Difference between List, Set and Map interface of Collection Framework

Every Java Student or Java Developer must know about Java Collection Framework. It has list of interfaces and classes which are widely used in most java application. List, Set and Map are most often used interfaces when it comes to storing elements and difference between them is one of the legacy question asked at typical Java Developer interview on collection framework. Your answer for this question will help interviewer to measure your knowledge about fundamentals of collection framework.

Differences based on properties of an Interface.

All collection classes have different underlying data structures and that is the primary reason for their behavior and properties.

Collection classes can be easily differentiate and identified from their characteristics. Here are some key measurable characteristics for collection classes

- Duplicate Objects
- Null Objects
- Preserve order of elements.
- Cursors used to iterate over data.
- Underlying data structure.

Now, we will compare List, Set and Map on this characteristics so we can easily identify right Interface to use in specific situation.

Duplicate Objects

List and Set are identical interface which stores objects, while map stores key value pairs. One primary difference between list and set is List allow duplicate objects while Set contract says that “No duplicate objects allowed”. While in case of Map, It can have duplicate values but key should be unique.

So it’s straight forward statement, if you want unique values to be stored in a collection then use Set interface.

equals() methods is being used to detect duplication of object.

Please note that Sorted Set uses compareTo() method for the same purpose.

Null Objects

Another significant different between them is on allowing null objects. List allows null objects and you can have multiple null elements in list because it allows duplicates too. Set just allow one null element as value must be unique in set. Map allows one null key and multiple null values.

Preserve Order of Elements

If you want to access elements by an index then you should consider List over other collection interface. List maintains order of elements so you can retrieve any element through its index. Set is an unordered collection which means element orders are not guaranteed when you retrieve them. Few set implementation classes like LinkedHashSet, SortedSet, SortedMap maintains order. Map is an unordered collection too. Set and Map does not preserve insertion order while List does.

We can use get method with List but we cannot use it with Set and Map due to nature of unordered collection.

Consider an example, where we have a student object:

```
Student firstStudent = new Student(1,"A");
```

Now we will see implementation example of all three interface by using above object.

List:

```
List<Student> studentsList = new ArrayList<>();
Student firstStudent = new Student(1,"A");
studentsList.add(firstStudent);
// null insertion
studentsList.add(null);
studentsList.add(firstStudent);
// duplicate value insertion
studentsList.add(null);
```

We used ArrayList implementation of List interface in above example and inserting multiple duplicate and null values in the list.

Set:

Now we will implement same example using HashSet.

```
Set<Student> studentsSet= new HashSet<>();
studentsSet.add(firstStudent);
// example of null
studentsSet.add(null);
studentsSet.add(firstStudent);
// duplicate value insertion
studentsSet.add(null);
```

Map:

```
Map<Integer, String> studentMap = new HashMap<>();
studentMap.put(firstStudent.getSid(), firstStudent.getSname());
studentMap.put(1, "B");
```

When you enter multiple duplicate values in Set or duplicate keys in map, It won't throw any compilation error. It will just return you one value.

Cursor used to iterate over data

For List, we have special ListIterator, which can be used to perform iteration over list and allow us to execute various functions on elements. We can also use regular Iterator with list but ListIterator is rich and provides methods for additional operations on list.

In case of set, it just returns Iterator object, we don't have any special iterator for it.

Map is collection of key value pair so we have multiple ways to access map. We can access it by keys, by values and through entry set as well.

Now, Lets iterate over above example and compare results in each case.

List:

```
ListIterator listIterator = studentsList.listIterator();
System.out.println("List");
while (listIterator.hasNext()){
    System.out.println(listIterator.next());
}
```

Output:

```
List
Student@1540e19d
null
Student@1540e19d
null
```

We can also use get method as below for List interface implementation.

```
System.out.println("Using get index " + studentsList.get(2));
```

Set:

```
Iterator iterator = studentsSet.iterator();
System.out.println("Set");
while(iterator.hasNext()){
    System.out.println(iterator.next());
}
```

Output:

```
Set
null
Student@1540e19d
```

Map:

```
System.out.println("Iterate using Entry Set");
for(Map.Entry entry : studentMap.entrySet()){
    System.out.println(entry.getKey() + ":" + entry.getValue());
}

System.out.println("Iterate using Key Set");
for(int key : studentMap.keySet()){
    System.out.println(key + ":" + studentMap.get(key));
}
```

Output:

```
Iterate using Entry Set
1:B
Iterate using Key Set
1:B
```

From output of Set and Map, we can see that both of them allow only unique values and keys respectively.

Underlying Data Structures

Lists are based on variable size array and it uses same approach for indexing an elements.

Set uses internal implementation of Map and it is not index based structure

Map uses various hashing techniques to store key-value pairs.

Due to different underlying data structures, their behavior and purpose are quite different from each other.

Other Differences

List and Set both have Collection as parent interface, while Map is not extending Collection interface.

Implementation classes for each of them are as below:

List - ArrayList, Vector, LinkedList

Set – HashSet, LinkedHashSet, TreeSet

Map – HashTable, LinkedHashMap, HashMap, TreeMap

Summary Table

Interface	Duplicates Allowed?	Null Values Allowed?	Insertion order preserved?	Iterator	Data Structure
List	Yes	Yes, Multiple	Yes and can	Iterator,	Array

		null values are allowed	retrieve using index	ListIterator	
Set	No	Yes but only once	No	Iterator	Underlying Map implementation
Map	Not for keys	Yes but only once for keys, Multiple allows for value	No	Through keyset, value and entry set	Hashing techniques

All Implementation classes of these interfaces follows same contract and all of them has both synchronized and unsynchronized implementations.

So now you know the difference between List, Set and Map interfaces and you can easily decide when to use them. But remember all of them might be different in behavior but their basic use is same to store elements and perform basic operations on it.