Explain the Java Collection framework

The key interfaces used by the collection framework are **List**, **Set, Queue** and **Map**. The List, Set and Queue extends the **Collection** interface.

Should not confuse the Collection interface with the **Collections** class which is a utility class.

A **Set** is a collection with unique elements and prevents duplication within the collection.

**HashSet** and **TreeSet** are implementations of a Set interface.

A **List** is a collection with an ordered sequence of elements and may contain duplicates.

**ArrayList, LinkedList** and **Vector** are implementations of a List interface.

The Collection API also supports maps, but within a hierarchy distinct from the Collection interface.

A **Map** is an object that maps keys to values, where the list of keys is itself a collection object. A map can contain duplicate values, but the keys in a map must be distinct.

**HashMap, TreeMap**and **Hashtable** are implementations of a Map interface.

**How to implement collection ordering?**

**SortedSet** and **SortedMap** interfaces maintain sorted order.

The classes, which implement the **Comparable** interface, impose natural order.

For classes that don’t implement comparable interface, or when one needs even more control over ordering based on multiple attributes, a **Comparator** interface should be used.

**Design pattern**:

**What is an Iterator?** An Iterator is a use once object to access the objects stored in a collection.

**Iterator design pattern** (aka Cursor) is used, which is a behavioural design pattern that provides a way to access elements of a collection sequentially without exposing its internal representation.

What is the difference between list and set?

What is the difference between arraylist and linkedlist

Arraylist internally uses a dynamic array to store the elements. Manipulation in the arraylist is very slow as it internally uses array. If any element is removed from the array then all bits has to be shifted in memory. It is better in storing and iterating data

Linked List internally uses doubly linked list to store elements. Manipulation with linkedlist will be faster then the arraylist as there is no shifting is required, it is better in manipulating data

HashMap vs Concurrenthashmap

Non syncronised Syncronised

Not thread safe Thread Safe

Failfast – throws concurrentmodification exception failsafe will modify the hashmap

It allows store null key -- this will not store null key

Faster -- slower then hashmap but faster then hashtable

Concurrenthashmap apply lock on segment level (segment locking or bucket locking)

HashTable will apply lock on entire objects

Comparable vs Comparator

Comparable interface can be used to provide single way of sorting whereas Comparator interface is used to provide different ways of sorting. For using Comparable, Class needs to implement it whereas for using Comparator we don't need to make any change in the class. When we use Comparable then we need to implement the compareTo method and for comparator we need to implement compare()

If a class implements the Comparable interface, objects created from that class can be sorted using Java's sorting algorithms.

In Java, both Comparable and Comparator interfaces are used to sort collections, but there are some differences between them:

Comparable:

It is a part of java.lang package.

It 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 compareTo(Object obj) method is used for sorting elements.

We can sort the list elements of Comparable type by Collections.sort(List) method.

To use Comparable, a class needs to implement this interface and override compareTo() method.

Comparator:

It is a part of java.util package.

It provides multiple sorting sequences. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price.

The compare(Object obj1, Object obj2) method is used for sorting elements.

We can sort the list elements of Comparator type by Collections.sort(List, Comparator) method.

Comparator is typically used when we want to provide multiple ways to sort objects, or when we want to sort objects of a class that did not implement Comparable.

In summary, if sorting of objects needs to be based on natural order then use Comparable whereas if you need to sort objects based on different attributes then use Comparator in Java.

How to make a class immutable?

To create an immutable class in Java, you need to follow these steps:

Declare the class as final so it can’t be extended.

Make all fields private so that direct access is not allowed.

Don't provide setter methods for variables.

Make all mutable fields final so that their value can be assigned only once.

Initialize all fields via a constructor.

Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

Here's an example of an immutable class:

make Defensive copy in the constructor parameters and the accessors they do not leak referneces to your internal state that could be modifieable.

Check String, Integer, Long, and all the wrapper classes are non modifiable

29What is the difference between Collection and a Set?

Collection and Set are the interfaces of Collection framework and Set extends the Collection interface and has a unique futre it will not add duplicate elements.

31 What were the main features of Java 8

Lamda expression, Updated collection framework , Stream API (clean implentation of map/filter/reduce),

CompleteableFuture API - asyncronous programming module. realesd in March 2014

32Do Streams and Collections have common methods?

yes and no -- foreach(

35What is a groupingBy()?

34.On what kind of source can you build a stream?

38.How can you create a Comparator?

Use the factory method of Comparator and this interface has bunch of factory method to create comparator that compare objects using one of their fields it also has default methods to modify the behaviour of the given comparator. It has aslo nullFirst and nullsLast method.

Comarator.comparing(User::lastName).thenCoparing(User::firstName).reversed().

39. Can you cite functional interfaces from before Java 8?

Runnable, Comparator, Iterable

Functional interfaces are backword compatible after java 8 we can implement functional interface using lamda It means that the old interfaces in your application that qualify for functional interfaces and you can use lambad to implement them without having to recomile them that is why the @FunctionalInterface annotation is not mandatory on the functional interface. we have more like Executor, Comparable functional interface.

41. Difference between a sorted and an ordered collection?

Sorted means from smallest to largest.

Ordered means from the first added to the last -means we can access them using an index

How can you print an array on the console?

44. How does a SortedSet work?

It keeps the element in the sorted order. Its a set so it doesn't have any duplicate, the elements we add must be comparable or if they are not must provide a comparator when we create a sorted set this will be used even if your elements not comparable. when iterate over them they will be sorted in the increasing order as java 6 you should favour navigable set instead of sorted set that gives you more method then sorted sets. Default implementation is treeset which implements red black tree

What is the difference between a Collection and a Stream?

How can you join Strings with a separator?

Difference between and intermediate and a terminal operation?

What differences between String and char[]?

How do you the sum the elements of an array?

How can you sort an array?

How can you duplicate an array?

What is a Map?

48, What is a bucket in a Map?

A cell in an array, Hashmap class from the collection framework is backed by an array. when a add a key to hashmap a special hashcode is evaluated for that key to choose which cell of the array will recive the key value pair it has already a different key value pair in this cell, this is called the collision and then linked list is created to add the second pair replaced by your red black tree to minimise the collisions when the amount of the key value pairs reaches 75% of the size of the array it is copied in new larger array and this copying incurs to rehashing all the keys present int the map which may be costly.

What is an ArrayList?

Collection vs Set

LinkedList vs ArrayList

LinkedList are good when you add or remove elements from the list vs arraylist are good in randmly accessing the element and it give good performance in iterating.

What is a sealed type in Java?

sorted vs ordered collection

What is an Iterator?

What is the difference between a Collection and a Stream?

Collections contains objects and A Stream is empty to connect to the source of the object and this can be collection or array many other things.If you are using distnct or sorted then your stream will not be empty.

Collection carries objects around with the methods to handle them

53 What is a Stream?

An in-momery implementation of map/filter/reduce pattern. Stream consumes elemements from source and it can do several of these operations and are organisated in a pipeline where each element is passed from one step of the pipeline to the next one. this is used to filter based on the predicate.It may consists many things summing the elements extracting min and max or adding them to a list. Stream API allows you to conduct these computation in paralel.

58.How many objects can you put in a Collection?

It depends on the implementation!.

Arraylist is backed by an Array so the limit will be Integer.MAX\_VALUE and LinkedList is also same.

same limit is also exists for maps but dont worry there is little chance that you can reach that limit witout reaching other limitation in your applications

59What is the maximum length of a String in Java?

Int.MAXVALue

How does a Set knows that it already contains an element? how to set will ditect duplicates?

Set will not add duplicates, this will be done by using hashcode of the object if it finds something then it will call the equal method. hasset stores its objects in cells of an array, index of the cell for the given object is computed from the hashcode of the of the object. If two different object has same hashcode there is a mecanisam to handle it . it has and don't forget to add equal and hashcode both in your object and do not mutate the objects once it is added cus it will never find that object

61.How is Arrays.asList() working?

It wraps an array and exposes it as an List, we can pass input as vararg and also pass an array, avoid passing an array of premitive as it will create single element with array itself It, if you modify the array you will pass then it will modify the list itself because there is no defensive copy. you can modify the elements of it but you can't add or remove the elements as it is an array.

62.Can you cite some methods from the Stream API?

map(), filter and reduce. you will not use reduce much cus we have many specialized method for it like toList(), forEach(), findAny(), findFirst() are all intermediate methods.

flatMap(), distict(), sorted()

2 things we need avoid using peek() unless you need debug your stream, Don't do any side effect in any mehod of steam api

and also you can also reduce your stream with the collector aPI there are bunch available collectors in collectors factory calss and you can even create your own.

76. How can you remove elements from a Stream?

Use Stream.filter() method. Stream doesn't contain any objects, It consumes the objects from the soruce so removing objects from a stream is actully not correct. we are asking stream to not process certain elements and filtering uses the predicate that test certain properties of the object and decide it should be part of the process or not. there are 2 other patterns does same thing like flatMap() nand mapMulti() these are not really ment for this but in certain cases they can work better filter.

63. What is the var keyword in Java?

this keyword is used in method to specify local variable. var is for local variables only you cann't use it for fields for instance you may use it for anonymous classes and in that case you will have access to your methods you add in your anonymous class that are not defined in original type and the type of an anonymous calss is called non-denotable type you can add fileds and methods to it and access them if you declare your variable with var keyword. you can use your IDE to see the type in third by the compiler and sometimes the result is unexpected

64.How can you create an unmodifiable list?

using List.of()

using Collections.unmodifiableList() will builds and unmodifiable view of the list but the list you give as an argument is not defensive copied, so if you modify it then this modification will be seen throgh the view

68.How can you find duplicates in a list?

It depends on what you need. First check your list has duplicates putting them to hashSet and comparing there size. Stream can be used for this using distinct() and count() functions and this will build the set internally if you need to find the duplicate elements themselves then you need to build an instragram of these eleements stream on the list and grouping the elements by themselves counting them with the couing charcater and these technics are all using Maps a set is infact a map with maps can solve this problem in one pass over your data but you consume more memory as usual there is a trade-off between computing time and memory

70. How is List.of() working?

It will takes the stream of characters or var arg and creates unmodifiable List of defensive copy.

We can use it with var arg to create prefilled list but you can also pass an array in that case remember that if it is an array of premitive types it will produce a list with one element which is your array and this list implementaion does not accept null values

Unmodifiablem, defensive values and null values not allowed -> it throws an exceptions

77How is Set.of() working?

It creates an array of the right size and immediatly fails if there are duplicates

If duplicate is identified illigal argument exception is immidelty thrown. this method only requires when pass pver your data because the duplicates are not allowed in tyour argulments it can create a an array of right size upfront and making the memory conception minimal so all this is pretty optimal, if you are using premitives then you get only one elements in an array whihc is usually not what you wanted

72What is the difference between map() and flatMap()? -

these are 2 methods from stream api which are used to convert objects of given types to another type.

maps can produce the arraylist where as flatmap produce one to many relationships, the mapping function should itself return stream of something and these stream are flattened by flat map method for instance given stream of contries can be flatmap to list of cities and mapping doesn't change the number of objects where flatmap can produce even zero element

83.What list can you pass to a method that takes a List<Number>?

List<Numbers> is not equal to List<Integer> when you use the list<Number> you can add float number also init then that will be converted into integer

87.Difference between toList() and Collectors.toList()?

Both are termial operations of the stream API and Collectors.toList() produces the arraylist and toList() will produce non-modifiable list. toList may be faster.

88.Comparing Doubles with a comparator of Numbers?

Yes, Usable for arrylist of wrapper types. Difing a method whihc takes 2 parameters of type number and it is fine to call it any/ we can use such a comparator to compare integer and floaing point number for instance can be used to sort any list of integers or floating point numbers beacuse all these types extends number. done make use this comparator with ?extends Number. will not be able call any other wrapper types and there is good reason for that

97. What is a ConcurrentModification Exception?

The exception raised when we modify the your collection while iterating over it. Removing elements from the collection while iterating it is not good idea. there is remove method on the iterator that can use just make sure it is supported by your implementation, if you need to remove elements from the predicate for instnace you can use the remove method that exaclty that. Some implementaion of the collections can be ok like blokingQueue or concurrentLinkedQueue.

116What is an enumeration?

A class that declares its only instances at compile time. these are intresting when you know that a class can only have a limited number of instance or objects. it can have a state, even a mutable state, why would you do that and you can declare a constructor for them your enum class is final and extends the class enum so your so your enumeration cannot extend anything and cannot be extended but it can implement interfaces. as you cannot create any more instance of an enumeration at one time using enumeration is the preferred pattren to create singleton instance. example is the natural order factory method of the comparator

117. How is the List.sublist() method working?

It creates a mutable view of your list and its actully a nifty feature and provba

So you can use it to manipulate the first list, for instance you can insert a list in another one at a given place by calling sublist passing the same index for the in from index in the to index and calling add all on this or you can clear a range of indexes from the list with the sublist from to do clear. what you need to select a range indexes and don't need the rest of the list anymore then you can copy

112.what is Iterable?

An interfaces and this interface is extend by collection so it is actully the base interface of many interfaces of the collections framework it has only one abstract method iterator so you can implement it with a lamda expression. All iterables can be used in a for each pattern and this is very handy if you already have an Iterator in your application you can use to create an iterable and then go throught all the values of this iterator using your iteratble in this using foreach pattern. Itrable also declares a for each method so you can also go through the values generated by the iterator using this pattern

Iterable<>

115. What are the four fundamental functional interfaces in Java?

Consumer<String> consumer = s-> System.out.println(s); take a argument and return nothing

Suppiler<Double> supplier = () -> Math.PI; doesn't take anything and return something

Function<String, Integer> length = s-> s.length(); takes something and return something else

Predicate<Integer> predicate = i -> i% 2 == 0; takes argument and return boolean

Runnable task = () -> "I am the fifth"; doesn't take any argument and doesn't return anything

BiConsumer<String, String> biCons = (s1,s2) -> System.out.println(s1+s2)

BiFunction<String, String, Integer>

BiFunction<String, String, Integer>

BiPredicate

takes two arguments

UnaryOperator<String> unary =

BinaryOperator<String> binary = (S1,s2) -> s1 + "" + s2

specilization for premitive type

106. What does Comparable mean?

98.What is a collector?

The object provided by stream api that can use to reduce the streams using the collect method. Passed to Stream.collect() method. Collector is implementaion of the collector interface, you can implement this interface and also have a collecors factory calss with planty of pre build connectors that should cover most of your needs.

we can do anything with collectors like create any sort of collections or maps and group the stream elements using keys and gather them in maps or join them in strings of cahateres. you can even have collecots that can map, filter or flatmap the elements of the stream. the stream API uses downstream collectors is the way to compose collectors.

103.What is a downstream collector?

A Collector , can be crated by collectors factory emthods, some of these take the collector as the last parameter called the downstream collector this collector is used for further process, for ex. groupingBy collector groups your eleemtns using the key

That you can pass to the methods of Collectors

Some factory methods always take downstream collector this is the case needed for mapping,filtering, flatmapping.

Needed for mapping filtering flatMapping

107.How can you create a prefilled map?

using method Map.of() using this we can create the map of 10 elements or you can use Map.entry() takes key value pair to create map using Map.ofEntries

Map.of(1, "one", 2, "aa". -- until 10 pattern) you can't overload this kind of method with vararg

var e1=Map.entry(1, "")

var e2=Map.entry(2,"")

Map.ofEntries(e1,e2)

Map and entrier are non-modifiable, you can not pass the same key

you can't use null key or values

108,What is the return type of max()

An Optional

and this is what you need

8445455640

Null or Integer.MIN\_Value

118 How can you create a Map to hold 20 elements?

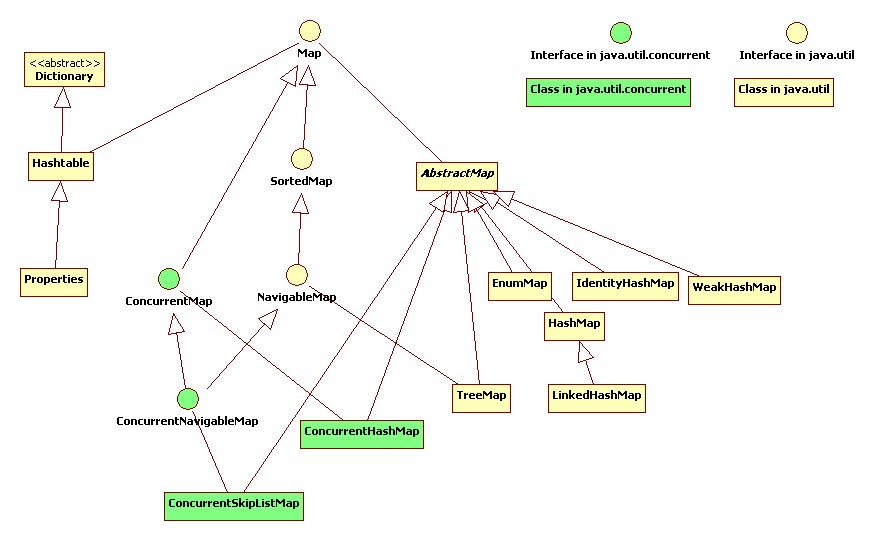
there is factory method to this Hashmap.newHashmap(24), LinkedHashmap.newHashmap(36)

creating a map with the given capacity is more complex then it seems, hashmap carries an internal array when the number is used cells reaches certain lavel then the content of the array is copied on larger array which is copy this is the reason why it may be intresting to create an array that is large enough upfront and then there is some math to do get this size that has been returns in this factory method. one very imp becarefull hasmap has also a constructor which takes that takes an int that is a size but this size is actully the size of internal array which may be too small to store all the key value pairs .. Map hashmap = new Hashmap(20)

119.How can you use flatMap to filter a stream?

Flatmap should return empty stream if the result can't be computed is probably better choice

another method to do the same thing that will save the consturction of this stream object its map multimethod added in jdk 16



https://en.wikipedia.org/wiki/Java\_collections\_framework#Extensions\_to\_the\_Java\_collections\_framework

What is the main difference between an ArrayList and a Vector?

What is the main difference between Hashmap and Hashtable?

Original classes before the introduction of Collections API. *Vector* & *Hashtable* are synchronized. Any method that touches their contents is thread-safe. So if you don’t need a thread safe collection, use the *ArrayList* or *Hashmap*. Why pay the price of synchronization unnecessarily at the expense of performance degradation.

**So which is better?** As a general rule, prefer *ArrayList/Hashmap* to *Vector/Hashtable*. If your application is a

multithreaded application and **at least one of the threads either adds or deletes an entry into the collection**

then use new Java c*ollection* API‘s external synchronization facility as shown below to **temporarily synchronize**

your collections as needed:

Map myMap = **Collections.synchronizedMap**(myMap);

List myList = **Collections.synchronizedList**(myList);

Java arrays are even faster than using an *ArrayList/Vector* and perhaps therefore may be preferable.

*ArrayList/Vector* internally uses an array with some convenient methods like add(..), remove(…) etc.

**What are the benefits of the Java collection framework?**

Collection framework provides flexibility,performance, and robustness.

􀂃**Polymorphic algorithms** – sorting, shuffling, reversing, binary search etc.

􀂃**Set algebra** - such as finding subsets, intersections, and unions between objects.

􀂃**Performance** - collections have much better performance compared to the older *Vector* and *Hashtable*

classes with the elimination of synchronization overheads.

􀂃**Thread-safety** - when synchronization is required, wrapper implementations are provided for temporarily

synchronizing existing collection objects.

􀂃**Immutability** - when immutability is required wrapper implementations are provided for making a collection

immutable.

􀂃**Extensibility** - interfaces and abstract classes provide an excellent starting point for adding functionality and

features to create specialized object collections.

What are some of the best practices relating to Java collection?

􀂃Use ArrayLists, HashMapetc as opposed to Vector, Hashtableetc, where possible to avoid any

synchronization overhead. Even better is to use just arrays where possible. If multiple threads concurrently

access a collection and **at least one of the threads either adds or deletes an entry into the collection**,

then the collection must be externally synchronized. This is achieved by:

Map myMap = **Collections.synchronizedMap**(myMap);

List myList = **Collections.synchronizedList**(myList);

􀂃Set the initial capacity of a collection appropriately (e.g. ArrayList, HashMapetc). This is because collection

classes like ArrayList, HashMapetc must grow periodically to accommodate new elements. But if you have a

very large array, and you know the size in advance then you can speed things up by setting the initial size

appropriately.

**For example**: HashMaps/Hashtables need to be created with sufficiently large capacity to minimise

**rehashing**(which happens every time the table grows). HashMap has two parameters initial capacity and

load factor that affect its performance and space requirements. Higher load factor values (default load factor

of 0.75 provides a good trade off between performance and space) will reduce the space cost but will

increase the lookup cost of myMap.get(…) and myMap.put(…) methods. When the number of entries in the

HashMap exceeds the **current capacity \* loadfactor**then the capacity of the HasMap is roughly doubled by

calling the rehash function. It is also very important not to set the initial capacity too high or load factor too

low if iteration performance or reduction in space is important.

􀂃**Program in terms of interface not implementation**: For example you might decide a LinkedList is the best

choice for some application, but then later decide ArrayList might be a better choice for performance reason.

**CO**

**Use**:

**List list**= new ArrayList(100); //program in terms of interface & set the initial capacity.

**Instead of**:

ArrayList list = new ArrayList();

􀂃**Avoid storing unrelated or different types of objects into same collection**: This is analogous to storing

items in pigeonholes without any labelling. To store items use **value objects** or **data objects** (as oppose to

storing every attribute in an ArrayList or HashMap). Provide wrapper classes around your collection API

classes like ArrayList, Hashmapetc as shown in better approach column. Also where applicable consider

using**composite design pattern**, where an object may represent a single object or a collection of objects.

Refer **Q52** in Java section for UML diagram of a composite design pattern.

When providing a user defined key class for storing objects in the Hashmaps or Hashtables, what methods do you

have to provide or override (i.e. **method overriding**)?

You should override the **equals()** and **hashCode()** methods from the *Object* class. The default implementation of

the equals() and hashcode(), which are inherited from the java.lang.Object uses an object instance’s memory

location (e.g. MyObject@6c60f2ea). This can cause problems when two instances of the car objects have the

samecolour but the inherited equals() will return false because it uses the memory location, which is different for

the two instances. Also the **toString()** method can be overridden to provide a proper string representation of your

object. ***Points to consider:***

• If a class overrides **equals()**, it must override **hashCode()**.

• If 2 objects are equal, then their hashCode values must be equal as well.

• If a field is not used in equals(), then it must not be used in hashCode().

• If it is accessed often, hashCode() is a candidate for caching to enhance performance.

**Note**: Java 1.5 introduces enumerated constants, which improves readability and maintainability of your code.

Java programming language enums are more powerful than their counterparts in other languages. E.g. A class

like*Weather* can be built on top of simple enum type *Season* and the class *Weather* can be made immutable, and

only one instance of each *Weather can* be created, so that your *Weather* class **does not have to override**

**equals()** and **hashCode()** methods.

public class Weather {

**public enum Season {WINTER, SPRING, SUMMER, FALL}**

private final Season season;

**private static final List<Weather>listWeather = new ArrayList<Weather> ();**

private Weather (Season season) { this.season = season;}

public Season getSeason () { return season;}

static {

for (Season season : Season.values()) {

listWeather.add(new Weather(season));

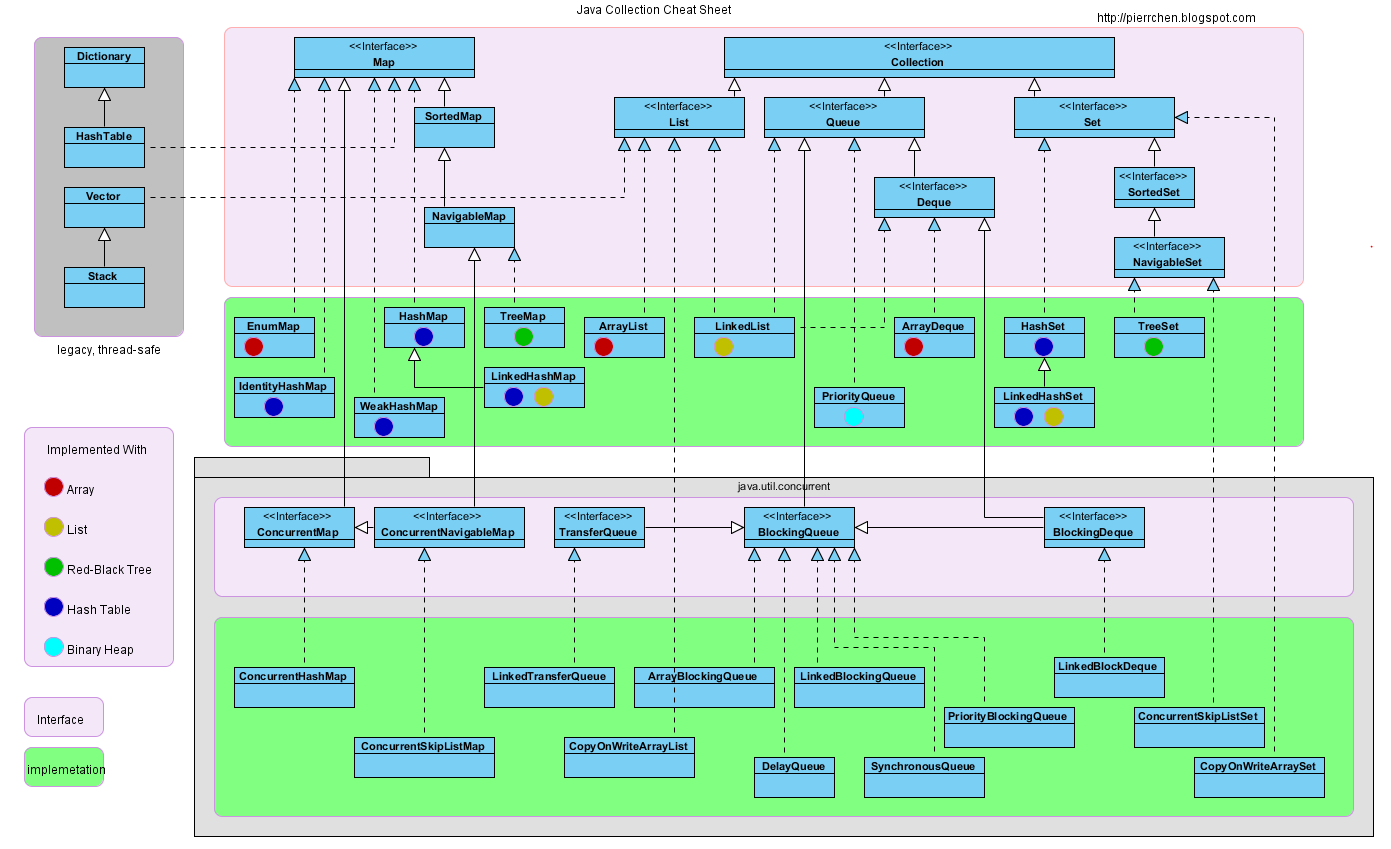
}

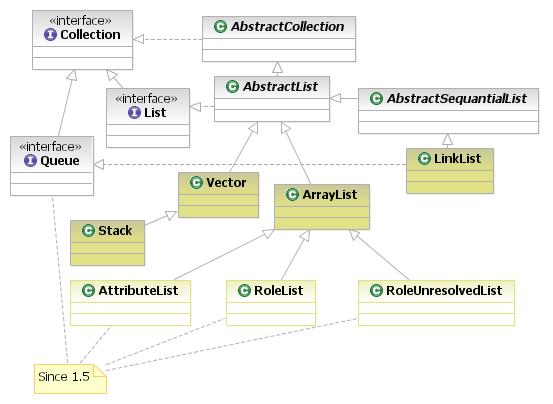
}

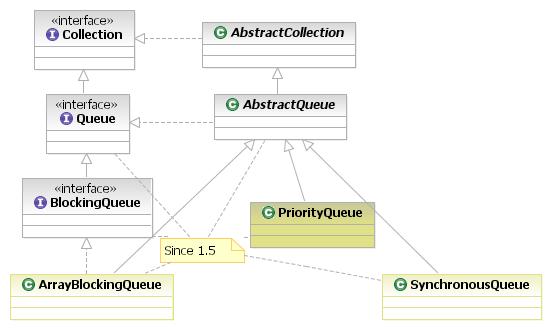
public static ArrayList<Weather>getWeatherList () { return **listWeather;** }

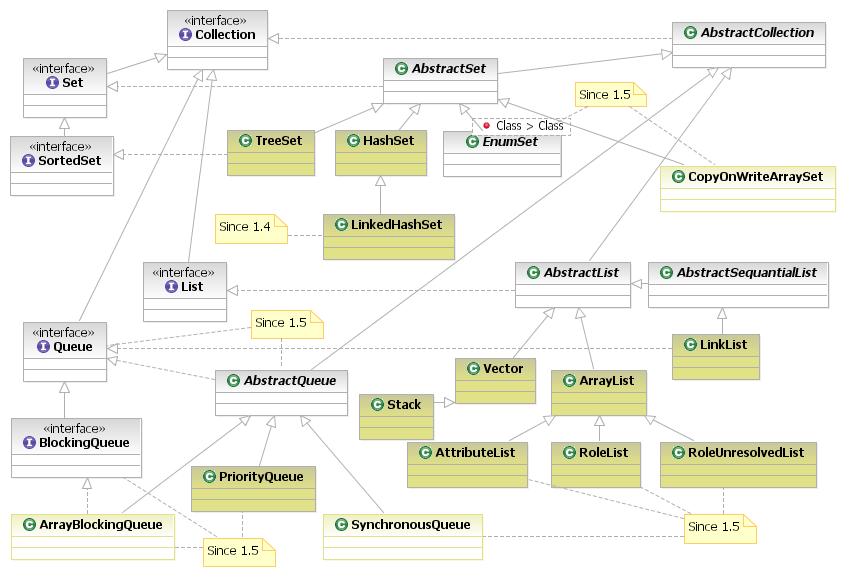
public String toString(){ return season;} // takes advantage of toString() method of Season.

}









List interface

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two examples for concrete classes that implement List are:

java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it.

java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.[10]

Stack class

Stacks are created using java.util.Stack. The stack offers methods to put a new object on the stack (method push()) and to get objects from the stack (method pop()). A stack returns the object according to last-in-first-out (LIFO), e.g. the object which was placed latest on the stack is returned first. java.util.Stack is a standard implementation of a stack provided by Java. The Stack class represents a last-in-first-out (LIFO) stack of objects. It extends class java.util.Vector with five operations that allow a vector to be treated as a stack. The usual push and pop operations are provided, as well as a method to peek at the top item on the stack, a method to test for whether the stack is empty, and a method to search the stack for an item and discover how far it is from the top. When a stack is first created, it contains no items.

Queue interfaces

The java.util.Queue interface defines the queue data structure, which stores elements in the order in which they are inserted. New additions go to the end of the line, and elements are removed from the front. It creates a first-in first-out system. This interface is implemented by java.util.LinkedList, java.util.ArrayDeque, and java.util.PriorityQueue. LinkedList, of course, also implements the List interface and can also be used as one. But it also has the Queue methods. ArrayDeque implements the queue as an array. Both LinkedList and ArrayDeque also implement the java.util.Deque interface, giving it more flexibility.[11]

java.util.Queue can be used more flexibly with its subinterface, java.util.concurrent.BlockingQueue. The BlockingQueue interface works like a regular queue, but additions to and removals from the queue are blocking. If remove is called on an empty queue, it can be set to wait either a specified time or indefinitely for an item to appear in the queue. Similarly, adding an item is subject to an optional capacity restriction on the queue, and the method can wait for space to become available in the queue before returning.[12]

java.util.PriorityQueue implements java.util.Queue, but also alters it. Instead of elements being ordered by the order in which they are inserted, they are ordered by priority. The method used to determine priority is either the compareTo() method in the elements or a method given in the constructor. The class creates this by using a heap to keep the items sorted.[13]

Double-ended queue (deque) interfaces

The java.util.Queue interface is expanded by the java.util.Dequesubinterface. Deque creates a double-ended queue. While a regular queue only allows insertions at the rear and removals at the front, the deque allows insertions or removals to take place both at the front and the back. A deque is like a queue that can be used forwards or backwards, or both at once. Additionally, both a forwards and a backwards iterator can be generated. The Deque interface is implemented by java.util.ArrayDeque and java.util.LinkedList.[14]

The java.util.concurrent.BlockingDeque interface works similarly to java.util.concurrent.BlockingQueue. The same methods for insertion and removal with time limits for waiting for the insertion or removal to become possible are provided. However, the interface also provides the flexibility of a deque. Insertions and removals can take place at both ends. The blocking function is combined with the deque function.[15]

Set interfaces

Java's java.util.Set interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet, java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a java.util.HashMap to store the hashes and elements and to prevent duplicates. java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. java.util.TreeSet uses a red-black tree implemented by a java.util.TreeMap. The red-black tree makes sure that there are no duplicates. Additionally, it allows TreeSet to implement java.util.SortedSet.[16]

The java.util.Set interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet[17]

java.util.SortedSet is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.[18]

Map interfaces

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map.[19]

The java.util.Map interface is extended by its subinterface, java.util.SortedMap. This interface defines a map that's sorted by the keys provided. Using, once again, the compareTo() method or a method provided in the constructor to the sorted map, the key-value pairs are sorted by the keys. The first and last keys in the map can be called. Additionally, submaps can be created from minimum and maximum keys. SortedMap is implemented by java.util.TreeMap.[20]

The java.util.NavigableMap interface extends java.util.SortedMap in various ways. Methods can be called that find the key or map entry that's closest to the given key in either direction. The map can also be reversed, and an iterator in reverse order can be generated from it. It's implemented by java.util.TreeMap.[2

Java collections framework is extended by the [Apache Commons](https://en.wikipedia.org/wiki/Apache_Commons) Collections library, which adds collection types such as a bag and bidirectional map, as well as utilities for creating unions and intersections.[[22]](https://en.wikipedia.org/wiki/Java_collections_framework#cite_note-22)

Google has released its own collections libraries as part of the [guava libraries](https://en.wikipedia.org/wiki/Google_Guava).