JAVA Collections

AN INTRODUCTION...

- PRABHANJAN

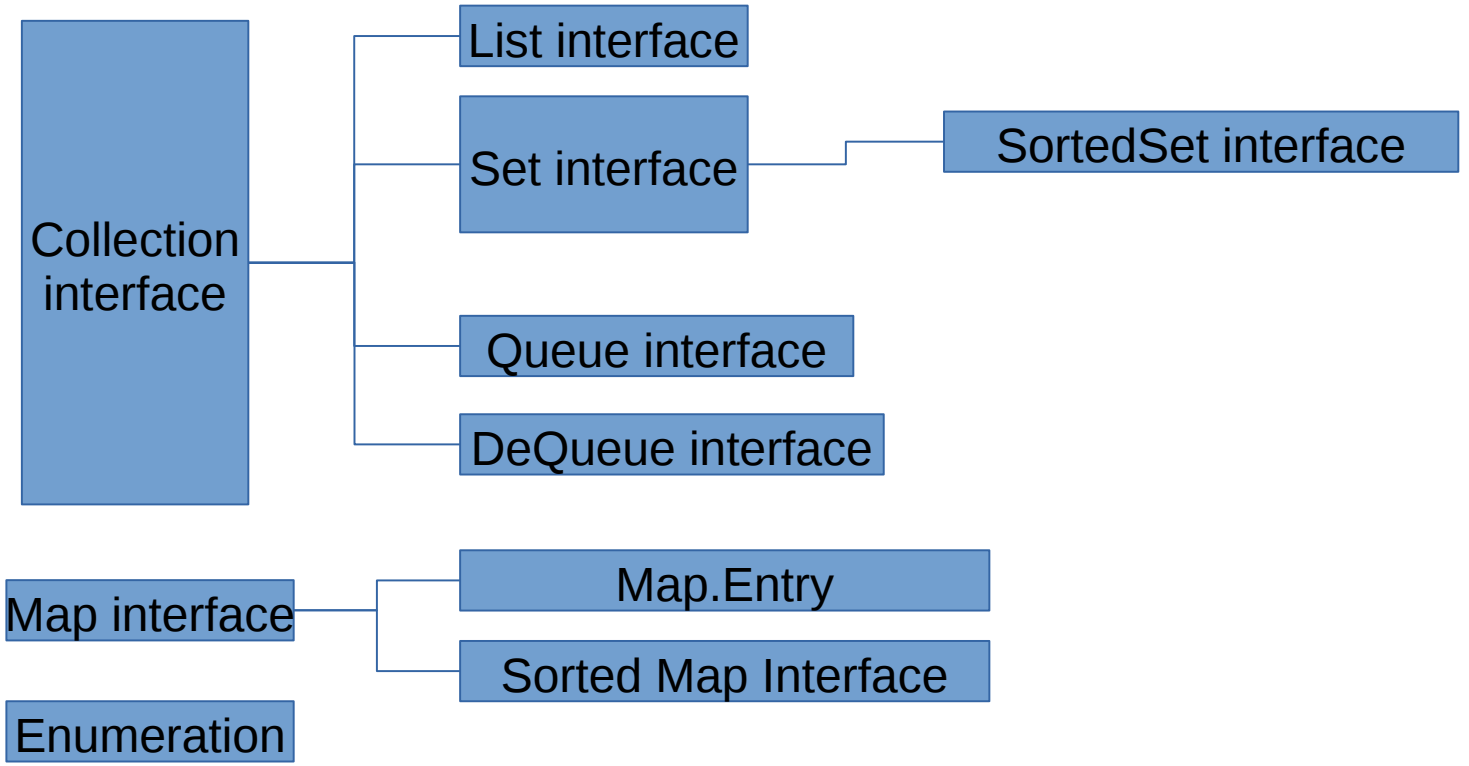
- Prior to Java 2
  - Dictionary
  - Vector
  - Stack
  - Properties
- Quite useful, but...?

The Collections Framework

- Design goals
  - High performance
  - Work in similar manner
  - Extending a collection should be easy.
  - You can have your own collection (Good News!!)
- The framework Contains
  - Interfaces
  - Classes
  - Algorithms

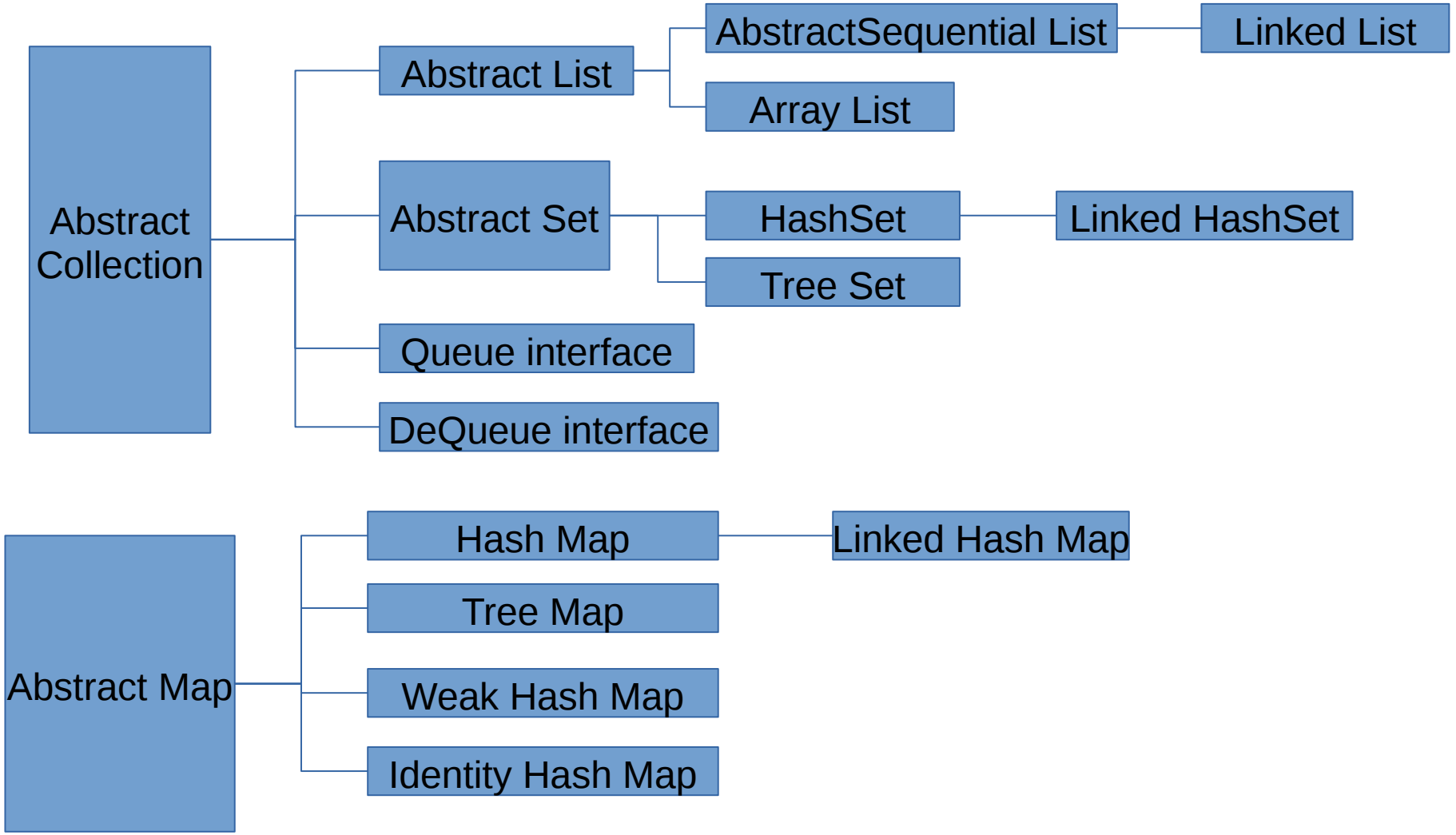
(P.S Just like C++ Standard Template Library)

The Interfaces



(Replaced by Iterator)

The Collection Classes



Algorithms

- Binary Search
- Copy
- Fill
- IndexOfSubList
- LastIndexOfSubList
- Ma/Minx Element
- Max /Min Element in Collection
- Copy
- Reverse
- Shuffle
- Singleton
- Sort
- Swap
- Synchronized operations (Operates on Maps and Lists and Sets)
- UnModifiable (Operates on Lists, Maps, Sets and returns an immutable object)

The Java.util.Collection

- The basic skeleton/Framework that other implementations extend.
- Contains the common core methods that is available

Methods	Methods
boolean add(Object obj)	boolean isEmpty( )
boolean addAll(Collection c)	Iterator iterator( )
void clear( )	boolean remove(Object obj)
boolean contains(Object obj)	boolean removeAll(Collection c)
boolean containsAll(Collection c)	boolean removeAll(Collection c)
boolean equals(Object obj)	boolean retainAll(Collection c)
int hashCode( )	int size( )
Object[ ] toArray()	Object[ ] toArray(Object array[ ])

A comparison

C++ STL	Java Collections
Vectors	Vector
List	List, ArrayList
SList	LinkedList
Stack	Stack
Map	AbstractMap, HashMap, TreeMap, WeakHashMap
Deque	Deque
Set	AbstractSet, HashSet, LinkedHashSet, TreeSet
hash_set, hash_map	Dictionary/Hashtable

Design Patterns in Java.Util.Collection

- Has the powerful design patterns that can be found in LISP, Smalltalk-80, and SCHEME.
- Structural Patterns
  - Adapter
    - Example: Arrays.asList(StringArray)
  - Bridge
    - Example: LinkedHashMap(LinkedHashSet<typeLHS>, List<typeL>)
  - Decorator
    - Example: Collection syncedColl = SynchronizedCollection(Collection C)

      ...

      synchronized(syncedColl) // MUST USE THIS STATEMENT

      {

      Iterator i = syncedColl.Iterator();

      ....

      }
    - Example: Collections.checkedCollection(new HashSet<String>(), String.class)
    - Example: Collections.unmodifiableCollection(list);
- Behavioral Patterns
  - Iterator
    - Example: Iterator i = new ArrayList<int>(Arrays.asList([1,2,3,4])).iterator();

Examples

- **Adapter Pattern Example**

```
String a[] = new  String[ {"abc","klm","xyz","pqr"};
 List list1 = Arrays.asList(a);
System.out.println("The list is:" + list1); // prints [abc, klm, xyz, pqr]
```

- **Bridge Pattern Example**

```
Map<String, Boolean> map = new WeakHashMap<String, Boolean>();
Set<String> set = Collections.newSetFromMap(map);

set.add("Java");
set.add("C");
set.add("C++");
System.out.println("Set is: " + set); // Java, C, C++
System.out.println("Map is: " + map); // {Java=true, C++=true, C=true}

set.add("python");
System.out.println("Set is: " + set); // Java, C, C++, python
System.out.println("Map is: " + map); // {Java=true, C++=true, C=true, python=true}
```

Examples (Contd..)

- **Decorator Pattern Example**

```
ArrayList<String> arlst = new ArrayList<String>();
arlst.add("ABC");
arlst.add("EFGH");
arlst.add("123");
arlst.add("4567");

Collection<String> tslst;
tslst = Collections.checkedCollection(arlst,String.class);
System.out.println("Type safe view is: "+tslst); // prints [ABC, EFGH, 123, 4567
```

- **Iterator Pattern Example**

```
Iterator i = new ArrayList<int>(Arrays.asList([1,2,3,4])).iterator();
while (i.hasNext())
{
int num = i.next();
System.out.println(“num = “ + num);
}
// prints num = 1, num = 2, num = 3, num = 4
```

References

1) Java Collections
http://www.tutorialspoint.com/java/java_collections.htm

2)Design Patterns used in Java Collections
http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns-in-javas-core-libraries

3) Dynamic Memory Management done in Java
http://stackoverflow.com/questions/23245386/how-does-memory-allocation-of-an-arraylist-work

THANK YOU