**Topic 1:**

Collection

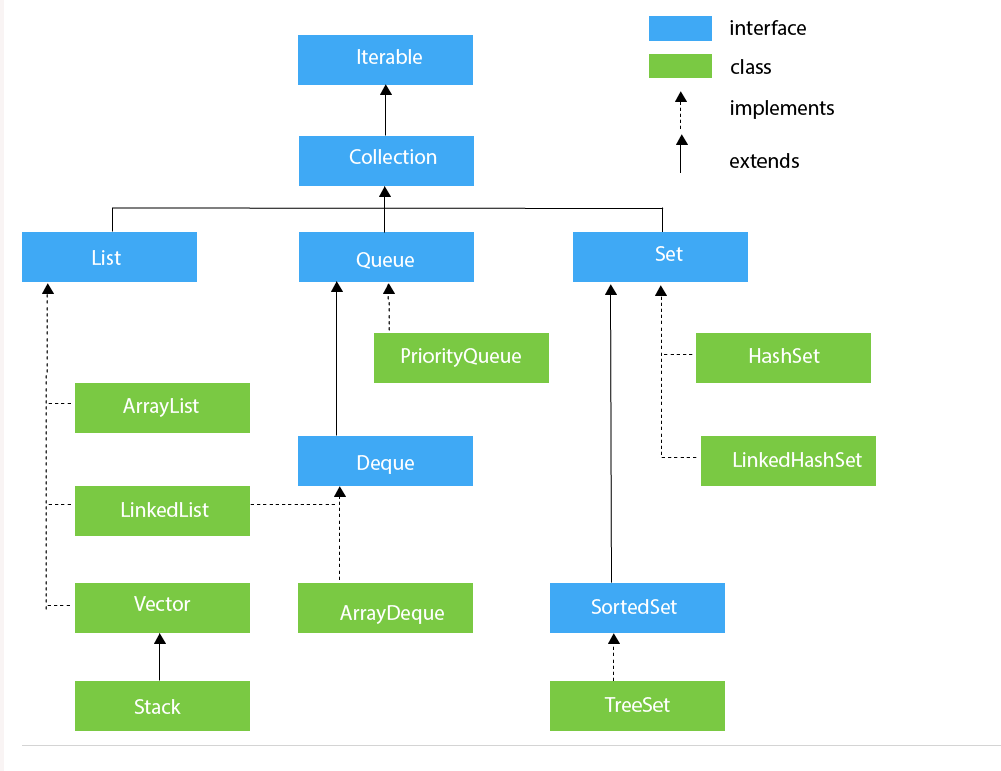
Sort element in collection

Comparable

Comparator

**Answer**

Hierarchy of Collection framework:

****

**The difference between “Collection” and “Collections”:**

+ Collection is a interface present in java.util package. It is used to represent a group of objects as a single unit. It is similar to the container in C++. It provides several classes and interfaces to represent a group of individual objects.

The List, Set, Queue are the main sub-interfaces of the collection interface. The add(), remove(), clear(), size(), and contains () are the important methods of the Collection interface.

+ Collections is a utility class present in java.util.package. It provides several utility methods like sorting and searching which is used to operate on collection.

+ It has a method sort() to sort the collection elements according to default sorting order. And it has a method min(), and max() to find the minimum and maximum value respectively in the collection elements.

+ Collections.sort( List , Comparator)

* Note that sort() can’t have Set, Map, Queue parameter.

+ method Min() or Max() are one of Collections’ methods.

Min( Collection, Comparator).

Max( Collection, Comparator).

* Comparable and Comparator

All of them are two interfaces.

- **Difference:**

+ **Comparable**: Interface compares “this” reference with the object specified

+ **Comparator**: compares two different class objects provided.

+ **Comparable :**  If any classes implement Comparable interface in Java, then collection of that object either List or Array can be sorted automatically by using Coleection.sort() or Array.sort() and objects will be sorted based on there natural order defined by CompareTo method.

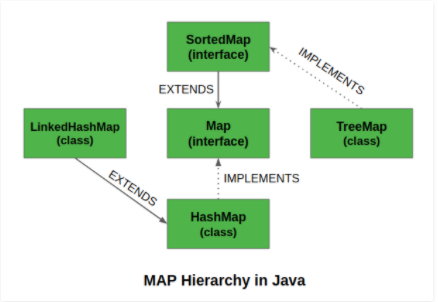
+**Comparator:** By using comparator, we are able to use different attributes by writing more than one custom comparators as we want. Whereas, we only can use one comparison by using comparable.

**Topic 2:**

Collection categories: sets, lists …

Interface Map its implementations.

Compare set and others set

**Answer: **

**Map interface belongs to Java Collection Framework but It doesn’t extend Collection interface.**

**Map :** A Map cannot contains duplicate keys and each key can map to at most one value.

+ Some implementations allow null key and null value like the HashMap and LinkedHashMap, but some do not like the TreeMap.

+ The order of a map depends on the specific implementations. For example, TreeMap and LinkedHashMap have predictable order, while HashMap does not.

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

**- HashMap** : Store (key, value). This class uses a technique called Hashing. Hashing is a technique of converting a large String to small String that represents the same String. A shorter value helps in indexing and faster searches.

**+ Some methods:**

Put(key, value)

Remove(key)

Get(object key)

containsKey(object Key)

containsValue(object value)

clear()

size()

values() : return a Collection

keyset(): return a Set

-**LinkedHashMap:** is just like HashMap with an additional feature of maintaining an order of elements inserted into it. HashMap provided the advantage of quick insertion, search and deletion but it never maintained the track and order of insertion which the LinkedHashMap provides where the elements can be accessed in their insertion order.

**TreeMap:** The TreeMap in Java is used to implement Map interface and NavigableMap along with the abstract class. Treemap is a **Red-Black tree** (same as C++) based NavigableMap implementation . The map is sorted according to the natural ordering of its key, or by a Comparator provided at map creation time.

**Some special method:**

CeilingEntry(K key) : return a key-value with the least key that greater than or equal to given key

CeilingKey( K key) : return a least key greater or equal to given key, or null if so such key

HigherEntry( K key): return a key-value with the least key that greater than the given key

HigherKey ( K key) : return a least key that greater than the given key

FloorEntry(K key) : return a key-value with the least key less than or equal to the given key.

FloorKey(K key): return a least key that less than or equal to the given keey

HeadEntry(K key): return a key-value with the least key strictly less than the given key

HeadKey(K key): return a least key that strictly less than the given key.

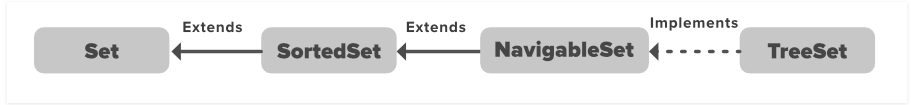
FirstEntry(K key): return a key-value that is the least key in this map

LastEntry(K key): return a key-value that is the greatest key in this map

**Set interface :**

+ Set interface present in Java.util package and extends the Collection interface is an unordered collection of objects in which duplicate values cannot be stored. This interface contains the methods inherited from the Collection interface and adds a feature which restricts the insertion of the duplicate elements.

+ There are two interfaces which extend the set implementation namely SortedSet and NavigableSet.



+ The set interface allows the users to perform the basic mathematical operation on the set.

A Union B ~ A.addAll(B)

A Intersection B ~ A.retainAll(B)

A Difference B ~ A.removeAll(B)

- **HashSet:** HashSet is a class that implements **Set interface**, is an inherent implementation of the **hash table data structure**. The object that we insert into the HashSet **does not guarantee** to be inserted in the same order. The objects are inserted based on their **hashCode**. NULL elements are **allowed**.

- Some methods:

+ add()

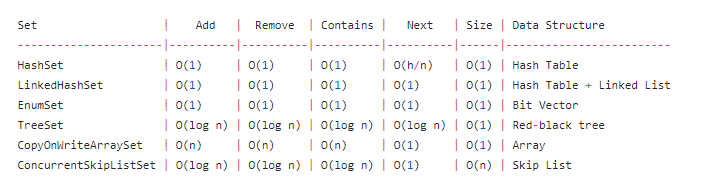
+ contains()

+ remove()

-**EnumSet:** Enumset class is implemented in the collections framework is one of the specialized implementation of Set interface for use with the **enumeration type.** It is much faster than HashSet.

-**LinkedHashSet:** LinkedHashSet is upgraded version of HashSet. It’s is an ordered version of HashSet that maintains a doubly-linked List across all elements. ( Nó lưu 2 con trỏ về trước và về sau, vì thế nên là sẽ lưu lại được thứ tự insert vào set).

- **TreeSet**: TreeSet class is implemented in the collections framework, and implementation of the SortedSet Interface and SortedSet extends Set interface. It behaves like a simple set but it stores elements in a sorted format. TreeSet uses a tree data structure for storage. Objects are stored in sorted, ascending order (by default).



https://gist.github.com/psayre23/c30a821239f4818b0709

**Topic 3:**

Parameterized types. Creation of parameterized classes. Wildcard parameters.

https://gpcoder.com/2868-huong-dan-su-dung-java-generics/

**Topic 4:**

Wrapper classes. Purpose, scope, advantages and disadvantages. Autoboxing and Unboxing.

**Answer**

* **Wrapper Classes:**

+ In Java, There is a Wrapper Class for every Primitive data type.

For example : Interger ~ int Long ~ long Boolean ~ boolean …

+ These classes are known as Wrapper Classes, because they “wrap” the primitive data type into an object of that class. The Wrapper Classes are part of the java.lang package

* **Purpose:**

+ To provide mechanism to “wrap” primitive type values in an object so that primitives can do activities reserved for the objects like, being added to ArrayList, Hashset, HashMap , …

+ Wrapper Classes provide many utility functions for primitives, Most of these functions are related to various conversions. For example: Convert an integer number to string …

* **Advantages**:

+ Can be used with generics

+ supports many additional methods.

* **Disadvantages**:

+ Use more cpu

+ Use more memory

+ Implies the value can be null

+ is longer to type in some cases

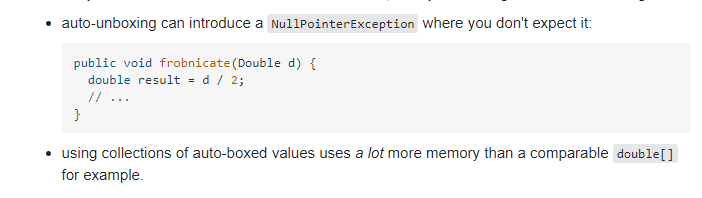
* Autoboxing : convert primitives type to Wrapper classes
* Unboxing : convert Wrapper Classes to primitives type.

**Advantages of Autoboxing / Unboxing:**

* Autoboxing and unboxing lets developers write cleaner code, making it easier to read.
* The technique let us use primitive types and Wrapper class objects interchangeably and we do not need to perform any typecasting explicitly.
* Ví dụ là phép += và phép %, compiler tự ngầm wrap từ kiểu primitives sang wrap để thực hiện

Disadvantage:

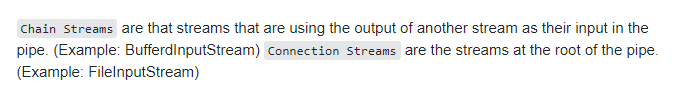
* Primitive types always are faster than wrapper tpes



**Topic 5:**

I / O streams in Java. Byte and character streams. Stream Chains.

<https://www.geeksforgeeks.org/java-io-input-output-in-java-with-examples/>



**Topic 6:**

Working with files in Java. The java.io.File class.

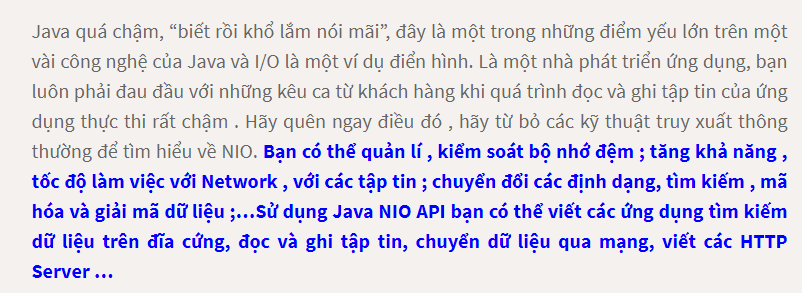
https://www.geeksforgeeks.org/file-class-in-java/

**Topic 7:**

Java.nio package - purpose, main classes and interfaces.

<https://edwardthienhoang.wordpress.com/2014/06/04/java-nio-qua-mot-vai-vi-du/>

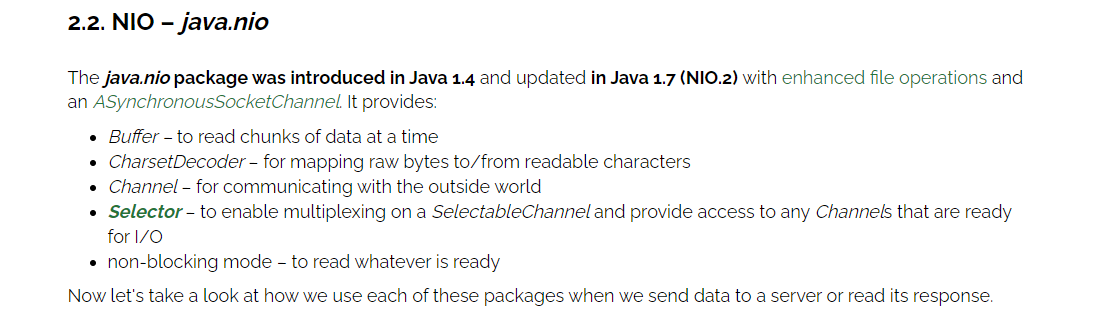
http://tutorials.jenkov.com/java-nio/nio-vs-io.html



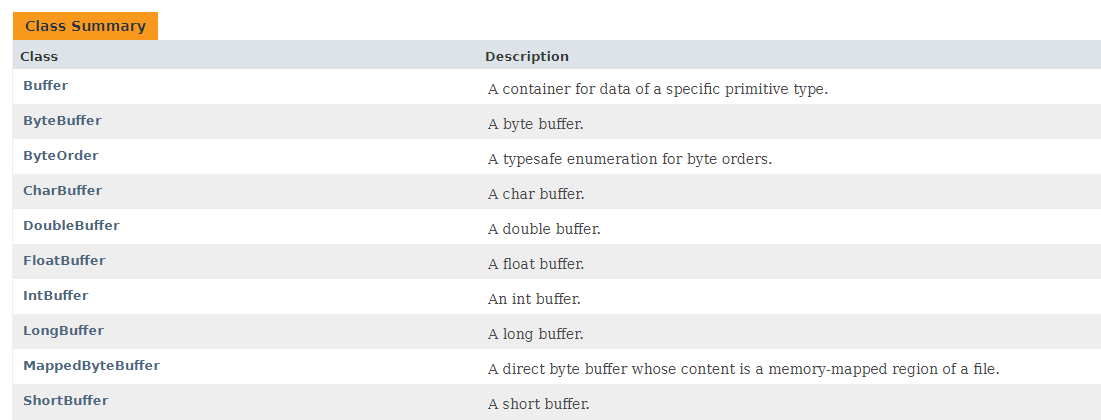
gói java.io cùng 4 gói nhỏ trong nó là java.nio.channels, java.nio.channels.spi, java.nio.charset và java.nio.charset.

Gói Java.nio : Gói này chứa các lớp buffer cung cấp các kỹ thuật để lưu trữ dữ liệu ở dạng nguyên thủy trong bộ nhớ trong.

Gói java.nio.charset : Chứa các charset, mã hóa, giải mã dữ liệu, các lớp chuyển đối dữ liệu.



**Important classes:**



**Important interface  
Java NIO path interface ( the most popular)**

****

**http://tutorials.jenkov.com/java-nio/path.html**

**Topic 8:**

The javadoc utility. Features of automatic code documentation in Java.

JavaDoc tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations ad documentation in a set of source file describing classes, methods, constructors, and fields.

Javadoc features include the following: Javadoc search

is producing the HTML pages. It's a fairly fast process, but slows down a little as the number of classes in a project increases. You can do documentation builds as frequently as you require them. Since it is an automated process, there's no need to rewrite documentation as changes to a class are made. Simply generate the documentation again, and the changes will be incorporated.

Well asked what are byte / character streams, what are the main classes there, asked what is a buffer reader

And asked about my collection

And, about generics and parameterized types, he also asked, for sure