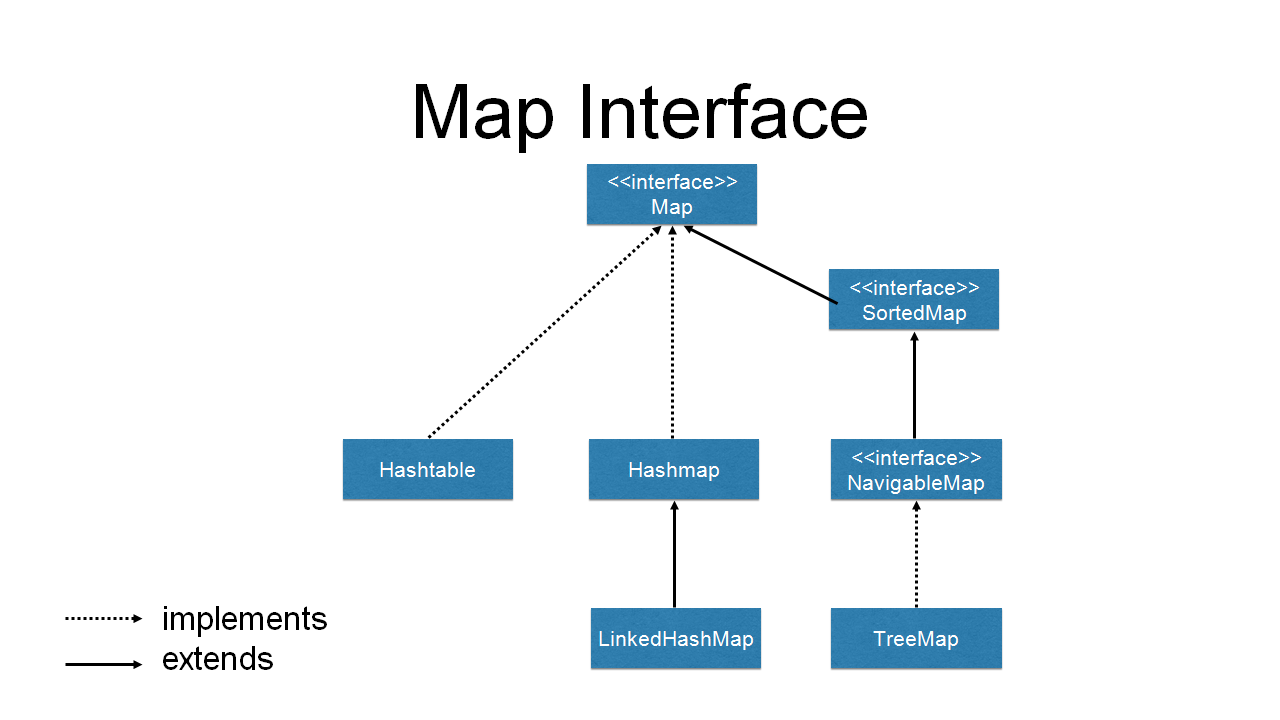
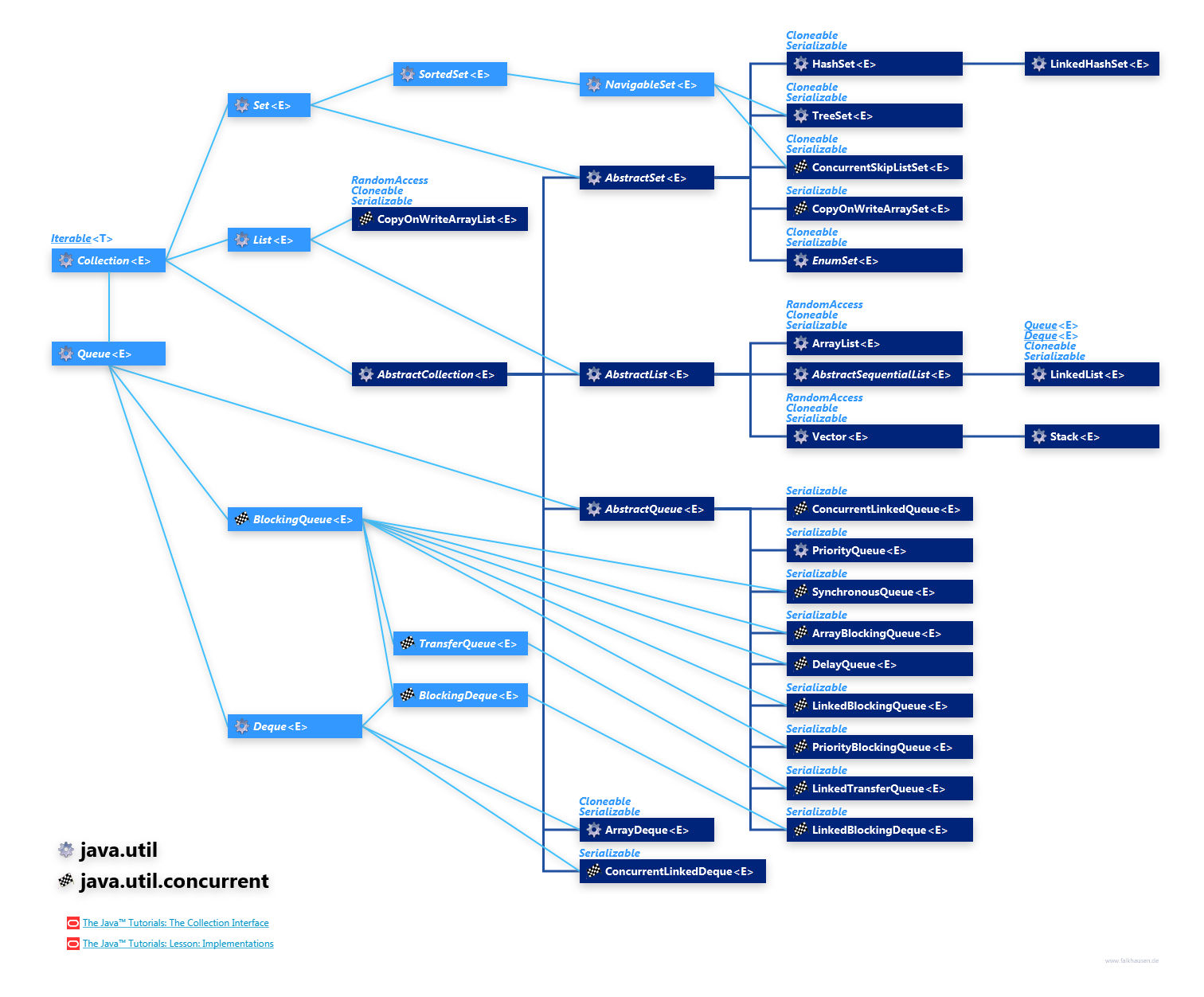
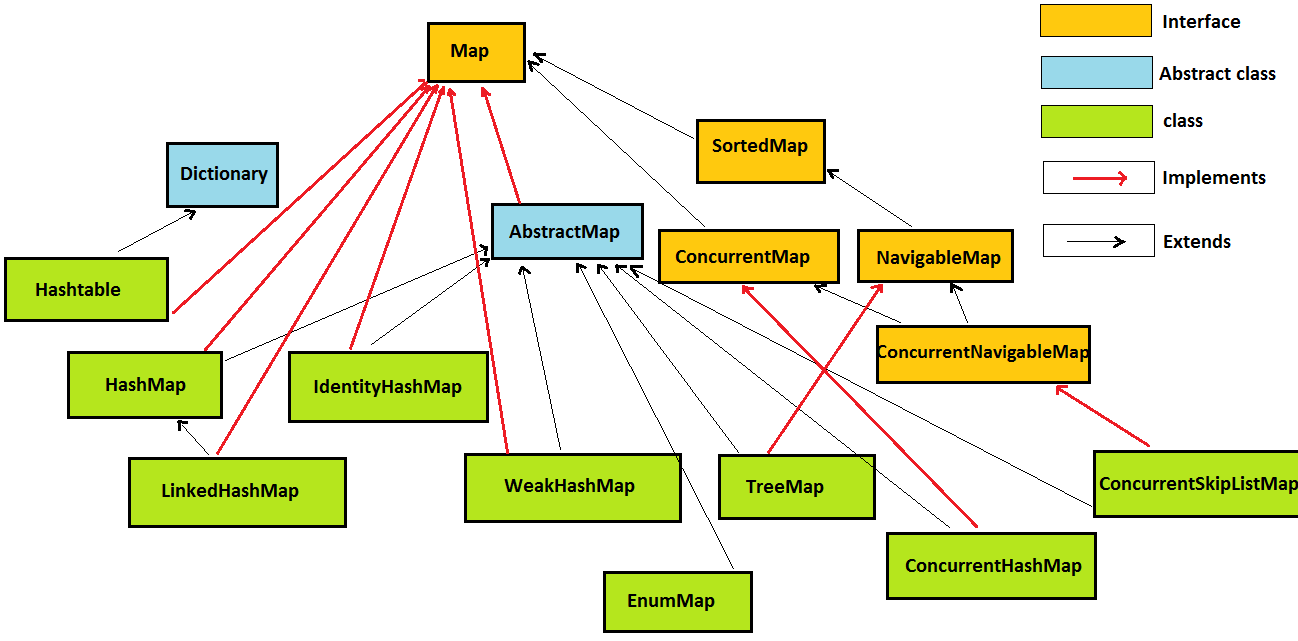


\\





Q. Why Map hierarchy is different than Collection.

=> Collection store single object where Map stores two objects in the form of key and value. Therefore Map is not a Collection.

What is Collection?

Group of individual objects

Q. What is difference between Array and Collection?

fixed in size-------growable in nature

wrt performance Array is required.Because in collection entire collection is copied in another collection...

Array -no underlying DS

Collection - standard DS

Array- manually searching, sorting

Collection - readymade methods like contains, isEmpty

Q What is Collection Framework

If defines several classes and interfaces which can be used a group of objects as single entity.

Just use readymade api and perform the operation.

What is difference between Collection and Collections?

=>

Collection - interface

Collections - utility class present in java.util package which defines several methods like searching, sorting for collection objects.

About version:

1.0 Enumeration, Stack, Vector

1.2 = Collection Framework introduced

1.4 = LinkedHashSet, LinkedHashMap

1.5 = ConcurrentHashMap, CopyOnWriteArrayList, CopyOnWriteArraySet

1.6 = NavigableMap, NavigableSet

1.8 = underlying DS algorithm is changed for HashMap

= many api and methods introduced from Java 8

e.g. Optional - isPresent

Sorting Map directly with Comparators

Iterate over map easily with forEach

Get rid off ugly if-else condition, use getOrDefault method.

Replace and Remove utilities

Do not override keys accidentally use putIfAbsent

operate directly on values.

To merge maps use merge method.

Difference between LinkedList and ArrayList?

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| Underlying DS – growable array | Underlying DS – Doubly LinkedList |
| Implements RandomAccess | Easier to add, modify, delete |
| Use case – good for searching |  |
|  |  |
|  |  |

Difference between HashSet and LinkedHashSet ?

HashSet =>unordered, introduced in 1.2 version

LinkedHashSet=>ordered, introduced in 1.4 version

What are the threadsafe classes in Collection Framework?

* Vector, Hashtable, ConcurrentHashMap, CopyOnWriteArrayList, CopyOnWriteArraySet

What is generic?

**Generics** means **parameterized types**

**Output?**

**import** java.util.HashSet;

**public** **class** HashCodeDemo {

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

HashSet<Employee> emps=**new** HashSet<>();

emps.add(**new** Employee("101","Sachin"));

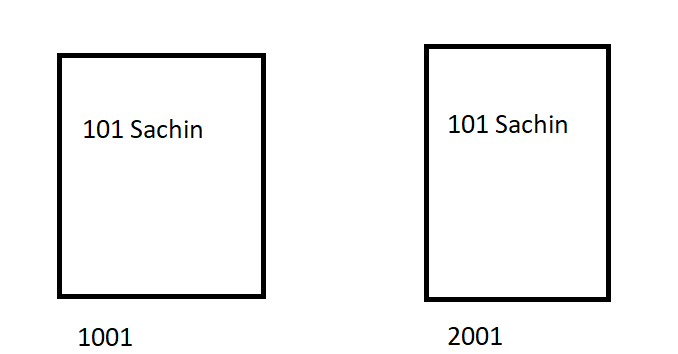
emps.add(**new** Employee("101","Sachin"));

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

}

}

* 2
* Why => HashSet consider unique object based on memory address and not base on contents.
* E.g. in below diagram though the contents are same, memory address is different



Q. If we don’t want to add the additional object with same content, what to do?

=> override hashcode and equals methods

hashCode and equals methods belonging to which class?

* Object class

What is purpose of hashCode?

* Returns a hash code value for the object.

What is purpose of equals?

=>Indicates whether some other object is "equal to" this one.

Signature of hashcode method:

public int hashCode(){…}

Signature of equals method

public boolean equals(Object obj){…}

Is it compulsory to override both the methods?

As far as the syntax is concern, you can skip any method but at runtime you won’t get expected output.

Therefore it is compulsory to override both the methods.

Can we return 420 number from hashCode method

* Yes, (worst algorithm)

How many times hashCode method is called?

* Each time you add the object, hashCode method is called

How many times equals method is called?

* hashCode method decides whether to call equals or not based on hashcode matching of two objects.

If two hashcodes are same, their object may not be equal

But if two objects are same their hashcode must be same

How many null values we can add as a key?

* One

If we try to add multiple null values as a key what happens? E.g.

h.put(**null**, "AB");

h.put(**null**, "BA");

* It considers the latest one i.e. “BA”

What is HashMap?

* Hash table based implementation of the Map interface.
* permits null values and the null key
* unsynchronized
* unordered

What is initial capacity of HashMap?

default initial capacity (16) and the default load factor (0.75).

how to calculculat hashcode?

* “AB”.hashCode()

65\*31 ^1 + 66\*31^0

2015 + 66 \*1

2015 + 66

=2081

Debug

How to attach source code?

Use any predefined class then ctrl+click then attach source code

What is Node in HashMap?

Node is static inner class of HashMap.

It contains 4 fields:

final int hash;

final K key;

V value;

Node<K,V> next;

final int hash – to store hash key. Generates hash key using key in HashMap.

final K key – to store key of HashMap

V value – to store value of HashMap

Node<K,V> next : to store address of next node, by default value is null

How the index position is calculated in HashMap?

index = hashCode(key) & (n-1).

where n is number of buckets or the size of array.

Default size of bucket = 16

How HashMap works internally in Java?

AB – 2081

Index = hashCode(key)&(n-1)

Index = 2081 & (16-1)

Index = 2081 & 15

Index = 1

Null objects are stored on 0th position.

You should use immutable object as a key in HashMap.

If immutable, the object's hashcode wont change and it allows caching the hashcode of different keys which makes the overall retrieval process very fast. Also for mutable objects ,the hashCode() might be dependent on fields that could change, if this happens you won’t be able to find the key (and its value) in the HashMap since hashCode() returns different value.

All the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable.

How many ways are there to print List object?

* Directly printing object( as toString method is already overridden)
* Index position (get(index))
* Enhanced/advanced for loop
* iterator()
* listIterator()
* Java 8

Iterator vs listiterator

Iterator:

Helps to traverse Map, List and Set.

Iterator can traverse only in forward direction

Cannot modify or replace elements present in Collection

Cannot add elements and it throws ConcurrentModificationException

Listiterator:

Can only traverse List and not the other two.

ListIterator traverses both in forward and backward directions.

We can modify or replace elements with the help of set(E e)

Can easily add elements to a collection at any time.

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price. | The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc. |
| 2) Comparable **affects the original class**, i.e., the actual class is modified. | Comparator **doesn't affect the original class**, i.e., the actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is present in **java.lang** package. | A Comparator is present in the **java.util** package. |
| 5) We can sort the list elements of Comparable type by **Collections.sort(List)** method. | We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method. |

