**Is Java platform independent?**

* Yes. Java is a platform independent language.
* We can write java code on one platform and run it on another platform.
* e.g. we can write and compile the code on windows and can run it on Linux or any other supported platform. This is one of the main features of java.

**What is javac ?**

* It produces the java byte code from \*.java file.
* It is the intermediate representation of your source code that contains instructions.

**What is class?**

* Class is nothing but a template that describes the data and behavior associated with instances of that class

**What is the base class of all classes?**

java.lang.Object

**What is Unicode?**

* Java uses Unicode to represent the characters.
* Unicode defines a fully international character set that can represent all of the characters found in human languages.

**What is Type casting in Java?**

* To create a conversion between two incompatible types,
* we must use a cast.
* There are two types of casting in java: automatic casting (done automatically) and explicit casting (done by programmer).

**Abstract class?**

An abstract class is a class which can’t be instantiated (we cannot create the object of abstract class), we can only extend such classes. It provides the generalized form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details. We can achieve partial abstraction using abstract classes, to achieve full abstraction we use interfaces.

**Q) What is Interface in java?**

An interface is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. [Read more about interface here](https://beginnersbook.com/2013/05/java-interface/).

**Q) What is the difference between abstract class and interface?**

1) abstract class can have abstract and non-abstract methods. An interface can only have abstract methods.

2) An abstract class can have static methods but an interface cannot have static methods.

3) abstract class can have constructors but an interface cannot have constructors.

What is Collection ? What is a Collections Framework ? What are the benefits of Java Collections Framework ?

Collection : A collection (also called as container) is an object that groups multiple elements into a single unit.

Collections Framework : Collections framework provides unified architecture for manipulating and representing collections.

Benefits of Collections Framework :

1. Improves program quality and speed

2. Increases the chances of reusability of software

3. Decreases programming effort.

Root Element in collection Hierarchy

Collection interface

What is the difference between Collection and Collections ?

Collection is an interface while Collections is a java class

Which collection classes are synchronized or thread-safe ?

Stack, Properties , Vector and Hashtable

The list of core collection interfaces are : just mention the important ones

Important : Collection , Set , Queue , List , Map

Other interface also in the list : SortedSet, SortedMap , Deque, ListIterator etc

What is the difference between List and Set ?

Set contain only unique elements while List can contain duplicate elements.

Set is unordered while List is ordered . List maintains the order in which the objects are added .

What is the difference between Map and Set ?

Map object has unique keys each containing some value, while Set contain only unique values.

Class implementing List interface : ArrayList , Vector , LinkedList ,

Class implementing Set interface : HashSet , TreeSet

Iterator is an interface . It is found in java.util package. It provides methods to iterate over any Collection.

Q10 What is the difference between Iterator and Enumeration ?

The main difference between Iterator and Enumeration is that Iterator has remove() method while Enumeration doesn't.

Hence , using Iterator we can manipulate objects by adding and removing the objects from the collections.

Enumeration behaves like a read only interface as it can only traverse the objects and fetch it

Which methods you need to override to use any object as key in HashMap ?

To use any object as key in HashMap , it needs to implement equals() and hashCode() method .

How to reverse the List in Collections ?

There is a built in reverse method in Collections class . reverse(List list) accepts list as parameter.

Convert array of string String[] wordArray = {"Love Yourself" , "Alive is Awesome" , "Be in present"};

List wordList = Arrays.asList(wordArray);

Difference between ArrayList and Vector

Synchronized :- Vector synchronized

ArrayList NotSync

Speed :- Vector slow

Arraylist Fast

Difference HashMap & HashSet

Null :- HashMap allows one null key and any number of null values while Hashtable does not allow null keys and null values.

Synchronized :- HashMap synchronized and Thread Safe

HashSet NotSync and Not Thread safe

Iterating the values: Hashmap object values are iterated by using iterator .

HashTable is the only class other than vector which uses enumerator to iterate the values of HashTable object.

Performance : Hashmap is much faster and uses less memory than Hashtable as former is unsynchronized

Queue :- Peek() , Poll() and remove()

Both poll() and remove() method is used to remove head object of the Queue.

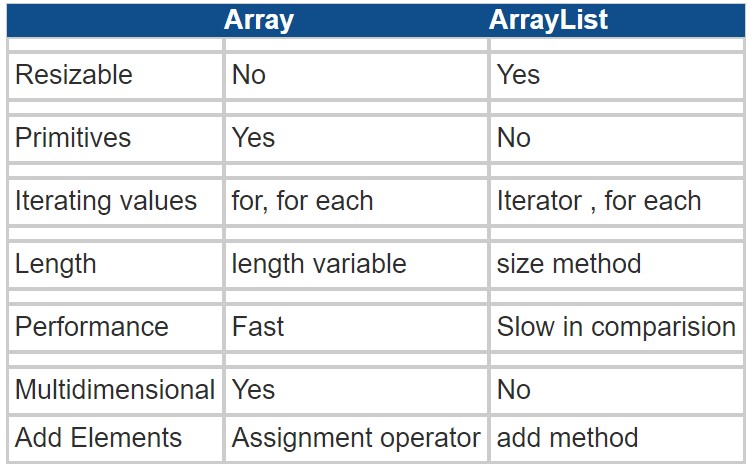
Return Type

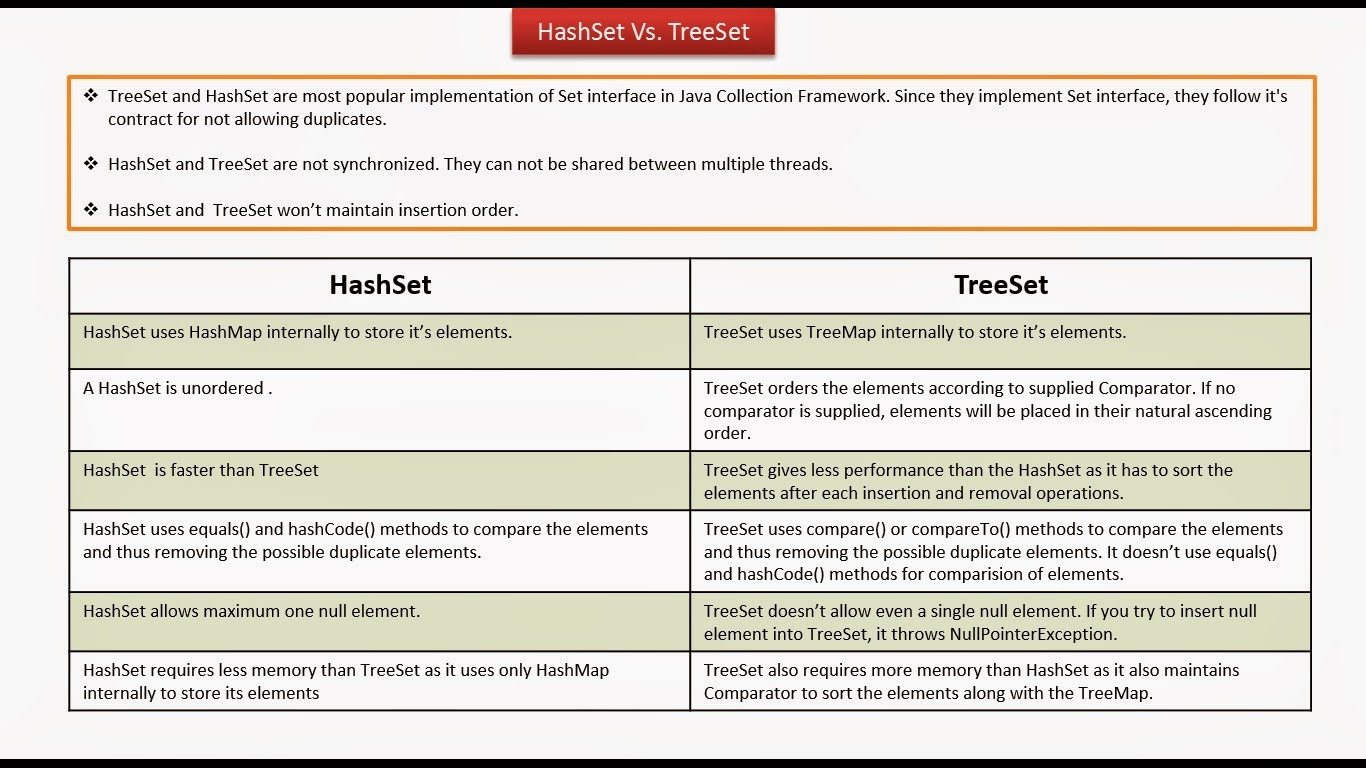
queque empty :-

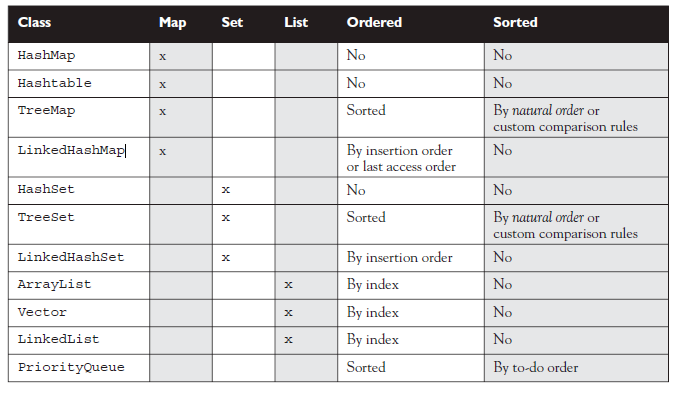
Poll() ==> Null

remove() ==> remove() method will throw NoSuchElementException .

peek() method==> retrieves but does not remove the head of the Queue. If queue is empty then peek() method also returns null.







How to make Map Ordered

* LinkedHashMap

Iteration order for above implementations:

* HashSet - undefined
* HashMap - undefined
* LinkedHashSet - insertion order
* LinkedHashMap - insertion order of keys (by default), or 'access order'
* ArrayList - insertion order
* LinkedList - insertion order
* TreeSet - ascending order, according to Comparable / Comparator
* TreeMap - ascending order of keys, according to Comparable / Comparator
* Collections.synchronizedList(new ArrayList<YourClassNameHere>())

**Principal features of non-primary implementations:**

* HashMap has slightly better performance than LinkedHashMap, but its iteration order is undefined
* HashSet has slightly better performance than LinkedHashSet, but its iteration order is undefined
* TreeSet is ordered and sorted, but slower
* TreeMap is ordered and sorted, but slower
* LinkedList has fast adding to the start of the list, and fast deletion from the interior via iteration

### **How HashMap works in Java?**

* HashMap stores key-value pair in Map.Entry static nested class implementation.
* HashMap works on hashing algorithm and uses hashCode() and equals() method in put and get methods.
* When we call put method by passing key-value pair, HashMap uses Key hashCode() with hashing to find out the index to store the key-value pair
* The Entry is stored in the LinkedList, so if there are already existing entry, it uses equals() method to check if the passed key already exists, if yes it overwrites the value else it creates a new entry and store this key-value Entry.
* When we call get method by passing Key, again it uses the hashCode() to find the index in the array and then use equals() method to find the correct Entry and return it’s value. Below image will explain these detail clearly.
* The other important things to know about HashMap are capacity, load factor, threshold resizing. HashMap initial default capacity is **16** and load factor is 0.75. Threshold is capacity multiplied by load factor and whenever we try to add an entry,
* if map size is greater than threshold, HashMap rehashes the contents of map into a new array with a larger capacity. The capacity is always power of 2,
* so if you know that you need to store a large number of key-value pairs, for example in caching data from database, it’s good idea to initialize the HashMap with correct capacity and load factor.

### **What is the importance of hashCode() and equals() methods?**

HashMap uses Key object hashCode() and equals() method to determine the index to put the key-value pair. These methods are also used when we try to get value from HashMap. If these methods are not implemented correctly, two different Key’s might produce same hashCode() and equals() output and in that case rather than storing it at different location, HashMap will consider them same and overwrite them.

Similarly all the collection classes that doesn’t store duplicate data use hashCode() and equals() to find duplicates, so it’s very important to implement them correctly. The implementation of equals() and hashCode() should follow these rules.

* If o1.equals(o2), then o1.hashCode() == o2.hashCode()should always be true.
* If o1.hashCode() == o2.hashCode is true, it doesn’t mean that o1.equals(o2) will be true.

https://www.journaldev.com/1330/java-collections-interview-questions-and-answers