JAVA: COLLECTIONS

EVER HINOJOSA AGUIRRE

GENERICS

To understand collections in java we must have an undesrtand of generics, What is generics?

Generics is a feature in JAVA that allows us to write code that can handle multiple types of objects, it allows to define clases, interfaces and methods with placeholder for the type that they will use. This avoids code duplication.

GENERICS

Type Variable	Meaning
E	Element type in a collection
K	Key type in a map
V	Value type in a map
T	General type
S, U	Additional general types

To define a generic type we have to use brackets <> with a placeholder usually “T”.

In this generic class, we define a container that Works as a placeholder that Works for any type when creating an instance of “Container”.

```
public class Container<T> {
    private T content;

    public void setContent(T content) {
        this.content = content;
    }

    public T getContent() {
        return content;
    }
}
```

GENERICS

With the generic class generic “Container” we can instance it with a string or with an integer:

```
Container<String> cs = new Container<>();
stringBox.setContent("COOL BEANS");

Container<Integer> ci = new Container<>();
integerBox.setContent(619);
```

GENERICS

When working with generics we can use wildcards which are represented with <?>, these three types of wildcards are:

1. Unbounded Wildcard <?>

   Represents unkown, doesnt care the type of object used and reads from collection.

2. UpperBound Wildcard <? extends T>

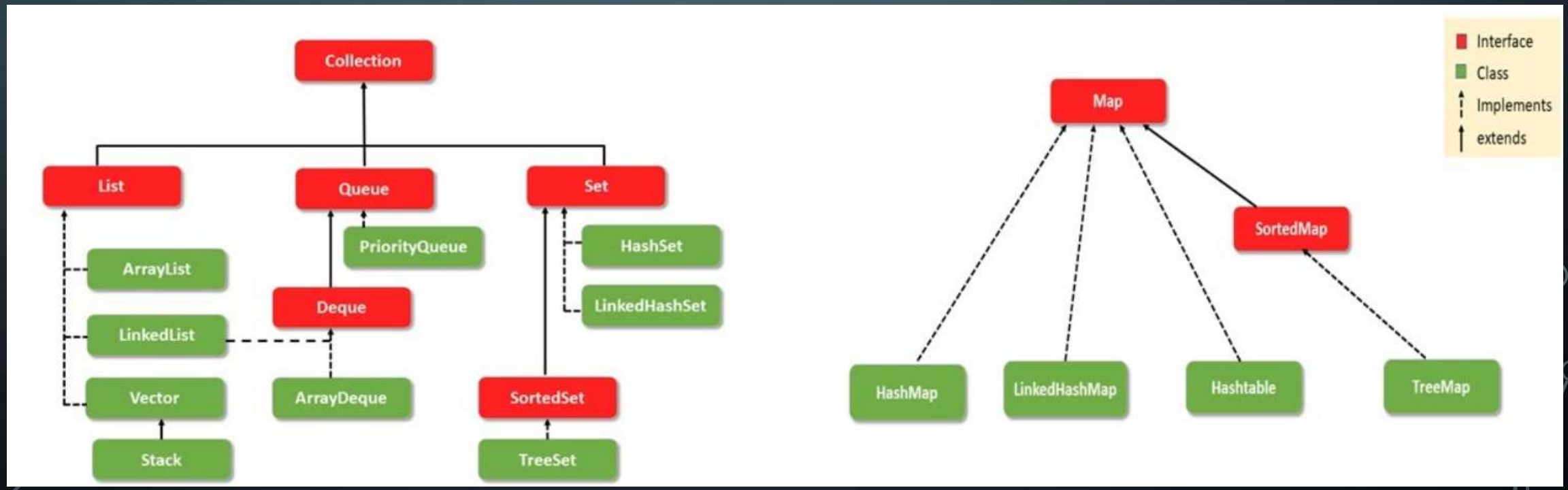
   When the object you want must be a subtype of the specified type, represents the class “T” and any of its subclasses, it also reads from collection.

3. LowerBound Wildcard<? super T>

   Allows the insertion of elements in the collection as long as they are “T” type elements or any of the subclasses.

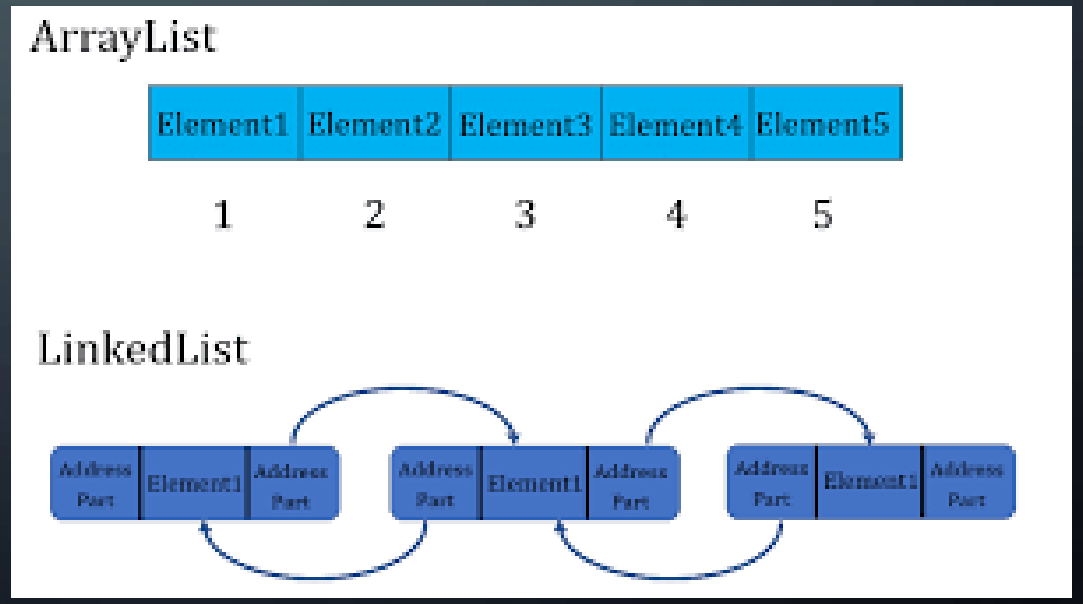
COLLECTIONS

Collections in Java refers to a frameworkthat provides clases and interfaces to handle groups of objects, includes such data structures as lists, sets and maps.



COLLECTIONS

- List: An ordered list that allows duplicates which can be accessed from their index.
- ArrayList: A resizable array, we can obtain the elements from their index.
- LinkedList: A doubly-linked list, each element in the list knows the previous element and the next element.



COLLECTIONS: LIST

KEY METHODS:

add(E e)	Adds an element to the list.
get(int index)	Gets the element of the specified index location.
remove(int index)	Removes the element from the specified index location.
set(int index, E element)	Replaces the element at the specified index location.
size()	Returns the number of elements in the list.
contains(Object o)	Checks the list to see if it contains the element.
clear()	Removes all elements from the list.

COLLECTIONS

- Set: A collection that does not allow duplicates elements and does not guarantee any order.
- HashSet: uses a hash table, doesnt have order.
- LinkedHashset: maintains a linked list of entries, preserving insertion order.
- Treeset: A sorted set, ordered according to the natural ordering of it´s elements or by comparator.

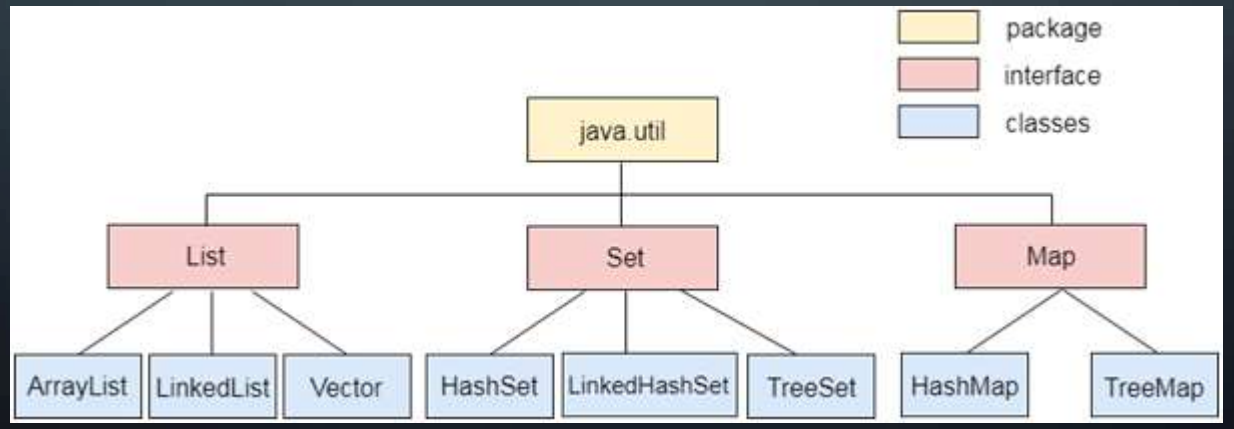
COLLECTIONS: SET

KEY METHODS:

add(E e)	Adds an element to the set (No duplicates).
remove(Object o)	Removes the specified element.
size()	Returns the number of elements in the set.
contains(Object o)	Checks the set to see if it contains the element.
clear()	Removes all elements from the set.
Iterator()	Returns an iterator over the elements in the set.

COLLECTIONS

- Map: a collection of key-value pairs, in which the key is unique like the primary key in a db.
- Hashmap: It doesnt maintan an order and implements the “map” interface.
- LinkedHashMap: Maintains insertion order or Access order.
- TreeMap: implements a sorted map, ordered by keys.



COLLECTIONS: MAP

put(K key, V value)	Links the specified value with the specified key
get(Object key)	Gets the value associated with the key.
remove(Object key)	Removes the value associated with the key.
containsKey(Object key)	Checks the map for the specified key.
containsValue(Object value)	Checks the map for the specified value.
size()	Returns the value of the key valued pairs in the map.
keySet()	Returns a set of all the keys in the map.
values()	Returns a collection of all the values in the map.
entrySet()	Returns a set of all key-value pairs (entries) in the map.
Clear()	Removes all key-values entires from the map.

CONCLUSIONS

The generics and collection framework in java provides a fantastic way to deal with large volumes of distinct data, in many ways that allows more effective way to manage data and makes it more effective to not re-write code. Even if i have used hashmaps before i wasn´t really aware of the innerworkings of it and in the collection spectrum the use of lambdas in it.