S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, NIDASOSHI.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Permanently Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Subject: Object Oriented Concepts (18CS45)

Module 2: Introduction to Java

Dr. Mahesh G. Huddar

Dept. of Computer Science and Engineering

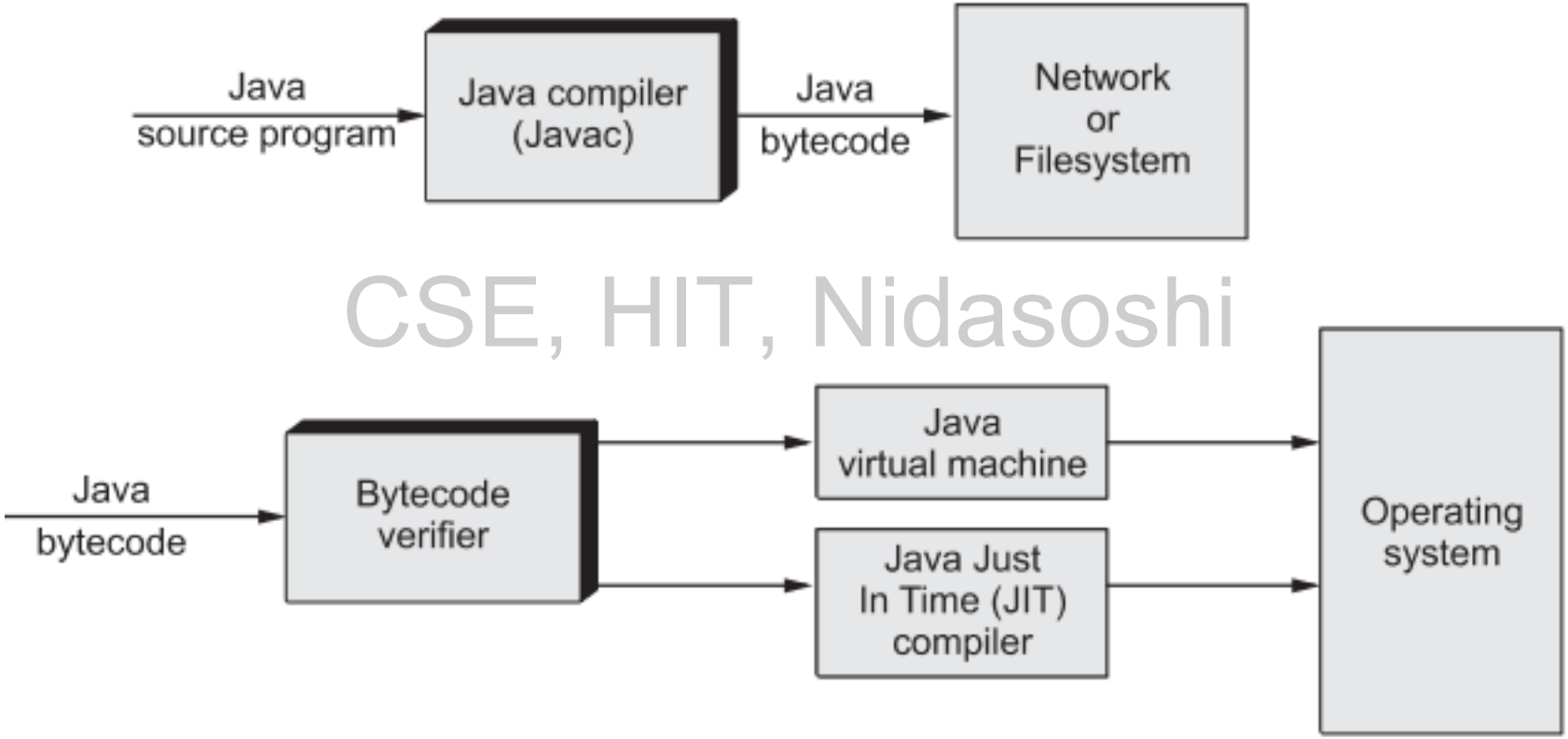
Java's Magic : the Byte Code

- Bytecode is an intermediate form of java programs.
- Bytecode consists of optimized set of instructions that are not specific to processor.
- We get **bytecode** after compiling the java program using a compiler called **javac**.
- The bytecode is to be executed by Java runtime environment which is called as Java Virtual Machine.
- Sometimes JVM is also called as interpreter for bytecode.
- The programs that are running on JVM must be compiled into a binary format which is denoted by .class files.

Java's Magic : the Byte Code

- Sometime for ease of distribution multiple class files are packaged into one jar file.
- The JVM executes .class or .jar files, by either interpreting it or using **a just-in-time compiler (JIT)** .
- The JIT is used for compiling and not for interpreting the file.
- It is used in most JVMs today to achieve greater speed. The bytecode verifier verifies all the bytecode before it is executed. This verification helps to prevent the crashing of host machine.

Java's Magic : the Byte Code



Java Development Kit (JDK)

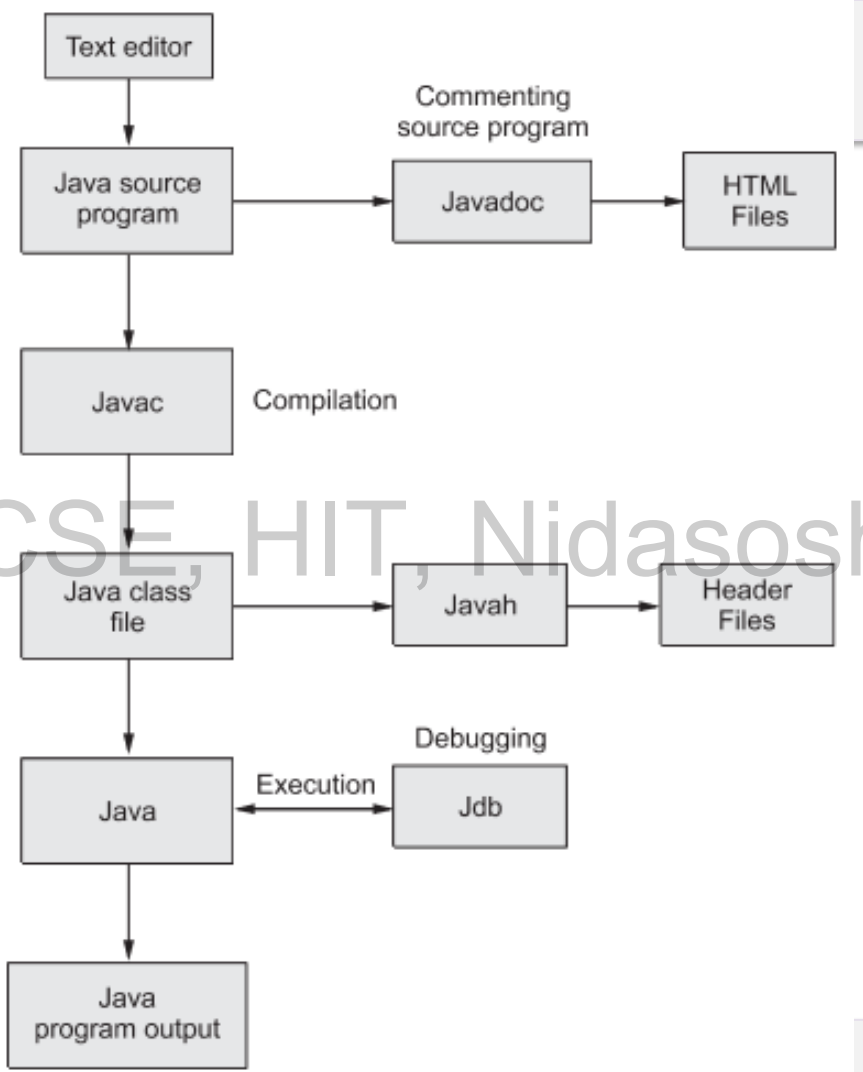
- The Java Development Kit is nothing but a collection of tools that are used for development and runtime programs. It consists of –

1. **javac** - The Java compiler which translates the source code to the bytecode form and stores it in a separate class file.
2. **java** - The Java interpreter, which interprets the bytecode stored in the class file and executes the program to generate output.
3. **javadoc** - For creating the HTML document for documentation from source code file.
4. **javah** - It produces header files for the use of native methods.
5. **jdb** - The Java debugger which helps to find the errors in the program.
6. **appletviewer** - For executing the Java applet.

Java Development Kit (JDK)

Following are the steps that illustrate execution process of the application program :

1. The user creates the Java source code in the text editor.
2. The source code is compiled using the javac command.
3. The javadoc tool can be used to create the HTML files that document the source program.
4. On compiling the source program a class file gets generated which consists of the byte code.
5. The developer may use javah tool for generating the required header files.
6. The class file produced by javac can be interpreted using Java in order to produce an executable.



Java the Buzzword – Features of Java

Following are the features of Java which makes it as a revolutionary programming language -

1. Java can be compiled and interpreted

Normally programming languages can be either compiled or interpreted but Java is a language which can be compiled as well as interpreted. First, Java compiler translates the Java source program into a special code called bytecode. Then Java interpreter interprets this bytecode to obtain the equivalent machine code. This machine code is then directly executed to obtain the output.

2. Java is a platform independent and portable programming language

Platform independence is the most exciting feature of Java program. That means programs in Java can be executed on variety of platforms. This feature is based on the goal - write once, run anywhere, and at anytime forever.

Java Buzzword – Features of Java

3. Java is known as an object oriented programming language

Java is a true object oriented language as everything in java is an object. In Java, all the code and data lies within the classes and object.

4. Java is robust and secure

Java ensures the reliable code. The data types are strictly checked at the compile time as well as run time in Java. The memory management technique is supported by garbage collection technique. This technique eliminates various memory management problems. The exception handling feature of Java helps the programmer to handle the serious errors delicately without crashing the overall system.

Java Buzzword – Features of Java

5. Java is a designed for distributed systems

This feature is very much useful in networking environment. In Java, two different objects on different computers can communicate with each other. This can be achieved by Remote Method Invocation(RMI). This feature is very much useful in Client-Server communication.

6. Java is simple and small programming language

Java is very simple programming language. Even though you have no programming background, you can learn this language very easily. The programmers who have worked on C++ can learn this language very efficiently.

Java Buzzword – Features of Java

7. Java is a multithreaded and interactive language

Java supports multi-threaded programming which allows a programmer to write such a program that can perform many tasks simultaneously. This allows the programmer to develop the interactive systems.

8. Java is known for its high performance, scalability, monitoring and manageability

Due to the use of bytecode the Java has high performance. The use of multi-threading also helps to improve the performance of the Java. The J2SE helps to increase the scalability in Java. For monitoring and management Java has large number of Application Programming Interfaces(API). There are tools available for monitoring and tracking the information at the application level.

Java Buzzword – Features of Java

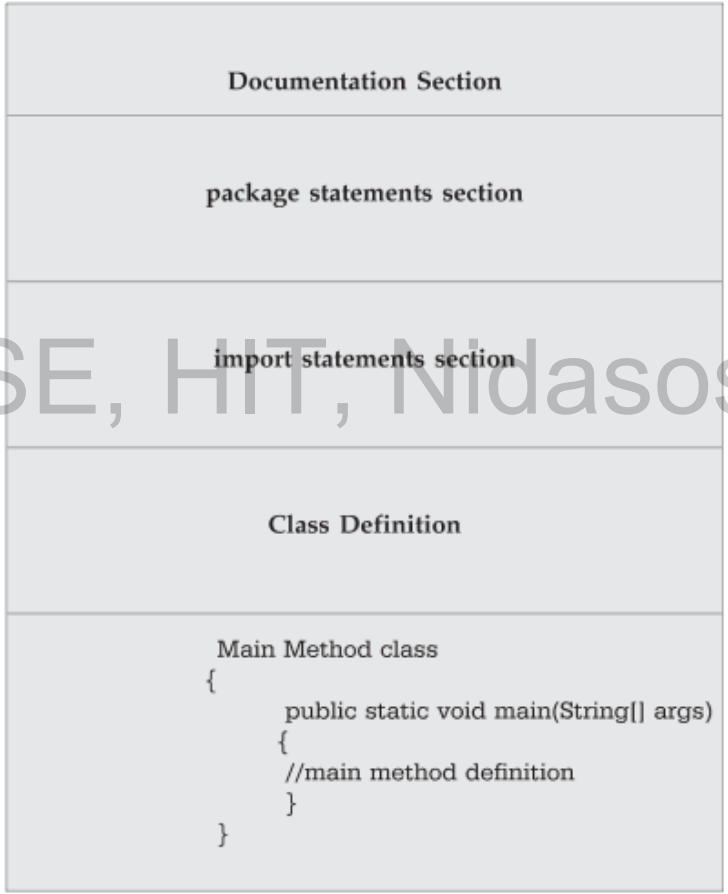
9. Java is a dynamic and extensible language

This language is capable of dynamically linking new class libraries, methods and objects. Java also supports the functions written in C and C++. These functions are called native methods.

10. Java can be developed with ease

There are various features of Java such as Generics, static import, annotations and so on which help the Java programmer to create a error free reusable code.

Java – Program Structure



Java – Program Structure

1. Documentation section: The documentation section provides the information about the source program. This section contains the information which is not compiled by the Java. Everything written in this section is written as comment.

2. Package section: It consists of the name of the package by using the keyword package. When we use the classes from this package in our program then it is necessary to write the package statement in the beginning of the program.

Java – Program Structure

3. Import statement section: All the required java API can be imported by the import statement. There are some core packages present in the java. These packages include the classes and method required for java programming. These packages can be imported in the program in order to use the classes and methods of the program.

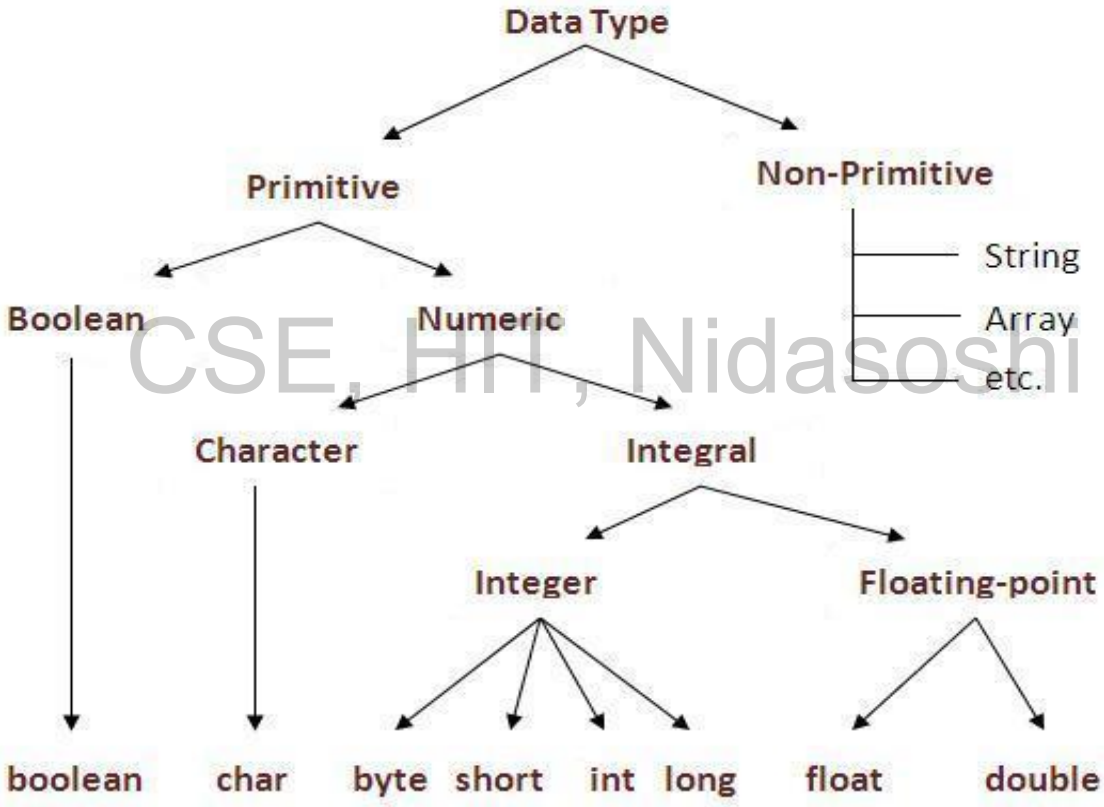
4. Class definition section: The class definition section contains the definition of the class. This class normally contains the data and the methods manipulating the data.

5. Main method class: This is called the main method class because it contains the main() function. This class can access the methods defined in other classes.

Java – Program Structure

```
/*  This is my first Java Program */

import java.io.*;

class FirstProgram

{

        public static void main(String args[])

        {

                System.out.println("Welcome to Java Programming");

        }

}
```

Data Types in Java



Data Types in Java

Various data types used in Java are **byte, short, int, long, char, float, double and boolean**.

byte

- This is in fact smallest integer type of data type. Its width is of 8-bits with the range - 128 to 127. The variable can be declared as byte type as
- Example: byte I, j;

Data Types in Java

short

- This data type is also used for defining the signed numerical variables with a width of 16 - bits and having a range from -32,768 to 32,767. The variable can be declared as short as
- Example:
- short a, b

Data Types in Java

int

- This is the most commonly used data type for defining the numerical data. The width of this data type is 32-bit having a range -2,147,483,648 to 2,147,483,647. The declaration can be
- Example:
- int p, q;

Data Types in Java

long

Sometimes when int is not sufficient for declaring some data then long is used.

The range of long is really very long and it is

-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

The declaration can be

Example:

long x, y;

Data Types in Java

float

- To represent the real number(i.e. number that may have decimal point) float data type can be used. The width is 32-bit and range of this data type is $1.4e^{-45}$ to $3.4e^{+38}$.
- Example:
- float x, y

Data Types in Java

double

- To represent the real numbers of large range the double data type is used. Its width is 64-bit having the range $4.9e^{-324}$ to $1.8e^{+308}$.
- Example:
- double a, b;

Data Types in Java

char

- This data type is used to represent the character type of data. The width of this data type is 16-bit and its range is 0 to 65,536.
- Example:
- char ch;

Data Types in Java

boolean

- Boolean is a simple data type which denotes a value to be either true or false.
- Example:
- boolean a, b;

Data Types in Java – Example Program

```
/* This program introduces use of various data type in the program */
class DatatypeDemo
{
        public static void main(String[] args)
        {
                byte a=10;
                short b=10*128;
                int c=10000* 128;
                long d=10000*1000*128;
                double e=99.9998;
                char f='a';
                boolean g=true;
                boolean h=false;
                System.out.println("The value of a=: "+a);
                System.out.println("The value of b=: "+b);
                System.out.println("The value of c=: "+c);
                System.out.println("The value of d=: "+d);
                System.out.println("The value of e=: "+e);
                System.out.println("The value of f=: "+f);
                System.out.println("The value of g=: "+g);
                System.out.println("The value of h=: "+h);
        }
}
```


OUTPUT

```
The value of a=: 10
The value of b=: 1280
The value of c=: 1280000
The value of d=: 1280000000
The value of e=: 99.9998
The value of f=: a
The value of g=: true
The value of h=: false
```

Variables in Java

- A variable is an identifier that denotes the storage location.
- Variable is a fundamental unit of storage in Java.
- The variables are used in combination with identifiers, data types, operators and some value for initialization.
- The variables must be declared before its use.
- The syntax of variable declaration will be –

Data_type name_of_variable [=initialization][,=initialization][,...];

Variables in Java

Following are some rules for variable declaration –

- The variable name should not begin with digits.
- No special character is allowed in identifier except underscore.
- There should not be any blank space with the identifier name.
- The identifier name should not be a keyword.
- The identifier name should be meaningful.

Variables in Java

For Example :

- int a, b;
- char m='a';
- byte k=12, p, t=22;

The variables have a scope which defines their visibility and a lifetime.

Operators in Java

Various operators that are used in Java are enlisted in following table

Type	Operator	Meaning	Example
Arithmetic	+	Addition or unary plus	c=a+b
	–	Subtraction or unary minus	d= – a
	*	Multiplication	c=a*b
	/	Division	c=a/b
	%	Mod	c=a%b

Operators in Java

Various operators that are used in Java are enlisted in following table

Relational	<	Less than	a<4
	>	Greater than	b>10
	<=	Less than equal to	b<=10
	>=	Greater than equal to	a>=5
	==	Equal to	x==100
	!=	Not equal to	m!=8

Operators in Java

Various operators that are used in Java are enlisted in following table

Logical	&&	And operator	0&&1
	||	Or operator	0 ||1
Assignment	=	is assigned to	a=5
Increment	++	Increment by one	++i or i++
Decrement	– –	Decrement by one	– – k or k– –

Conditional Operators in Java

The conditional operator is "?"

The syntax of conditional operator is

Condition?expression1:expression2

Where expression1 denotes the true condition and expression2 denotes false condition.

For example: **a >b?true:false**

This means that if **a** is greater than **b** then the expression will return the value **true** otherwise it will return **false**.

Operator Precedence in Java

- The precedence relation specifies which operation must be done first during the expression evaluation.
- Following table denotes the precedence relation.

- The operators are arranged from highest priority to lowest priority.
- The operators in the same row indicate the equal precedence.

Name	Operators
Post increment/decrement	++,- -
unary operator	unary minus, ~,!
Multiplicative	*,/
Additive	+,-
Shift	<<,>>, >>>
Relational	<,>,<=,>=,instanceof
Equality	==,!=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
Logical AND	&&
Logical OR	||
ternary	?:

Program for demonstrating arithmetic operators

```
class ArithOperDemo
{
        public static void main(String[] args)
        {       System.out.println("\n Performing arithmetic operations ");
                int a=10, b=20, c;
                System.out.println("a= "+a);
                System.out.println("b= "+b);
                c=a+b;
                System.out.println("\n Addition of two numbers is "+c);
                c=a-b;
                System.out.println("\n Subtraction of two numbers is "+c);
                c=a*b;
                System.out.println("\n Multiplication of two numbers is "+c);
                c=a/b;
                System.out.println("\n Division of two numbers is "+c);
        }
}
```


Output

```
Performing arithmetic operations
a= 10 b= 20
Addition of two numbers is 30
Subtraction of two numbers is -10
Multiplication of two numbers is 200
Division of two numbers is 0
```

Program for demonstrating relational operators

```
class RelOperDemo
{
        public static void main(String[] args)
        {
                System.out.println("\n Performing Relational operations ");
                int a=10,b=20,c=10;
                System.out.println("a= "+a);
                System.out.println("b= "+b);
                System.out.println("\n The a<b is "+(a<b));
                System.out.println("\n The a>b is "+(a>b));
                System.out.println("\n The a= =c is "+(a==c));
                System.out.println("\n The a< =b is "+ (a< =b));
                System.out.println("\n The a> =c is "+(a> =c));
                System.out.println("\n The a!=b is "+(a!=b));
        }
}
```


Output

Performing Relational operations

a= 10

b= 20

The a<b is true

The a>b is false

The a= =c is true

The a< =b is true

The a> =c is true

The a! =b is true

Program for demonstrating increment and decrement

```
class IncrDecrOperDemo
{
        public static void main(String[] args)
        {
                System.out.println("\n Performing increment and
decrement operations ");
                int a=11,b=22;
                System.out.printin("a= "+a);
                System.out.println("b= "+b);
                System.out.println(An Pre-Incrementing a "+ + +a);
                System.out.println("\n Post-Incrementing b "+ b++);
                System.out.println(An After this statement the value of
b is "+ b);
        }
}
```


Output

```
Performing increment and

decrement operations
a= 11 b= 22

Pre-Incrementing a 12

Post-Incrementing b 22

After this statement the value of b is

23
```

Short Circuit Operator in Java

- The && and || are the short circuit operators in Java.
- These operators are called short-circuited operators because the outcome of the expression can be determined by the left operand only then the right operand won't be evaluated and hence the order of evaluation will always be from left to right irrespective of whether you place the operand within parentheses or not.

Short Circuit Operator in Java

For example:

- Suppose we have an expression as **'operand1 && operand2'** and 'operand1' is evaluated to be **false**, then there is no point evaluating the right operand 'operand2' as irrespective of whether that operand is **true** or **false** the expression will always be **false** only.
- Suppose we have an expression as **'operand1 || operand2'** and 'operand1' is evaluated to be **true**, then there is no point evaluating the right operand 'operand2' as irrespective of whether that operand is **true** or **false** the expression will always be **true** only.

Shift Operators in Java

- There are left, right shift and unsigned right shift operators in Java.
- **Left shift operator:**
- The left shift operator is denoted by <<.
- The syntax of left shift is **value << n** where the bits in value are shifted by n positions to the left.
- For example
- 5 << 2 will result in 0000 0101 << 2 → 0000 1010 → 0001 0100 = 20

Short Circuit Operator in Java

- **Right shift operator:**
- The right shift operator is denoted by >>.
- The syntax of right shift is **Value >> n** where the bits in value are shifted by n positions to the right.
- For example
- 5 >> 2 will result in 0000 0101 >> 2 → 0000 0010 → 0000 0001 = 1
- -5 >> 2 will result in 1111 1011 >> 2 → 11111101 → 11111110 = -2

Short Circuit Operator in Java

- **Unsigned right shift operator:**
- The unsigned right shift operator is denoted by >>>.
- The syntax of unsigned right shift operator is **Value >>> n** where the bits in value are shifted by n positions to the right and fills MSB with 0.
- For example 5 >>> 2 will result in
- 0000 0101 >>> 2 → 0000 0010 → 0000 0001 = 1

Single Dimensional Array in Java

- Array is a collection of similar type of elements. Thus grouping of similar kind of elements is possible using arrays. Typically arrays are written along with the size of them.
- The syntax of declaring array is -

```
data_type array_name[ ];
```


- and to allocate the memory –

```
array_name=new data_type[size];
```

- where array_name represent name of the array, new is a keyword used to allocate the memory for arrays, data_type specifies the data type of array elements and size represent the size of an array. based on arrays.

Single Dimensional Array in Java

- For example :

a=new int[10];

- After this declaration the array a will be created as follows Array a[10]

- That means after the above mentioned declaration of array, all the elements of array will get initialized to zero.
- Note that, always, while declaring the array in Java, we make use of the keyword new, and thus we actually make use of dynamic memory allocation. Therefore arrays are allocated dynamically in Java.

Single Dimensional Array in Java - Program

```
class SampleArray
{
        public static void main(String[] args)
        {       int a[];
                a=new int[5];
                System.out.println("Storing the numbers in array");
                a[0]=1;
                a[1]=2;
                a[2]=3;
                a[3]=4;
                a[4]=5;
                System.out.println("The element at a[2] is: " +a[2]);
                System.out.println("The element at a[4] is: " +a[4]);
        }
}
```


Output

```
Storing the numbers in array

The element at a[2] is : 3

The element at a[4] is : 5
```

Single Dimensional Array in Java - Program

```
class SampleArray
{
        public static void main(String args[])
        {
                int a[] = {1, 2, 3, 4, 5};
                System.out.println("Storing the numbers in array");
                System.out.println("The element at a[2] is: " +a[2]);
                System.out.println("The element at a[4] is: " +a[4]);
        }
}
```


Output

```
Storing the numbers in array
The element at a[2] is : 3
The element at a[4] is : 5
```

Single Dimensional Array in Java

- Another way of initialization of array is
- int a[ ]= {1,2,3,4,5};
- That means, as many number of elements that are present in the curly brackets, that will be the size of an array.
- In other words there are total 5 elements in the array and hence size of array will be 5.

Single Dimensional Array in Java

- Write a program that creates and initializes a four integer element array.
- Calculate and display the average of its values.

Single Dimensional Array in Java

```
class AvgDemo
{
        public static void main(String[] args)
        {
                int a[]= {10,20,30,40};
                int sum=0,avg;
                for(int i=0;i<4;i++)
                {
                        sum=sum+ a[i];
                }
                System.out.println("Sum= "+sum);
                avg=sum/4;
                System.out.println("\nAverage=" +avg);
        }
}
```


Output

Sum= 100

Average= 25

Single Dimensional Array in Java

- Write a program to calculate the average among the elements (8, 6, 2, 7) using for each in Java. How for each is different from for loop ?

Single Dimensional Array in Java

```
class Test
{       public static void main(String[] args)
        {       int elements[] = {8,6,2,7};
                int sum=0;
                int count=0;
                for (int num : elements)      // for each loop
                {
                        sum=sum+num;
                        count++;
                }
                System.out.println("The sum is ="+sum);
                float avg=sum/count;
                System.out.println("The average is ="+avg);
        }
}
```


Output:

The sum is = 23

The average is = 5.0

Single Dimensional Array in Java

- In case of simple for loop, the index value is known to programmer explicitly at each iteration.
- But in case of for-each loop, during its execution the index is not known explicitly to the programmer.

Ragged Array in Java

- Ragged array is more than one dimension array in which each dimension have different size.
- For example {{1,2,3,4}, {5,6}, {7,8,9}} is a ragged two dimensional array.
- Following is a simple Java program that represents the use of ragged array.

Ragged Array in Java

```
public class RaggedArray
{
        public static void main(String[] args)
        {       int A[] = new int[3][];
                A[0] = new int[4];
                A[1] = new int[2];
                A[2] = new int[3];
                System.out.println("Total Number of Rows: " + A.length);
                A[0][0] = 1;
                A[0][1] = 2;
                A[0][2] = 3;
                A[0][3] = 4;
                // 2nd row
                A[1][0] = 5;
                A[1][1] = 6;
```

Ragged Array in Java

```
                // 3rd row
                A[2][0] = 7;
                A[2][1] = 8;
                A[2][2] = 9;
                System.out.println("\nArray Representation");
                for(int i = 0; i < A.length; i++)
                {
                        for(int j = 0; j < A[i].length; j++)
                        {
                                System.out.print(A[i][j] + " ");
                        }
                        System.out.println(" ");
                }
        }
}
```


Output

Total Number of Rows: 3

Array Representation

1 2 3 4

5 6

7 8 9

Two Dimensional Array in Java

- The two dimensional arrays are the arrays in which elements are stored in rows as well as in columns.
- For example Rows The two array can be declared and initialized as follows

Syntax

datatype array_name[][]=new data_type[size][size];

- For example:

int a[][]=new int[3][3];

- Let us demonstrate a java program that is using two dimensional arrays.

Two Dimensional Array in Java

```
class Sample2DArray
{
        public static void main(String args[])
        {
                int a[][]=new int[3][3];
                int k=0;
                System.out.println("\tStoring the numbers in array");
                for(int i=0;i<3;i++)
                {
                        for(int j=0;j<3;j++)
                        {
                                a[i][i]=k+10;
                                k=k+10;
                        }
                }
```

Two Dimensional Array in Java

```
                System.out.println("You have stored...");
                for(int i=0;i<3;i++)
                {
                        for(int j=0;j <3;j++)
                        {
                                System.out.print(" "+a[i][j]);
                        }
                        System.out.println();
                }
        }
}
```


OUTPUT:

Storing the numbers in array

You have stored...

10 20 30

40 50 60

70 80 90

Two Dimensional Array in Java

- The typical application of two dimensional arrays is performing matrix operations.
- Let have some simple java program in which various matrix operations are performed.

Two Dimensional Array in Java

- Write a program for following series $1+1/2+1/2^2+1/2^3...+1/2^n$.

Two Dimensional Array in Java

```
import java.io.*;
import java.lang.*;
import java.math.*;
public class Series1
{
        public static void main(String args[])
        {
                int n;
                double val;
                double sum=0;
                n=5;
                System.out.println("Program for displaying the sum of series");
```

Two Dimensional Array in Java

```
                for(int i=0;i<=n;i+ +)
                {
                        val= 1/(Math.pow(2,i));
                        sum=sum+val;
                }
                System.out.println("Sum of the series is
"+sum);
        }
}
```

OUTPUT:

Program for displaying the sum of series

Sum of the series is 1.9375

Control Statements in Java

Programmers can take decisions in their program with the help of control statements. Various control statements that can be used in Java are

1. if statement
2. if else statement
3. while statement
4. do... while statement
5. switch case statement
6. for loop

Let us discuss each of the control statement in details.

Control Statements in Java

1. if statement

- The if statement is of two types

  Simple if statement: The if statement in which only one statement is followed by that is statement.

  Syntax


```
if(apply some condition)
    statement
```

  For example

```
if(a>b)
    System.out.println("a is Bning!");
```

Control Statements in Java

Compound if statement: If there are more than one statement that can be executed when if condition is true. Then it is called compound if statement. All these executable statements are placed in curly brackets.

Syntax


```
if(apply some condition)
{
        statement 1
        statement 2
        statement n
}
```

Control Statements in Java

2. if...else statement

The syntax for if...else statement will be –

```
if(condition)
        statement
else
        statement
```

Control Statements in Java

For example

```
if(a>b)
        System.out.println("a is big")
else
        System.out.println("b :big brother")
```

Control Statements in Java

The if...else statement can be of compound type even.

For example

```
If(raining= =true)
{
        System.out.println("I won't go out");
        System.out.println("I will watch T.V. Serial");}
else
{
        System.out.println("I will go out");
        System.out.println("And will meet my friend");
}
```

Control Statements in Java

if...else if statement

The syntax of if...else if statement is

```
if(is condition true?)
        statement
else if(another condition)
        statement
else if(another condition)
        statement
else
        statement
```

Control Statements in Java

```
class ifelsedemo
{
        public static void main(String[] args)
        {
                int x=111, y=120, z=30;
                if(x>y)
                {
                        if(x> z)
                                System.out.println("The x is greatest");
                        else
                                System.out.println("The z is greatest");
                }
```

Control Statements in Java

```
                else
                {
                        if(y> z)
                                System.out.printin("The y is greatest");
                        else
                                System.out.println("The z is greatest");
                }
        }
}
```

Control Statements in Java

3. while statement

This is another form of while statement which is used to have iteration of the statement for any number of times.

The syntax is

```
while(condition)
{
        statement 1;
        statement 2;
        statement 3;
        • • •
        statement n;
}
```

Control Statements in Java

```
class whiledemo
{
        public static void main(String args[])
        {
                int count=1,i=0;
                while(count< =5)
                {
                        i=i+1;
                        System.out.println("The value of i= "+i);
                        count+ +;
                }
        }
}
```

Output

The value of i= 1

The value of i= 2

The value of i= 3

The value of i= 4

The value of i= 5

Control Statements in Java

4. do... while statement

- This is similar to while statement but the only difference between the two is that in case of do...while statement the statements inside the do...while must be executed at least once.
- This means that the statement inside the do...while body gets executed first and then the while condition is checked for next execution of the statement, whereas in the while statement first of all the condition given in the while is checked first and then the statements inside the while body get executed when the condition is true.

Control Statements in Java

Syntax

```
do {
        statement 1;
        statement 2;
        statement 3;
        • • •
        statement n;
} while(condition);
```

Control Statements in Java

```
class dowhiledemo
{
        public static void main(String args[])
        {
                int count=1,i=0;
                do
                {
                        i=i+1;
                        System.out.println("The value of i= "+i);
                        count++;
                }while(count< = 5);
        }
}
```

Output

```
The value of i= 1
The value of i= 2
The value of i= 3
The value of i= 4
The value of i= 5
```

Control Statements in Java

5. switch statement

You can compare the switch case statement with a Menu-Card in the hotel.

You have to select the menu then only the order will be served to you. Here is a sample program which makes use of switch case statement -

Control Statements in Java

```
class switchcasedemo
{
        public static void main(String args[]) throws java.io.IOException
        {
                char choice;
                System.out.println(" \tProgram for switch case demo');
                System.out.println("Main Menu");
                System.out.println("1. A");
                System.out.println("2. B');
                System.out.println("3. C');
```

Control Statements in Java

```
        System.out.println("4. None of the above");
        System.out.println("Enter your choice'):
        Choice = (char)System.in.read();
        switch(choice)
        {
                case '1':System.out.println('You have selected A'); break;
                case '2':System.out.println('You have selected B'); break;
                case '3':System.out.println('You have selected C'); break;
                default:System.out.println("None of the above choices made');
        }
}
```

Output

```
Program for switch case demo
Main Menu
1. A
2. B
3. C
4. None of the above
Enter your choice 2
You have selected B
```

Control Statements in Java

6. For loop

for is a keyword used to apply loops in the program.

Like other control statements for loop can be categorized in simple for loop and compound for loop.

Simple for loop :

```
for (statement 1; statement 2; statement 3)
        execute this statement;
```

Control Statements in Java

Compound for loop :

```
for(statement 1; statement 2; statement 3)
{
        execute this statement;
        execute this statement;
        execute this statement;
}
```


Here Statement 1 is always for initialization of conditional variables, Statement 2 is always for terminating condition of the for loop, Statement 3 is for representing the stepping for the next condition.

Control Statements in Java

```
class forloop
{
        public static void main(String args[])
        {
                for(int i=0;i< =5;i+ +)
                        System.out.println("The value of i: "+i);
        }
}
```

Output

```
The value of i: 0
The value of i: 1
The value of i: 2
The value of i: 3
The value of i: 4
The value of i: 5
```

Break and Continue Statements in Java

Break and Continue

Sometimes when we apply loops we need to come out of the loop on occurrence of particular condition. This can be achieved by break statement.

Following program illustrates the use of break in the for loop.

Break and Continue Statements in Java

```
class breakdemo
{
        public static void main(String args[])
        {
                for(int i=1;i< =20;i+ + )
                {
                        if(i%10==0)
                        {
                                System.out.println("number "+i+" is divisible by 10");
                                break;                  //come out of for loop
                        }
                        else
                        {       System.out.println("number "+i+" is not divisible by 10");
                        }
                }
        }
}
```

Break and Continue Statements in Java

Output:

The number 1 is not divisible by 10
The number 2 is not divisible by 10
The number 3 is not divisible by 10
The number 4 is not divisible by 10
The number 5 is not divisible by 10
The number 6 is not divisible by 10
The number 7 is not divisible by 10
The number 8 is not divisible by 10
The number 9 is not divisible by 10
The number 10 is divisible by 10

Break and Continue Statements in Java

Program Explanation:

Note that in the above program the numbers after 10 will not be considered at all.

This is because when we reach at 10 then if condition becomes true and we encounter break statement.

This forces us to come out of for loop. Just look at the output of corresponding program!!!

Similarly we can use the return statement which is useful for returning the control from the current block.

Type Casting in Java

- For incompatible data types we have to do explicit type conversion. This mechanism is called casting.
- Type casting means explicit conversion between incompatible data types. For casting the values we use following syntax :

(type) variable

Example :

- double a;
- int b;
- b=(int) a;

Type Casting in Java

```
class TypeCastProg
{       public static void main(String[] args)
        {
                double x;
                int y;
                x=25.50;
                System.out.println(Value of [double] x: "+x);      //output 25.5
                System.out.println("Conversion of double to integer");
                y=(int)x;
                System.out.println(Value of [integer' y: "+y);     //output:25
```

Type Casting in Java

```
        int m;
        double n;
        m=10;
        n=(double)m;
        System.outprintln(Value of [integer] m: "+m);    //output:10
        System.out.println("Conversion of integer to double");
        System.out.println("Value of [double] n: "+n);    //output:10.0
    }
}
```