IntelliGrape | TO THE NEW

Agenda

- ***Collections***
- ***Lists***
- Sets
- Ranges
- Maps

Prerequisites

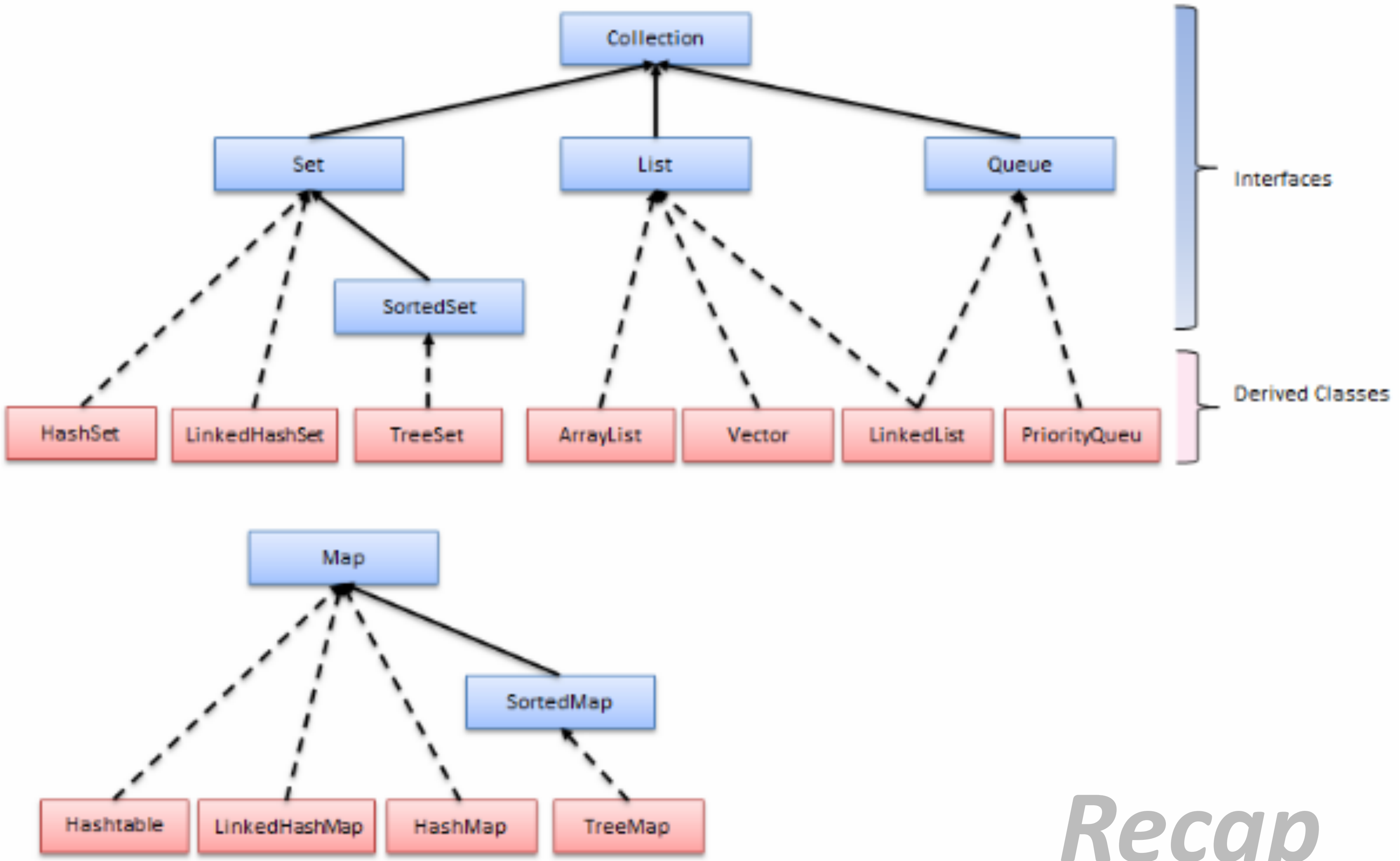
- Knowledge of Java collections.
- Basic working of groovy.
- Working knowledge of Closure.

Recap

Groovy Collections

- Data structure that helps in dealing with a number of objects.
- A wide variety of methods available for easy manipulation.

Recap


Collection
Set
List
Queue
SortedSet
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueu
Interfaces
Derived Classes
Map
SortedMap
Hashtable
LinkedHashMap
HashMap
TreeMap

Recap

Lists

- A list cares about the index
- Elements are assigned indices on the basis of how they are added
- Has methods related to the index
- Index starts from 0

Creating list

```
 //create empty list with ech element of type 'def'
List list = [ ]
//create empty list with elements of type 'type'
List<type> list = [ ]
List<type> list = new ArrayList()
```

Recap

Sets

- A Set cares about uniqueness - it doesn't allow duplicates
- It can be considered as a list with restrictions, and is often constructed from a list.

```
Set set = [1,3,3,4] as Set     // [1,3,4]
```

- No Ordering; element positions do not matter
- Most methods available to lists, besides those that don't make sense for unordered items, are available to sets

Eg. - `getAt`, `putAt`, `reverse`

Ranges

- Ranges allow you to create a list of sequential values.
- These can be used as Lists since Range extends java.util.List.
- Used for looping, switch, lists etc
- Ranges defined with the “..” notation are inclusive (that is the list contains the from and to value).
- Ranges defined with the “..<” notation are exclusive, they include the first value but not the last value.

Ranges

```
Range range = 1..10
range = -10..<10
range =  'a'..'z'
Range='#'..'~'
```

Methods

`range.from` – Get the lower limit of range

`range.to` – Get upper limit of range

`range.contains(value)` – Does range contain value?

Maps

- A Map cares about unique identifiers.
- Each key can map to at most one value. Keys and values can be of any type, and mixed together.

Initializing a Map :

```
Map map = [:]
Map map = new LinkedHashMap()
Map<KeyType, ValueType> map = [:]
Map<KeyType, ValueType> map = new LinkedHashMap()
```

```
Map m = [1:'a', 2:'b', (true):'p', (false):'q',
         null:'z']
```

Maps

Adding an element:

```
map.put(key, value)
map.putAll(Map)
map.key = value
map[key] = value
```

Fetching elements:

```
map[key] / map.get(key) / map.key
```

Maps

Removing elements:

`map.remove(key)` : Remove key value pair

Adding Two Maps:

```
Map map3 = map1 + map2
```

Maps

Operations on keys:

```
map.containsKey(key)
map.keySet()
```

Operations on values:

```
map.containsValue(value)
map.values()
```

Maps

`map.find { }` - Find first occurrence of element being searched

`map.findAll { }` - Return map of all occurrences of element being searched

`map.each { }` - Perform action with all elements

```
map.eachWithIndex {entry,index->
     println entry.key + ". " + entry.value
}
```

More Map Methods...

`isEmpty()` - Is map empty?

`toMapString()` - Return map as a string

Some more List methods

`groupBy{condition}` - Group a list into a map using some criteria.

Eg.

```
List l = (1..100)
println l.groupBy { it %2 }
```

References

http://groovy.codehaus.org/Collections

http://groovy.codehaus.org/groovy-jdk/java/util/List.html

http://groovy.codehaus.org/api/groovy/lang/Range.html

http://groovy.codehaus.org/JN1035-Maps

http://groovy.codehaus.org/JN1015-Collections (sets)