Collections

Managing Common Data by Using Collections

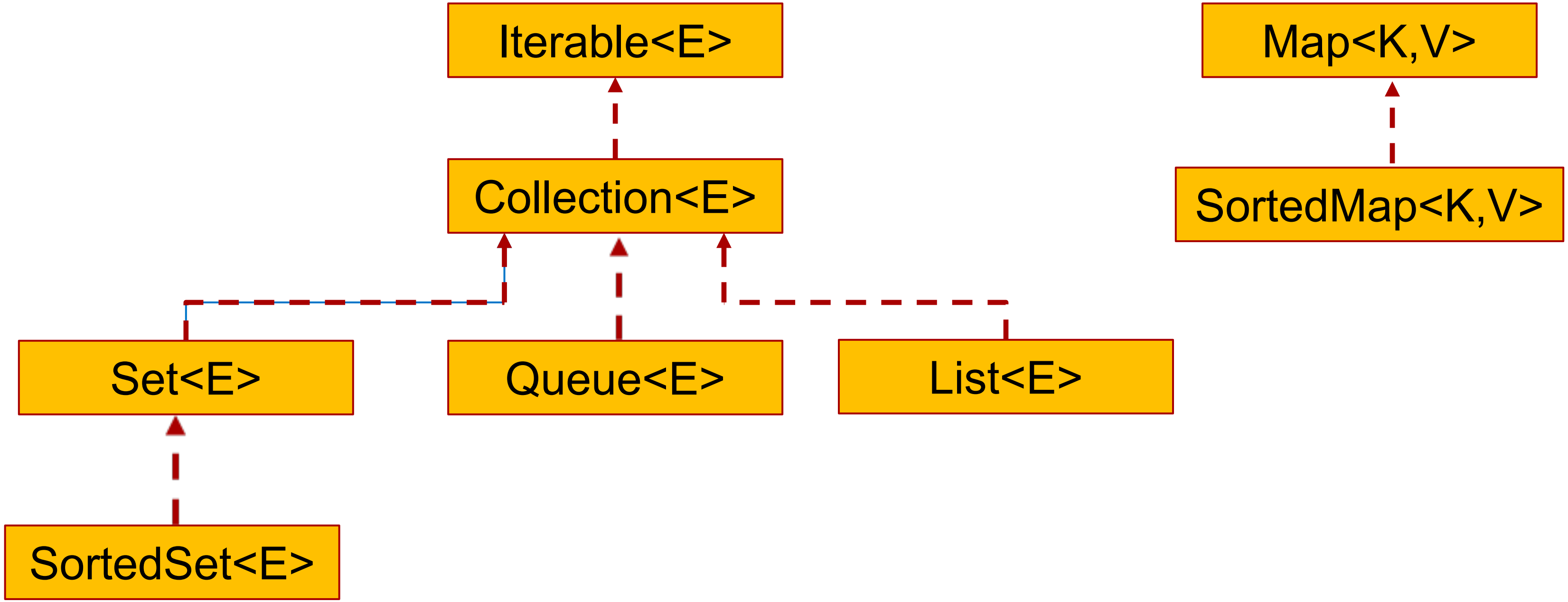
Overview

- Collections
- Collection-related interfaces
- Interface description
- Collections
- Main Underlying Structures

Collections

- Dynamic set of items of the same type.
- Choice based on
  - Performance (Add, Retrieve, Insert, Resize, Search, Sort)
  - Retrieval order (LIFO, FIFO, Random)
  - Retrieval by key or index
  - Sorted
  - Specialized

Collection-Related Interfaces



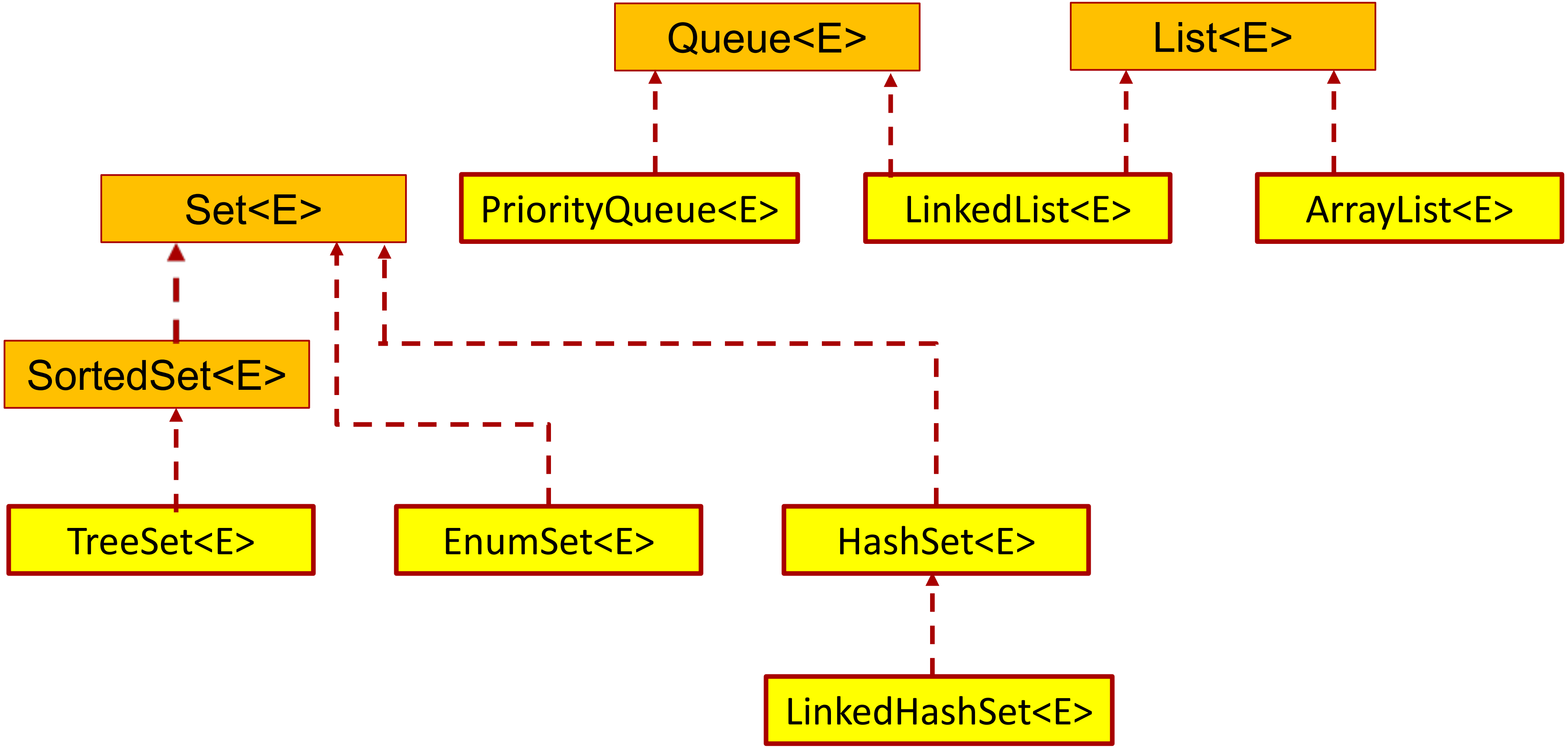
Interface description

Interface	Description
Iterable<E>	Provides an Iterator and so can be used by the enhanced for statement
Collection<E>	Root interface for collections. Provides methods as add, remove, size, toArray.
Set<E>	Collection, no duplicate elements
SortedSet<E>	Set whose elements are sorted
Queue<E>	Collection with FIFO structure

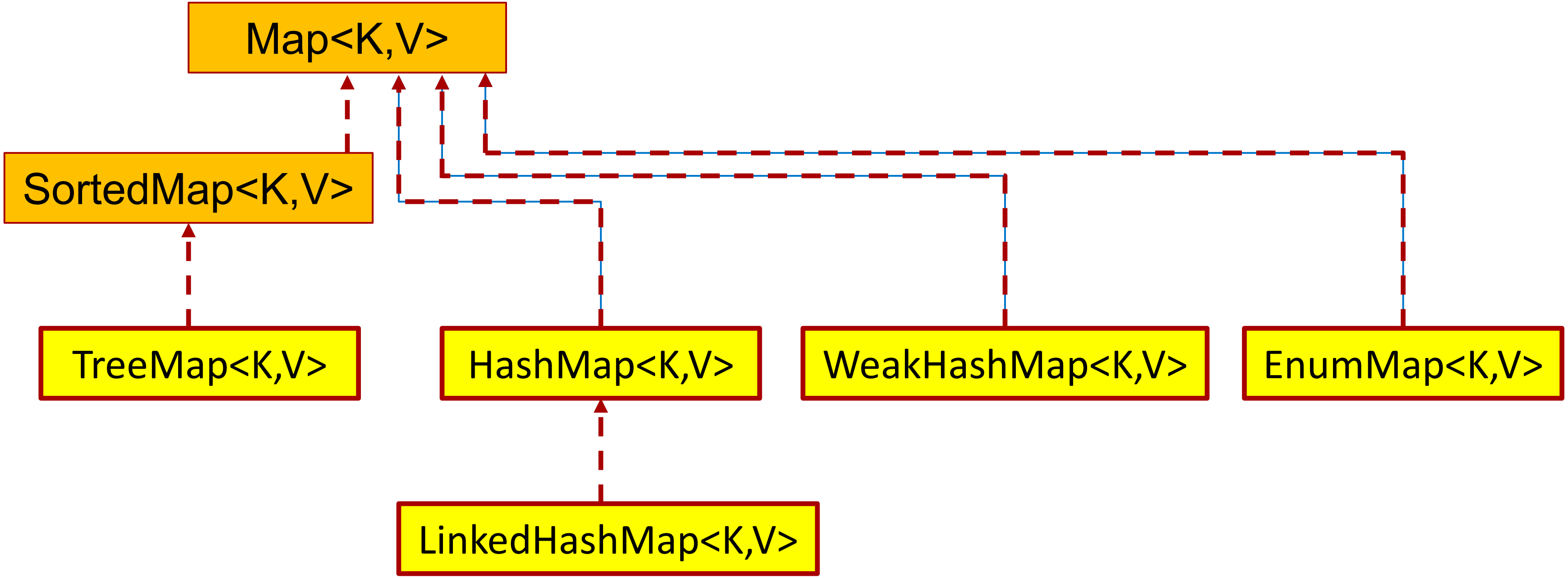
Interface description

Interface	Description
Map<K,V>	Mapping of Keys to a Value
SortedMap<K,V>	Map with sorted Keys

Collections



Collections



Main Underlying Structures

- Array
- HashTable
- LinkedList
- BinaryTree

Array

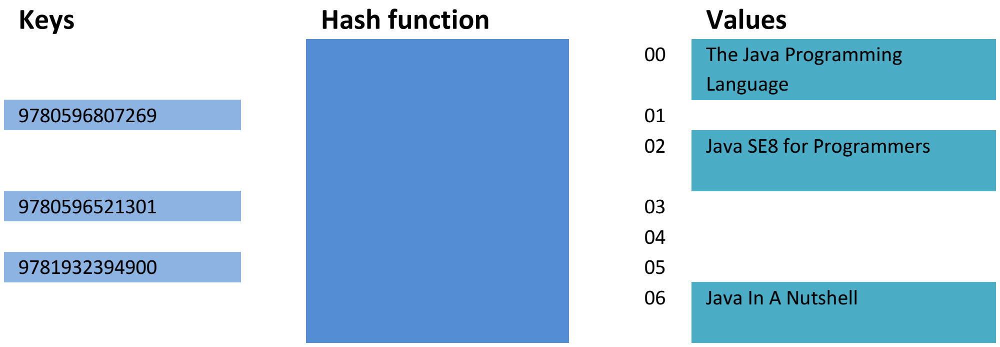
- Uses indices.
- Indexed based access is O(1)
- Append is O(1),Insert is O(n), delete is O(n).
- Searches are of O(n), when sorted O(log n)
- Sorting is of O(n.log n).
- Fixed size. Size adjustments come at high costs (reallocation and copying).
- Index based insertion is complex.
- Continuous block allocation in memory.
- Size must be known in advance.

00	The Java Programming Language
01	Head First Java, 2nd Edition
02	Windows PowerShell in Action
03	Java Programming
04	Java SE8 for Programmers
05	JavaScript and HTML5 Now
06	Java In A Nutshell

Array implementation

```java
ArrayList<String> cities=new ArrayList<String>
(Arrays.asList("Veenendaal", "Utrecht", "Amersfoort" ));
cities.add("Ede");
for (int i = 0; i < cities.size(); i++) {
  System.out.println(cities.get(i));
}
if(cities.contains("Utrecht")){
  System.out.println("Found Utrecht");
}
```

HashTable



Keys

9780596807269

9780596521301

9781932394900

Hash function

Values

00	The Java Programming Language
01	
02	Java SE8 for Programmers
03	
04	
05	
06	Java In A Nutshell

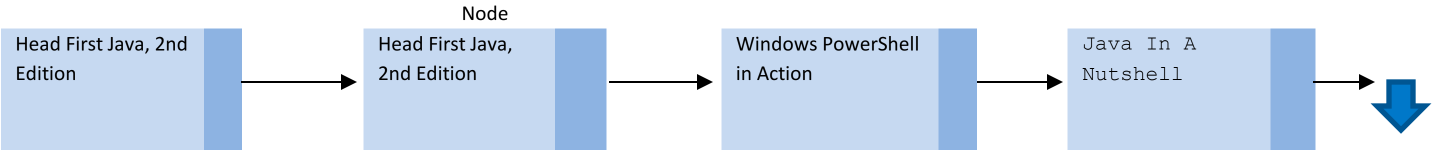
Maps Keys to their associated Values. The Hash function transforms the Key into the Index

- No (external) index.
- Arbitrary insertions and deletions.
- Searches are fast O(1), so are inserts and deletes.
- Resizing is a costly operation.

HashTable implementation

```java
Map<String, String> books = new HashMap<String, String>();
books.put("9780596807269", "The Java Programming Language");
books.put("9780596521301", "Windows PowerShell in Action");
books.put("9781932394900", "Java In A Nutshell");
Set<String> keys = books.keySet();
for (String key : keys) {
  System.out.println(books.get(key));
}
if(books.containsKey("9780123743190")==false){
  books.put("9780123743190", "DW2.0");
}
String searchISBN = "9780596807269";
System.out.printf("Book with ISBN:%s has title:%s”
                    ,searchISBN,books.get(searchISBN));
```

LinkedList

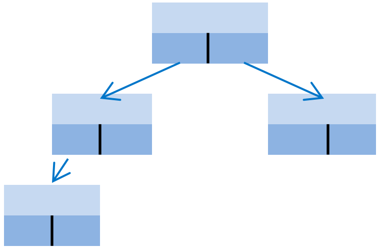

Node
Head First Java, 2nd Edition
Head First Java, 2nd Edition
Windows PowerShell in Action
Java In A Nutshell

- No Indices.
- Searches are (sequential) slow O(n).
- Insert is O(1), delete is of O(1) when based on position.
- Resizable at minor costs.
- Can use memory fragments.
- Size not known in advance.
- Relatively complex in comparison to Arrays.

LinkedList implementation

```java
LinkedList<String> cities = new LinkedList<String>
              (Arrays.asList("Veenendaal", "Utrecht"));
cities.addLast("Ede");
int nodeIndex = cities.indexOf("Utrecht");
cities.add(nodeIndex, "Amersfoort");//add before Utrecht
for (String city : cities) {
  System.out.println(city);//Veenendaal Amersfoort Utrecht Ede
}
cities.remove("Utrecht");
cities.removeFirst();
for (String city : cities) {
  System.out.println(city);// Amersfoort Ede
}
```

Binary Tree



- No indices
- Complex structure
- Balancing is complex.
- Each node has at most two child nodes.
- Search is on average O(log n).
- Insert is of O(log n), Delete is of O(log n)
- Insertions and deletions are simple.
- Arranging data in a hierarchy.

BinaryTree implementation

```
TreeMap<String, String> orderedDictionary =
                    new TreeMap<String, String>();
orderedDictionary.put("9780596807269", "The Java Programming Language");
orderedDictionary.put("9780596521301", "Windows PowerShell in Action");
orderedDictionary.put("9781932394900", "Java In A Nutshell");
Set<String> keyCollection = orderedDictionary.keySet();
for (String item : keyCollection) {
  System.out.printf("Key: %s, Value %s%n", item,
  orderedDictionary.get(item));
}
```

Lab: Collections

- Exercise 1 : Collections