HashMap, HashSet- 2

Strings-3

Interfaces-4

Comparator-5

Exception-6

Arraylist, set, List-7

Collections framework-8

File Handling-9

Oops-10,11

Miscellaneous-At last

**HashMap and HashSet**

HashMap<I,I>- contiansKey, contiansValue, isEmpty, keySet- gives Set<>, size, put, get

Split-method-String Method

HashSet-add, remove,contains

Iterator<String> i = h.iterator();

Where **h** is the name of HashSet or can be set.

And i.hasNext() and i.next

Map.entrySet=return hashmap in list form

Map.Entry-interface for hashmap provides certain function equals <https://www.geeksforgeeks.org/map-entry-interface-java-example/>

Sorting in hashmap using above tech and comparator

Load Factor- filled/total. Rehashing inc array size(1.5 or 2) and again hashing all values

HashSet uses **HashMap** for storing its object internally. Value is constant variable.

Strings

String(char array) return string

**Integer.toString**- to concat int with String.

Interfaces

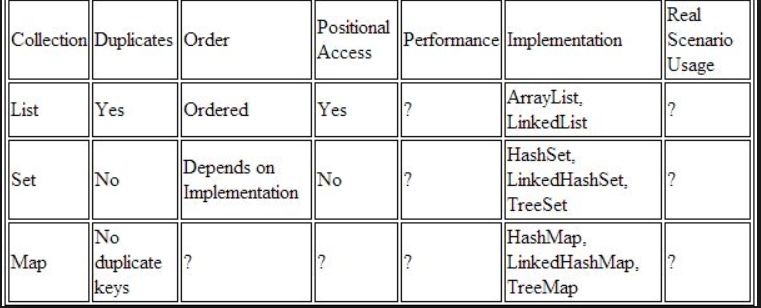
**Set** is implemented by HashSet, LinkedHashSet or TreeSet and extends collection

List- also by vector and stack- <https://www.geeksforgeeks.org/list-interface-java-examples/>

Set-<https://www.geeksforgeeks.org/set-in-java/> .All set operation like union, sub etc. **not sorted**.

SortedSet-Interface, extends Set <https://www.geeksforgeeks.org/sortedset-java-examples/>

List vs Set both extends collection interface



Comparator

Use to sort objects

Implements comparator generic interface.

compare function and equals(one argument) function, public methods

-1 when a<b

1 when a>b sort happens

0 a==b

Then Collections.sort(collection, comparator object)/Arrays.sort()

**Comparable**-compareTo()-we can use only once, so we can only do for a particular attribute/members of class. Overriding interface function. Implemented by item class.

[comparable vs comparator](https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/)

# Exception

Util package

And for creation an expection one must extend **Exception** class

ArrayList

Subtype of list

<https://www.geeksforgeeks.org/arraylist-in-java/>

add(it is used to add to specific location), by shifting all by one location

List vs ArrayList-interface and class

Interface implementation support change

**Collections Framework**

Collections of classes or data structures.

Collection-interface, A Collection represents a single unit of objects, i.e., a group

Collections-utility class.<https://docs.oracle.com/javase/6/docs/api/java/util/Collections.html>

File Handling

6 ways using

BufferReader, Scanner, fileReader

And for delete- delete()- return true or false.

File obj with path as parameter.

Oops

A **final class** is simply a **class** that can't be extended.

Wrapper class Integer is a final class. If that class is not final, then any one can extend Integer into his own class and change the basic behaviour of integer class. To avoid this, java made all wrapper classes as final classes. Derived final class (can extend non final) or function can be final.

**Method final-** nooverriding by derived(error), fixed functionality, Final methods are faster than instance methods, as there is no use of virtual table concept for final and private methods. Derived function can be final

**constructor** can't be final. They can be protected but **class** can't

The **protected** specifier allows access by all subclasses of the class in a program, whatever package they reside in, as well as to other code in the same package. The **default** specifier allows access by other code in the same package, but not by code that is in subclasses residing in different packages.

**Static vs non static(instance)**

Instance method are methods which require an object of its class to be created before it can be called. Static methods are the methods in Java that can be called without creating an object of class. Static method is declared with static keyword. Instance method is not with static keyword. Static only one copy for class. There is no cross usage ie instance can’t use static or vice versa. No opposite overriding.

Instance-dynamic binding-**overriding**, overloading

Static- static binding-**overloading**, no overriding

**Polymorphism**

It describes the concept that different classes can be **used** with the same interface

Virtual-overridden

This is known as [Polymorphism](http://java67.blogspot.com/2012/10/difference-between-polymorphism-overloading-overriding-java.html) because any virtual method will be executed from subclass only, even though they were called from super type.

All non-static methods are virtual except final ones. Overridden methods are virtual methods.

Base wala override ho jata hai. (runtime polymorphism)

**Dynamic Method Dispatch**-Polymorphism overriding ie at runtime.

**Ad-hoc** polymorphism is also known as overloading.

If virtual method in derived class(subclass)is private, it will raise an error.

Or assess modifier of derived is equal or more than of base.

**Inheritance in Java**

is a mechanism in which one object acquires all the properties and behaviours of a parent object, hierarchical- one base many derived. No multiple inheritance

**Composition-** They are absolutely different. Inheritance is an ***"is-a"*** relationship. Composition is a ***"has-a"***. Package or by creating obj instead of extend. class object to be used inside another class. Car has wheel(composition) and student is human(inheritance).

**Abstraction**: Abstraction is the act of representing essential information without including background details and explanations. **Data Hiding-** hiding internal information.

**Encapsulation**: Encapsulation is the act of **wrapping** up of attributes (represented by data members) and operations (represented by functions) under one single unit (represented by class).

Ex-Vending machine or tic tac toe-

Abstraction-we are only concerned with i/o not internal processing

Encapsulation-Vending machine is a class- data- items and functions take input/ deliver/ make payment etc They are wrapped.

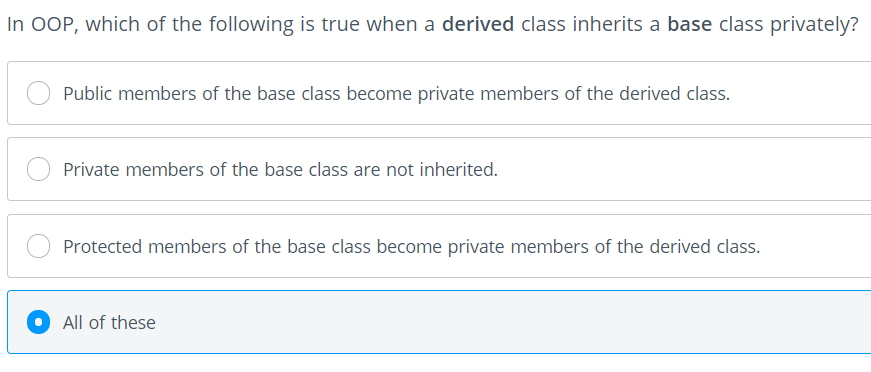
Data Hiding- how many cans we have/ or how many variables

**Encapsulation is a way to implement Abstraction**

**C++**

Constructor- can be private, protected, copy constructor but not friend

**Friend Function-** not in java as it will break the design. It is not the member function but can access private variable. Can be called using objs.



**Miscellaneous**

Java heap size-<http://net-informations.com/java/cjava/limit.htm>