**Java Introduction:-**

Language:- To Communicate we need Some Language. If we want to communicate with Machine we need Some Language.

Programming Language:- We need to use Some Languages C,C++, Java, Python

How can we communicating with a machine without programming Language

Any machine can understand 2 electronic pulses like on and off true/false and 0/1.

Programming Language:- One Person wants to communicating with the machine we need programming Language.

Types of Programming Language:-

1. Machine Understandable Language:- Binay Format
2. Assembly level Language:- Low Level Langugage needs compiler and Interpreter which helps to convert into Machine Language

MS Dos , Assembly

1. High Language

Why do we need to communicate with Machines:- To perform some Tasks

Project vs Product:- Series of Steps to prepare a product is called a project. Developers involves to build a product.

Product end result is called product which is deliverable to the market by a company.

Procedure Oriented Programming Language:- Model which is derived from structured programming. Pascal, COBOL, C

Object Oriented Programming:- Programming model which based upon the concepts of Objects. Objects contain data in the form of attributes and code in the form of Methods. Java, C++, C#, Python,

PHP, JavaScript, Ruby, Perl, Objective-C, Dart, Swift, Scala.

Overview of Java:- Java is a high-level, robust, Object-oriented Programming Language but it is not pure object oriented Language because it supports Primitive data types.

Platform independent Language.

Java codes are compiled into byte code or machine-independent code. This byte code is run on JVM (Java Virtual Machine)

The syntax is Java is almost the same as C/C++. But java does not support low-level programming functions like pointers.

Java is one of the most popular programming languages in use, especially for client-server web applications.

Java was developed by James Ghosling, Patrick Naughton, Mike Sheridan at Sun Microsystems Inc. in 1991. It took 18 months to develop the first working version.

The initial name was **Oak** but it was renamed to **Java** in 1995 as OAK was a registered trademark of another Tech company. It doesn’t have any acronym its just a name.

History of Java:-

1995 java 1.0

1996 1.0

1997 1.1

1998 1.2

2000 1.3

2002 1.4

2004 1.5

2006 1.6

2011 1.7

2014 1.8

2021 Sep 17

Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java :

1. Developing Desktop Applications
2. Web Applications like Linkedin.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems
5. Robotics and games etc.

https://www.oracle.com/java/technologies/downloads/

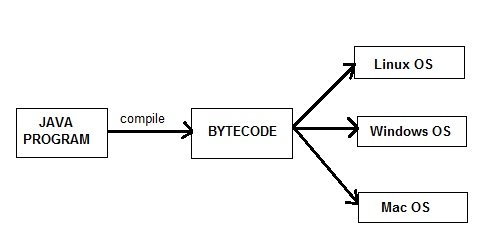
**Features of Java**

1. Simple:- Easy to learn and its syntax is quite simple, clean and easy to understand
2. Object Oriented:- In Java Everything is an object which has some data and behavior.

It has some basic oops principles.

Object,Class,Inheritance,Polymorphism,Abstraction,Encapsulation.

1. Robust:- Robust. Java is robust as it is **capable of handling run-time errors**, supports automatic garbage collection and exception handling, and avoids explicit pointer concept. Java has a strong memory management system. automatic **Garbage Collector** and **Exception Handling**.
2. Platform Independent:- Write once Run Anywhere.



1. Secure When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.
2. Architectural Neutral:- You don't have to recompile your Java source code for 32-bit or 64-bit. (So, "architecture" refers to the CPU architecture).
3. Multithreading:- It makes it possible to program that can do many tasks simultaneously. Error checking, It uses the same memory and other resources to execute the multiple threads simultaneously.
4. Portable:- Java program can be carried to any platform means

d) Portable

e) High performance

f) Robust:- Robust. Java is robust as it is **capable of handling run-time errors**, supports automatic garbage collection and exception handling, and avoids explicit pointer concept. Java has a strong memory management system.

g) Secure:- Automatic Garbage Collection and Type safety means that the compiler will validate types while compiling, and throw an error if you try to assign the wrong type to a variable.

h)Multithreaded

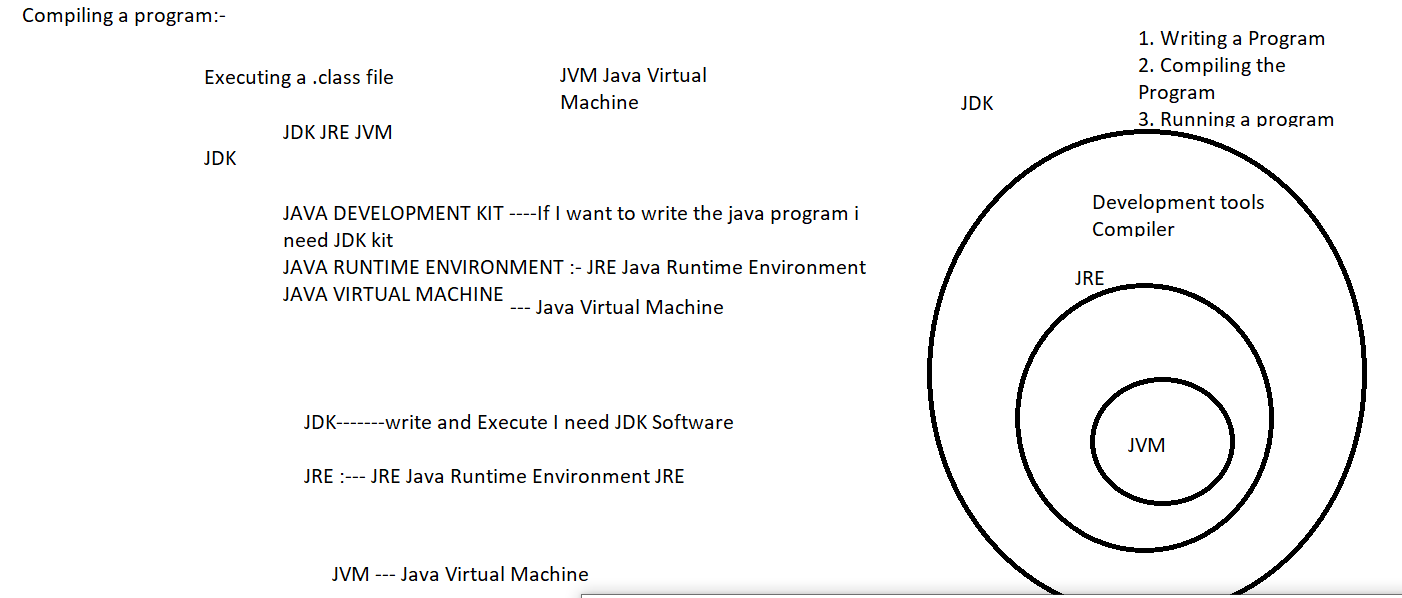
i) Distributed

j) Dynamic

k) Oops

**Java JDK, JRE, JVM**

**JVM:-** Java Virtual Machine which provides the runtime environment to execute the Java byte code. JVM does not understands the java code. We need to compile the code and provide .class file(byte code) which is understandable by jvm. It controls the execution of every program.



JVM Architecture:-



Once we compile it will be loaded into the class loader subsystem.

I have five memory Areas. Runtime Area

Class loader:- It loads the class for Execution.

Method Area:- It stores the pre class structure as constant pool.

Heap :- Objects are allocated.

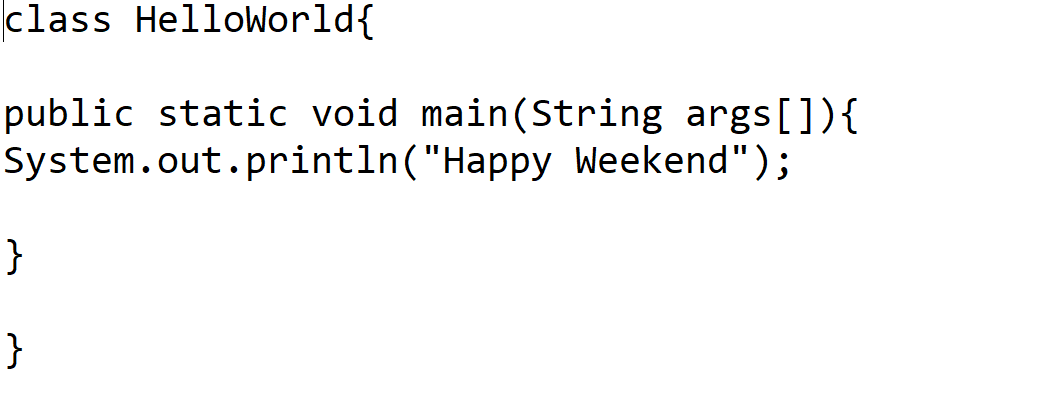
Stack:- Local variables and Partial values are stored here.

Program Register:- It stores the address of the current instruction

Native Method Stack:- Native methods stack will be available

Execution Engine:- It has multiple components. Interpreter, Compiler and native method libraries.

Java Program:-



class:- class key word is used to declare classes in java.

Public:- It is an access specifier this function is visible to all.

Static:- It is a keyword which is used to run the method without creating any object for the class. The main() method will be called by JVM.

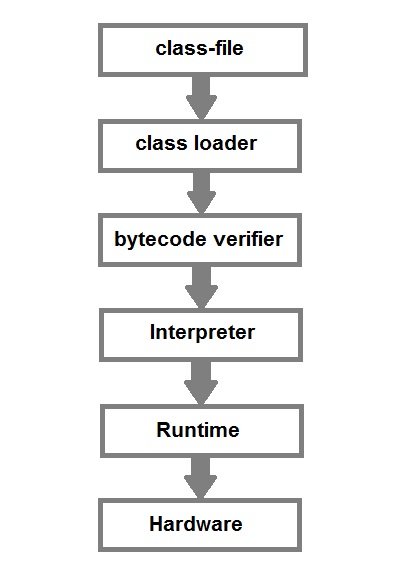
Void:- return time this method will not return anything.

Main:- This is the main() method which is used to execute all the logic from here. This method is executed hence all the logic should be written inside the method.

String args[]:- This represents an Array whose type is String and name is args

System.out.println:- This is used to print anything on the console like printf in C language.

What happens while executing a Program:-



Variables:- When we write a program it will store in the address of a computer. We cant remember the complex address so we represent those address in the name of Variables.

How to declare a variable:-

Datatype variablename;

Here datatype is what type of data we are going to store in a variable. Like float, int, char, double, long, Boolean.

Types of Variables:- Three types of Variables we have in java.

1. Instance Variables:- Instance Variables Declared
2. Local Variables:- Method level Variables declared inside a method or block.
3. Static variables:- Class Variables declared with static keyword. They are also used to declare constants along with final Key word.
4. class Student
5. {
6. String name;
7. int age;
8. static int instituteCode=1101;
9. }

It need not to be called from object.

Data Types In Java:- Datatypes specify size and Type of Values stored in an identifier.

Scope of the Variables:- Accessing the variables in a program.

Public, protected, default, private

public class HelloWorld {

public static void main(String[] args) {

int a = 10;

for(int i = 0; i<5; i++) {

System.out.println(i);

}

System.out.println("a = "+a);

System.out.println("i = "+i); // error

}

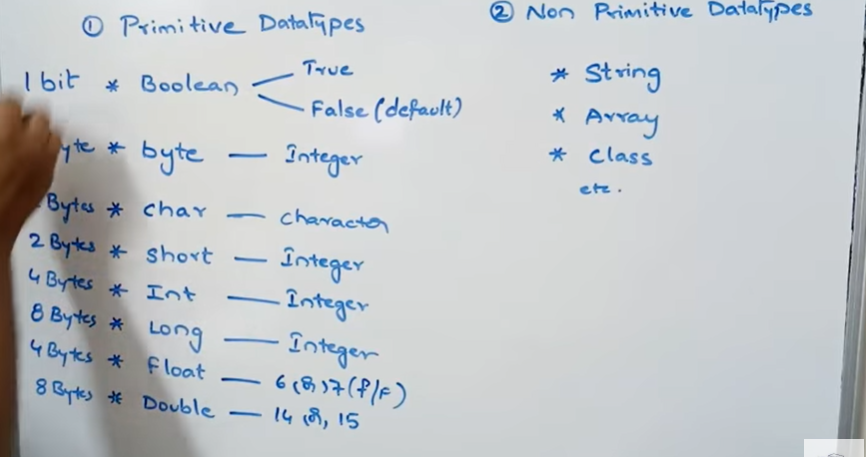
Data Types are classified into two Categories:- Type of data holding by a variable

1. Primitive Datatypes
2. Non-Primitive Datatypes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primitive Data types** | | | | | | | |
| char | boolean | byte | short | int | long | float | double |

Integer

This group includes byte, short, int, long

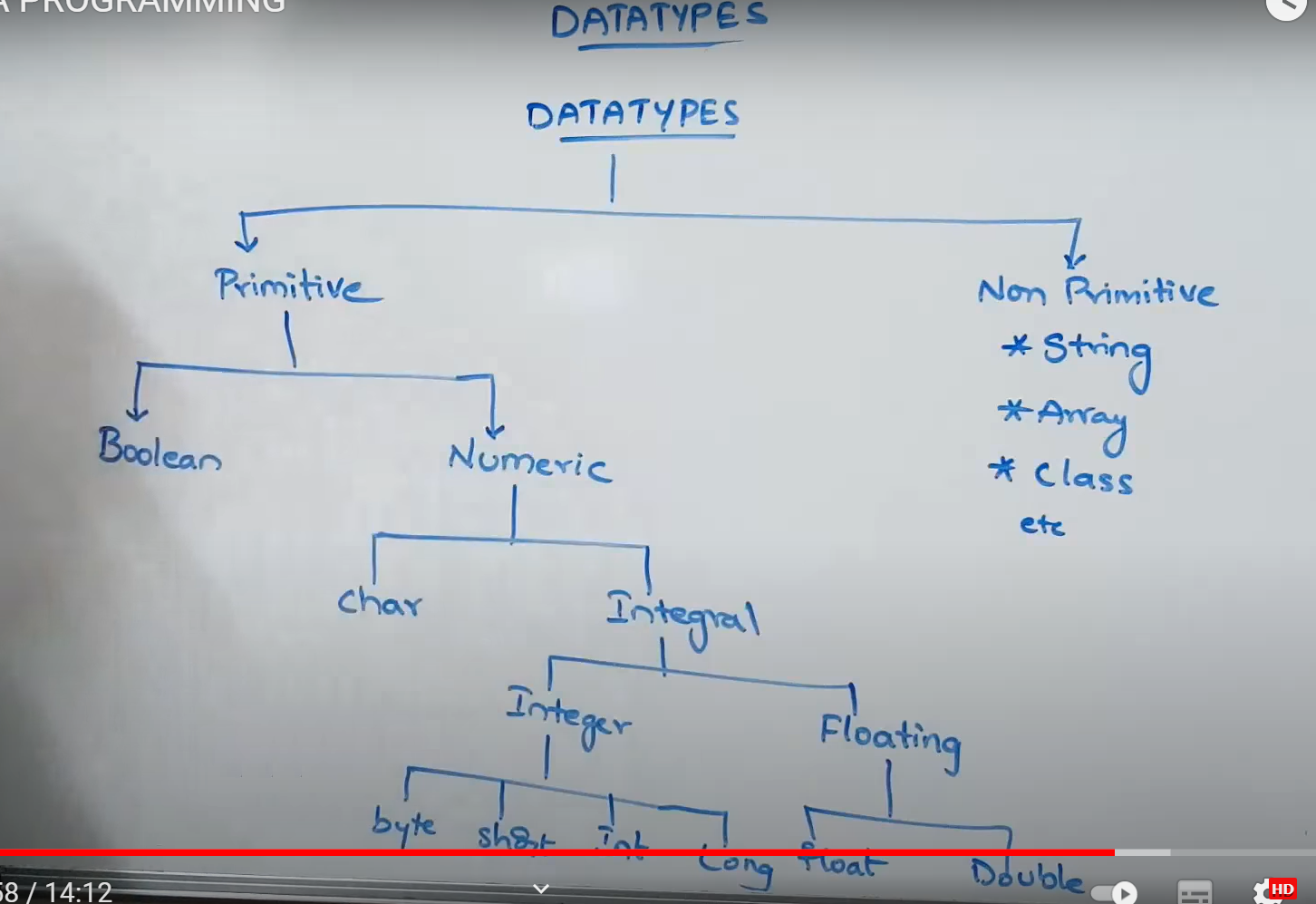


**byte :** It is 1 byte(8-bits) integer data type. Value range from -128 to 127. Default value zero. example: byte b=10;

**short :** It is 2 bytes(16-bits) integer data type. Value range from -32768 to 32767. Default value zero. example: short s=11;

**int :** It is 4 bytes(32-bits) integer data type. Value range from -2147483648 to 2147483647. Default value zero. example: int i=10;

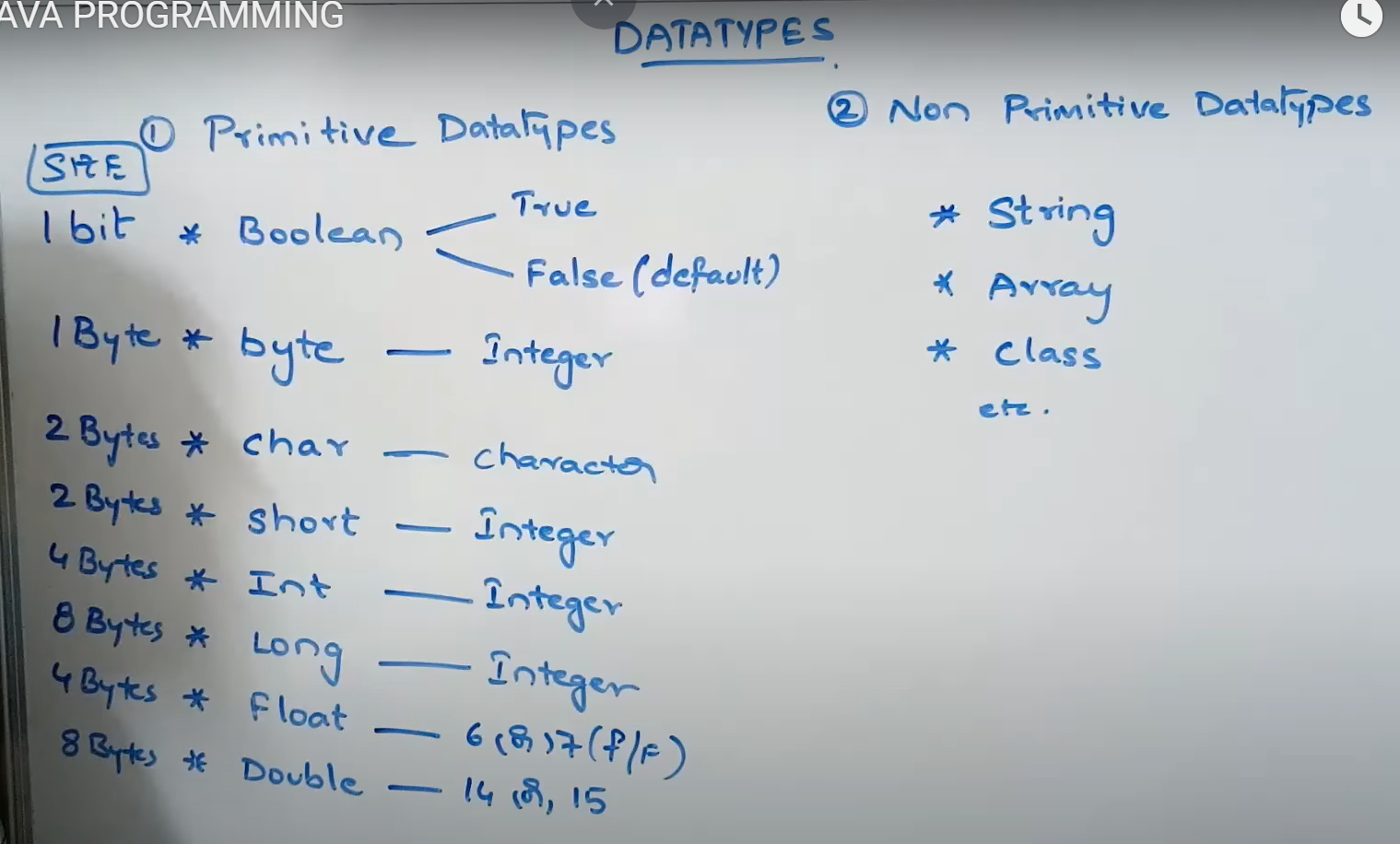
**long :** It is 8 bytes(64-bits) integer data type. Value range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. example: long l=100012;



Float:- Having Decimal part and fractional part.

Single Precision (6 or 7)

Double has double precision. (14 or 15 )



Identifiers are names that are helpful in uniquely recognizing a class, a method, or a variable. Certain rules are to be followed in java while we are defining an identifier or the compiler.

public class MainClass {

public static void main(String[] args) {

int var1 = 99;

double var2 = 2.0;

System.out.println("Hello World!");

}

}

Rule 1:

Identifiers can only contain alphanumeric characters [a-z] [A-Z] [0-9] , dollar sign ($) and underscore ( \_ ). No other character is allowed.

Rule 2:-

Cant start with numeric

Rule3:- Should not contain a space in their name.

Rule 4:- Case Sensitive

Rule 5:- length 4-15 characters

Rule 6:- Reserve key words cant be a Identifiers

Below is the table of references for all the 53 keywords in Java:

|  |  | **Keywords** |  |  |
| --- | --- | --- | --- | --- |
| abstract | Default | goto | package | this |
| assert | Do | if | private | throw |
| boolean | Double | implements | protected | throws |
| break | Else | import | public | transient |
| byte | Enum | int | return | true |
| catch | Extends | interface | short | try |
| char | False | instanceof | static | void |
| class | Final | long | strictfp | volatile |
| const | Finally | native | super | while |
| continue | Float | new | switch |  |
| case | For | null | synchronized |  |

Static Block:- Static key is mainly used to manage the memory management. It can be used with Variables, methods, classes Blocks and nested classes.

Static block in a program is set of statements those are executed by the jvm before main method.

### **Initializer Block in Java**

Typecasting:- Converting one data type to another datatype value is called Typecasting.

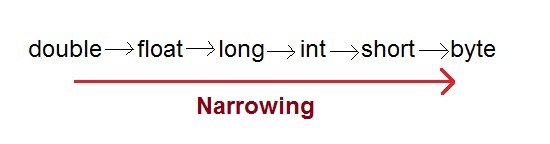
Int x=10;

Byte y=(byte)x;

Two types of Casting will be happen in java

1. Widening Casting( Implicitly will be done)
2. Narrowing casting (Explicitly done)

Byte->short-> int-> long-> float-> double



**Widening or Automatic type converion**

Automatic Type casting take place when, the two types are compatible

the target type is 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);

}

}

public class Test

{

public static void main(String[] args)

{

double d = 100.04;

long l = (long)d; //explicit type casting required

int i = (int)l; //explicit type casting required

System.out.println("Double value "+d);

System.out.println("Long value "+l);

System.out.println("Int value "+i);

}

}

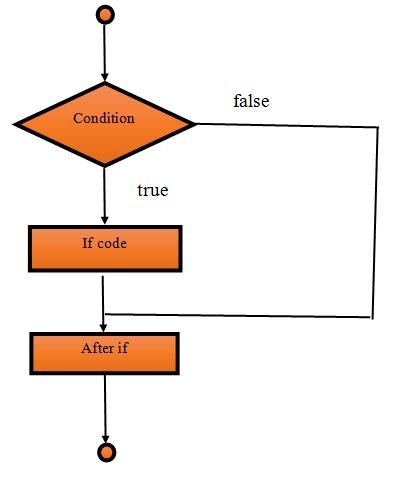
Conditional Statements:-

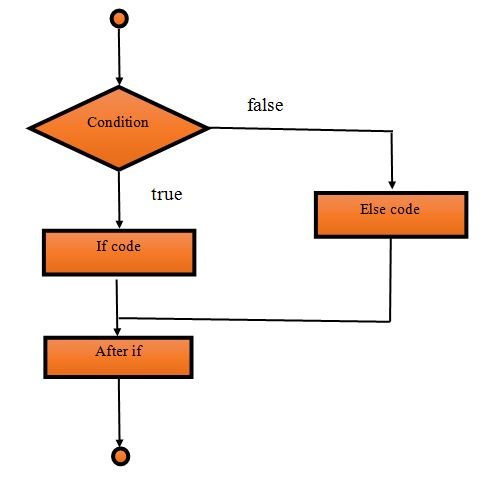
If statement is used for testing the condition . If the condition matches the statement it returns true else return false.

There are four types of if statement in Java:

1. if statement
2. if-else statement
3. if-else-if ladder
4. nested if statement

if statement is single condition statement it executes when condition is true.





Operators in Java:-

Operator is a symbol which tells the compiler to perform some operation.

Operators are always an essential part of a programming language. Java operators are divided into the following categories

1. Arithmetic Operators (+,-,\*,/,%,++,--)
2. Relational Operators ( ==, !=, >, <, <=, >=)
3. Logical Operators (&&,||, !)
4. Bitwise Operators ( &, |, ^, <<, >>) ^ It returns true only when both operands are diff otherwise return false.
5. a = 0001000
6. b = 2
7. a << b = 0100000
8. a >> b = 0000010
9. Assignment Operators (=,+=,-=,\*=,/=,%=,
10. Conditional Operators (Ternary Operator)

It is short alternate of if-else statement

Condition ?true ? false;

class Conditional\_operators1{

public static void main(String as[])

{

int a, b;

a = 20;

b = (a == 1) ? 30: 40;

System.out.println( "Value of b is : " + b );

b = (a == 20) ? 30: 40;

System.out.println( "Value of b is : " + b );

}

}

1. Misc Operators :- instanceOf operator which returns true or false to check the instance of a particular class or interface.

class instanceof\_operators1{

public static void main(String as[])

{

String a = "Studytonight";

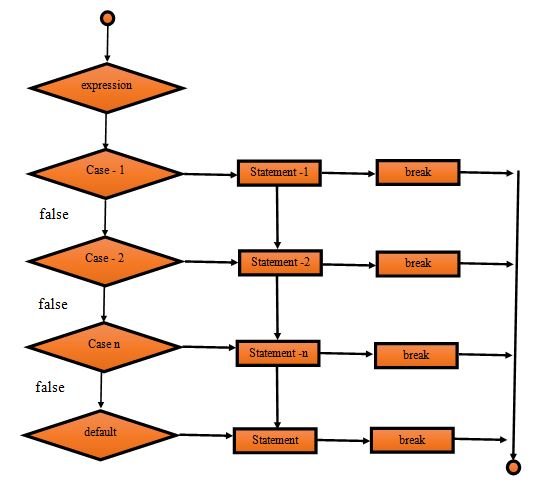
boolean b = a instanceof String;

System.out.println( b );

}

}

Switch Statement:- Switch statement is used to execute the one statement from multiple conditions. It is similar to if-else-if ladder



switch(expression)

{

case value1:

//code for execution;

break; //optional

case value2:

// code for execution

break; //optional

......

......

......

......

Case value n:

// code for execution

break; //optional

default:

code for execution when none of the case is true;

}

public class SwitchDemo1{

public static void main(String[] args)

{

int day = 3;

String dayName;

switch (day) {

case 1:

dayName = "Today is Monday";

break;

case 2:

dayName = "Today is Tuesday";

break;

case 3:

dayName = "Today is Wednesday";

break;

case 4:

dayName = "Today is Thursday";

break;

case 5:

dayName = "Today is Friday";

break;

case 6:

dayName = "Today is Saturday";

break;

case 7:

dayName = "Today is Sunday";

break;

default:

dayName = "Invalid day";

break;

}

System.out.println(dayName);

}

}

Loops In Java:-

Loops allows to iterate no of statements in no of times.

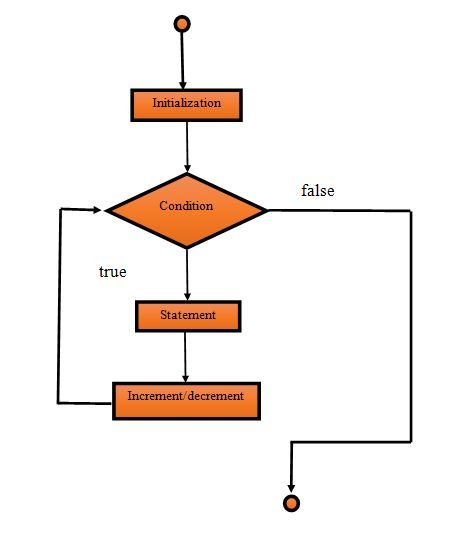
Looping is executing a block of code till the specific condition is true.

The common use of looping statements is to execute the repetitive tasks.

Java provides three types of loops

1. For loop
2. While loop
3. Do while loop

For loop if we know the repetitive times is fixed we will go for loop



For each loop:- It is used to traverse the list of elements.

for( type variable: list){

syso(i);

}

public class ForEachDemo1

{

public static void main(String[] args)

{

inta[]={20,21,22,23,24};

for(int i:a)

{

System.out.println(i);

}

}

}

While Loop:-

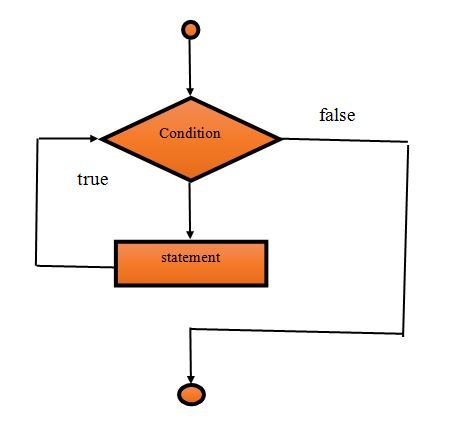
If we don’t know the specific termination No of iterations is not fixed we will go for while loop.

while(condition)

{

//code for execution

}



Do-while loop:-

In this the block of code executed atleast once before checking the condition. If the condition fais.

Do{

}while(condition);

Break and Continue Statement;\_

Break;

Continue;

Java Arrays:- It is a collection of similar data types. Array is a container that holds the homogenious elements . Array starts with zero index and go to n-1 when it has length of n.

Features of Array

It is always indexed. Index begins from 0.

It is a collection of similar data types.

It occupies a contiguous memory location.

It allows to access elements randomly.

Single Dimensional Array:-

datatype[] arrayName;;

or

datatype arrayName[];

public class ForDemo1

{

public static void main(String[] args)

{

int n, i;

n=2;

for(i=1;i<=10;i++)

{

System.out.println(n+"\*"+i+"="+n\*i);

}

}

}

Ways to create an Object in Java:- Object is an real time entity. Java is an object oriented Language everything revolves into an object. An object is an runtime entity of class and it is used to access the variables and methods of the class.

1. New Keyword
2. New instance
3. Clone method
4. Deserialization
5. NewInstance() method

New keyword:- It is the popular and common way to create an object

It is used to instantiate the instance variables. Using this method user or system defined default constructor will be called that initializes the instance variables. A new keyword creates a memory in heap area.

public class NewKeyword

{

String s = "studytonight";

public static void main(String as[])

{

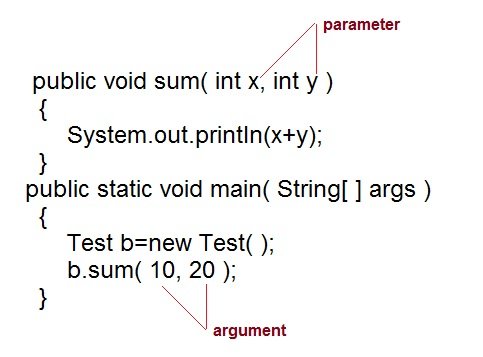
NewKeyword a = new NewKeyword();

System.out.println(a.s);

}

}

**Parameter Vs Argument in a method:-**



**Constructor:-** Constructor is a special type of method which is used to initialize the object. Every class has its own Constructor either implicitly or explicitly.

A constructor of same name as Class name.

Public Car(){};

Constructor must have no explicit return type.

Constructor in Java can not be abstract, static, final or synchronized. These modifiers are not allowed for constructor.

Type of Constructors:-

1. Default constructor
2. Parameterized Constructor

Default constructor it does not have any parameters. Default constructor does not have any parameter. Default constructor can be either user defined or provided by JVM.

Constructor Chaining:- Calling one constructor from another constructor is called Constructor chaining. We will use the this() keyword to call one constructor to another constructor.

class Test

{

Test()

{

this(10);

}

Test(int x)

{

System.out.println("x="+x);

}

public static void main(String arg[])

{

Test object = new Test();

}

}

Private Constructor:- Private constructors are used to prevent class being instantiated. Declaring private constructor restricts to create object of that class.

We use private constructor to create a Singleton Class.

Singleton Class:- A class which has only one 0bject is called Singleton class.

class Test

{

Test()

{

this(10);

}

Test(int x)

{

System.out.println("x="+x);

}

public static void main(String arg[])

{

Test object = new Test();

}

}

How can we make a class as a singleton class:-

**Access Modifiers:-** Access Modifiers are keywords those are used to set the Accessibility. An Access modifier restricts a class, method, constructor, data member and method in another class

Default: Default has scope only inside the same package

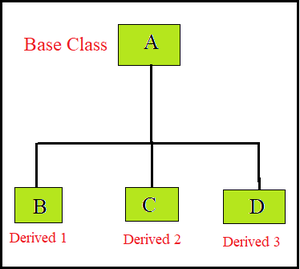
Public: Public has scope that is visible everywhere

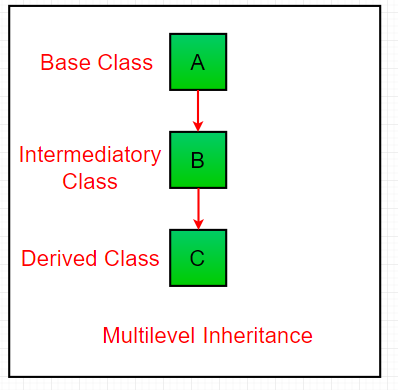
Protected: Protected has scope within the package and all sub classes outside the class as well.

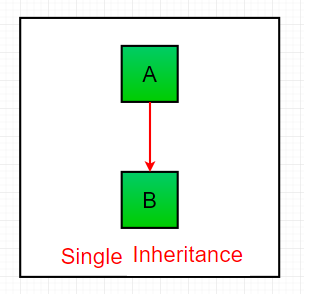
Private: Private has scope only within the classes

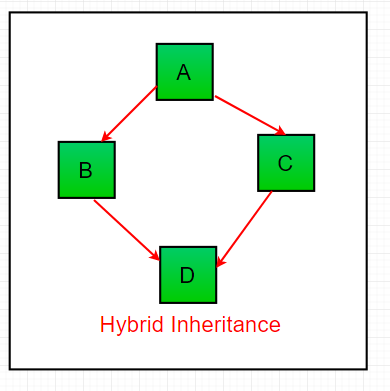
Java also supports many non-access modifiers, such as **static, abstract, synchronized, native, volatile, transient** etc. We will cover these in our other tutorial.

Inheritance:-

 Hierarchial Inheritance







Private Memebers Inheritance:-

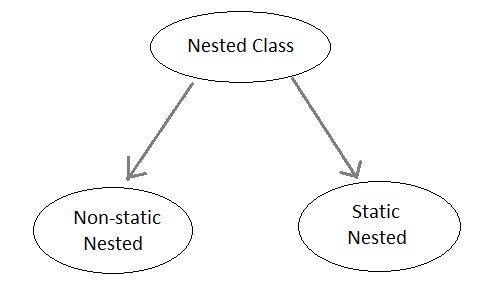
Private members of super class can’t be inherited from Subclass.

Nested classes:- A class which is defined inside a class is called Nested class.

Two types:-

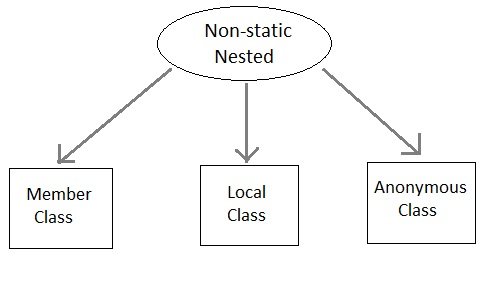
Static nested class:- declared with static keyword.

Non-static Inner class:- Member class, Local class, Anonymous Inner class



Static Nested Class

If the nested class i.e the class defined within another class, has static modifier applied in it, then it is called as static nested class. Since it is, static nested classes can access only static members of its outer class i.e it cannot refer to non-static members of its enclosing class directly. Because of this restriction, static nested class is rarely used.



**String Handling:-** String is an object that represents sequence of characters. Which is located in java.lang package

In java whatever we are creating is an object of String class. String is immutable once it is created cant be changed.





What is an immutable object?

An object whose state cant be changed once it is created known as immutable object.

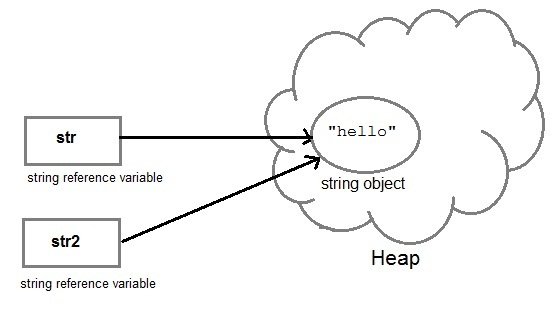
Creating a String

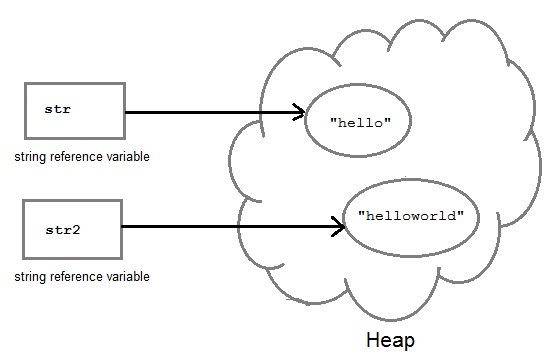
Create in no of ways String literal

String str=”AshokIT”;

Using new keyword:- String s1 = new String("Hello Java");

Strings are stored in special memory String constant pool inside the heap memory.





Concat() method:-

public class Demo{

public static void main(String[] args) {

String s = "Hello";

String str = "Java";

String str1 = s.concat(str);

System.out.println(str1);

}

}

String comparison:-

Equals() method:-It compares the content of the Strings

===

compareTo() method

### **Using == operator**

The double equal (==) operator compares two object references to check whether they refer to same instance. This also, will return **true** on successful match else returns false.

public class Demo{

public static void main(String[] args) {

String s1 = "Java";

String s2 = "Java";

String s3 = new String ("Java");

boolean b = (s1 == s2); //true

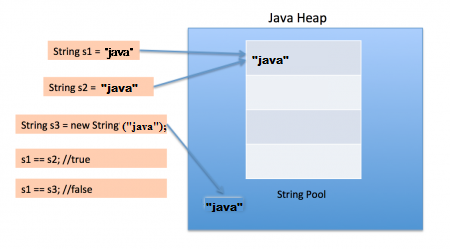
System.out.println(b);

b = (s1 == s3); //false

System.out.println(b);

}

}



compareTo() method returns integer value.

Functions:- charAt(), euqalsIgnoreCase(),indexOf(),length(),replace(),substring(),toLowercase(),toUppercase(),valueOf(),toString(),trim(),

public class Demo {

public static void main(String[] args) {

String a1 = String.format("%d", 125);

String a2 = String.format("%s", "studytonight");

String a3 = String.format("%f", 125.00);

String a4 = String.format("%x", 125);

String a5 = String.format("%c", 'a');

System.out.println("Integer Value: "+a1);

System.out.println("String Value: "+a2);

System.out.println("Float Value: "+a3);

System.out.println("Hexadecimal Value: "+a4);

System.out.println("Char Value: "+a5);

}

}

isEmpty()

public class JoinDemo1

{

public static void main(String[] args)

{

String s = String.join("\*","Welcome to studytonight.com");

System.out.println(s);

String date1 = String.join("/","23","01","2020");

System.out.println("Date: "+date1);

String time1 = String.join(":", "2","39","10");

System.out.println("Time: "+time1);

}

}

StringBuffer(): It creates an empty string buffer and reserves space for 16 characters.

StringBuffer(int size): It creates an empty string and takes an integer argument to set capacity of the buffer.

StringBuffer(String str): It creates a stringbuffer object from the specified string.

StringBuffer(charSequence []ch): It creates a stringbuffer object from the charsequence array.

**Exception Handling:-** Mechanism to handle the exception while executing the program. There are multiple ways to have exceptions while handling with files, programmer error, hardware failures.

Exception is a condition that occurs during program execution and lead to program termination abnormally.

Exception will abrupt the termination . Handling exception will help to run the remaining code without any problem.

Java exception handling provides a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user.

A Java Exception is an object that describes the exception that occurs in a program. When an exceptional events occurs in java, an exception is said to be thrown. The code that's responsible for doing something about the exception is called an exception handler.

Java provides to handle the exception in program.

* Try : It is used to enclose the suspected code.
* Catch: It acts as exception handler.
* Finally: It is used to execute necessary code.
* Throw: It throws the exception explicitly.
* Throws: It informs for the possible exception.

### **Types of Exceptions**

In Java, exceptions broadly can be categories into checked exception, unchecked exception and error based on the nature of exception.

* **Checked Exception**

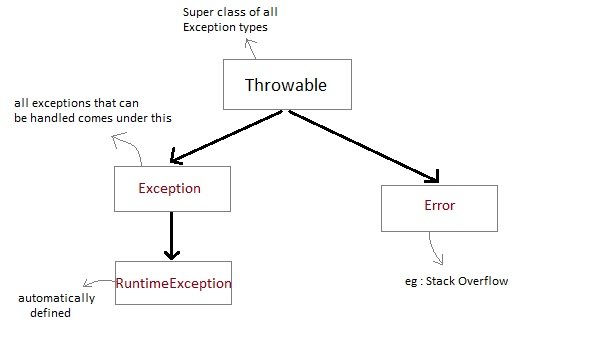
The exception that can be predicted by the JVM at the compile time for **example** : File that need to be opened is not found, SQLException etc. These type of exceptions must be checked at compile time.

* **Unchecked Exception**

Unchecked exceptions are the class that extends RuntimeException class. Unchecked exception are ignored at compile time and checked at runtime. For **example** : ArithmeticException, NullPointerException, Array Index out of Bound exception. Unchecked exceptions are checked at runtime.

* **Error**

Errors are typically ignored in code because you can rarely do anything about an error. For **example**, if stack overflow occurs, an error will arise. This type of error cannot be handled in the code.



Explain about try catch block and multi catch block.

Try with resource:-

class Demo

{

public static void main(String[] args)

{

try {

String str;

//opening file in read mode using BufferedReader stream

BufferedReader br = new BufferedReader(new FileReader("d:\\myfile.txt"));

while((str=br.readLine())!=null)

{

System.out.println(str);

}

br.close(); //closing BufferedReader stream

}

catch(IOException ie)

{

System.out.println("I/O Exception "+ie);

}

}

}

import java.io.\*;

class Demo

{

public static void main(String[] args)

{

try(BufferedReader br = new BufferedReader(new FileReader("d:\\myfile.txt")))

{

String str;

while((str = br.readLine()) != null)

{

System.out.println(str);

}

}

catch(IOException ie)

{

System.out.println("I/O Exception "+ie);

}

}

}

|  |  |
| --- | --- |
| **throw** | **throws** |
| throw keyword is used to throw an exception explicitly. | Throws keyword is used to declare an exception possible during its execution. |
| Throw keyword is followed by an instance of Throwable class or one of its sub-classes. | Throws keyword is followed by one or more Exception class names separated by commas. |
| Throw keyword is declared inside a method body. | throws keyword is used with method signature (method declaration). |
| We cannot throw multiple exceptions using throw keyword. | We can declare multiple exceptions (separated by commas) using throws keyword. |
|  |  |

class MyException extends Exception

{

private int ex;

MyException(int a)

{

ex = a;

}

public String toString()

{

return "MyException[" + ex +"] is less than zero";

}

}

class Demo

{

static void sum(int a,int b) throws MyException

{

if(a<0)

{

throw new MyException(a); //calling constructor of user-defined exception class

}

else

{

System.out.println(a+b);

}

}

public static void main(String[] args)

{

try

{

sum(-10, 10);

}

catch(MyException me)

{

System.out.println(me); //it calls the toString() method of user-defined Exception

}

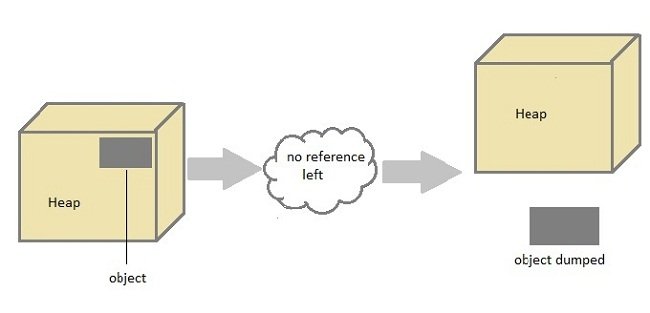
}

}

**Garbage Collection:**- Garbage collection is the process of releasing the unused memory occupied by unused objects. This is done by JVM automatically.

When a java program runs on JVM memory will be created on Heap. When there is no reference to an object it will be assumed to be no longer needed

It is an automatic process we don’t need to do anything but we can request explicitly to the JVM



How can we call

System.gc()

public class Test

{

public static void main(String[] args)

{

Test t = new Test();

t=null;

System.gc();

}

public void finalize()

{

System.out.println("Garbage Collected");

}

}

**Collections:-**

An hierarchy of multiple interfaces and classed used to manipulate the whole set of objects.

Collection in java referred to an object that collects multiple elements into a single unit.

Components of Collection:-   
It consists of

Interfaces:- Set, List, Maps

Classes:- It has multiple classes.

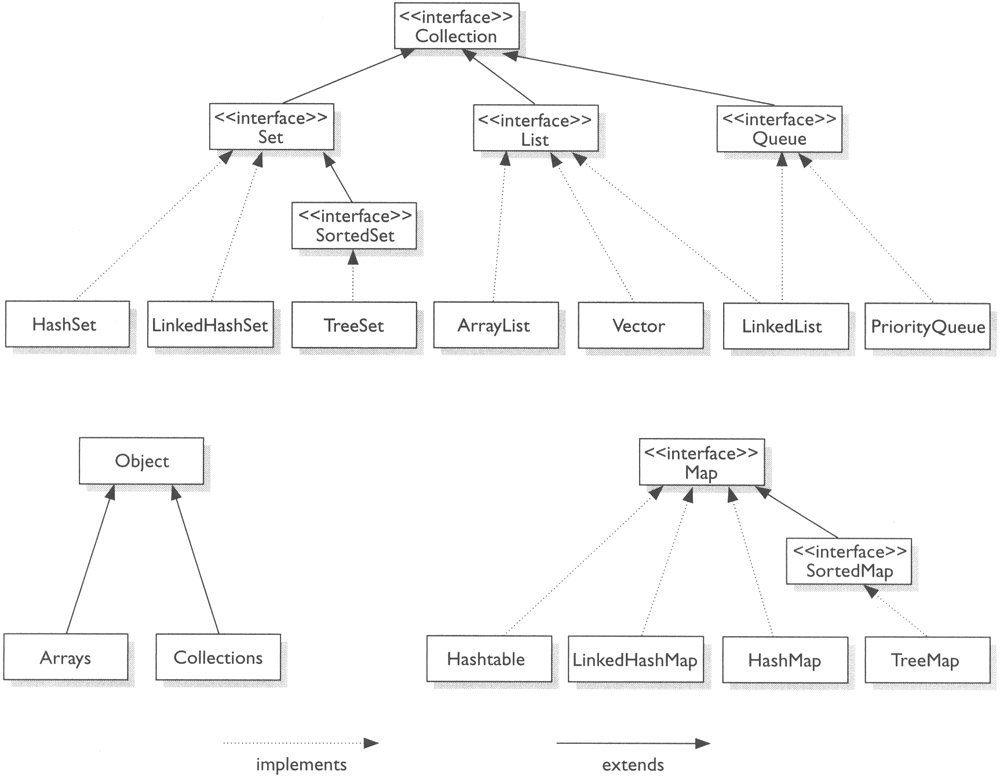
Concurrent implemented classes for highly concurrant usage.

Interfaces:-

|  |  |
| --- | --- |
| **Collection** | Enables you to work with groups of object; it is at the top of Collection hierarchy |
| **Deque** | Extends Queue to handle double ended queue. |
| **List** | Extends Collection to handle sequences list of object. |
| **Queue** | Extends Collection to handle special kind of list in which element are removed only from the head. |
| **Set** | Extends Collection to handle sets, which must contain unique element. |
| **SortedSet** | Extends Set to handle sorted set. |

### **Important Interfaces of Collection API**

|  |  |
| --- | --- |
| **Interface** | **Description** |
| **Collection** | Enables you to work with groups of object; it is at the top of Collection hierarchy |
| **Deque** | Extends Queue to handle double ended queue. |
| **List** | Extends Collection to handle sequences list of object. |
| **Queue** | Extends Collection to handle special kind of list in which element are removed only from the head. |
| **Set** | Extends Collection to handle sets, which must contain unique element. |
| **SortedSet** | Extends Set to handle sorted set. |
|  |  |



List Interface:-

ArrayList

LinkedList

Vector

Stack

Iterable

Collection

Duplicate Unique Elements FIFO Key,value(AaadharcardNo,Person)

List Set Queue (FIFO) Map (key,value)

HashSet PriorityQueue HashMap

ArrayList LinkedHashSet LinkedHashMap

LinkedList TreeSet TreeMap

Vector SortedSet HashTable

Stack( NavigableSet SortedMap

(LIFO)

When ur Collection is going allow the Duplicate elements then i will go for List Interface

10 elements --->

ArrayList<String> list=new ArrayList<String>(); 10 spaces

5 elements ---- 10+5 ====15 elements

list.add("Satti");

list.add("babu");

list.add("Ravi");

list.add("Ramesh");

list.add("Srujana");

list.remove(3);

list.get(3);

String [] list2=new String[15];

list2=list;

LinkedList<String> linkedList=new LinkedList<String>

|  |
| --- |
| //A Java program to demonstrate Comparator interface  import java.io.\*;  import java.util.\*;    // A class 'Movie' that implements Comparable  class Movie implements Comparable<Movie>  {      private double rating;      private String name;      private int year;        // Used to sort movies by year      public int compareTo(Movie m)      {          return this.year - m.year;      }        // Constructor      public Movie(String nm, double rt, int yr)      {          this.name = nm;          this.rating = rt;          this.year = yr;      }        // Getter methods for accessing private data      public double getRating() { return rating; }      public String getName()   {  return name; }      public int getYear()      {  return year;  }  }    // Class to compare Movies by ratings  class RatingCompare implements Comparator<Movie>  {      public int compare(Movie m1, Movie m2)      {          if (m1.getRating() < m2.getRating()) return -1;          if (m1.getRating() > m2.getRating()) return 1;          else return 0;      }  }    // Class to compare Movies by name  class NameCompare implements Comparator<Movie>  {      public int compare(Movie m1, Movie m2)      {          return m1.getName().compareTo(m2.getName());      }  }    // Driver class  class Main  {      public static void main(String[] args)      {          ArrayList<Movie> list = new ArrayList<Movie>();          list.add(new Movie("Force Awakens", 8.3, 2015));          list.add(new Movie("Star Wars", 8.7, 1977));          list.add(new Movie("Empire Strikes Back", 8.8, 1980));          list.add(new Movie("Return of the Jedi", 8.4, 1983));            // Sort by rating : (1) Create an object of ratingCompare          //                  (2) Call Collections.sort          //                  (3) Print Sorted list          System.out.println("Sorted by rating");          RatingCompare ratingCompare = new RatingCompare();          Collections.sort(list, ratingCompare);          for (Movie movie: list)              System.out.println(movie.getRating() + " " +                                 movie.getName() + " " +                                 movie.getYear());              // Call overloaded sort method with RatingCompare          // (Same three steps as above)          System.out.println("\nSorted by name");          NameCompare nameCompare = new NameCompare();          Collections.sort(list, nameCompare);          for (Movie movie: list)              System.out.println(movie.getName() + " " +                                 movie.getRating() + " " +                                 movie.getYear());            // Uses Comparable to sort by year          System.out.println("\nSorted by year");          Collections.sort(list);          for (Movie movie: list)              System.out.println(movie.getYear() + " " +                                 movie.getRating() + " " +                                 movie.getName()+" ");      }  } |

Output :

Sorted b