Name:Pedro Jahir Hinojosa Garcia
Date: 08/24/2024

The objective of this document is to resume the main characteristics of collection group in java 17.

The collection group in java are useful interfaces that behaves like an object of objects. There are four principal collection:

1. List: A list is allows duplicate elements as an ordered collection. The elements can be accessed by an index wich is an int.
2. Set: Does not allow duplicate entries
3. Queue: The elements are ordered by a given priority
4. Map: A map contains objects with objects in it, and are called keys and values. The keys can't repeat but the keys can have the same values. As well a key can have more than one value.

Every single collection type has a different set of functions and different ways to declare them.

List: List<Integer> list = new ArrayList<Integer>t();

Set: Set<String> mySet = new HashSet<>();

Queue: Queue<String> myQueue = new LinkedList<>();

Map: Map<String, Integer> myMap = new HashMap<>();

In the next section the methods that different collections use, wil be presented.

List Method	Description
add	Add elements to the end or to an specific given index
get	Returns element at specific index
remove	Remove elements at specific index
replaceAll	Replace elements in the list resulting from an operator
set	Replace a number at an index and return the original
sort	Accommodate the data to an specific task

Table 1: List's methods

List Method	Description
add	Add elements to the end or to an specific given index
check	Check if an element is on the set
remove	Remove elements at specific index

isEmpty	Check if the set is empty
size	Give an int with the size of the set
clear	Clear the set

Table 2: Set's methods

List Method	Description
add	Add elements at the back of the queue
peek	Read from the front to the back
remove	Get the element from the front and remove it
offerFirst	Add element to front
peekLast	Reed the set backwards
pollLast	Remove from back

Table 3: Queue's methods

List Method	Description
forEach	Loops to every key value pair
get	Returns a value given a key
getOrDefault	REturns a mapped value or default of none is mapped
keySet	Returns a set with the keys
put	Replace key/values pair. Returns previous values or none
values	Return a collection of values

Table 4: Queue's methods

The following definitions are the last comments about the collection classes.
ArrayList: Definition: ArrayList is a resizable array implementation of the List interface. Characteristics: It allows duplicate elements, maintains the insertion order, provides fast random access with O(1) time complexity for getting an element, but is slower for insertion and deletion operations except when adding or removing at the end.

List:
Definition: List is an interface that represents an ordered collection (sequence) of elements. Characteristics: It allows duplicate elements, maintains the order of elements based on insertion, and provides methods to access elements by their index. Common implementations include ArrayList, LinkedList, Vector, and Stack.

LinkedList:
Definition: LinkedList is a doubly linked list implementation of the List and Deque interfaces.
Characteristics: It allows duplicate elements, maintains insertion order, is faster for insertion and deletion operations compared to ArrayList (O(1) time complexity for adding or removing elements at the beginning or end), but slower for random access with O(n) time complexity for getting an element by index.

PriorityQueue:
Definition: PriorityQueue is a queue implementation that orders its elements according to their natural ordering or a provided comparator.
Characteristics: It does not allow null elements, does not guarantee the order of elements, ensures the element at the head of the queue is the least (or greatest, depending on the comparator), and provides O(log n) time complexity for insertion and removal operations.

ArrayDeque:
Definition: ArrayDeque is a resizable array implementation of the Deque interface.
Characteristics: It does not allow null elements, provides fast access with O(1) time complexity for insertion and deletion operations at both ends, and is more memory-efficient than LinkedList when used as a stack or queue.

HashSet:
Definition: HashSet is a hash table implementation of the Set interface.
Characteristics: It does not allow duplicate elements, does not maintain any order of elements, and provides O(1) time complexity for insertion, deletion, and lookup operations.

LinkedHashSet:
Definition: LinkedHashSet is an ordered version of HashSet that maintains a linked list of the entries in the set.
Characteristics: It does not allow duplicate elements, maintains the insertion order of elements, and provides O(1) time complexity for basic operations.

SortedSet:
Definition: SortedSet is an interface that extends Set and provides a total ordering on its elements.
Characteristics: It does not allow duplicate elements, maintains elements in a sorted order according to their natural ordering or a specified comparator.

TreeSet:
Definition: TreeSet is a red-black tree-based implementation of the SortedSet interface.
Characteristics: It does not allow duplicate elements, maintains elements in their natural order (ascending) or according to a specified comparator, and provides O(log n) time complexity for insertion, deletion, and lookup operations.

Map:
Definition: Map is an interface that represents a collection of key-value pairs.

Characteristics: It does not allow duplicate keys, each key maps to exactly one value, and provides methods to add, remove, and retrieve elements by key. Common implementations include HashMap, LinkedHashMap, TreeMap, and Hashtable.

SortedMap:
Definition: SortedMap is an interface that extends Map and maintains the keys in a sorted order.
Characteristics: Keys are sorted according to their natural order or a specified comparator, and it does not allow duplicate keys.

TreeMap:
Definition: TreeMap is a red-black tree-based implementation of the SortedMap interface.
Characteristics: It maintains keys in their natural order (ascending) or according to a specified comparator, does not allow duplicate keys, and provides O(log n) time complexity for insertion, deletion, and lookup operations.

Hashtable:
Definition: Hashtable is a hash table implementation of the Map interface.
Characteristics: It is synchronized (thread-safe), does not allow null keys or values, does not maintain any order of elements, and is slower than HashMap.

LinkedHashMap:
Definition: LinkedHashMap is an ordered version of HashMap that maintains a linked list of the entries in the map.
Characteristics: It maintains the insertion order of elements, allows null keys and values, and provides O(1) time complexity for basic operations.

HashMap:
Definition: HashMap is a hash table-based implementation of the Map interface.
Characteristics: It allows null keys and values, does not maintain any order of elements, and provides O(1) time complexity for insertion, deletion, and lookup operations.