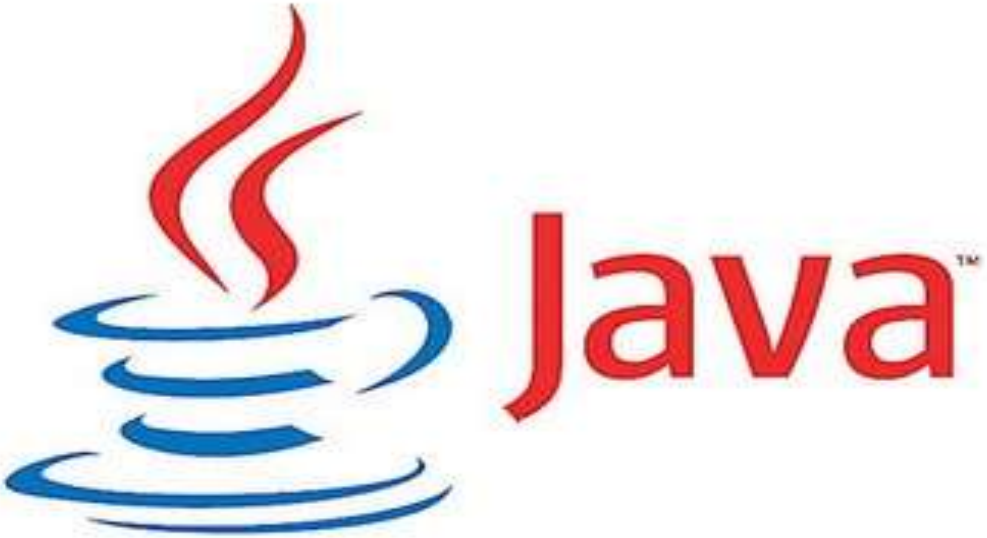
Bharathidasan University

Centre for Differently Abled Persons
Tiruchirappalli - 620024.

- Programme Name : Bachelor of Computer Applications
- Course Code : 23UCACC04
- Course Title : Programming in Java
- Semester : IV
- Unit : Unit I
- Compiled by : Dr. M. Prabavathy
  Associate Professor

  Ms. M. Hemalatha
  Guest Faculty


Java

INTRODUCTION TO JAVA

Introduction to JAVA

- Java is a **Object Oriented programming** language
- Java works on different platforms (Windows, Mac, Linux)
- It is easy to learn and simple to use
- It is open-source and free

Applications of JAVA

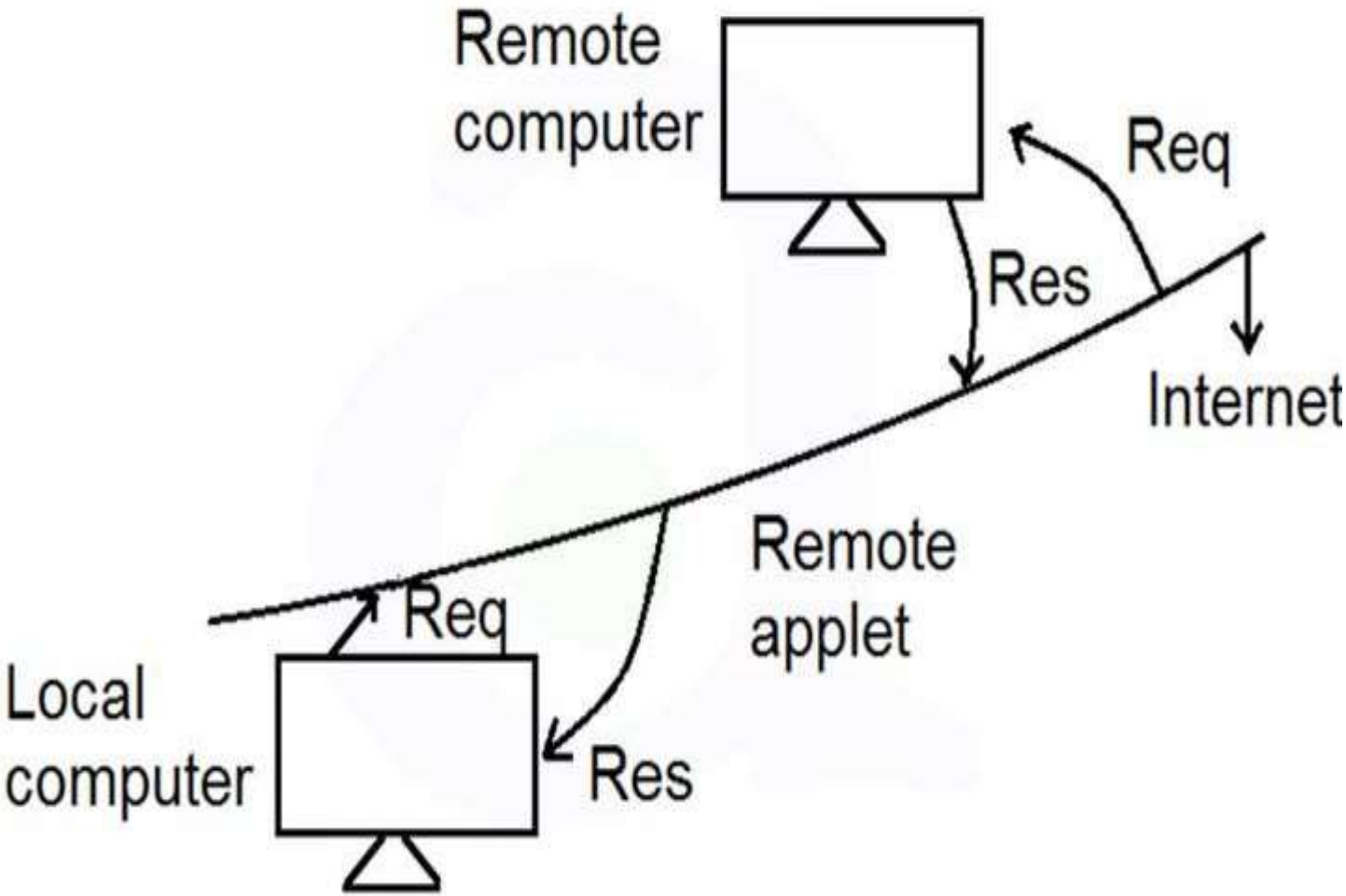
- Mobile application (Android applications)
- Desktop applications
- Web applications
- Games
- Database connection

Top Companies using Java

Java and Internet

- Java is associated with the internet because of the **first application program** is written in Java was **hot Java**.

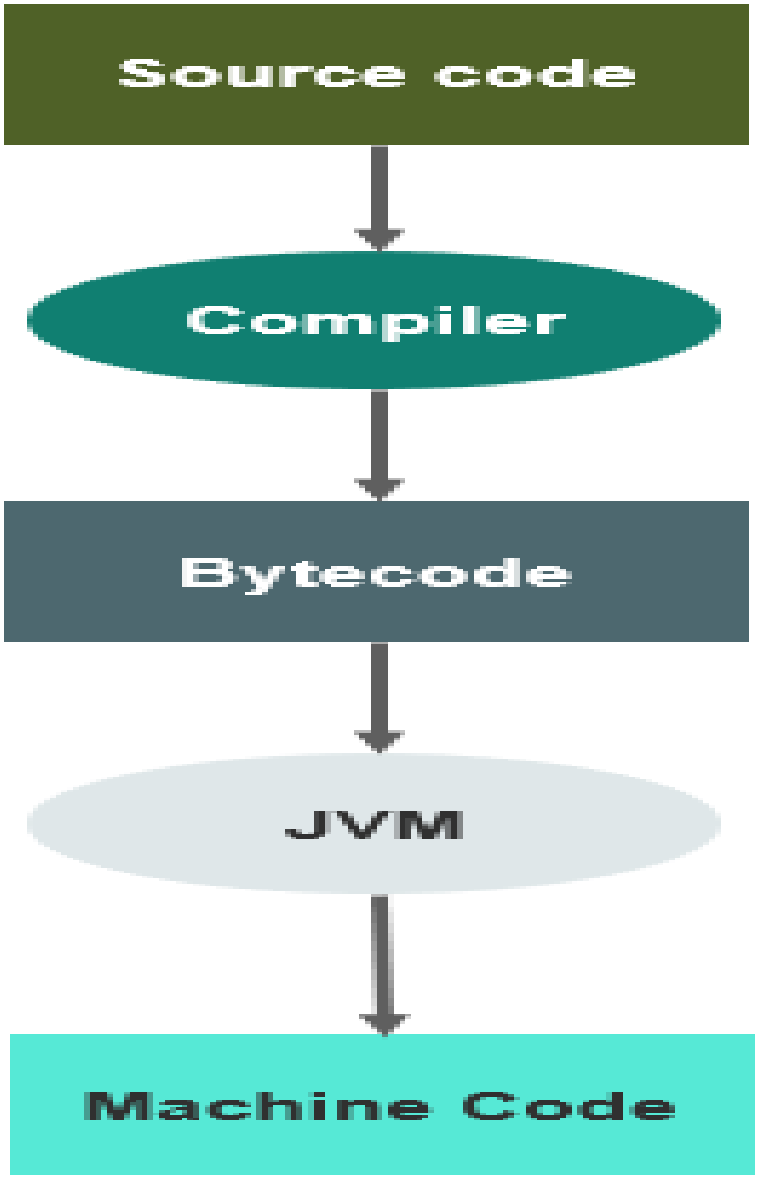
- Internet users can use Java to create **applet programs** & run then locally using a Java-enabled browser such as hot Java.

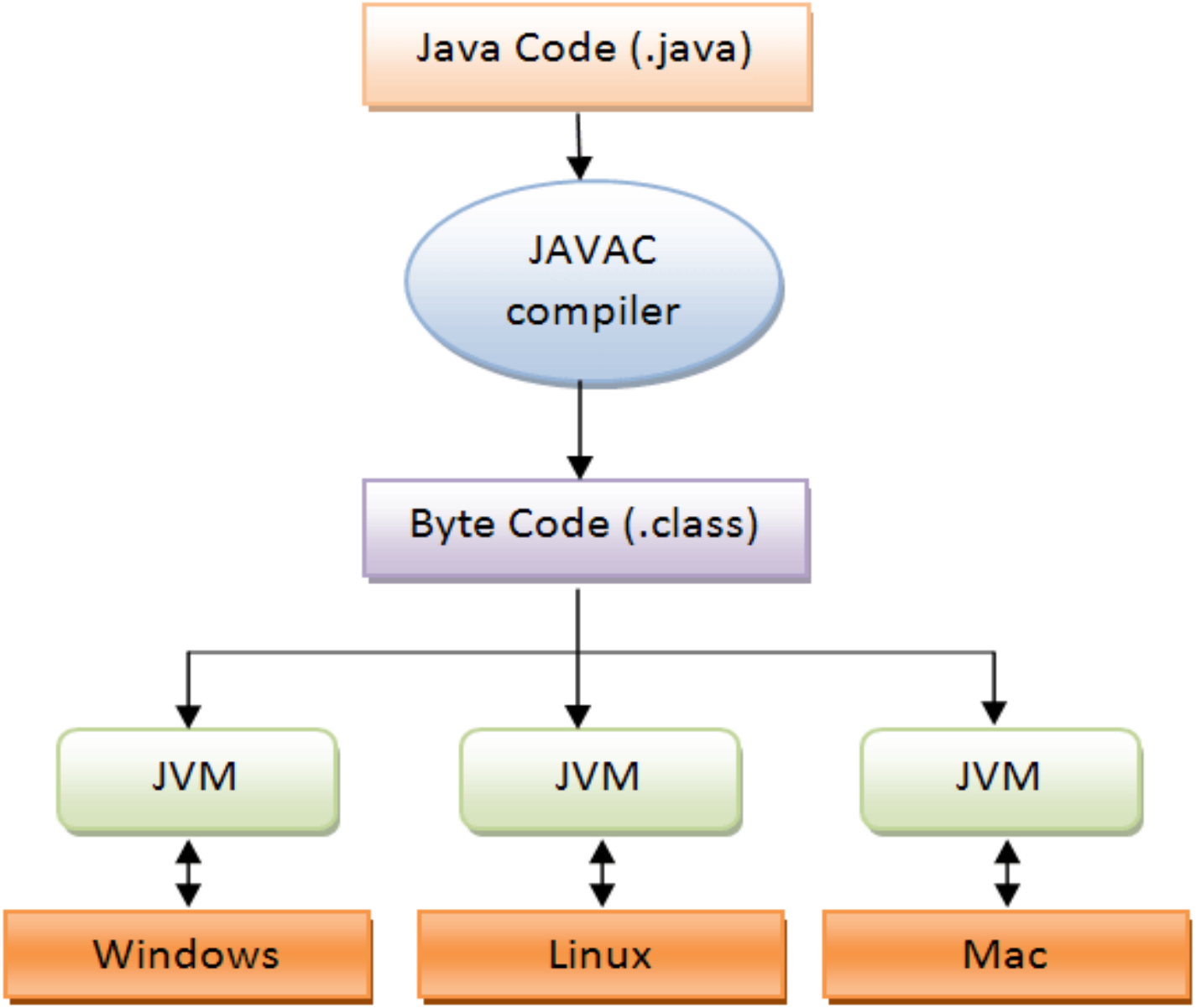

Remote
computer
Req
Res
Internet
Remote
applet
Req
Local
computer
Res

Byte Codes

- Byte code in Java is **platform-independent**
- When Java program is compiled, bytecode is generated.
- When a Java program is executed, the compiler compiles that piece of code.

- Thus Byte code generates **. class** file for every Java program.
- The bytecode is a non-runnable code that requires an interpreter.
- This is where JVM plays an important part.


Source code
Compiler
Bytecode
JVM
Machine Code


Java Code (.java)
JAVAC compiler
Byte Code (.class)
JVM
JVM
JVM
Windows
Linux
Mac

Features of Java

Java Development Kit(JDK)

- **The Java Development Kit** (JDK) is a software development environment which is used to **develop java applications and applets**.

- JDK = JRE + Development tool.
  (JRE →Java Runtime Environment)

- JDK is a package of tools for developing Java-based software

- The **JDK** is one of **three core** technology packages used in Java programming, along with the **JVM (Java Virtual Machine)** and the **JRE (Java Runtime Environment).**

- The **JVM** is the Java platform component that **executes programs**.
- The **JRE** is the on-disk part of Java that **creates the JVM**.
- The **JDK** allows developers to create Java programs that can be **executed and run by the JVM and JRE.**

The JDK contains

- a private Java Virtual Machine (JVM)
- interpreter/loader (Java),
- a compiler (javac),
- an archiver (jar),
- a documentation generator (Javadoc)

Java Character Set

- The Character set in java is a **set of alphabets, letters and some special characters.**

- The Character set consists of

  - Alphabets – a, b, c.....z
  - Digits – 0,1,2,3.....
  - Special Character - + _ ( ) { } []<>;
  - White spaces - tab, new line, Spaces.

OPERATORS

OPERATORS

- An operator is a symbol that operates on a value or a variable.
- There are different types of operators.

1. **Arithmetic Operators**

- It is used to perform **arithmetic/mathematical operations** on operands.

Operator	Description	Example
+	Addition	A + B
–	Subtraction	A – B
*	Multiplication	A * B
/	Division	A/ B
%	Modulus	A % B

2. Relational Operator

- A relational operator **checks the relationship** between two operands.
- If the relation is **true**, it returns **1**;
- if the relation is **false**, it returns value **0**.

Operator	Description	Example
==	Equal to	5 == 3
>	Greater than	5 > 3
<	Less than	5 < 3
!=	Not equal to	5 != 3
>=	Greater than or equal to	5 >= 3
<=	Less than or equal to	5 <= 3

3. Logical Operator

- Logical operator **returns either 0 or 1** depends on whether expression results true or false.

OPERATOR	MEANING	EXAMPLE
&&	Logical AND. True only if all operands are true	(c==5) &&(d>5)
||	Logical OR. True only if either one operand is true	(c==5) || (d>5)
!	Logical NOT. True only if the operand is 0	!(c==5)

4. Assignment Operator

- An assignment operator is used for **assigning a value to a variable.**

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

5. Increment / Decrement Operator

These operators are used to **increase or decrease by one value.**

Operator	Description	Example
++	Increment	A++
--	Decrement	A--

6. Bitwise Operator

- Bitwise operators are used to perform **bit-level operations.**

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

CONTROL STATEMENT

CONTROL STATEMENT

The statements that control the execution **flow of the program** are known as **control statements.**

1. Branching Statement
2. Looping Statement

BRANCHING STATEMENT

IF 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
- Else if Ladder

  (&)
- Switch Statement

(i) if Statement

If statement tests the condition.

It executes the if block if condition is true.

Syntax:

```
if(condition)
  {
      //code to be executed
  }
```

Example

```
int age=20;
if(age>18)
{
   System.out.println("Age is greater than18");
}
```

(ii) if-else statement

The if-else statement also tests the condition.

It executes the if block if condition is true otherwise else block is executed.

Syntax:

```
if (condition)
{
        statement 1;
}
else
{
        statement 2;
 }
```

Example

```
int number=13;
if(number %2 ==0)
{

    System.out.println("EVEN number");
}
else
{
    System.out.println("ODD number");

}
```

(iii) Else- If Ladder

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

Syntax:

```
if(condition1)
{
Statement 1;
}
else if(condition2)
{
Statement 2;
}
...
else
{
 Statement n;
}
```

Example

```
int number= -13;
if(number>0)
{
System.out.println("POSITIVE");
}
else if(number<0)
{
System.out.println("NEGATIVE");

}
else
{
  System.out.println("ZERO");
}
```

(iv) Switch Statement

Switch statement executes one statement from multiple conditions.

Syntax:

```
switch(expression)
{
case value1:
        Statement1;
        break;
case value2:
        Statement 2;
        break;
…….
default:
        Statement n;
}
```

Example

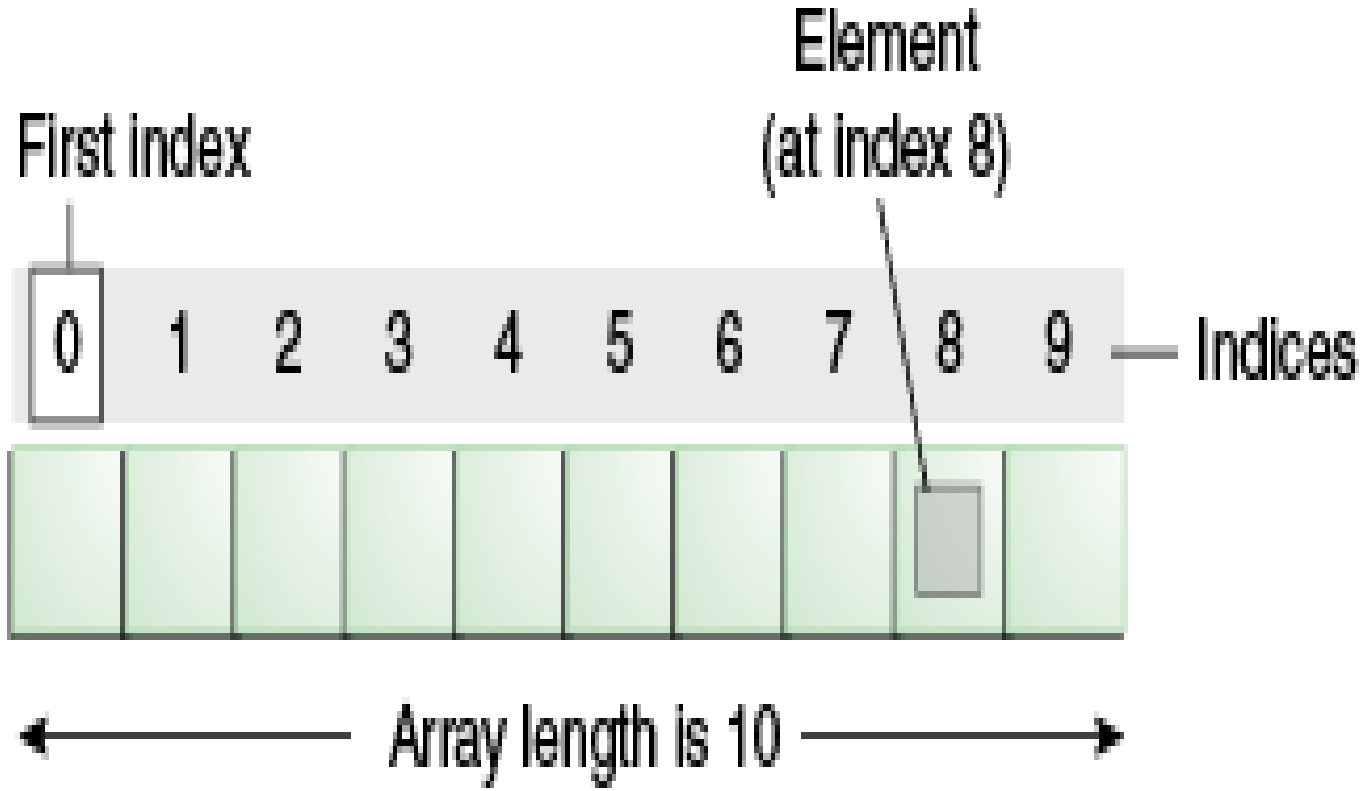
```
int number=20;
switch(number)
{
case 10:
        System.out.println("10");
        break;
case 20:
        System.out.println("20");
        break;
case 30:
        System.out.println("30");
        break;
default:
   System.out.println("Not in 10, 20 or 30");
}
```

ARRAY
&
VECTOR

ARRAY

- Array is an object which contains **elements of a similar data type**.

- Array is index-based,
  - the **first element** of the array is stored at the **0th index**
  - **2nd element** is stored on **1st index** and so on.


Element
(at index 8)
First index
0
1
2
3
4
5
6
7
8
9
Indices
Array length is 10

Types of Array

There are two types of array.

One Dimensional Array

Multidimensional Array

One Dimensional Array

A one-dimensional array behaves likes a **list of variables**.

Index value should be an integer.

Syntax:

dataType[] arr; (or)

dataType arr[];

EXAMPLE

```
public class Testarray
{
public static void main(String args[])
{
   int a[]=new int[3];
   a[0]=10;
   a[1]=20;
   a[2]=70;
   for(int i=0;i<a.length;i++)
         System.out.println(a[i]);
}
}
```

Multidimensional Array

A multidimensional array is an array containing **one or more arrays.**

Syntax:

```
dataType[][] arrayname;
```

EXAMPLE

```
public class Multiarray
{
public static void main(String args[])
{
   int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
   for(int i=0;i<3;i++)
   {
        for(int j=0;j<3;j++)
                System.out.print(arr[i][j]+" ");
   }
}
}
```

Vector

- Vector implements a **dynamic array**.
- Vector is a list that has one dimension.
- It is a **row of data.**

Vector Constructor

Vector( ) - creates a **default vector**, which has an initial size of 10.

Vector(int size) - creates a vector whose initial capacity is **specified by size.**

- **Vector(int size, int incr) -** creates a vector whose initial capacity is specified by **size and increment** is specified by incr.

- **Vector(Collection c) -** creates a vector that contains **the elements of collection** c.

```
import java.io.*;
import java.util.*;
class Vector
{
public static void main(String[] args)
{
int n = 5;
Vector<Integer> v = new Vector<Integer>(n);
    for (int i = 1; i <= n; i++)
            v.add(i);
            System.out.println(v);

            v.remove(3);
            System.out.println(v);
    for (int i = 0; i < v.size(); i++)
            System.out.println(v.get(i) + " ");
}
}
```

JAVA TOKENS

Java Tokens

- Java Tokens are the smallest individual building block or smallest unit of a Java program.
- The Java compiler uses it for constructing expressions and statements.
- Java program is a collection of different types of tokens, comments, and white spaces.

There are 5 types of tokens

1. Keywords

In java, there are **set of reserved words** that cannot be used as identifiers.

Those reserved words are called Keywords

Example:

int,

float…

2. Identifiers

Identifiers are used as **variable names**.

Example:

Sum,

Total.

3. Constants

A constant is a variable whose **value cannot change** once it has been assigned.

4. Separators

Separator is a token used to **separate two individual tokens**

Separators are also called as punctuators

Example: () {} ; ,

5. Operators

Operator is a symbol that represent specific **mathematical operation**

JAVA STATEMENT

Java Statement

- A statement **specifies an action** in a Java program.
- Java statements can be broadly classified into three categories:

1. Declaration statement

A declaration statement is used to **declare a variable.**

For Example:

int n, int a = 100.

2. Expression statement

An expression with a **semicolon at the end** is called an expression statement.

For example:

Sum = num1 + num2;

3. Control flow statement

The statements that control the execution **flow of the program** are known as control statements.

For Example:

If Statement, For Loop etc,..

LOOPING STATEMENT

LOOPING STATEMENT

Loops are used to execute a set of instructions/functions repeatedly when some conditions become true.

There are three types of loops in Java.

- while loop
- do..while loop
- for loop

While loop

- The While loop is used to iterate a part of the program several times.

Syntax:

```
Initialization;
while(condition)
{
        //code to be executed;
        Increment / decrement;
}
```

Example:

```
int i=1;
while(i<=10)
{
        System.out.println(i);
        i++;
}
```

Do..While Loop

- The do-while loop is used to iterate a part of the program several times.
- **Syntax:**

```
Initialization;
do
{
        //code to be executed;
        Increment / decrement;
}while(condition);
```

Example

```
int i=1;
do
{
        System.out.println(i);
        i++;
}while(i<=10);
```

For Loop

The for loop is used to iterate a part of the program several times.

Syntax:

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

Example

```
for(int i=1;i<=10;i++)
{
        System.out.println(i);
}
```

Break Statement

The Java break statement is used to break loop or switch statement.

It breaks the current flow of the program at specified condition.

Syntax:

```
// loop or switch case;
   Break;
```

Example

```
for (int i = 0; i < 10; i++)
{
  if (i == 4)
        {
                break;
        }
        System.out.println(i);
}
```

OUTPUT:

0

1

2

3

Continue Statement

- The Continue statement is used to continue the loop.
- It continues the current flow of the program and skips the remaining code at the specified condition.

Syntax:

```
// loop statement;
Continue;
```

Example

```
for(int i=1;i<=10;i++)
{
  if(i==5)
  {
      Continue;
  }
  System.out.println(i);
}
```

OUTPUT:

1

2

3

4

6

7

8

9

10

STRING
&
STRING BUFFER

STRING

In Java, string is a **sequence of character.** An array of characters works same as Java string.

Example:

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
String s="CDAP";
```

Some of the String Methods

1. Length()

The length of a string can be found with the length() method.

Example:

```
String txt = "CDAP";
System.out.println("The length of the txt string is: " + txt.length());
```

2. toUpperCase() and toLowerCase()

To display string in upper and lower case.

Example:

```
String txt = "Hello World";
System.out.println(txt.toUpperCase());
System.out.println(txt.toLowerCase());
```

3. Indexof()

The indexOf() method returns the index (the position) of the first occurrence of a specified text in a string (including whitespace)

Example:

```
String txt = "Have a great day";
System.out.println(txt.indexOf("great"));
```

4. String Concatenation

The + operator can be used between strings to combine them.

Example:

```
String firstName = "Good";
String lastName = "Morning";
System.out.println(firstName + " " +
                                    lastName);
```

STRING BUFFER

String Buffer class is used to create mutable (modifiable) string.

The String Buffer class in java is same as String class except it is mutable i.e. it can be changed.

Constructor of String Buffer:

(i) StringBuffer() - creates an empty string buffer with the initial capacity of 16.

(ii) StringBuffer(String str) - creates a string buffer with the specified string.

(iii) StringBuffer(int capacity) - creates an empty string buffer with the specified capacity as length

Methods in String – Buffer

- **append()** - concatenates the given argument with this string.
- **insert()** - inserts the given string with this string at the given position.
- **replace()** - replaces the given string from the specified beginIndex and endIndex.
- **delete()** - deletes the string from the specified beginIndex to endIndex.
- **reverse()** - reverses the current string.
- **capacity()** - returns the current capacity of the buffer.