-oriented programming language. Everything in Java is an object.

- Basic concepts of OOPs are:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

Platform Independent

Java code can be run 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).



Robust

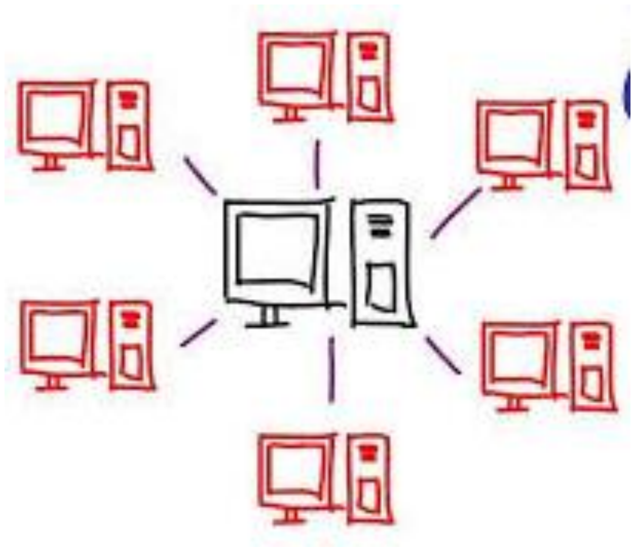
- Java is robust because:
  - It uses strong memory management.
  - There is a lack of pointers that avoids security problems.
  - There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.

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
  - Every time when a user compiles the Java program, the Java compiler creates a class file with Bytecode, which are tested by the JVM at the time of program execution for viruses and other malicious files.

Distributed

- Java is distributed because it facilitates users to create distributed applications in Java.
- RMI((**Remote Method Invocation**) and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.



Multi-threaded

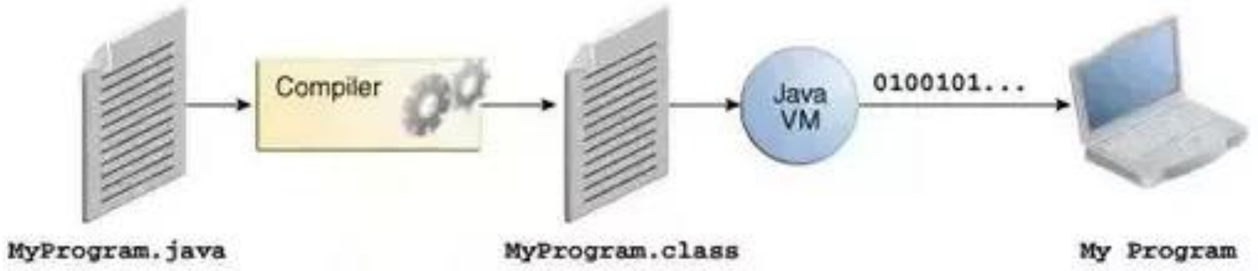
- A thread is like a separate program, executing concurrently. The main advantage of multi-threading is that it doesn't occupy memory for each thread.
- It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Portable

- Java is portable because it facilitates you to carry the Java bytecode to any platform.
- It doesn't require any implementation

Interpreted

- Typical Java environment consists of two programs: Java compiler and Java Virtual Machine.
- Java compiler takes the source code written in Java programming language, into class files containing Java byte code.
- The Java Virtual Machine takes the byte code prepared by the Java compiler and executes it. The byte-code itself is platform-independent, it the the responsibility of the Java Virtual Machine implementation to execute the program in the bytecode form on the real computer.



Difference between JAVA and C

C	JAVA
C is a Procedural Programming Language.	Java is Object-Oriented language.
In C declaration of variables is at the beginning of the block.	We can declare variables anywhere.
C does not supports Threading.	Java supports the concept of threading.
C supports pointers.	Java does not supports pointers.
C is platform dependent.	Java is a platform independent.
It follows a top-down approach.	Java follows a bottom-up approach.
C does not supports OOPS concept.	Java supports OOPS concept
C is a compiled language	Java is an Interpreted language
C is a low-level language	JAVA is a high-level language

Difference between JAVA and C++

C++ Programming	Java Programming
It support operator overloading.	It does not support operator overloading.
It support has template classes.	It does not have template classes as in java.
It supports multiple inheritances of classes.	It does not support multiple inheritances of classes. This is accomplished using a new feature called “Interface”.
It supports global variables.	It does not support global variables. Every variable and method is declared within classes and forms part of that class.
It supports pointers.	It does not use pointers.
It does not support destructor function with a finalize() function.	It has replaced the destructor function with a finalize() function.
There are header files in C++.	There are no header files in Java.

Web Browsers

- The browser application retrieves or fetches code, usually written in HTML (HyperText Markup Language) and/or another language, from a web server, interprets this code, and renders (displays) it as a Web page for you to view, on the computer or another Internet-enabled device that supports a browser.

An example of Web Browsers:

• Hot Java	Sun Microsystems
• Netscape Navigator	Netscape
• Internet Explorer	Microsoft
• Google Chrome	Google

Hardware Requirements

– Processor	- Any latest Processor(like Intel, AMD)
– RAM	- 4 gig RAM,
– Harddisk-space	- 10 gig disk space.

Software Requirements

– Operating System	- Any (Like Windows, Linux)
– JDK	- 1.5 version or heigher
– Editor	- Eclipse or Netbeans

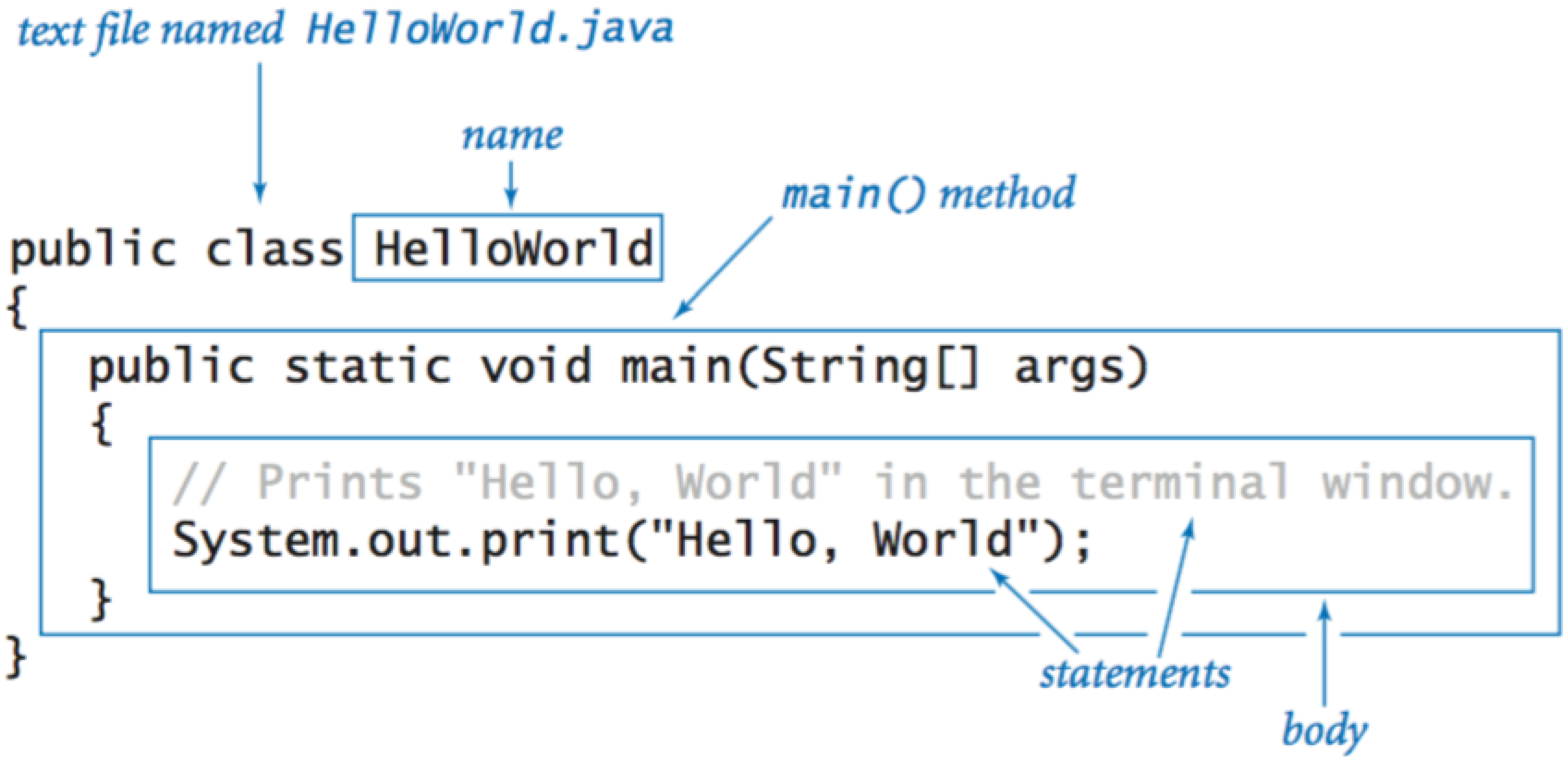
JDK Link: https://www.oracle.com/java/technologies/java-archive-javase5-downloads.html

Eclipse Link : https://www.eclipse.org/downloads/

Java Environment

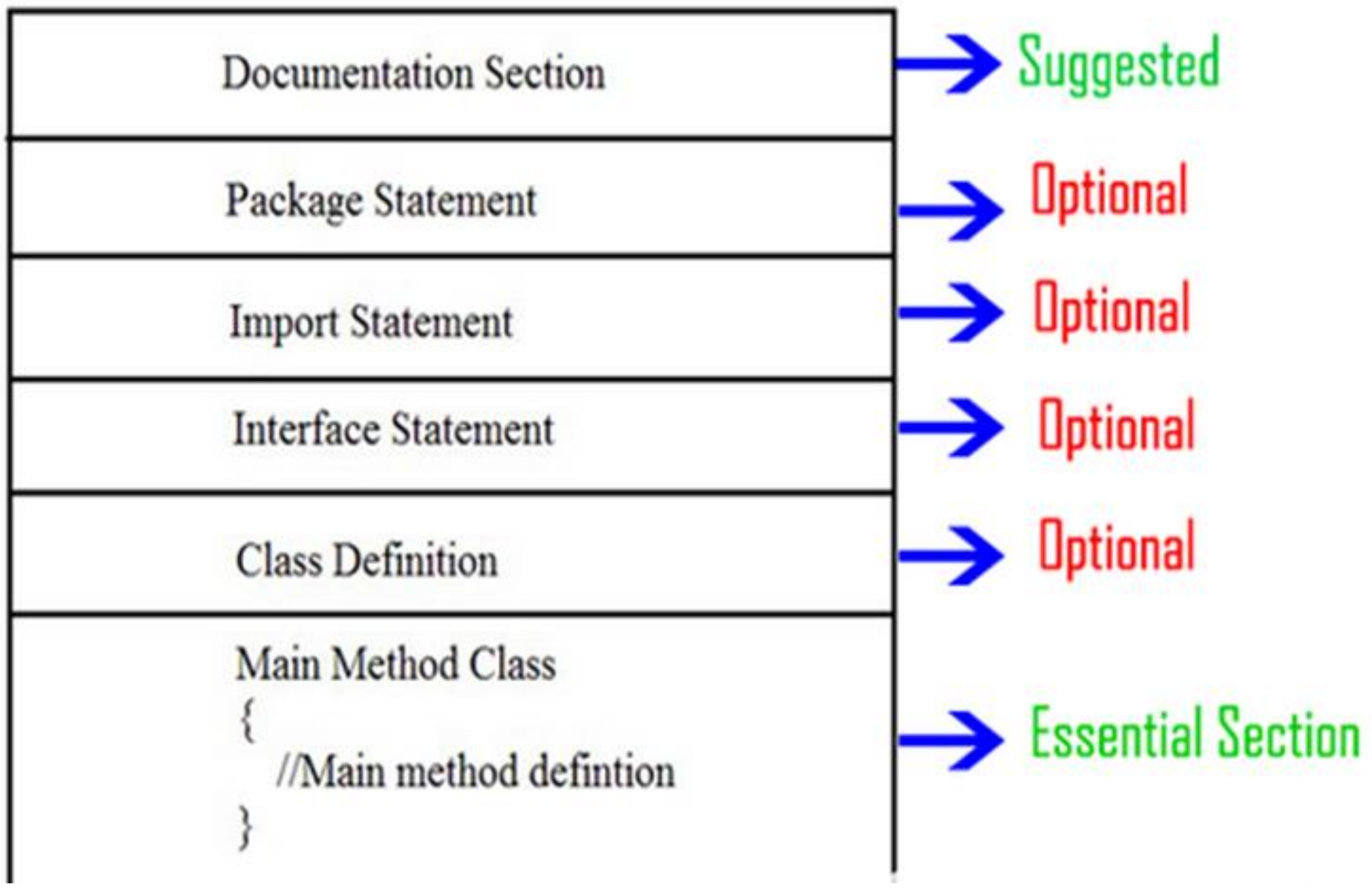
- From the desktop, right click the **Computer** icon.
- Choose **Properties** from the context menu.
- Click the **Advanced system settings** link.
- Click **Environment Variables**. In the section **System Variables**, find the PATH environment variable and select it. Click **Edit**. If the PATH environment variable does not exist, click New.
- In the **Edit System Variable** (or **New System Variable**) window, specify the value of the PATH environment variable. Click **OK**. Close all remaining windows by clicking **OK**.
- Reopen Command prompt window, and run your java code.

Simple Java Program



```
public class HelloWorld
{
    public static void main(String[] args)
    {
        // Prints "Hello, World" in the terminal window.
        System.out.print("Hello, World");
    }
}
```

Java Program Structure

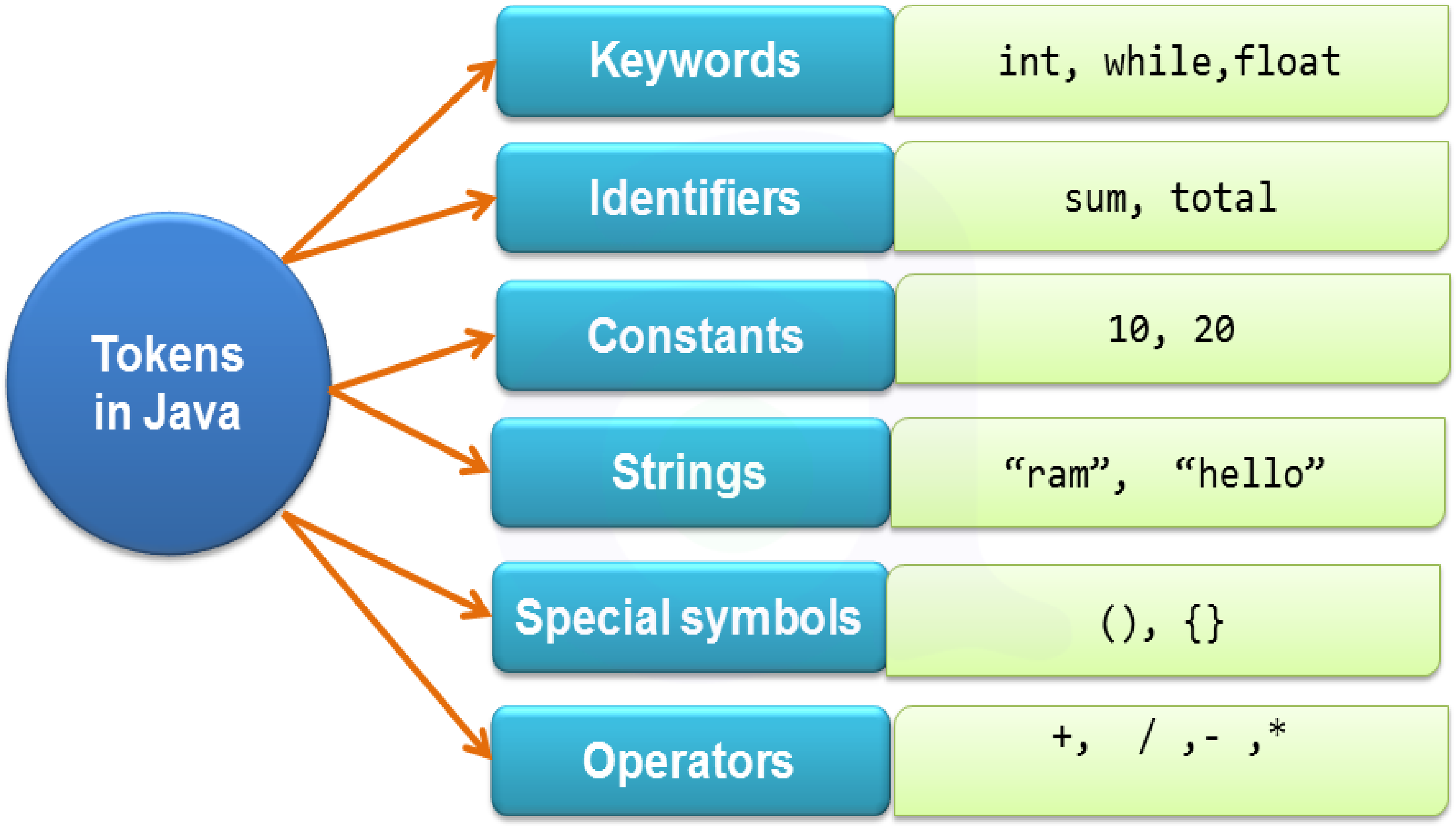


Two Classes Java Program

```
public class Computer
{
        void computer_method()
      {
         System.out.println("Power gone! Shut down your PC soon...");
       }
       public static void main(String[] args)
      {
         Computer c = new Computer();
         Laptop l = new Laptop();
         c.computer_method();
         l.laptop_method();
       }
}
class Laptop
 {
        void laptop_method()
      {
         System.out.println("99% Battery available.");
       }
}
```

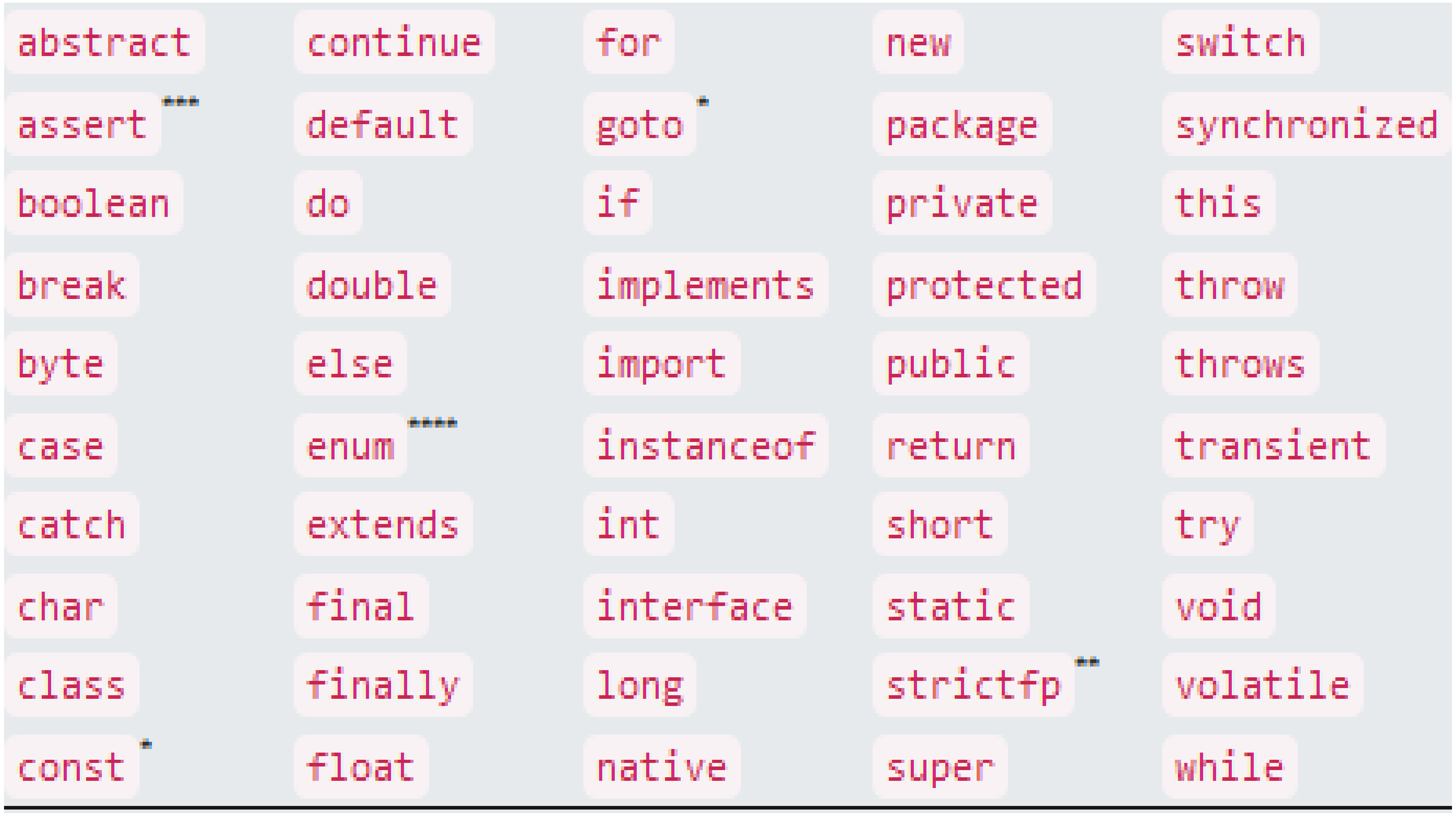
JAVA PROGRAMMING

Java program is a collection of different types of tokens, comments, and white spaces.



JAVA PROGRAMMING

Keywords



abstract	continue	for	new	switch
assert ***	default	goto *	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum ****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp **	volatile
const *	float	native	super	while

Identifiers

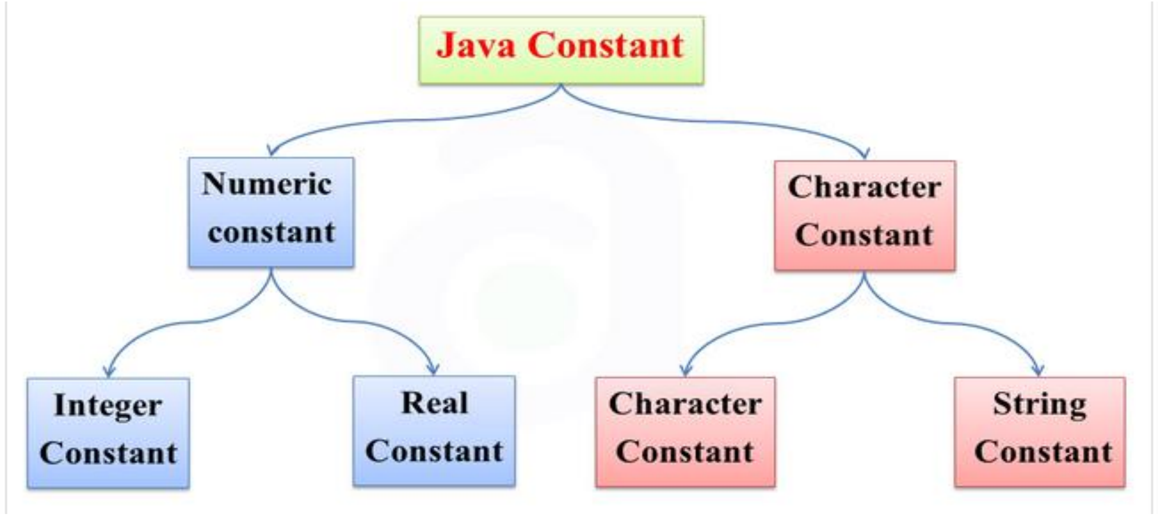
- Identifiers are the names of variables, methods, classes, packages and interfaces

Identifier must follow some rules. Here are the rules:

- All identifiers must start with either a letter(a to z or A to Z) or currency character($) or an underscore.
- They must not begin with a digit.
- Identifiers in Java are case sensitive, foo and Foo are two different identifiers.

Constants

- Constants in java are fixed values those are not changed during the Execution of program java supports several types of Constants those are:
  - Integer Constants
  - Real Constants
  - Single Character Constants
  - String Constants



Constants in Java are used when a **'static'** value or a permanent value for a variable has to be implemented. Java doesn't directly support constants. To make any variable a constant, we must use 'static' and 'final' modifiers in the following manner:

Syntax to assign a constant value in java:

static final datatype identifier_name = constant;

Example : static final int a=20;

- The **static modifier** causes the variable to be available without an instance of it's defining class being loaded
- The **final modifier** makes the variable unchangeable

String

In Java, string is basically an object that represents sequence of char values.

How to create a string object?

There are two ways to create String object:

- By string literal
  - String name=”Priya”;
- By new keyword
  - String s=new String(“Priya”);

```
public class StringExample{
public static void main(String args[]){
String s1="java";//creating string by java string literal

String s3=new String("example");System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

Special Symbol

Escape Sequences

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\\\	Insert a backslash character in the text at this point.

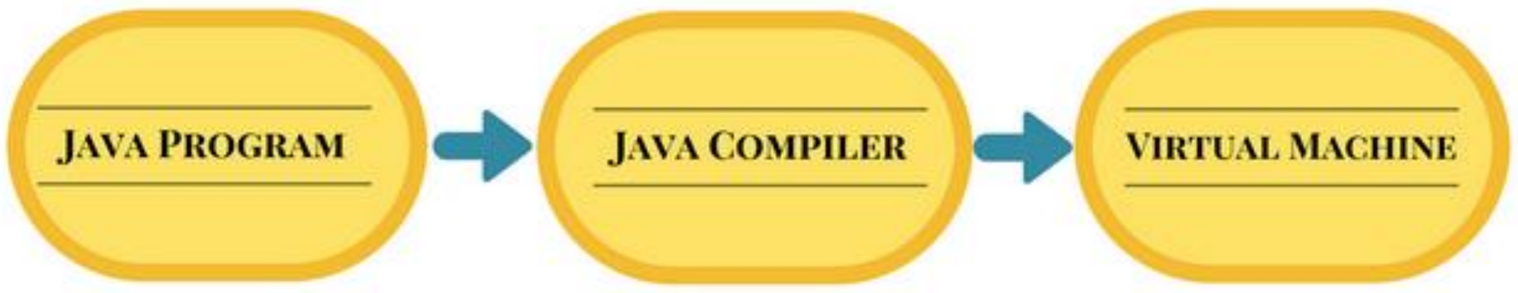
Java statements

are instructions that tell the programming language what to do. A basic statement, like an **assignment statement**, assigns a value to a variable.

JVM

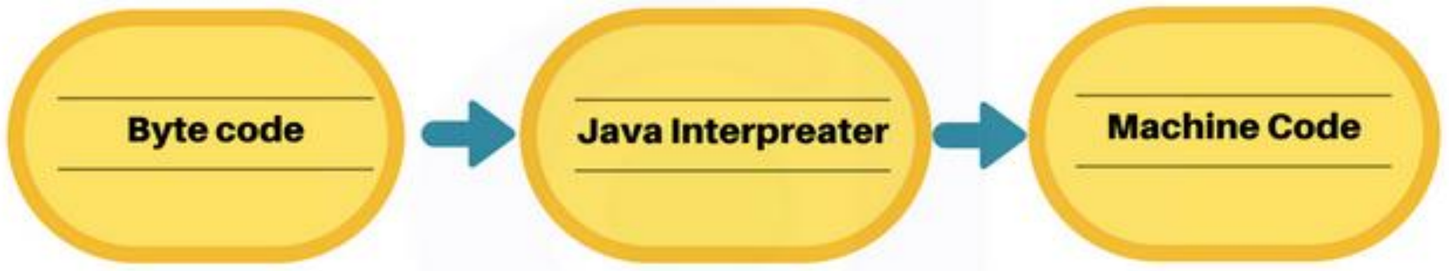
- **Java Virtual Machine (JVM)** is a engine that provides runtime environment to drive the Java Code or applications.
- It converts Java bytecode into machines language. JVM is a part of Java Run Environment (JRE).

JAVA PROGRAMMING



Source Code

Byte Code



Virtual Machine

Real Machine

Command Line Arguments

- Command Line Argument is the parameters or arguments passed to the program when you run the program.
- The passed parameters is stored as a string array in the main method.

Command Line Arguments

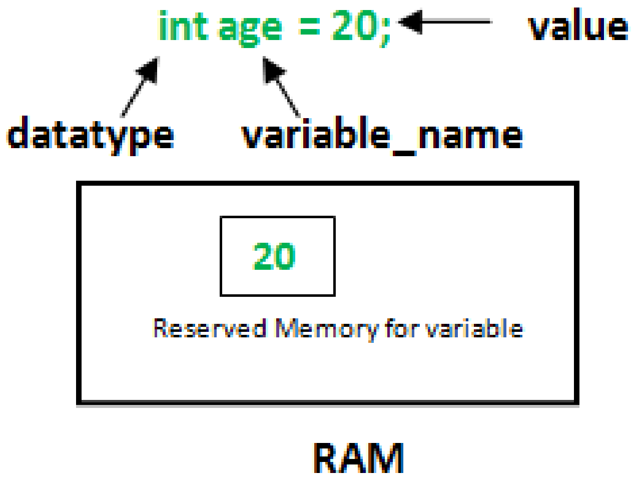
```
class ComLineEx
{
  public static void main (String args[])
     {
               for(int i=0;i<args.length;i++)
                              {
                                        System.out.println("Java is : "+args[i]);
                              }
     }
}
```

Output:

```
C:\jdk1.8.0_131\bin>javac ComLineEx.java

C:\jdk1.8.0_131\bin>java ComLineEx Simple Object-Oriented Robust Secure

Java is : Simple
Java is : Object-Oriented
Java is : Robust
Java is : Secure
```

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.



Rules for naming variables:

- They must not begin with a digit.
- Uppercase and Lowercase are distinct. This means that the variable Total is not the same as total or TOTAL.
- It should not be keyword.
- White spaces is not allowed.
- Variables name can be of any length.

Types of variables

There are three types of variables in Java:

- Local Variables
- Instance Variables
- Static Variables

Local Variables: A variable defined within a block or method or constructor is called local variable.

- These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function.
- The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block.

```
public class StudentDetails {
    public void StudentAge()
    {
        // local variable age
        int age = 0;
        age = age + 5;
        System.out.println("Student age is : " + age);
    }

    public static void main(String args[])
    {
        StudentDetails obj = new StudentDetails();
        obj.StudentAge();
    }
}
```

Instance(Gloable) Variables:

Instance variables are non-static variables and are declared in a class outside any method, constructor or block.

```
public class Add {
    int a=20, b=30, c; // globle variable age
    public void sum()
    {
            c=a+b;
            System.out.println("Sum  is : " + c);
    }

    public static void main(String args[])
    {
             Add obj = new Add();
            obj.sum();
    }
}
```

Static Variables:

- Static variables are also known as Class variables.
- These variables are declared similarly as instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.
- To access static variables, we need not create an object of that class, we can simply access the variable as
- class_name.variable_name;Sample Program:

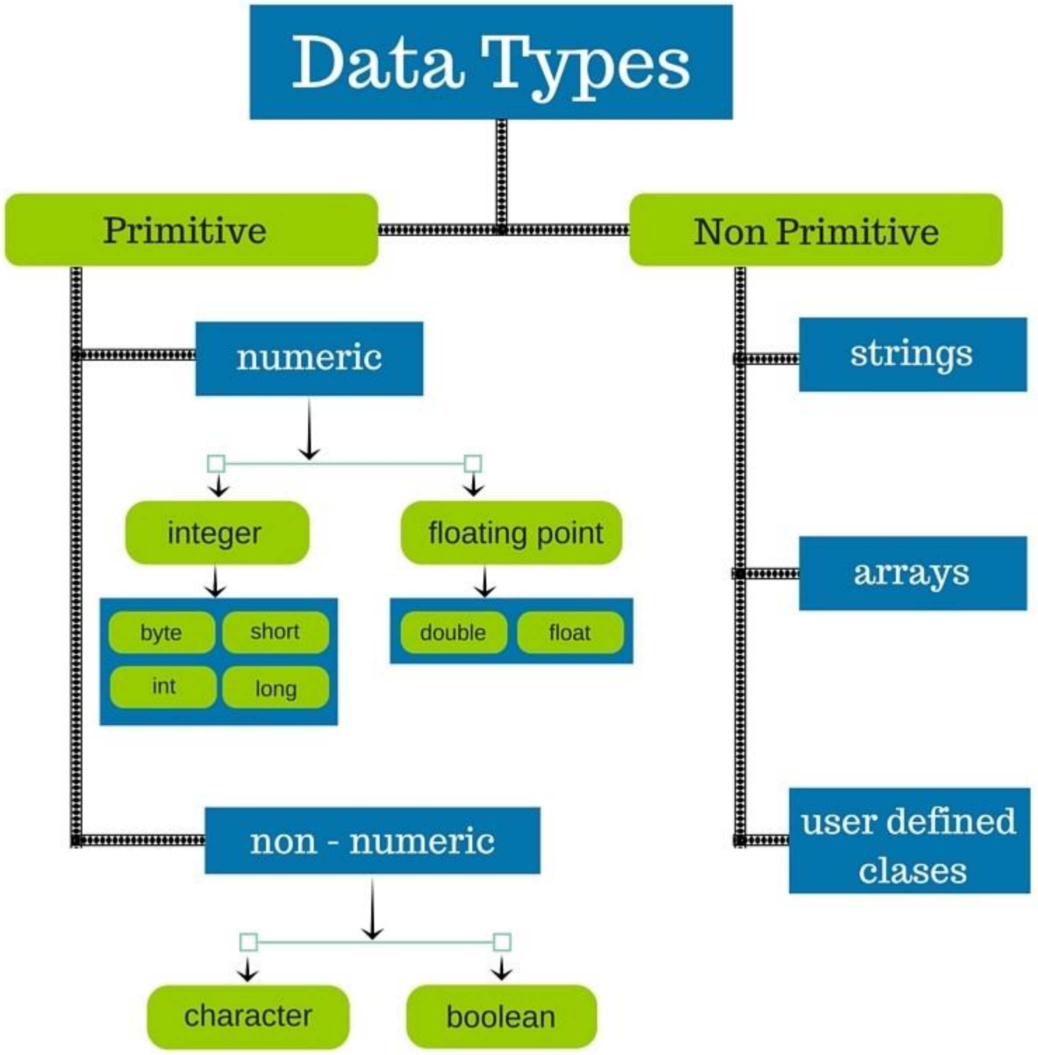
JAVA PROGRAMMING

```
import java.io.*;
class Emp {

    // static variable salary
    public static double salary;
    public static String name = "Harsh";
}

public class EmpDemo {
    public static void main(String args[])
    {

        // accessing static variable without object
        Emp.salary = 1000;
        System.out.println(Emp.name + "'s average salary:"
                                        + Emp.salary);
    }
}
```

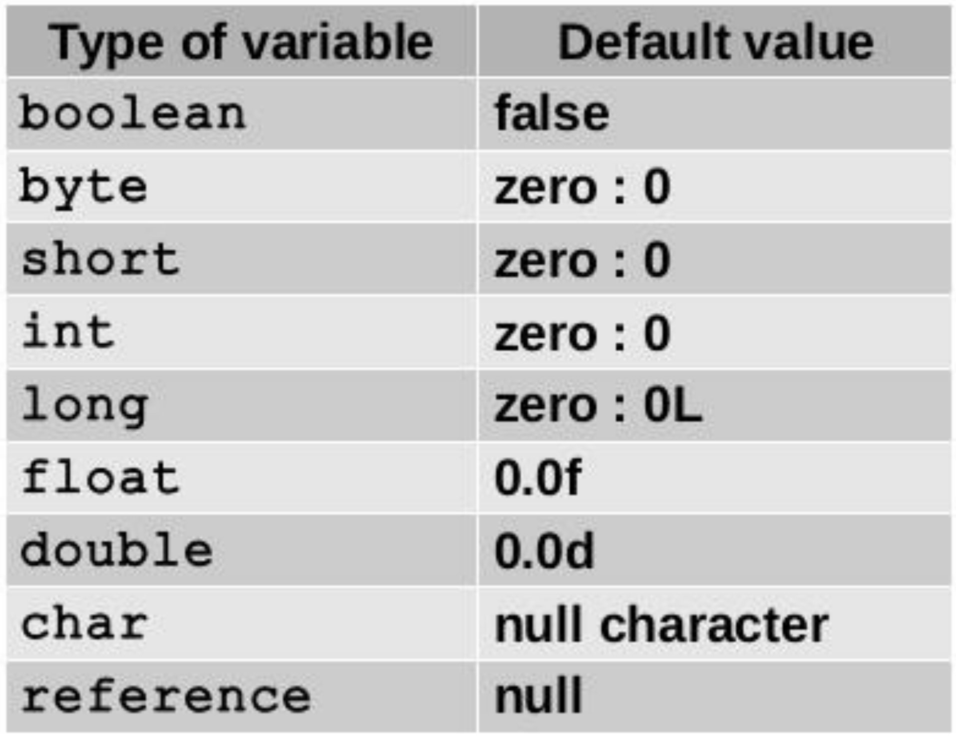
Data Types

Java Data types are classified into 2 type

- **Primitive Data Types:** A primitive data type is pre-defined by the programming language..
- **Non-Primitive Data Types:** These data types are not actually defined by the programming language but are created by the programmer.


Data Types
Primitive
Non Primitive
numeric
integer
floating point
byte
short
int
long
double
float
non - numeric
character
boolean
strings
arrays
user defined clases

Standard default values



Type of variable	Default value
`boolean`	false
`byte`	zero : 0
`short`	zero : 0
`int`	zero : 0
`long`	zero : 0L
`float`	0.0f
`double`	0.0d
`char`	null character
`reference`	null

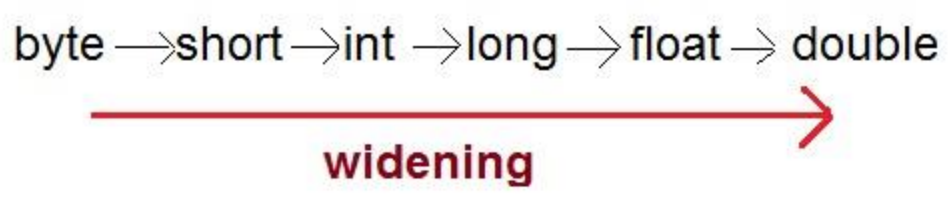
Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Type Casting

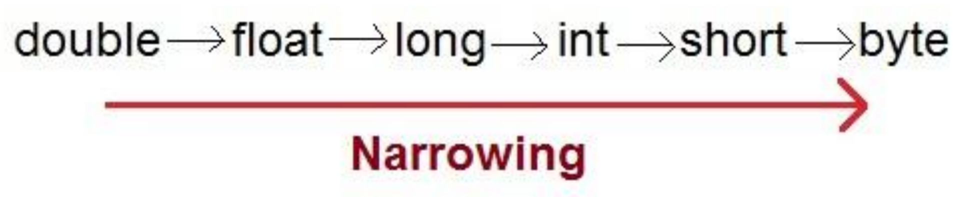
Casting is a process of changing one type value to another type. In Java, we can cast one type of value to another type. It is known as type casting.

In Java, type casting is classified into two types,

– Widening Casting(Implicit)



– Narrowing Casting(Explicitly done)



Type Casting

Converting a lower data type into a higher one is called **widening** type casting. It is also known as **implicit conversion** or **casting down**. It is done automatically. It is safe because there is no chance to lose data. It takes place when:

•Both data types must be compatible with each other.

•The target type must be larger than the source type.

```
public class Test
 {
     public static void main(String[] args)
     {
      int i = 100;
     long l = i;  //no explicit type casting required
     float f = l;  //no explicit type casting required
      System.out.println("Int value "+i);
     System.out.println("Long value "+l);
     System.out.println("Float value "+f);
     }
}
```

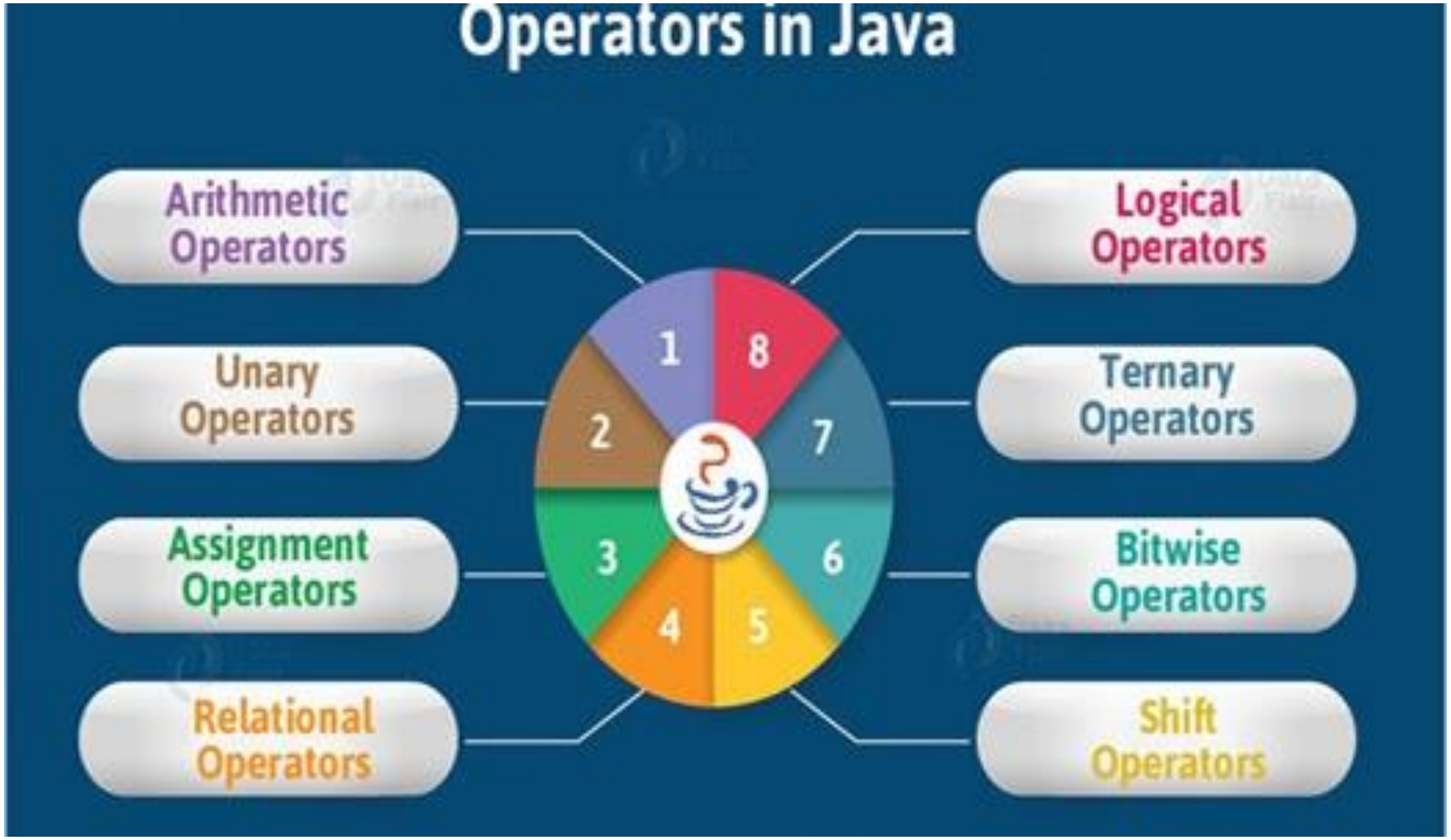
Narrowing or Explicit type conversion

When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.

If we don't perform casting then compiler reports compile time error.

```
public class Test
 {
    public static void main(String[] args)
    {
    double d = 100.04;
    long l = (long)d;
    int i = (int)l;
    System.out.println("Double value "+d);
    System.out.println("Long value "+l);
    System.out.println("Int value "+i);
    }
}
```

Operators are used to perform operations on variables and values



Assignment Operator

Assignment Operator (=) is used for Initializing a value to variable or we can say that an assignment operator is used for Assigning a value to the variable .

- Ex: int age;

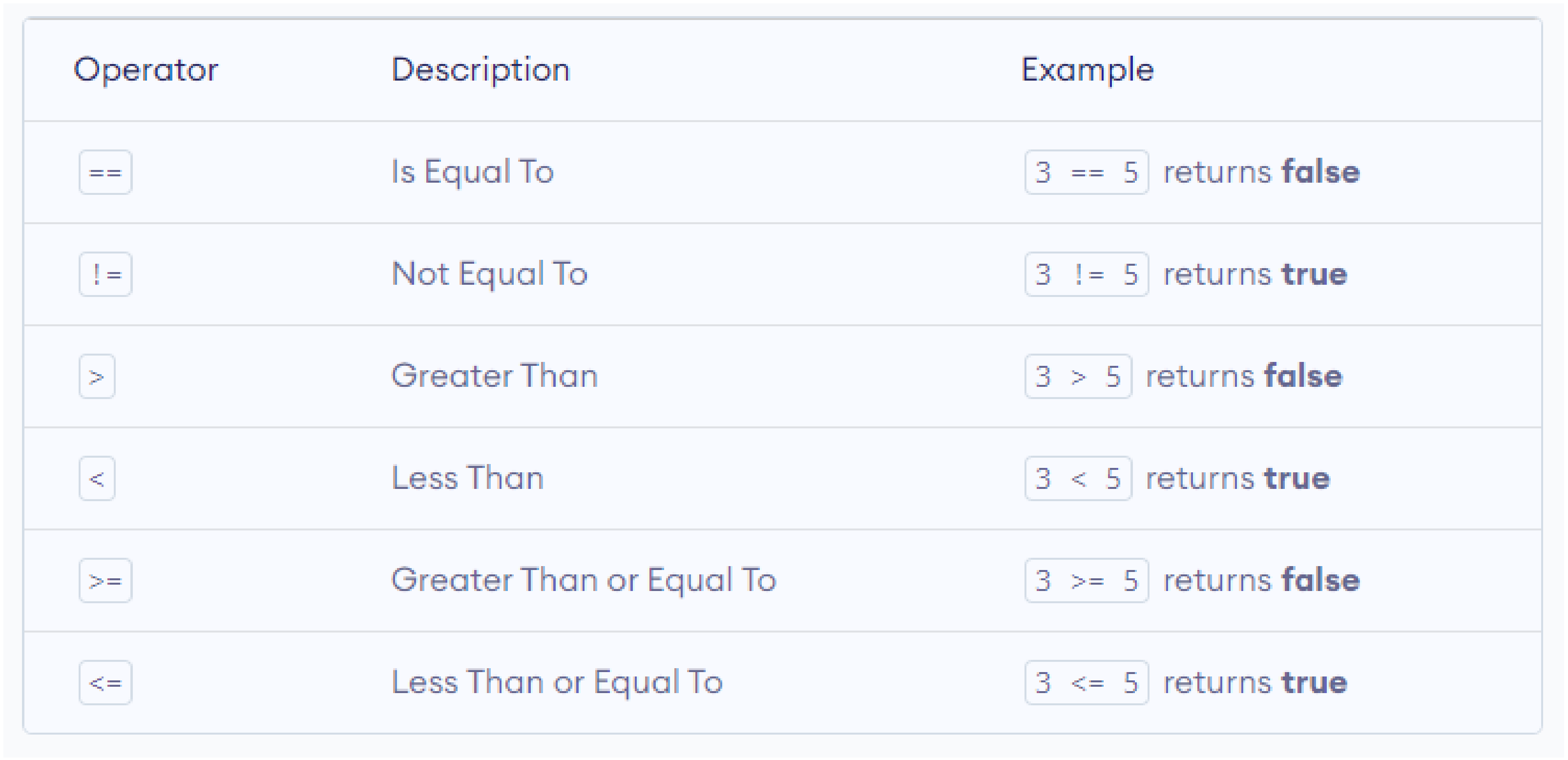
  age = 5;

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C – A
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A

```
class Main {
  public static void main(String[] args) {

    // create variables
    int a = 4;
    int var;

    // assign value using =
    var = a;
    System.out.println("var using =: " + var);

    // assign value using =+
    var += a;
    System.out.println("var using +=: " + var);

    // assign value using =*
    var *= a;
    System.out.println("var using *=: " + var);
  }
}
```

Relational Operators

The Relational Operators are used for Checking the Conditions or These operators are used for Controlling the Flow of the Execution to First Check the Condition it Contains.



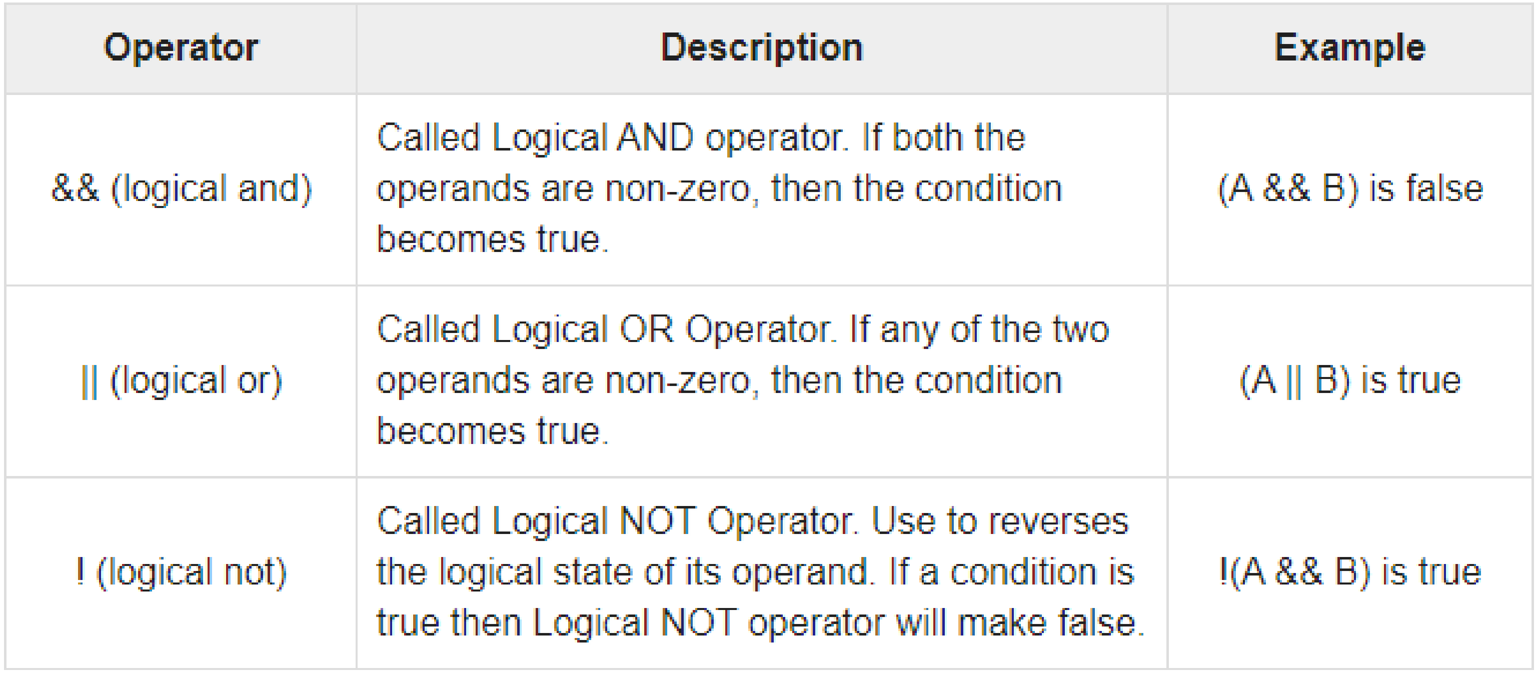
Operator	Description	Example
`==`	Is Equal To	`3 == 5` returns **false**
`!=`	Not Equal To	`3 != 5` returns **true**
`>`	Greater Than	`3 > 5` returns **false**
`<`	Less Than	`3 < 5` returns **true**
`>=`	Greater Than or Equal To	`3 >= 5` returns **false**
`<=`	Less Than or Equal To	`3 <= 5` returns **true**

```
class Main {
  public static void main(String[] args) {

    // create variables
    int a = 7, b = 11;

    // value of a and b
    System.out.println(“value of a  is " + a + " and value of  b is " + b);

    // == operator
    System.out.println(a == b);  // false

    // != operator
    System.out.println(a != b);  // true

    // > operator
    System.out.println(a > b);  // false

    // < operator
    System.out.println(a < b);  // true

    // >= operator
    System.out.println(a >= b);  // false

    // <= operator
    System.out.println(a <= b);  // true
  }
}
```

Logical Operators

These operators are used to perform logical "AND", "OR" and "NOT" operation, i.e. the function similar to AND gate and OR gate in digital electronics.

They are used to combine two or more conditions/constraints.



Operator	Description	Example
&& (logical and)	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false
|| (logical or)	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true
! (logical not)	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

```
int a = 10, b = 20, c = 20, d=0;
```

```
if ((a < b) && (b == c))
    {
        d = a + b + c;
        System.out.println("The sum is: " + d);
     }
     else
        System.out.println("False Conditions");
  }
```

Syntax of && :
condition1 && condition2

Syntax of ||
condition1 || condition2

```
int a = 10, b = 1, c = 10, d = 30;
```

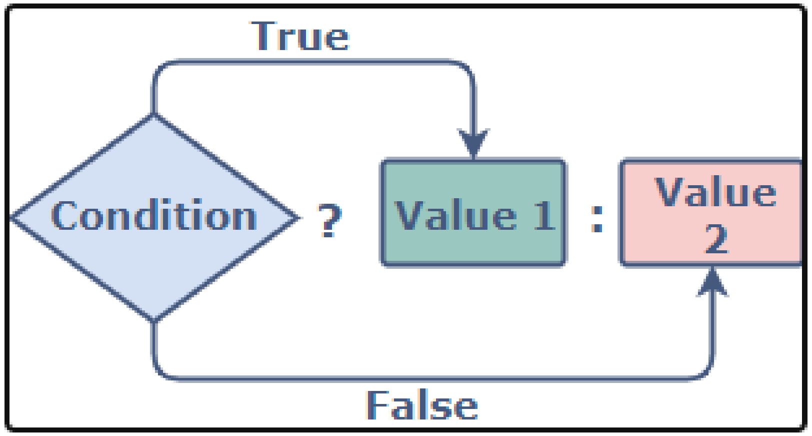
```
if (a > b || c == d)
         System.out.println("One or both the conditions are true");
      else
         System.out.println("Both the conditions are false");
  }
```

```
class Demo1 {
public static void main(String[] args)
{

int a=5,b=5;
if(a!=b)
   {
           System.out.println("true");
   }
    else
   {
           System.out.println("false");
    }
}
}
```

Syntax:
!(condition)

ternary operator

it is the only operator in Java consisting of *three operands*.



Syntax

The three operands in a ternary operator include:

- A boolean expression that evaluates to either true or false.
- A value to be assigned if the expression is evaluated to true.
- A value to be assigned if the expression is evaluated to false.

```
class CheckEvenNumber
{
     public static void main( String args[] )
    {
      int number = 3;
      String msg =  (number % 2 == 0) ? "The number is even!" : "The numbe
       r is odd!";
      System.out.println(msg);
     }
}
```

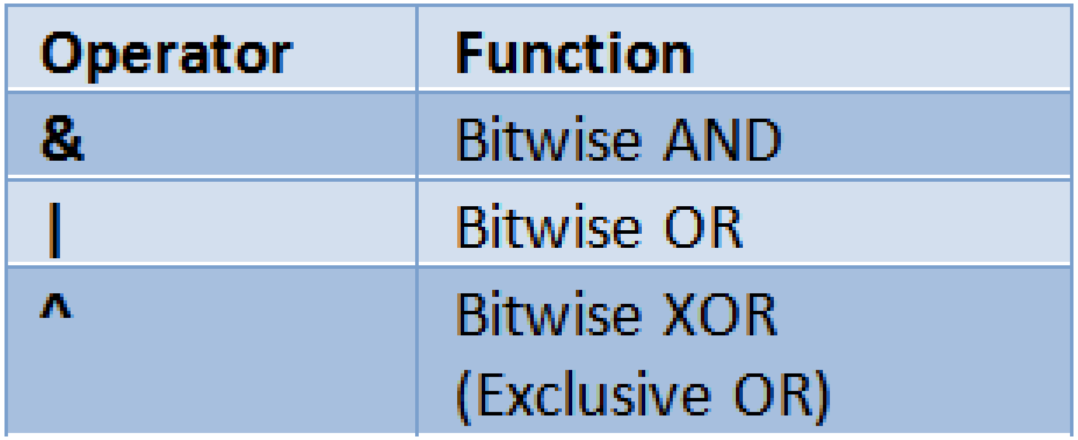
variable = Expression1 ? Expression2: Expression3

Output

```
The number is odd!
```

Bitwise operators

- Bitwise operators are used to perform manipulation of individual bits of a number.

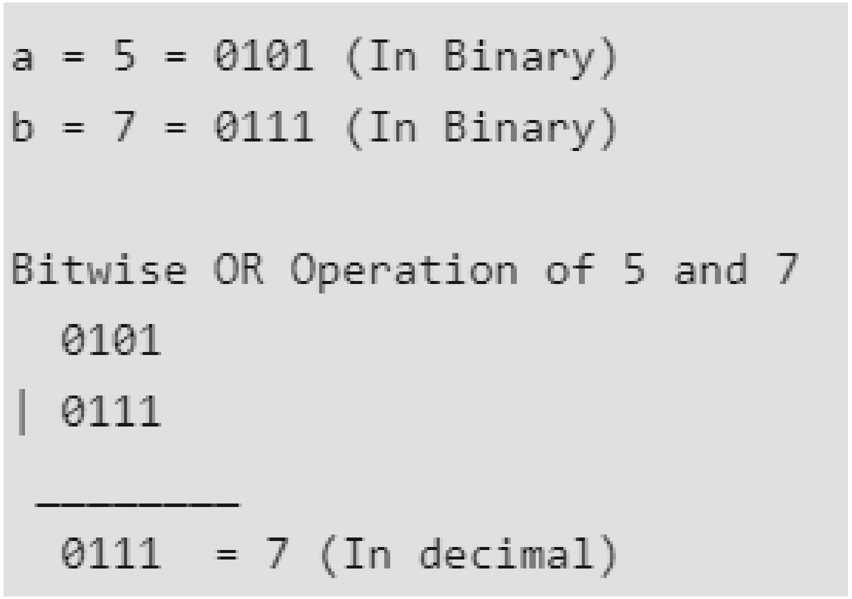


Operator	Function
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR (Exclusive OR)

Bitwise OR (|) –

This operator is binary operator, denoted by '|'. It returns bit by bit OR of input values, i.e, if either of the bits is 1, it gives 1, else it gives 0.

For example,



```
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise OR Operation of 5 and 7
  0101
| 0111
 ________
  0111  = 7 (In decimal)
```

Bitwise AND (&) –

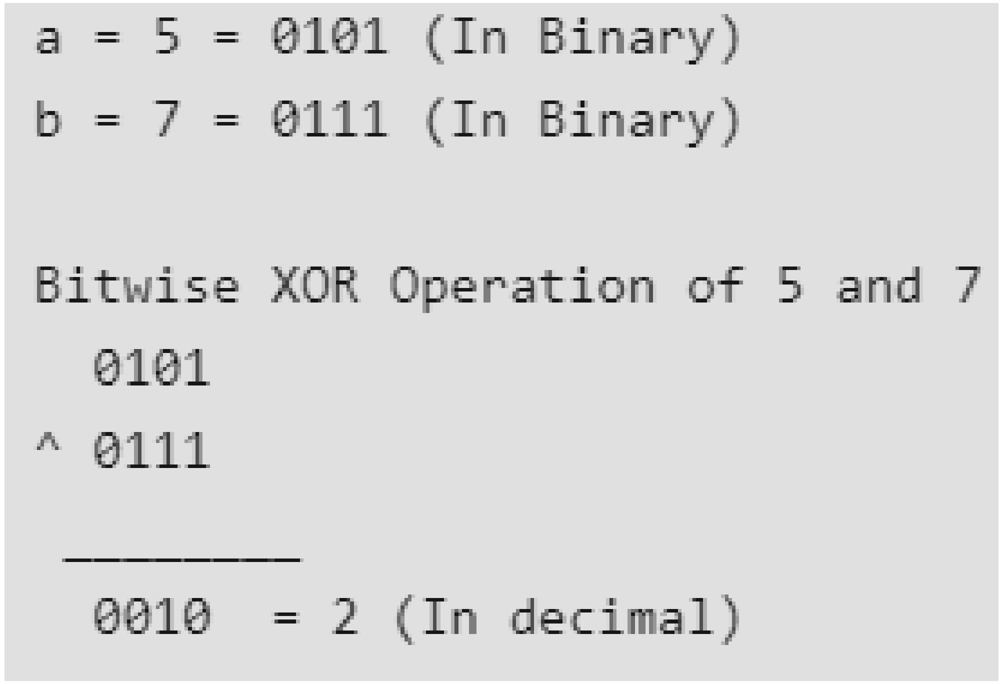
This operator is binary operator, denoted by '&'. It returns bit by bit AND of input values, i.e, if both bits are 1, it gives 1, else it gives 0.

```
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise AND Operation of 5 and 7
  0101
& 0111

 ________
  0101   = 5 (In decimal)
```

Bitwise XOR (^) –

This operator is binary operator, denoted by '^'. It returns bit by bit XOR of input values, i.e, if corresponding bits are different, it gives 1, else it gives 0.

For example



```
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise XOR Operation of 5 and 7
  0101
^ 0111
 ________
  0010  = 2 (In decimal)
```

```
// Java program to illustratebitwise operators
public class operators {
        public static void main(String[] args)
        {

                int a = 5;
                int b = 7;

                // bitwise and
                System.out.println("a&b = " + (a & b));

                // bitwise or
                System.out.println("a|b = " + (a | b));

                // bitwise xor
                System.out.println("a^b = " + (a ^ b));

                // bitwise not
                // ~0101=1010
                // will give  2's complement of 1010 = -6
                System.out.println("~a = " + ~a);
        } }
```

Arithmetic expressions

An arithmetic expression is a combination of variables, constants and operators arranged as per the syntax of the language.

Evaluation of Expression

Expression are evaluated using an assignment statement of the form

```
variable = expression;
```

Example

```
x=a*b-c;
```

Precedence of Arithmetic Operators

An arithmetic expression without any parentheses will be evaluated from left to right using the rules of precedence of operators.

There are two distinct priority levels of arithmetic operators in Java.

High priority * / %

Low priority + -

Example:

x = a – b / 3 + c * 2 - 1

When a=9,b=12 and c=3 the statement becomes

x = 9 – 12 / 3 + 3 * 2 – 1

Step1 : x = 9 – 4 + 3*2 – 1	(12/3)
Step2 : x = 9 – 4 + 6 – 1	(3 * 2)
Step3 : x = 5 + 6 – 1	(9 - 4)
Step4 : x = 11 - 1	(5+6)
Step5: x = 10	

Expression with parentheses assume the highest priority

Example

9 – 12 / (3 + 3) * (2 -1)

Stept1 : 9 - 12 / 6 * (2 -1)	(3 + 3)
Stept2 : 9 - 12 / 6 * 1	(2 -1)
Stept3 : 9 – 2 * 1	(12 / 6)
Stept4 : 9 – 2	(2 * 1)
Step 5 : 7	

Type Conversion

Type Conversion is that which converts the one data type into another for example

- converting a int into float
- converting a float into double

- The Type Conversion is that which automatically converts the one data type into another but remember we can't store a large data type into the small
  - for ex we can t store a float into int because a float is greater than int
- Type Conversion is Also Called as Promotion of data Because we are Converting one Lower data type into higher data type So this is Performed Automatically by java compilers
- Remember the type Conversion is performed by the compiler but a casting is done by the user

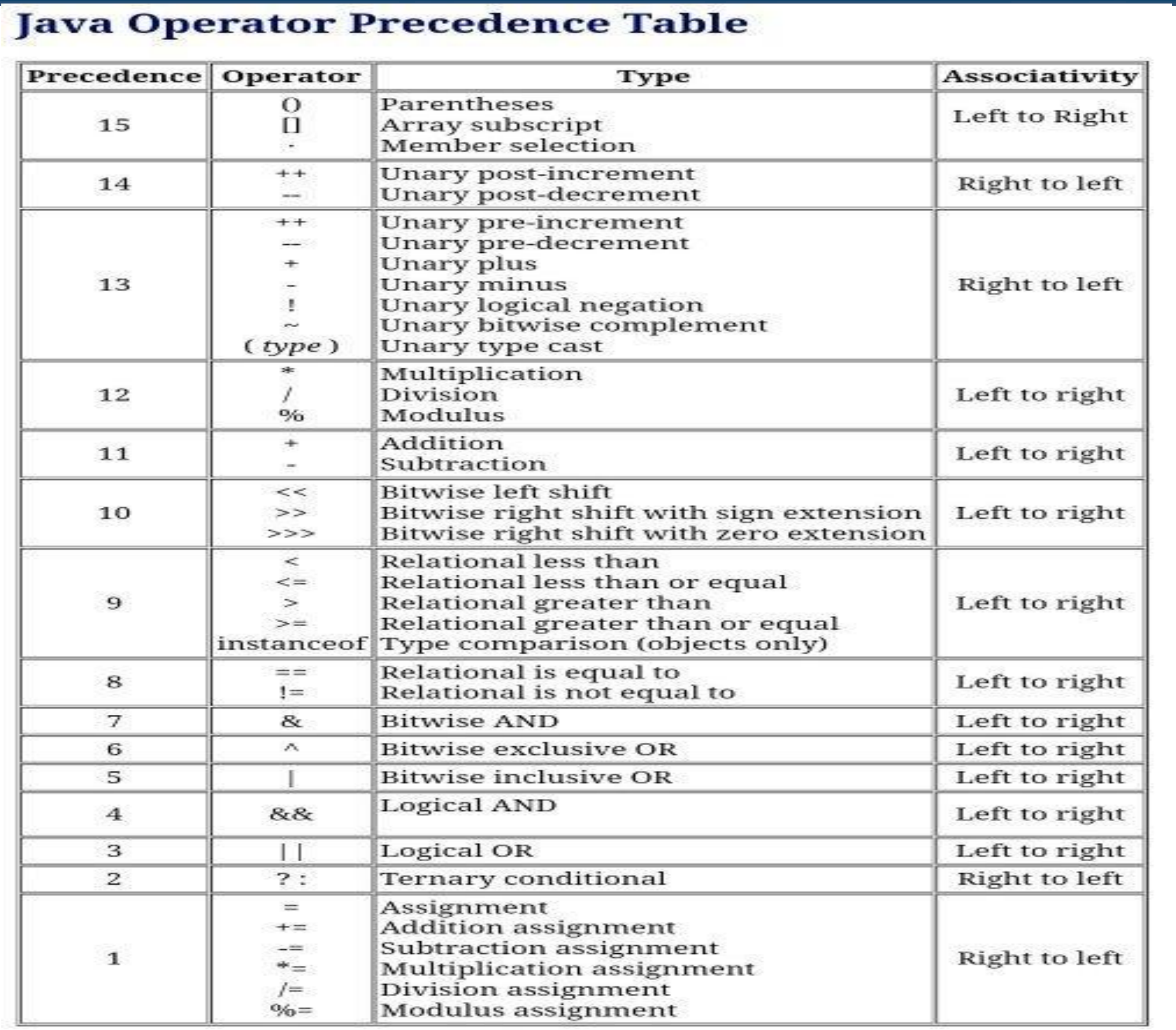
JAVA PROGRAMMING

Example:

```
class Test
{
    public static void main(String[] args)
    {
        int i = 100;

        //automatic type conversion
        long l = i;

        //automatic type conversion
        float f = l;
        System.out.println("Int value "+i);
        System.out.println("Long value "+l);
        System.out.println("Float value "+f);
    }
}
```

Output:

```
        Int value 100
         Long value 100
        Float value 100.0
```



Java Operator Precedence Table

Precedence	Operator	Type	Associativity
15	() [] ·	Parentheses Array subscript Member selection	Left to Right
14	++ --	Unary post-increment Unary post-decrement	Right to left
13	++ -- + - ! ~ (*type*)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
12	* / %	Multiplication Division Modulus	Left to right
11	+ -	Addition Subtraction	Left to right
10	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	? :	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

Mathematical Functions

- Mathematical functions such as cos,sqrt,log etc. are frequently used in analysis of real time problems.
- Java supports these basic math functions through Math class defined in java,lang package.
- Example
- `double y = Math.sqrt(x);`

- Math Class Methods include
  - abs(a) : Returns the absolute value of a.
  - ceil(a) : Returns the smallest whole number greater than a.
  - floor(a) : Returns the largest whole number less than a.
  - max(a , b) : Returns the larger of a and b.
  - min(a , b) : Returns the smaller of a and b.
  - pow(a , b) : Returns the number a raised to power b.
  - random() : Generates a random number less than or equal to 0.0 and less than 1.0 .
  - sqrt(a) : returns the square root of a.

```
public class JavaMathExample1
{
    public static void main(String[] args)
    {
        double x = 28;
        double y = 4;
                System.out.println("Maximum  number  of x and  y is: " +Math.max(x,  y));
                 System.out.println("Square  root of y is: " + Math.sqrt(y));
                 System.out.println("Power  of x and y is: " + Math.pow(x, y));
                 System.out.println("Logarithm  of x is: " + Math.log(x));
                System.out.println("Logarithm  of y is: " + Math.log(y));
               System.out.println(Math.PI);


         }
}
```

Decision Making in java

Java supports various selection statements, like if, if-else and switch . These statements allow us to control the flow of our program's execution based upon conditions known only during run time.

There are various types of if statement in java.

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

There are another types of selection or decision making statement in java.

- switch

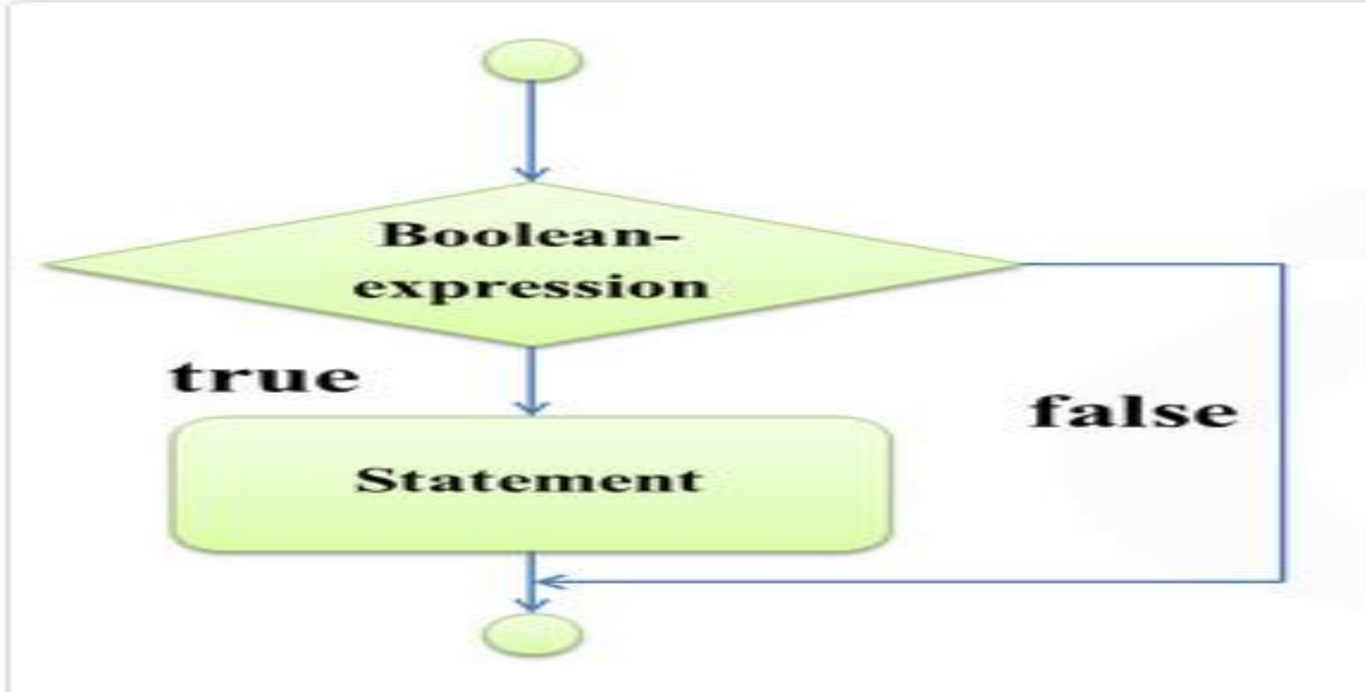
if statement

- The if statement is a conditional statement. It is used to test the condition.
- It tells your program to **execute a certain section of code only if a particular test evaluates to true**.
- It checks boolean condition: *true* or *false,* If **boolean Expression** evaluates to true, the statements in the block following the if statement is executed.
- If it evaluates to false, the statements in the **if the block is not executed**.

Syntax of if statement

```
if (boolean-expression)
      {
              statement(s);
      }
```

Flow Diageram



```
class IfStatementExample
      {
             public static void main(String[] args)
        {
             float radius, area, PI;
             PI=3.14f;
             radius = 2.1f ;
             if (radius >= 0)
             {
                          area = radius * radius * PI;
                          System.out.println("The area for the circle of radius "
    +radius                             +" is "+area);
             }
       }
  }
```

If-else statement

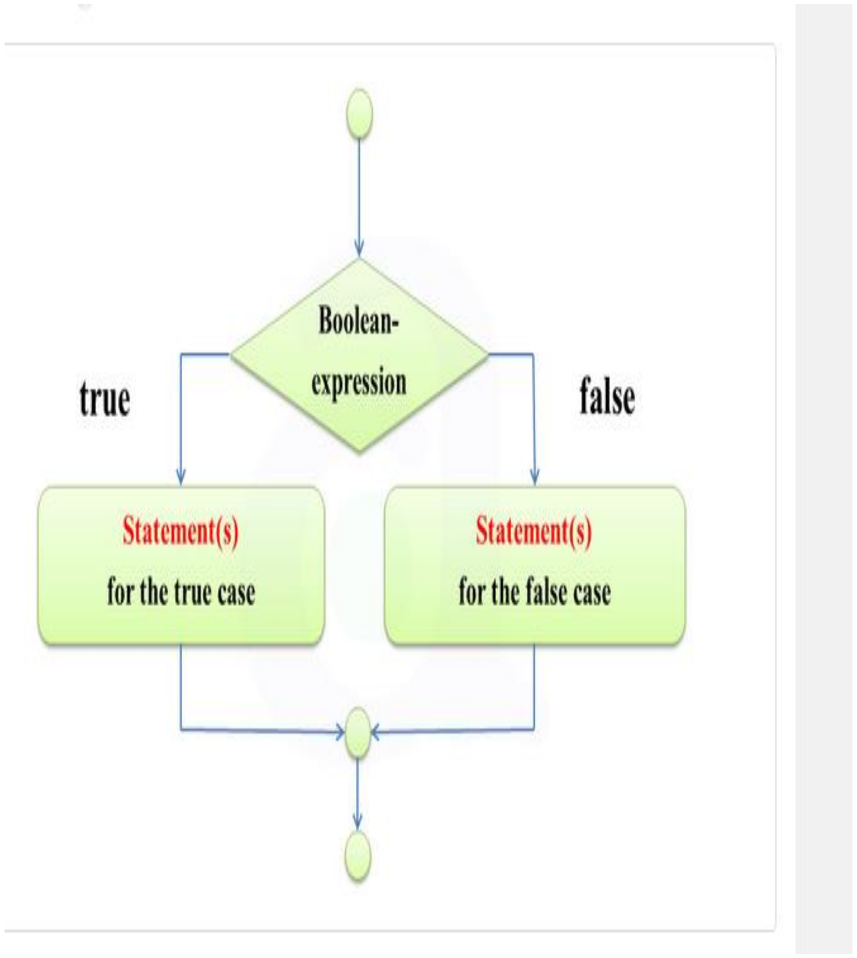
- If-else statement also tests the condition. It executes the if block if condition is true otherwise else block is executed.
- The *if else* statement is a conditional branch statement.

- It tells your program to **execute a certain section of code only if a particular test evaluates to true**.
- If **boolean Expression** evaluates to true, the statements in the block following the if statement is executed.

- If it evaluates to false, *else block* is executed.

- **Syntax if-else Statement**

```
if(condition)
        {

        statement1(s);
        }
 else
        {

        statement2(s);
        }
```


Boolean-expression
true
false
Statement(s) for the true case
Statement(s) for the false case

JAVA PROGRAMMING

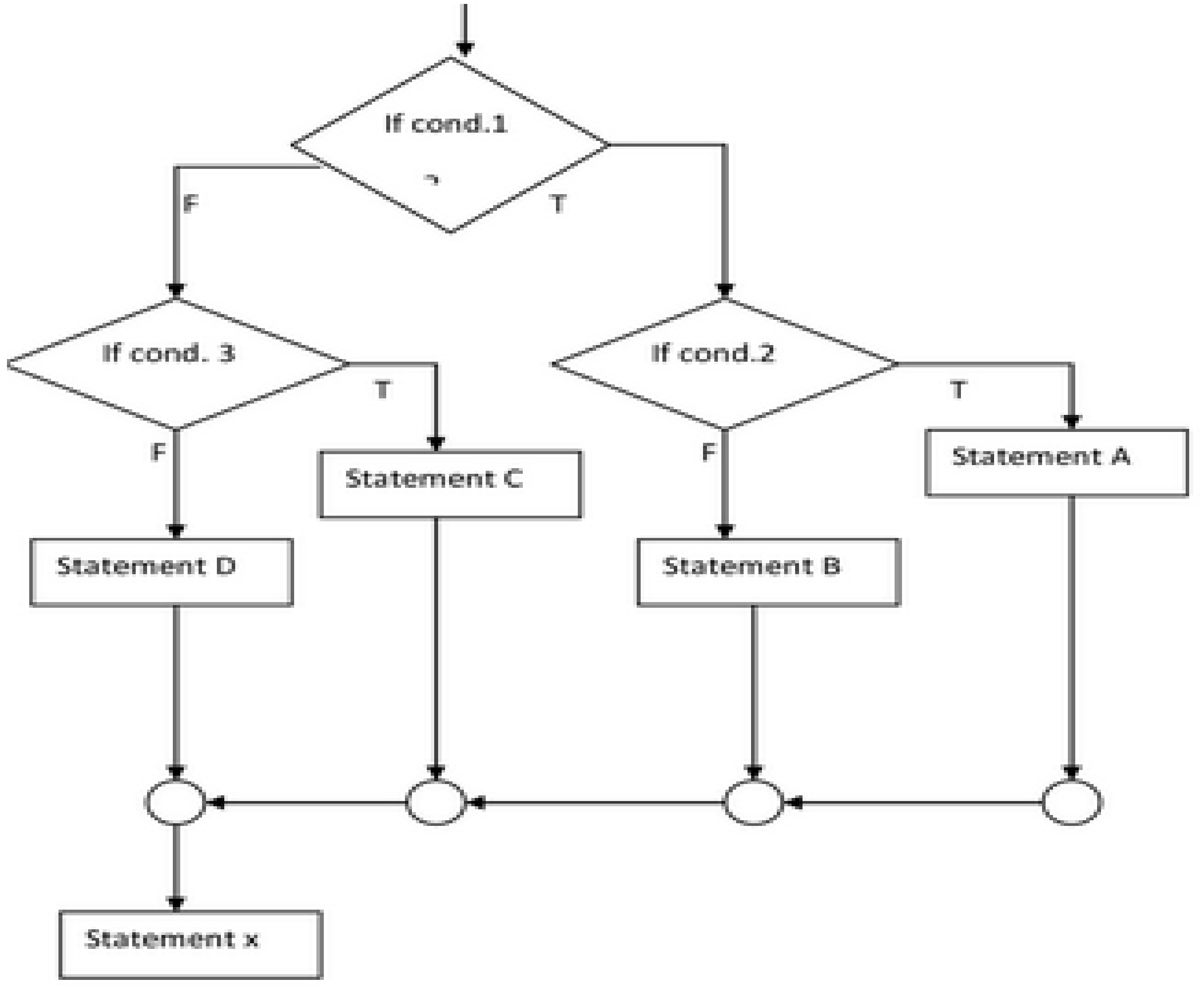
```
class IfStatementExample
    {
          public static void main(String[] args)
          {
                  int number[] = { 50,65,56,71,81};
                  int even=0,odd=0;
                  for(int i= 0;i<number.length;i++)
                   {
                          if((number[i] % 2)==0)
                          {
                                  even += 1;
                          }
                          else
                          {
                                  odd += 1;
                          }
                   System.out.println("\n Even Numbers  :  " + even + "Odd Numbers : " +odd);

                   }
          }
```

Nested if..ELSE Statement

A nested if …else is an statement which contains another if …else.

The syntax for a nested if...else is as follows –

```
If(test condition1 )
 {
        if(test condition2)
         {
                statement-1;
       }
        else
         {
                statement-2;
       }
else
       {
                statement-3;
       }
  }
```


If cond.1
F
T
If cond. 3
If cond.2
T
F
T
F
Statement C
Statement A
Statement D
Statement B
Statement x

```
class ifElseNesting
        {
            public static void main(String arg[])
            {
                 int  a = 325,b=712,c=478;
                System.out.println(“Largest value is :   “);
                 if ( a > b )
                  {
                            if (a > c)
                            {
                              System.out.println(a);
                           }
                         else
                          {
                            System.out.println(c);
                          }
                  }
                 else
                  {
                             if( c > b)
                           {
                      System.out.println(c);
                           }
                         else
                           {
                                             System.out.println(b);
                           }
                   }
        }
}
```

if else ladder statement

The if else ladder statement is used to work on multiple conditions.

Syntax

```
if(condition1)
        statement1;
else if(condition2)
        statement2;
else if(condition3)
        statement3;
 …….
else
        default-statement;
```

Example

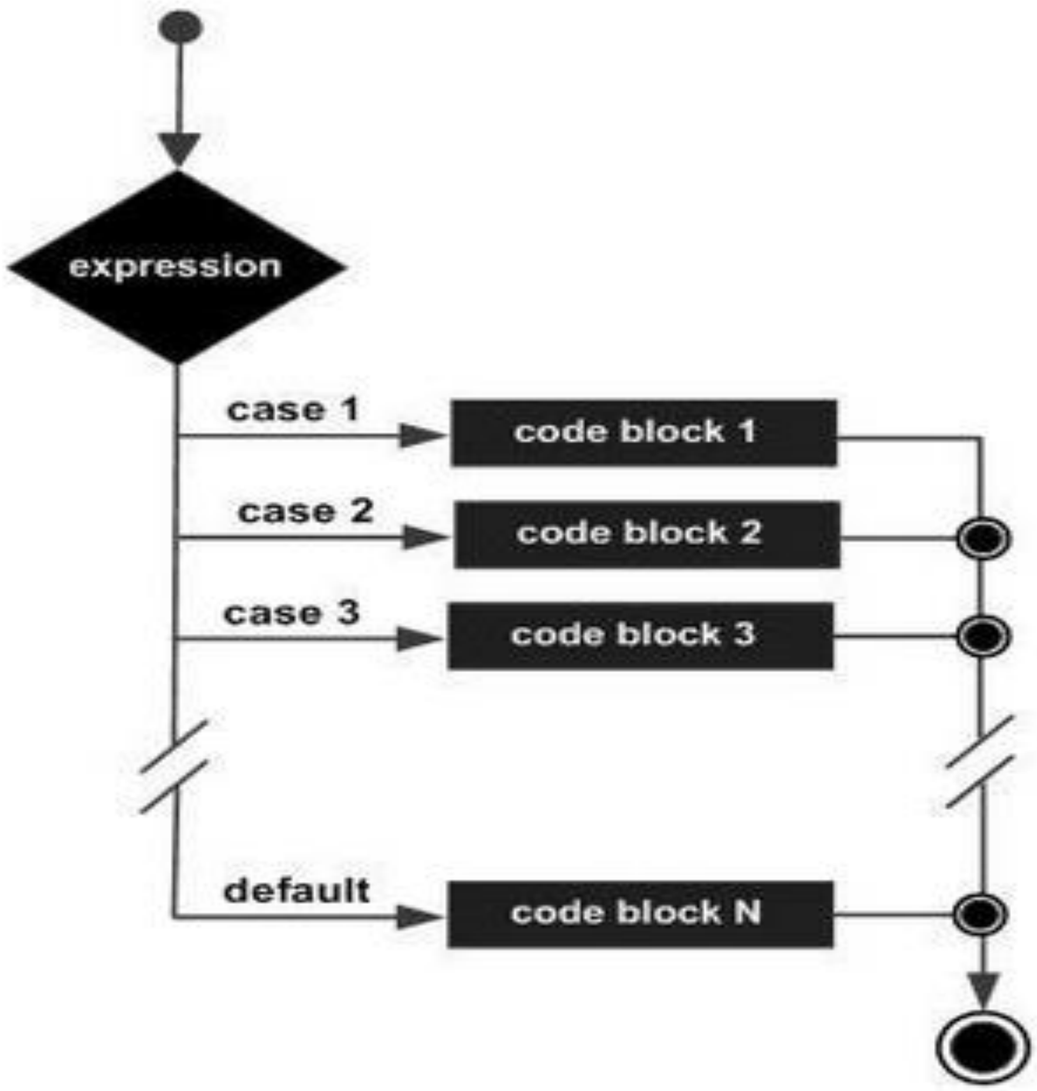
```
class ElseifLadder
 {
        public static void main(String args[])
         {
                 int rollnumber [] = {111,222,333,444};
                 int marks – { 81,75,43,58};
                 for(int i=0;i<rollNumber.length;i++)
                  {
                     if (marks[i] > 79)
                                System.out.println(rollnumber[i] + " "Honors");
                     else if (marks[i] > 59)
                                System.out.println(rollnumber[i] + " "I Division");
                     else if (marks[i] > 49)
                                System.out.println(rollnumber[i] + " "II  Division");
                else
                                System.out.println(rollnumber[i] + " "FAIL");
                   }
    }
}
```

THE SWITCH STATEMENT

- ❖ The if statement in java, makes selections based on a single true or false condition.
- ❖ switch case have multiple choice for selection of the statements or we can switch case is a multiway branch statement.
- ❖ The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement.
- ❖ It adds an easy way to dispatch execution to different parts of your code based on the value of an expression.

Syntax

```
    switch (expression)
    {
           case value1:
                       block-1
                       break;
           case value2:
                       block-2
                       break;
           ……………
           …………….
           default:
                       default-block
                       break;

    }
Statement-x;
```


expression
case 1
code block 1
case 2
code block 2
case 3
code block 3
default
code block N

JAVA PROGRAMMING

```
class SwitchStatement
{
        public static void main(String args[])
        {
                char ch;
                System.out.print("\n\tA.  Red");
                System.out.print("\n\tB.  Blue");
                System.out.print("\n\tC.  Yellow");
                System.out.print("\n\tEnter  Your Choice  (A-C) :\t");
                ch = (char) System.in.read();
                switch(ch)
                {
                        case 'A':
                        {
                                System.out.print("\n\tYour  Choice  is Red\n\n");
                                break;
                        }
                        case 'B':
                        {
                                System.out.print("\n\tYour  Choice  is Blue\n\n");
                                break;
                        }
                        case 'C':
                        {
                                System.out.print("\n\tYour  Choice  is Yellow\n\n");
                                break;
                        }
                        default:
                        {
                                System.out.print("\n\tYour  Choide  is Wrong...\n\n");
                                break;
                        }
                }     }}
```

The ? : Operator

Java has an operator that can be used an alternative to if statement. You are familiar with this operator, the conditional operator ?: This operator can be used to replace if-else statement of the general form:

if(expression !) expression 2; else expression 3;

The above form if can be alternatively written using ?: as follows:
expression 1 ? expression 2 : expression 3;
It works in the same way as given form of if does i.e. expression 1 is evaluated, if it ture, expression 2 gets executed (i.e its becomes the value of entire expression). For instance, the following if statement,

```
int c;
if (a > b)
c = a;
else
c = b;
```

Can be alternatively written as

```
int c = a > b ? a : b;
```

See how simple and compact your code has become.

```
class CheckEvenNumber
{
        public static void main( String args[] )
       {
         int number = 3;
         String msg =  (number % 2 == 0) ? "The number is even!" : "The numbe
          r is odd!";
         System.out.println(msg);
        }
}
```

Output

```
The number is odd!
```

Decision Making and Looping

- ❖ A Computer is used for performing many Repetitive types of tasks The Process of Repeatedly performing tasks is known as looping
- ❖ The Statements in the block may be Executed any number of times from Zero to Up to the Condition is True
- ❖ There is Either Entry Controlled loop or as Exit Controlled Loop
- ❖ We know that before Execution of Statements all Conditions are Checked these are Performed by Entry Controlled Loops Which First Checks Condition
- ❖ In Exit Controlled Loop it Checks Condition for Ending Loop Whether given Condition is False or not if a Loop
- ❖ First Checks Condition For Execution then it is called as Entry Controlled Loop and if a Loop Checks Condition after

In The loop generally there are three basic operations are performed

1) Initialization
2) Condition check
3) Increment

There are the three types of loops in the java

1) while
2) do-while
3) for

while Loop

- while Loop is Known as Entry Controlled Loop because in The while loop first we initialize the value of variable or Starting point of Execution and then we check the condition.

- if the condition is true then it will execute the statements and then after it increments or decrements the value of a variable. But if Condition is false then it will never Executes the Statement

Syntax

```
Initialization;
while(test condition)
  {
       Body of the loop
  }
```

JAVA PROGRAMMING

Example

```
import java.io.*;
class WhileLoop
{
    public static void main(String args[])
    {
        int i=1;

        while(i <=10)
        {
            System.out.print("\n\tI = "+i);
            i++;
        }
    }
}
```

do while Loop

- This is Called as Exit Controlled Loop
- In do while loop first it executes the statements and then it increments the value of a variable and then last it checks the condition So in this either the condition is true or not it Execute the statement at least one time.

Syntax

```
Initialization:
   {
         Body of the loop
   }
while(test condition);
```

Example

```
class DoWhileLoop
{
    public static void main(String args[])
    {
        int i=1;

        do
        {
            System.out.print("\n\tI = "+i);
            i++;
        }
        while(i <=10);
    }
}
```

for Loop

❖ for Loop is Known as Entry Controlled Loop

❖ In This loop all the basic operations like initialization ,condition checking and incrementing or decrementing all these are performed in only one line.

❖ this is similar to the while loop for performing its execution but only different in its syntax.

Syntax

```
for(Initialization:test condition;increment)
  {
        Body of the loop
 }
```

```
class ForLoop
{
    public static void main(String args[])
    {
            int i;

            for(i =1; i <=10; i++)
            {
                        System.out.print("\n\tI = "+i);
            }
    }
}
```

Jumping in Loops

Jumping Statements are also Called as Control Statements.

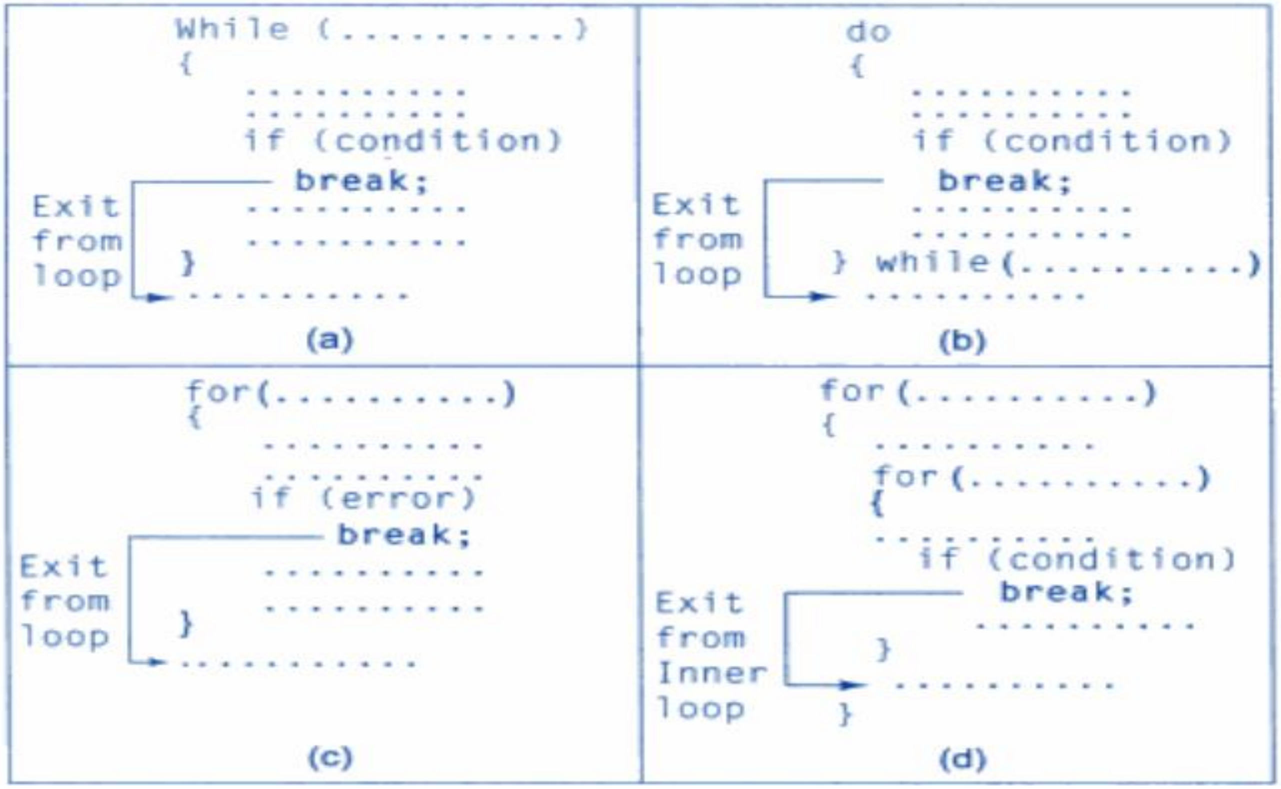
Jumping Statements are used for controlling the Execution of the program

There are the three control statements those are break , continue. And labelled loops

Break

In Java, break is majorly used for Terminate a sequence in a switch statement

To exit a loop.



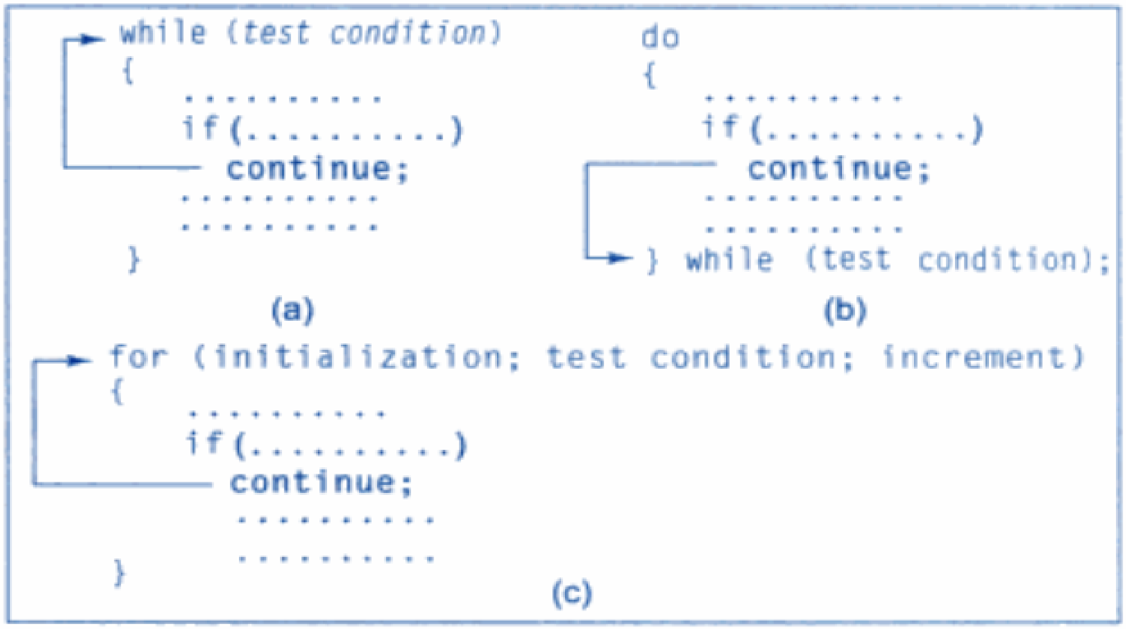
JAVA PROGRAMMING

```
class BreakLoopDemo
{
    public static void main(String args[])
    {
        // Initially loop is set to run from 0-9
        for (int i = 0; i < 10; i++)
        {
            // terminate loop when i is 5.
            if (i == 5)
                break;

            System.out.println("i: " + i);
        }
        System.out.println("Loop complete.");
    }
}
```

Continue

Unlike break which causes the loop to be terminated, the continue causes the loop to be continued with the next iteration.

The continue statement tells the compiler skip the following statements and continue the next iteration



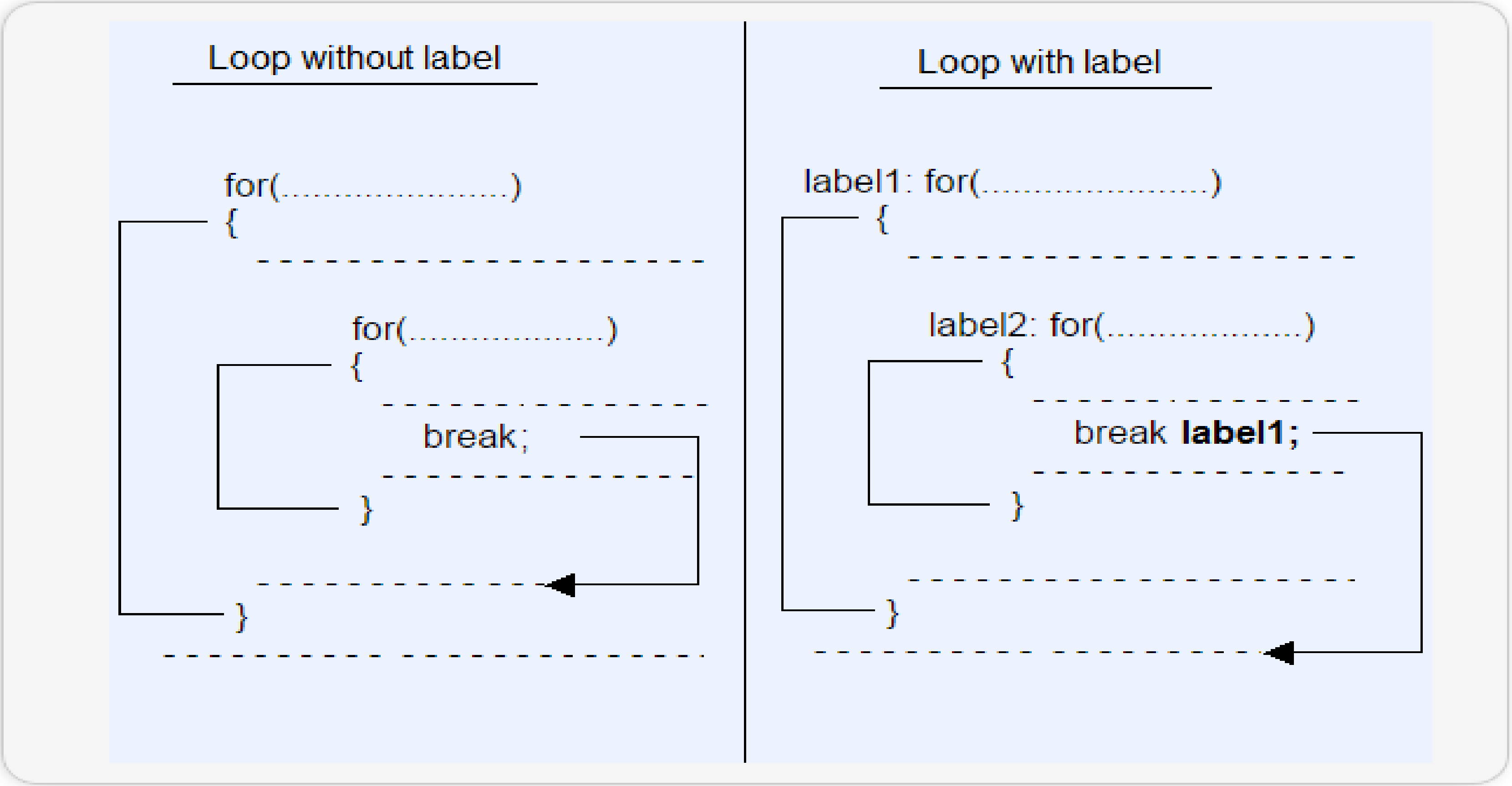
```
while (test condition)
{
    ..........
    if(..........)
        continue;
    ..........
    ..........
}
        (a)

do
{
    ..........
    if(..........)
        continue;
    ..........
    ..........
} while (test condition);
        (b)

for (initialization; test condition; increment)
{
    ..........
    if(..........)
        continue;
        ..........
        ..........
}
        (c)
```

```
class ContinueDemo
{
    public static void main(String args[])
    {
        for (int i = 0; i < 10; i++)
        {
            // If the number is even
            // skip and continue
            if (i%2 == 0)
                continue;

            // If number is odd, print it
            System.out.print(i + " ");
        }
    }
}
```

LABELED LOOPS

Labeled loops are used for giving a name to the loop which is also known as Label of Loop if you wants to give a name to Loop the Label Must be given before loop with a valid name and a Colon.

Syntax of Labelled loop


Loop without label
for(......................)
{
for(..................)
{
break;
}
}
Loop with label
label1: for(......................)
{
label2: for(...................)
{
break label1;
}
}

JAVA PROGRAMMING

```
class WithLabelledLoop
{
    public static void main(String args[])
    {
        int i,j;
        loop1: for(i=1;i<=10;i++)
                {
                        System.out.println();
                        loop2: for(j=1;j<=10;j++)
                          {
                                System.out.print(j + " ");
                                if(j==5) break loop1; //Statement 1
                          }
                }
    }
}
```