# Basic Java

## How u rate you in java?

I am quite comfortable in Core Java base on automation prospective. I consider myself 7 out of 10 in core Java. I am learning Java streams. I would give 6 out of 10 in Java streams. I am expert in Java collections, so I consider myself at 9 out of 10 in Java collections framework.

## What is Java?

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

## Mention some features of Java?

Some of the features which play important role in the popularity of java are as follows:

* Simple: Java is easy to learn. Even though Java is based on C++ , it was developed by eliminating poor programming practices of C++.
* Object-Oriented: Java is a object oriented programming language. Everything in Java is an Object.

[Learn more here.](https://www.softwaretestingmaterial.com/oops-concept-in-java/)

**Portable:**

Java run time environment uses a bytecode verification process to make sure that code loaded over the network doesn’t violate Java security constraints.

**Platform independent:**

Java is platform independent. Java is a write once, run anywhere language. Without any modifications, we can use a program in different platforms.

**Secured:**

Java is well known for its security. It delivers virus free systems.

**High Performance:**

Java enables high performance with the use of JIT (Just-In-Time) compilers

**Multithreaded:**

A Java Multithreaded feature allows us to write programs that can perform many tasks simultaneously. Multithreading concept of Java shares a

Common memory area. It doesn’t occupy memory for each thread.

## Why we need to compile a program in java?

To transform a [program](https://www.webopedia.com/TERM/P/program.html) written in a [high-level programming language](https://www.webopedia.com/TERM/H/high_level_language.html) from [source code](https://www.webopedia.com/TERM/S/source_code.html) into [object code](https://www.webopedia.com/TERM/O/object_code.html). [Programmers](https://www.webopedia.com/TERM/P/programmer.html) write programs in a form called source code. Source code must go through several steps before it becomes an executable program. The first step is to pass the source code through a [compiler](https://www.webopedia.com/TERM/C/compiler.html), which translates the high-level language [instructions](https://www.webopedia.com/TERM/I/instruction.html)into object code.

The final step in producing an executable program -- after the compiler has produced object code -- is to pass the object code through a [linker](https://www.webopedia.com/TERM/L/linker.html). The linker combines [modules](https://www.webopedia.com/TERM/M/module.html) and gives real values to all symbolic [addresses](https://www.webopedia.com/TERM/A/address.html), thereby producing [machine code](https://www.webopedia.com/TERM/M/machine_language.html).

## What is an Object in Java?

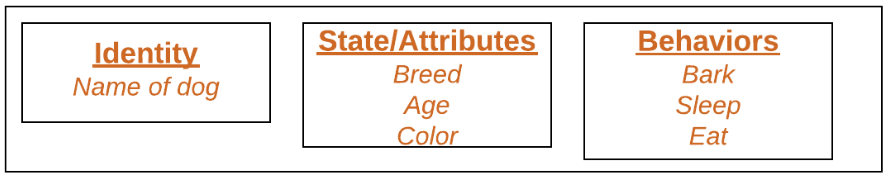
It is a basic unit of Object Oriented Programming and represents the real life entities. And it has some properties and can perform some task. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

**State:**

It is represented by attributes of an object. It also reflects the properties of an object.

1. **Behavior:** It is represented by methods of an object. It also reflects the response of an object with other objects.
2. **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

Example of an object : dog



An object is an instance of a class. Objects have state (variables) and behavior (methods).

Example: A dog is an object of Animal class. The dog has its states such as color, name, breed, and behaviors such as barking, eating, wagging her tail.

|  |  |
| --- | --- |
| 1  2  3  4  5 | public class MyClass{    //Class name (MyClass) declaration      public static void main(String[] args){          MyClass obj = new MyClass(); //Object Creation      }  } |

## What Is the Life Cycle of an Object in Java?



* **Created -**memory was allocated, object placed in Eden;
* **Initialized**- object is initialized with a value and becomes ready to be used;
* **In use -**object is used for some period of time, until no references to it remain;
* **Unreachable -**after there are no references to the object it enters this state, and can be collected by the GC;
* **Finalized -**object is garbage collected and memory it occupied is freed up.

## What is a Class in Java?

A class can be defined as a collection of objects. It is the blueprint or template that describes the state and behavior of an object.

|  |  |
| --- | --- |
|  | public class MyClass{    //Class name (MyClass) declaration  int a = 9;   // Variable declaration  int b = 99;      public void myMethod(){ //Method (myMethod) declaration          int sum=a+b;      }  } |

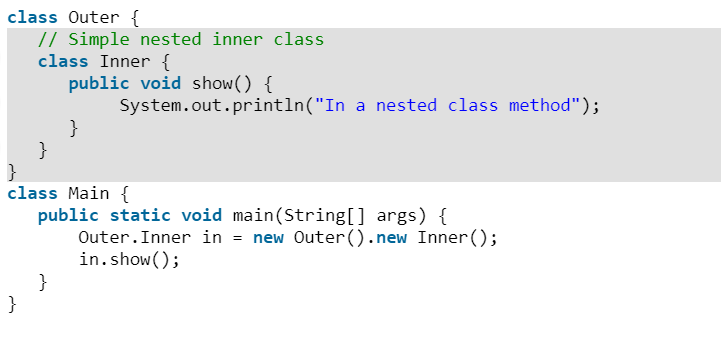
## What is Inner class in java?

Inner class means one class which is a member of another class. There are basically four types of inner classes in java.

* Nested Inner class.
* Method Local inner classes.
* Anonymous inner classes.
* Static nested classes.

## **Nested Inner class**

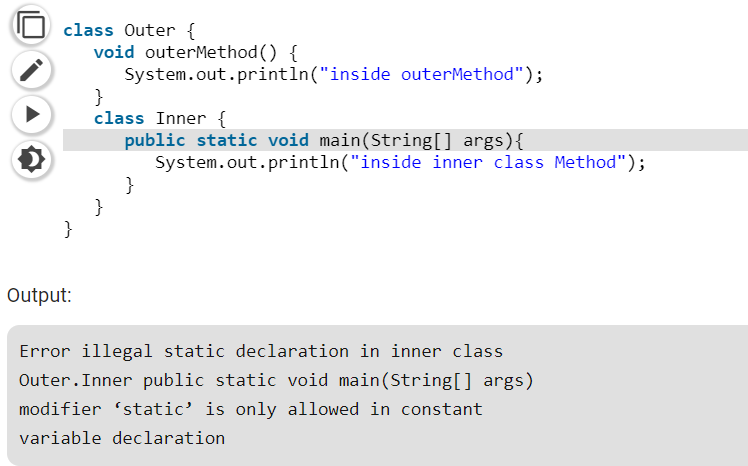
Can access any private instance variable of outer class. Like any other instance variable, we can have access modifier private, protected, public and default modifier.  
Like class, interface can also be nested and can have access specifies.



Following example demonstrates a nested class.

Output:

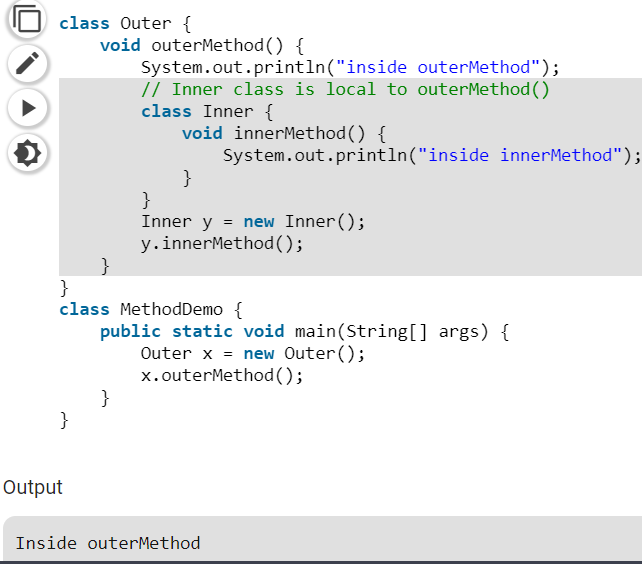
In a nested class method

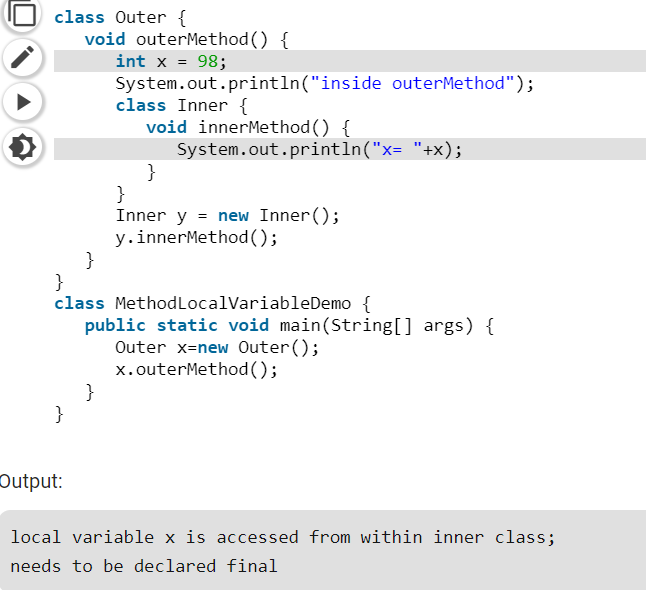
As a side note, we can’t have static method in a nested inner class because an inner class is implicitly associated with an object of its outer class so it cannot define any static method for itself. For example the following program doesn’t compile.

An interface can also be nested and nested interfaces have some interesting properties. We will be covering nested interfaces in the next post.

## **Method Local inner class’s**

Inner class can be declared within a method of an outer class. In the following example, Inner is an inner class in outerMethod().



Method Local inner classes can’t use local variable of outer method until that local variable is not declared as final. For example, the following code generates compiler error (Note that x is not final in outerMethod() and innerMethod() tries to access it)

**Note :**Local inner class cannot access non-final local variable till JDK 1.7. Since JDK 1.8, it is possible to access the non-final local variable in method local inner class.

But the following code compiles and runs fine (Note that x is final this time)

The main reason we need to declare a local variable as a final is that local variable lives on stack till method is on the stack but there might be a case the object of inner class still lives on the heap.  
Method local inner class can’t be marked as private, protected, static and transient but can be marked as abstract and final, but not both at the same time.

## **Static nested classes**

Static nested classes are not technically an inner class. They are like a static member of outer class.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| class Outer {     private static void outerMethod() {       System.out.println("inside outerMethod");     }       // A static inner class     static class Inner {       public static void main(String[] args) {          System.out.println("inside inner class Method");          outerMethod();       }     }    } |

Output

inside inner class Method

inside outerMethod

## **Anonymous inner classes**

Anonymous inner classes are declared without any name at all. They are created in two ways.  
***a)****As subclass of specified type*

In the above code, we have two class Demo and Flavor1Demo. Here demo act as super class and anonymous class acts as a subclass, both classes have a method show(). In anonymous class show() method is overridden.

## What is Interface in Java?

An interface in Java looks similar to a class but both the interface and class are two different concepts. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract. We can achieve 100% abstraction and multiple inheritances in Java with Interface.

## How many test cases can be automated in a day?

It Depends on Script and test Case and how much time we got from sprint base on that we divide to test case Executives expect answers such as, "We have 500 out of 10,000 test cases remaining," "On average, we can do 50 test cases per day, so about 10 days," or, "We are 95% complete."

## How you will avoid object cloneable?

Clone able interface and override clone () method from Object class. It is a good idea to prevent cloning in a singleton class. To prevent cloning on singleton object, let us explicitly throw CloneNotSupportedException exception in clone () method.

## What is Synchronization in Java

Synchronization in Java is an important concept since Java is a multi-threaded language where multiple threads run in parallel to complete program execution. In multi-threaded environment *synchronization of Java object or synchronization of Java class becomes extremely important*. Synchronization in Java is possible by usingJava keywords *"synchronized"* and *"volatile”*.

## What is synchronizing methods?

**Synchronized methods** enable a simple strategy for preventing the thread interference and memory consistency errors. If a Object is visible to more than one threads, all reads or writes to that Object's fields are done through the **synchronized method**.

## How to find out the part of the string from a string

## What is the final keyword in Java?

In Java, the **final keyword** can be used while declaring an entity. Using the final keyword means that the value can’t be modified in the future. This entity can be - but is not limited to - a variable, a class or a method.

## How u will return multiple values in a method?

If you are returning more than 1 value that is related, then it makes sense to encapsulate them into a class and then return an object of that class.

If you want to return unrelated values, then you can use Java's built-in container classes like Map, List, and Set etc. Check the [java.util](http://docs.oracle.com/javase/6/docs/api/java/util/package-summary.html) package's JavaDoc for more details.

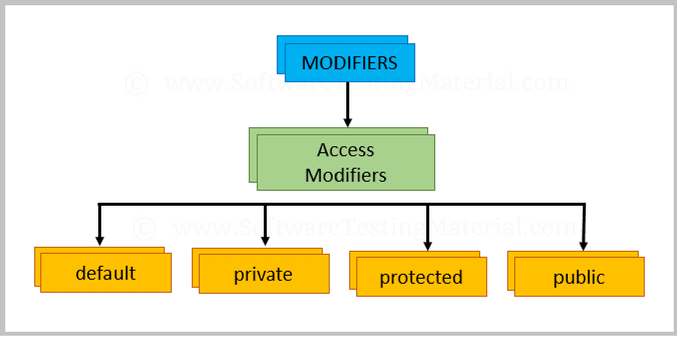
## What is the default package in java?

The default package is an unnamed package. The unnamed package contains java classes whose source files did not contain a package declaration. The purpose of default package is for convenience when developing small or temporary applications or when just beginning development.

## What datatype can be added to a List?

Any Data type can be accepted.

## What are the different access modifiers available in Java?

Access modifiers are subdivided into four types such as Default, Public, Private, Protected[](https://i1.wp.com/www.softwaretestingmaterial.com/wp-content/uploads/2018/03/Access-Modifiers.png?ssl=1)

## Default:

The scope of default access modifier is limited to the package only. If we do not mention any access modifier, then it acts like a default access modifier.

## Private:

The scope of private access modifier is only within the classes.

Note: Class or Interface cannot be declared as private

## Protected:

The scope of protected access modifier is within a package and also outside the package through inheritance only.

Note: Class cannot be declared as protected

## Public:

The scope of public access modifier is everywhere. It has no restrictions. Data members, methods and classes that declared public can be accessed from anywhere.

## What is Multiple Inheritance?

If a class implements multiple interfaces or an interface extends multiple interfaces then it is known as multiple inheritances.

We will update this post “Java Interview Questions for Selenium Testers” ASAP. Keep visiting.

## What is up casting in Java?

***Upcasting***is casting a subtype to a supertype, upward to the inheritance tree. Let’s see an example:

|  |  |
| --- | --- |
|  | Dog dog = new Dog();  Animal anim = (Animal) dog;  anim.eat();  Generally, upcasting is not necessary. However, we need upcasting when we want to write general code that deals with only the supertype. |

## What is singleTon class?

A singleton class is a class that can have only one object (an instance of the class) at a time.

To design a singleton class:

* Make constructor as private.
* Write a static method that has return type object of this singleton class. Here, the concept of [Lazy initialization](https://en.wikipedia.org/wiki/Lazy_initialization) is used to write this static method.

## Is it possible to call finally block after throwing an exception?

Yes. Finally, will always be executed.

## Is finally block always executed?

No. In case of System. exit() it won’t be executed.

## Is it possible to have auto widening and auto narrowing in java?

Auto widening is possible but auto narrowing is not.

## Why auto narrowing is not possible in java?

In auto narrowing there is always a loss of memory. So it’s not possible.

## What is recursion in java?

In Java, recursion is a process where a method with in a class calls itself inside the class.

## Why Main Method is void?

Because main method does not return a value

## How to Call Static Method or Variable?

* Direct Call
* 2 Calling By Class Name

## How to Call Non Static Method or Variable?

* First Create an object then call that method or variable.

## Can I access static method by using Object?

* Yes

# Code

## Write a program to iterate through a List?

Iterator it=myData.iterator();

**while**(it.hasNext())

{

**int** str=(**int**) it.next();

System.***out***.println(str);

}

## How we can specify data type of specific type in List?

ArrayList<String> myData=**new** ArrayList<String>();// array list can contains any value

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

{

myData.add(i);

System.***out***.println(myData.get(i));

}

System.***out***.println(myData);

## Where u has used the concept of interface and abstract in your framework?

## We are using interface on base class and use it for all test case and POM concept.

Abstraction we using on testNG Listeners.

## Count only the non-duplicate words in the string?

**public** **static** String *Str*="Test you how Test work and you";

**public** **static** **void** main(String[] args) {

String[] ar=*Str*.split(" ");

**for**(**int** i=0;i<ar.length;i++)

{

**int** count=1;

**for**(**int** j=i+1;j<ar.length;j++){

**if**(ar[i].equals(ar[j]))

{

count=count+1;

ar[j]="0";

}

}

**if**(count>1 && ar[i]!="0")

{

System.***out***.println("\*\* "+ar[i]+" \*\* This value has Total Count of "+ count);

}

}

## Write a program to print the pattern given below

|  |  |
| --- | --- |
|  | 1  1 2  1 2 3  1 2 3 4  1 2 3 4 5 |

Here is the program to print the pattern mentioned above

**for** (**int** x = 1; x <= 5; x++) {

**for** (**int** y = 1; y <= x; y++) {

System.***out***.print(y+" ");

}

System.***out***.println();

}

## Write a program to print Fibonacci Series up to count 10.

**int** n1=0,n2=1,n3,i,count=10;

System.***out***.print(n1+" "+n2);//printing 0 and 1

**for**(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

{

n3=n1+n2;

System.***out***.print(" "+n3);

n1=n2;

n2=n3;

}

## How to reverse a String in Java?

String Str="Test ffsg hhfgsk fgkkh";

String Nstr="";

**for**(**int** i=Str.length();i>0;i--)

{

System.***out***.print(Str.charAt(i-1));

}

**Or**

StringBuffer a = **new** StringBuffer(Str1);

System.***out***.println(a.reverse());

## How to find the largest value from the given Array.

**int** Highval=*Test*[0];

**int** Lowval=*Test*[0];

String res="";

**for**(**int** i=0;i<*Test*.length;i++)

{

**if**(*Test*[i]>Highval)

{

Highval=*Test*[i];

}

**if**(*Test*[i]<Lowval)

{

Lowval=*Test*[i];

}

}

System.***out***.println("Largest value in the Given Array is "+ Highval);

System.***out***.println("Largest value in the Given Array is "+ Lowval);

## How to display all the prime numbers between 1 and 100

**int** Num;

String PrimeNumber="";

**for**(**int** i=1;i<=100;i++)

{

**int** count=0;

**for**(Num=i;Num>=1;Num--)

{

**if**(i%Num==0)

{

count=count+1;

}

}

**if**(count==2)

{

PrimeNumber=PrimeNumber+i+" ";

}

}

System.***out***.println(PrimeNumber);

}

## How to Sort an Array in Ascending Order or Descending Order?

**public** **static** **int** []*as*= {21,2,5,55,2,47,8,9,52,-12};

**public** **static** **int** *a*,*b*,*c*;

**public** **static** **void** main(String[] args) {

**for**(**int** i=0;i<*as*.length;i++)

{

**for**(**int** j=0;j<*as*.length;j++)

{

**if**(*as*[i]<*as*[j])

{

*a*=*as*[i];

*as*[i]=*as*[j];

*as*[j]=*a*;

}

}

}

**for**(**int** i=0;i<*as*.length;i++)

{

System.***out***.println(*as*[i]);

}

}

## How to find out the length of the string without using length function?

System.***out***.println(*Str*.lastIndexOf(""));

System.***out***.println(*Str*.length());

## Write a program to get substring of string?

 public static void main(String[] args)

     {

         String blogName = "howtodoinjava.com";

    System.out.println(blogName.substring(3));  //todoinjava.com

            System.out.println("hello world".substring(6)); //world

     }

Write a script to print the multiples of 5?  
 **public** **static** **void** main(String[] args)

{

**int** a=5;

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

{

**int** b=a\*i;

System.***out***.print(b+" ");

}

}

## Which method is used to convert String to Char?

by using toCharArray();

## Which package is used for file reading and writing in java?

Java I/O package is used

## In java how to take input from user?

By using Scanner class

## How to split?

String expDate[] = expectedDate.split("/");

# Constructor

## What is Constructor in Java?

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

Constructor name should be same as class name. It looks like a method but it’s not a method. It won’t return any value. We have seen that methods may return a value. If there is no constructor in a class, then compiler automatically creates a default constructor.

## What is the different use of *“*this*”* statement in java?

“This” is a keyword in java used for referring same class instance variable, same class method, returning same class instance.

## What is the different type of constructor?

Java has default and parameterized constructor

# Collection

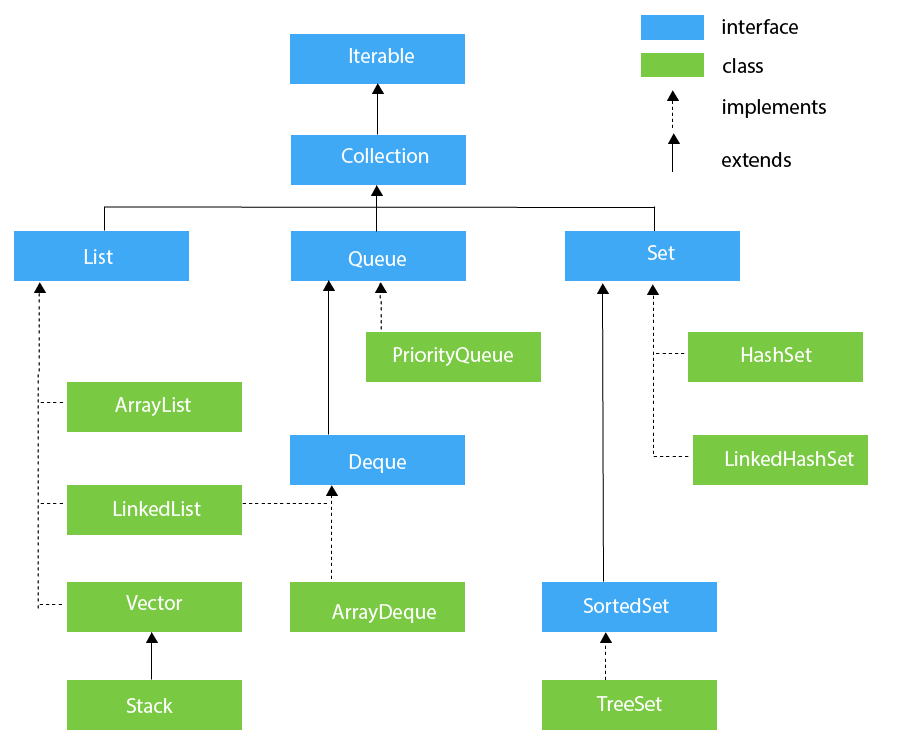
## What Is Collections in Java?

A Collection represents a single unit of objects. The Collection in Java is a framework that provides architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.javatpoint.com/java-arraylist), Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [PriorityQueue](https://www.javatpoint.com/java-priorityqueue), HashSet, LinkedHashSet, TreeSet).

**Flow Chart**



## Iterable Interface:

The Iterable interface is the root interface for all the collection classes. The Collection interface extends the Iterable interface and therefore all the subclasses of Collection interface also implement the Iterable interface.It contains only one abstract method. i.e.,

Iterator<T> iterator()

## List Interface:

List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

To instantiate the List interface, we must use:

List <data-type> list1= **new** ArrayList();

List <data-type> list2 = **new** LinkedList();

List <data-type> list3 = **new** Vector();

List <data-type> list4 = **new** Stack();

## ArrayList:

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed.

* It’s not in fixed Size.
* It’s automatically incise size.
* Two array list merging to one by using add all.
* Remove all if you want remove it.

Consider the following example.

ArrayList myData=**new** ArrayList();// array list can contains any value

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

{

myData.add(i);

System.***out***.println(myData.get(i));

}

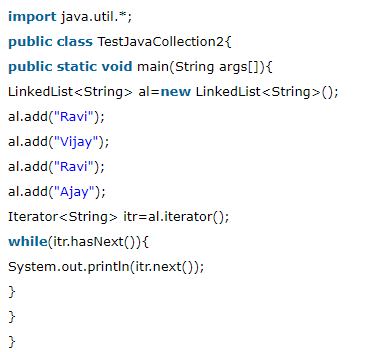
System.***out***.println(myData);

## LinkedList

LinkedList implements the Collection interface. It uses a doubly linked list internally to store the elements. It can store the duplicate elements. It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

LinkList Create two part of node first part store data and second part store next

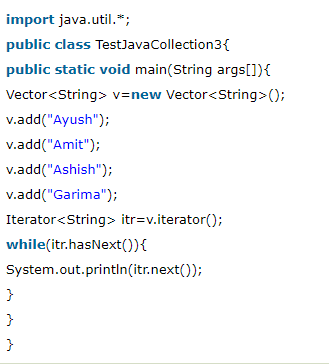
Consider the following example.



## Vector

Vector uses a dynamic array to store the data elements. It is similar to ArrayList. However, it is synchronized and contains many methods that are not the part of Collection framework.

Consider the following example.



## HashSet

HashSet class implements Set Interface. It represents the collection that uses a hash table for storage. Hashing is used to store the elements in the HashSet. It contains unique items.

Consider the following example.

HashSet<String> hs=**new** HashSet<String>();

hs.add("abc");

hs.add("usa");

hs.add("uas,ffhg,dfjd,shjdgfjs");

System.***out***.println(hs);

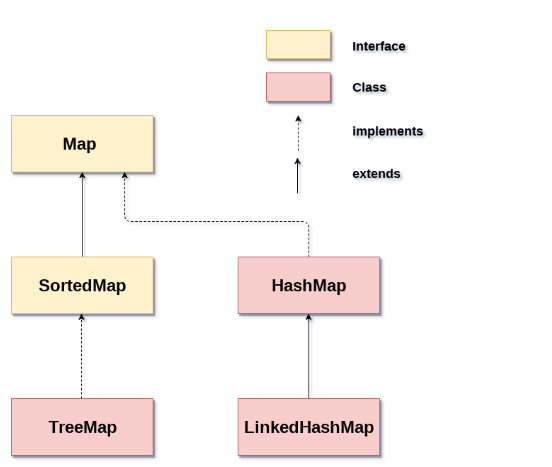
## What is Java Map Interface

A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.

A Map is useful if you have to search, update or delete elements on the basis of a key.

## Java Map Hierarchy

There are two interfaces for implementing Map in java: Map and SortedMap, and three classes: HashMap, LinkedHashMap, and TreeMap.



The hierarchy of Java Map is given below:

A Map doesn't allow duplicate keys, but you can have duplicate values. HashMap and LinkedHashMap allow null keys and values, but TreeMap doesn't allow any null key or value.

A Map can't be traversed, so you need to convert it into Set using keySet() or entrySet() method.

## Java HashMap class



* Java HashMap class implements the map interface by using a hash table. It inherits AbstractMap class and implements Map interface.

## Points to remember

* Java HashMap class contains values based on the key.
* Java HashMap class contains only unique keys.
* Java HashMap class may have one null key and multiple null values.
* Java HashMap class is non-synchronized.
* Java HashMap class maintains no order.
* The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.

Example Of Hashmap:

HashMap<Integer,String > hp= **new** HashMap<Integer,String>();

hp.put(1,"Test1");

hp.put(5,"Test5");

hp.put(4,"Test4");

hp.put(3,"Test3");

hp.put(2,"Test2");

System.***out***.println(hp);

//System.out.println(hp.get(4));

Set sn=hp.entrySet();

Iterator i=sn.iterator();

**while**(i.hasNext())

{

Map.Entry mp=(Map.Entry)i.next();

//System.out.println(mp.getKey());

System.***out***.println(mp.getValue());

}

## **Where all you have used ArrayList,Set and HashMap in your application.**

# Difference in Java

## Difference between Declaration and Definition in Java?

**Declaration:**

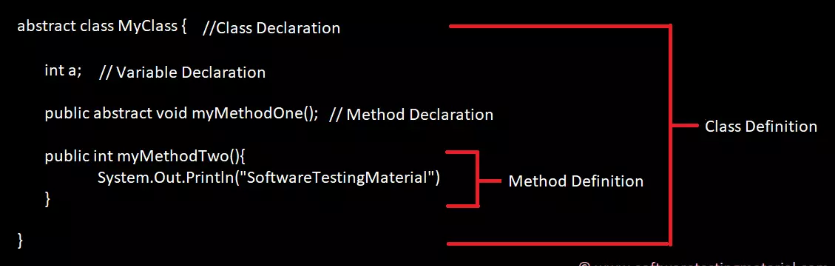
If you just declare a class or method/function or variable without mentioning anything about what that class or method/function or variable looks like is called as declaration in Java. Declaration there is no body of method

**Definition:**

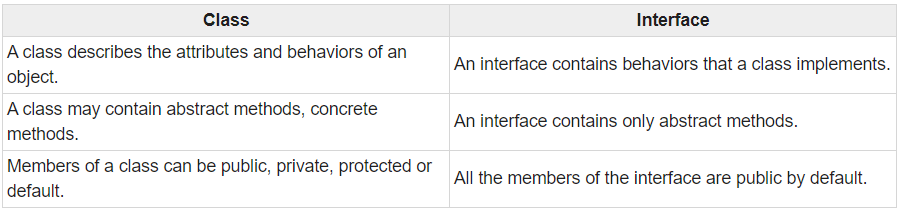
If you define how a class or method/function or variable is implemented then it is called definition in Java.

When we create an interface or abstract class, we simply declare a method/function but not define it.

For clear understanding, check the below image



## Difference between interface and a class?



## Difference between final,finally,finalize?

There are many differences between final, finally and finalize. A list of differences between final, finally and finalize are given below:

|  |  |  |
| --- | --- | --- |
| **Final** | **Finally** | **Finalize** |
| Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed. | Finally is used to place important code, it will be executed whether exception is handled or not. | Finalize is used to perform clean up processing just before object is garbage collected. |
| Final is a keyword. | Finally is a block. | Finalize is a method. |
| 1. **class** FinalExample{ 2. **public** **static** **void** main(String[] args){ 3. **final** **int** x=100; 4. x=200;//Compile Time Error 5. }} | 1. **class** FinallyExample{ 2. **public** **static** **void** main(String[] args){ 3. **try**{ 4. **int** x=300; 5. }**catch**(Exception e){System.out.println(e);} 6. **finally**{System.out.println("finally block is executed");} 7. }} | 1. **class** FinalizeExample{ 2. **public** **void** finalize(){System.out.println("finalize called");} 3. **public** **static** **void** main(String[] args){ 4. FinalizeExample f1=**new** FinalizeExample(); 5. FinalizeExample f2=**new** FinalizeExample(); 6. f1=**null**; 7. f2=**null**; 8. System.gc(); 9. }} |

## Difference between Call by value and call by reference?

Call by Value means calling a method with a parameter as value. Through this, the argument value is passed to the parameter.

While Call by Reference means calling a method with a parameter as a reference. Through this, the argument reference is passed to the parameter.

In call by value, the modification done to the parameter passed does not reflect in the caller's scope while in the call by reference, the modification done to the parameter passed are persistent and changes are reflected in the caller's scope.

## Difference between static binding and dynamic binding?

* Static binding is also known as early binding whereas dynamic binding is also known as late binding.
* Determining the type of an object at compile time is Static binding whereas determining the type of an object at run time is dynamic binding
* Java uses static binding for overloaded methods and dynamic binding for overridden methods.

## Difference between Abstract Class and Interface?

| **ABSTRACT CLASS** | **INTERFACE** |
| --- | --- |
| To declare Abstract class we have to use abstract keyword | To declare Interface we have to use interface keyword |
| In an Abstract class keyword abstract is mandatory to declare a method as an abstract | In an Interface keyword abstract is optional to declare a method as an abstract. Compiler treats all the methods as abstract by default |
| An abstract class contains both abstract methods and concrete methods(method with body) | An interface can have only abstract methods |
| An abstract class provides partial abstraction | An interface provides fully abstraction |
| An abstract class can have public and protected abstract methods | An interface can have only public abstract methods |
| An abstract class can have static, final or static final variables with any access modifiers | An interface can have only public static final variables |
| An abstract class can extend one class or one abstract class | An interface can extend any number of interfaces |
| Abstract class doesn't support multiple inheritance | Interface supports multiple inheritance |

## Difference between Array and ArrayList?

| **Array** | **ArrayList** |
| --- | --- |
| Array is static | ArrayList is dynamic |
| Size of the array should be given at the time of array declaration. We cannot change the size of array after creating it | Size of the array may not be required. It changest the size dynamically. Capacity of ArrayList increases automatically whenever we add elements to an ArrayList |
| Array can contain both primitive data types as well as objects | ArrayList cannot contain primitive data types. It contains only objects |
| Arrays are multidimensional | ArrayList is always single dimension |

## Difference between ArrayList and HashSet in Java?

| **ArrayList** | **HashSet** |
| --- | --- |
| ArrayList implements List interface | HashSet implements Set interface |
| ArrayList allows duplicates | HashSet doesn’t allow duplicates |
| ArrayList is an ordered collection and maintains insertion order of elements | HashSet is an unordered collection and doesn’t maintain insertion order |
| ArrayList is backed by an Array | HashSet is backed by an HashMap instance |
| ArrayList is an index based | HashSet is object based |
| In ArrayList, we can retrive object by calling get() method or remove object by calling remove() method | In HashSet, we can’t achieve get() method |

## Difference between array and string

|  |  |
| --- | --- |
| **Array** | **String** |
| Arrays are a sequential collection of element of similar data types | String refer to a sequence of characters represented as a single data type |
| Elements of arrays are stored contiguously in increasing memory location. | String can be stored in any manner in memory location |
| An array is a special variable that can hold more than one value at a time | String can only hold char data which are the most commonly used data types |
| Arrays are mutable meaning the fields can be modified | String are immutable meaning the value can’t be changed in memory once created |
| The length of an array is fixed | The size of string is not fixed |

## **Difference between throw and throws in java?**

**Throws clause** is used to declare an exception, which means it works similar to the try-catch block. On the other hand **throw** keyword is used to throw an exception explicitly.

If we see syntax wise than **throw** is followed by an instance of Exception class and **throws** is followed by exception class names.

## Difference between .equals (), (==) and compare-to ();

|  |  |  |
| --- | --- | --- |
| **.**equals () | **(==)** | **compare-**to (**)** |
|  |  |  |
| equals will take any Object as a parameter | equals will take any Object as a parameter | compareTo will only take Strings |
| equals() will only talk about the equalitys | It talks about logical equality | compareTo()**gives the ordering of objects** |
| **equals()** can be more efficient |  | compare-to () not efficient |
| Return value is boolean | Return True or False | Return value is 0 or 1 |

## [Difference between equals() and hashCode()](https://stackoverflow.com/questions/24446763/difference-between-equals-and-hashcode)

|  |  |
| --- | --- |
| **equals()** | **hashCode()** |
|  |  |
| Return value is Boolean | hashCode just returns int value for an Object |
| equals() checks if the two object references are same. |  |
|  |  |

## Difference between StringBuffer and StringBuilder?

|  |  |
| --- | --- |
| **String Buffer** | **String Builder** |
|  |  |
| StringBuffer is synchronized | StringBuilder is non-synchronized |
| StringBuffer is less efficient than StringBuilder. | StringBuilder is more efficient than StringBuffer. |
| thread safe | not thread safe |

## Difference between hashmap and hash table?

|  |  |
| --- | --- |
| **Hashmap** | **Hash table** |
|  |  |
| HashMap Does Not  is synchronized | HashTable  is synchronized |
| Its Thread Safe | Does Not Thread Safe |
| Its Allow Null Value | It’s does not allow Null value |

## Difference in List and Set in java?

List can have duplicate element but Set only have unique element.

## Difference in Collections and Collection in java.

Collection is an interface in java whereas Collections is a utility class has static java methods inside it.

# Data Type

## What are Local Variable and Instance Variable?

**Local Variable:**

Local variable is a variable which we declare inside a Method. A method will often store its temporary state in local variables.

**Instance Variable (Non-static):**

Instance variable is a variable which is declared inside a Class but outside a Method. We don’t declare this variable as Static because these variables are non-static variables.

**Class Variable (Static):**

Class variable is a variable which is declared as Static. Additionally, the keyword final could be added to include that the value will never change.

## What is variable scope in java?

Scope of a variable is the part of the program where the variable is accessible. Like C/C++, in Java, all identifiers are lexically (or statically) scoped, i.e. Scope of a variable can determined at compile time and independent of function call stack.  
Java programs are organized in the form of classes. Every class is part of some package. Java scope rules can be covered under following categories.

## In a class can a local variable be static?

No

## Why String is immutable?

As, once the String object is created cannot be modified

## Where the String objected created?

The string objects are created in String pool area.

## What is toString() in java?

toString() is used for String representation of objects.

# OOPs concepts

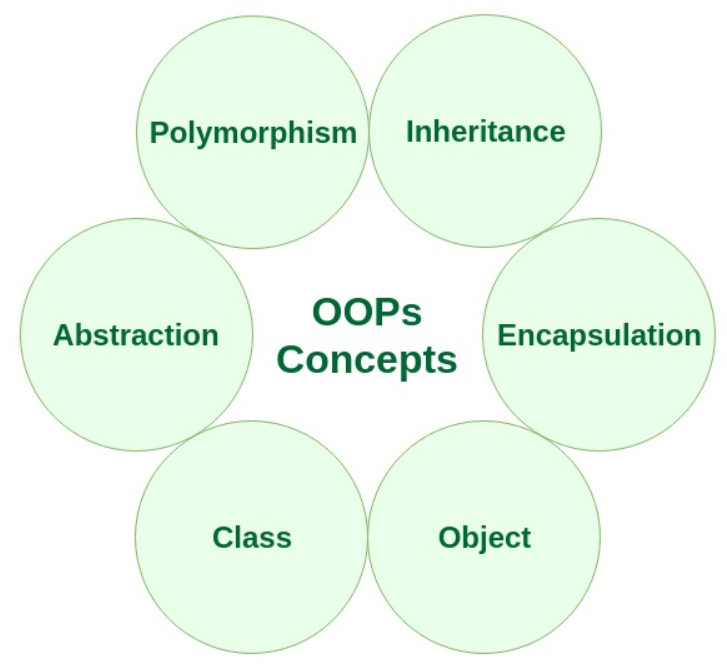
## Object Oriented Programming (OOPs) Concept in Java

**Object-oriented programming:**

As the name suggests, Object-Oriented Programming or OOPs refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

**OOPs Concepts:**

* [Polymorphism](https://www.geeksforgeeks.org/polymorphism-in-java/)
* [Inheritance](https://www.geeksforgeeks.org/inheritance-in-java/)
* [Encapsulation](https://www.geeksforgeeks.org/encapsulation-in-java/)
* [Abstraction](https://www.geeksforgeeks.org/abstraction-in-java-2/)
* [Class](https://www.geeksforgeeks.org/classes-objects-java/)
* [Object](https://www.geeksforgeeks.org/classes-objects-java/)
* [Method](https://www.geeksforgeeks.org/methods-in-java/)
* [Message Passing](https://www.geeksforgeeks.org/message-passing-in-java/)



Let us learn about the different characteristics of an Object-Oriented Programming language:

## [Polymorphism](https://www.geeksforgeeks.org/polymorphism-in-java/):

It’s Performing Same method or task in different ways.

Polymorphism refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities.

**For example:**

|  |
| --- |
| public class Sum {        // Overloaded sum().      // This sum takes two int parameters      public int sum(int x, int y)      {          return (x + y);      }        // Overloaded sum().      // This sum takes three int parameters      public int sum(int x, int y, int z)      {          return (x + y + z);      }       // Driver code      public static void main(String args[])      {          Sum s = new Sum();          System.out.println(s.sum(10, 20));          System.out.println(s.sum(10, 20, 30));      }  } |

**Output:**

30

60

Polymorphism in Java is mainly of 2 types:

* + [Overloading in Java](https://www.geeksforgeeks.org/overloading-in-java/)
  + [Overriding in Java](https://www.geeksforgeeks.org/overriding-in-java/)

## [Inheritance](https://www.geeksforgeeks.org/inheritance-in-java/):

Acquiring the properties one class to another class.

Inheritance is an important pillar of OOP (Object Oriented Programming). It is the mechanism in java by which one class is allows inheriting the features (fields and methods) of another class.  
**Important terminology:  
Super Class:**The class whose features are inherited is known as superclass

(or a base class or a parent class).

**Sub Class:**

The class that inherits the other class is known as subclass(or a derived

class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

**Reusability:**

Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

The keyword used for inheritance is **extends**.  
**Syntax:**

class derived-class **extends** base-class

{

//methods and fields

}

**There** are three type of Inheritance available

* + Single Inheritance.
  + Multilevel Inheritance.
  + Hierarchal Inheritance.

## Can an Interface have complete method?

Yes. After java ver 8, Interface can have complete methods but the method should be static and default.

## [Encapsulation](https://www.geeksforgeeks.org/encapsulation-in-java/):

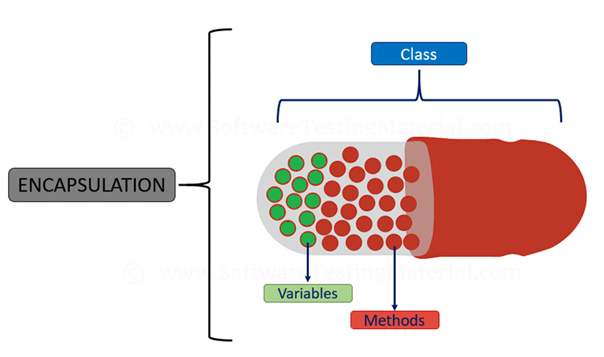
## How to Achieve Encapsulation?

Encapsulation is binding variable and method under single unity by declaring Variable as private and method as public.

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.

* + As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.
  + Encapsulation can be achieved by declaring all the variables in the class as
  + Private and writing public methods in the class to set and get the values of Variables.



## [Abstraction](https://www.geeksforgeeks.org/abstraction-in-java-2/):

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user.

Ex:

A car is viewed as a car rather than its individual components.

Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

In java, abstraction is achieved by [interfaces](https://www.geeksforgeeks.org/interfaces-in-java/) and [abstract classes](https://www.geeksforgeeks.org/abstract-classes-in-java/). We can achieve 100% abstraction using interfaces.

## [Method](https://www.geeksforgeeks.org/methods-in-java/):

A method is a collection of statements that perform some specific task and return result to the caller. A method can perform some specific task without returning anything. Methods allow us to **reuse** the code without retyping the code. In Java, every method must be part of some class which is different from languages like C, C++ and Python.  
Methods are **time savers**and help us to **reuse** the code without retyping the code.

**Method Declaration**

In general, method declarations have six components:

## [Access Modifier](https://www.geeksforgeeks.org/access-modifiers-java/):

Defines **access type** of the method i.e. from where it can be accessed in your application. In Java, there 4 type of the access specifies.

**Public:**

Accessible in all class in your application.

**Protected:**

Accessible within the package in which it is defined and in its **subclass (es)(including subclasses declared outside the package)**

**Private:**

accessible only within the class in which it is defined.

**Default**

**declared/defined without using any modifier:** accessible within

Same class and package within which its class is defined.

The return type:

The data type of the value returned by the method or void if does not return a value.

Method Name:

The rules for field names apply to method names as well, but the convention is a little different.

**Parameter list**:

Comma separated list of the input parameters are defined, preceded with their data type, within the enclosed parenthesis. If there are no parameters, you must use empty parentheses ().

**Exception list:**

The exceptions you expect by the method can throw, you can specify these exception(s).

**Method body:**

it is enclosed between braces. The code you need to be executed to perform your intended operations.

[](https://media.geeksforgeeks.org/wp-content/uploads/methods-in-java.png)

[**Message Passing**](https://www.geeksforgeeks.org/message-passing-in-java/)**:**

Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

## What are the types of Polymorphism?

There are two types of Polymorphism in Java

* Compile time polymorphism (Static binding) – Method overloading.
* Runtime polymorphism (Dynamic binding) – Method overriding

We can perform polymorphism by ‘Method Overloading’ and ‘Method Overriding’

## What is Method Overloading?

A class having multiple methods with same name but different parameters is called Method Overloading

There are three ways to overload a method.

* Parameters with different data types.
* Parameters with different sequence of a data types.
* Different number of parameters.

## What is Method Overriding?

Declaring a method in child class which is already present in the parent class is called Method Overriding.

In simple words, overriding means to override the functionality of an existing method.

In this case, if we call the method with child class object, then the child class method is called. To call the parent class method we have to use **super**keyword.

## Can we override static methods in java?

No we cannot override static methods in java.

## Is it possible to have static method in method overloading?

Yes it’s possible as method overloading depends only on the number of the argument and data type.

## In a class can a local variable be static?

No

## What is Abstract Class in Java?

We can easily identify whether a class is an abstract class or not. A class

which contains abstract keyword in its declaration then it is an Abstract Class.

Syntax:

|  |  |
| --- | --- |
|  | abstract class <class-name>{} |
|  |  |

Points to remember:

* Abstract classes may or may not include abstract methods.
* If a class is declared abstract then it cannot be instantiated.
* If a class has abstract method then we have to declare the class as abstract class.
* When an abstract class is subclasses, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

## What is Abstract Method?

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

|  |  |
| --- | --- |
|  | abstract void myMethod(); |
|  |  |

In order to use an abstract method, you need to override that method in sub class.

## Why we use interface why not abstract class?

Interface is acquiring one class properties to another class so we can use any method but abstraction we can’t .

## What is run time polymorphism in java?

In java, calling to a method is defined based on the type of object created in runtime. That is run time polymorphism.