

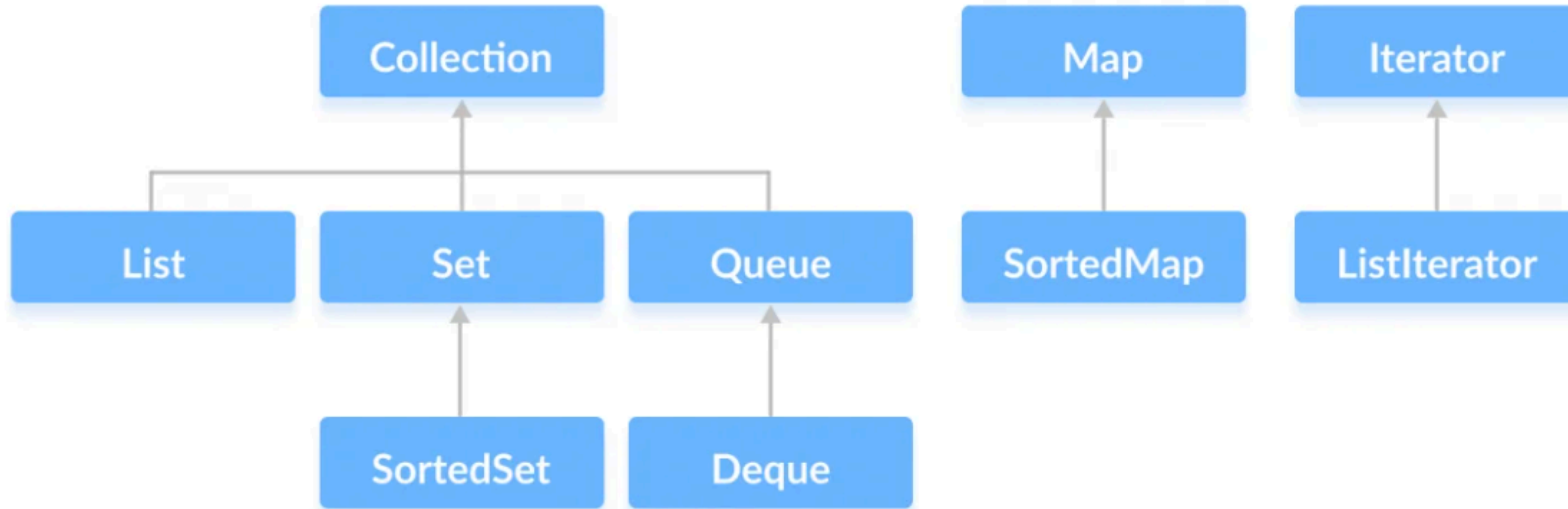
Java Course

Collections

Java Collections Framework

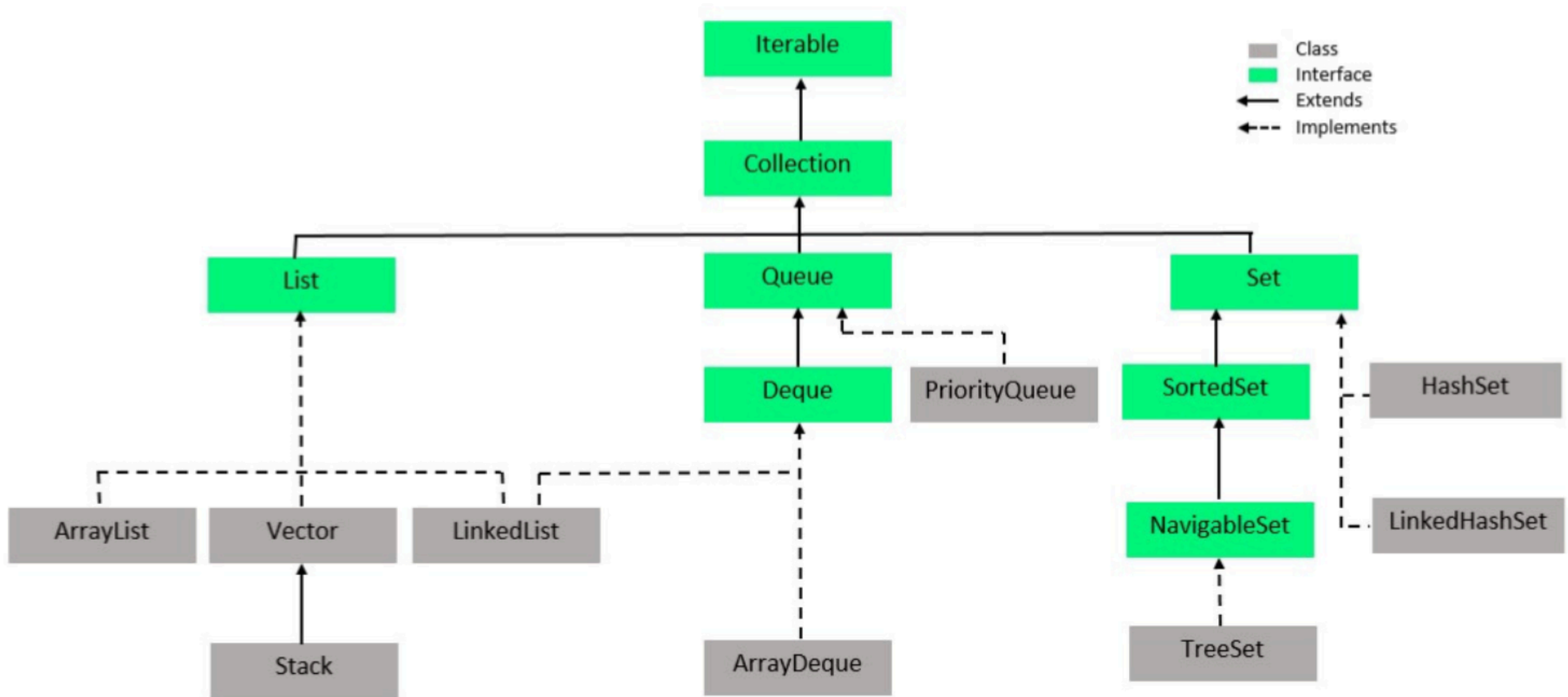
- Java Collections Framework provides a set of interfaces and classes to implement various data structures and algorithms
- These interfaces include several methods to perform different operations on collections
- Advantage of using Collections Framework:
 - we do not have to implement these data structures and algorithms manually
 - efficiency - highly optimized

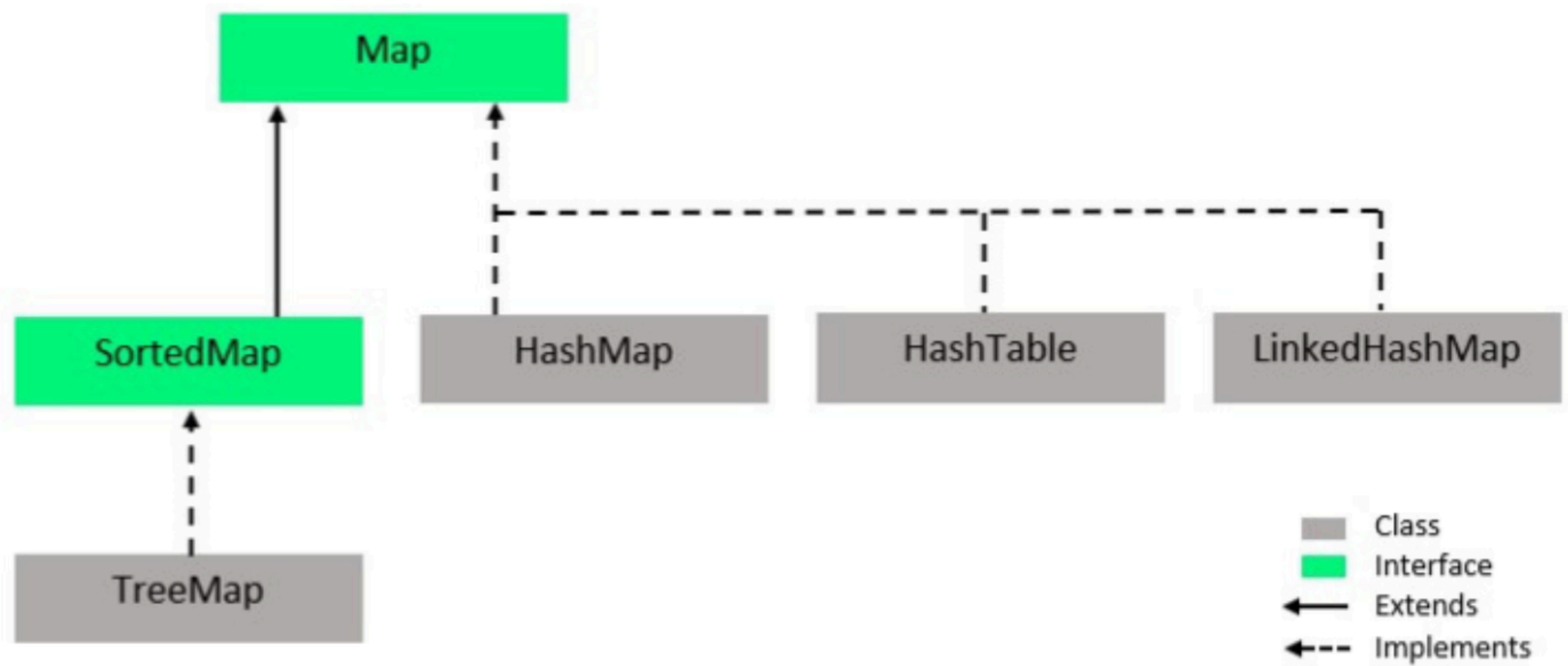
Java Collections Framework



Java Collection interface

- The `Collection` interface is the root interface of the Collections framework hierarchy
- Java does not provide direct implementation of the `Collection` interface but provides implementations of its subinterfaces like `List`, `Set`, `Queue`
- Java Collections Framework includes other interfaces as well: `Map` and `Iterator`





List

- List is an ordered collection of objects, where same object may appear more than once
- Methods:
 - `add()` - adds an element to a list
 - `get()` - accessing elements
 - `remove()` - removes an element from the list
 - `size()` - returns length of a list
 - `contains()` - check if the list contains a certain element
- Implementations:
 - `ArrayList`, `LinkedList`, `Vector`,..

```
7
8 public static void main(String[] args) {
9
10     // initialize empty list of Strings
11     List<String> listOfNames = new ArrayList<String>();
12
13     // print length of the list
14     System.out.println("Length of list: " + listOfNames.size());
15
16     // add elements to a list
17     listOfNames.add("Pera");
18     listOfNames.add("Marko");
19     listOfNames.add("Ana");
20
21     // print length of the list
22     System.out.println("Length of list: " + listOfNames.size());
23
24     // get n-th element of a list (indexes start from 0)
25     System.out.println("2nd element: " + listOfNames.get(1));
26
27     // check if list contains an element
28     System.out.println("Does list contains Ana? " + listOfNames.contains("Ana"));
29
30     // remove element - removing Ana from the list
31     listOfNames.remove(2);
32
33     System.out.println("Does list contains Ana? " + listOfNames.contains("Ana"));
34
35     // iterating over list elements
36     for (String name : listOfNames) {
37         System.out.println("Name: " + name);
38     }
39
40 }
```


Queue



- A queue is an ordered collection of elements with a functionality of a queue (FIFO - **F**irst **I**n **F**irst **O**ut)
- Methods:
 - `add()` - insert element into the queue (add to the end of a queue)
 - `remove()` - remove element from the queue (remove from the front of the queue)
 - `element()` - returns head element of the queue
- Implementations:
 - `PriorityQueue`, `ArrayDeque`,...

```
5
6 public class QueueExample {
7
8     public static void main(String[] args) {
9
10         // initialize a Queue - PriorityQueue
11         Queue<String> queueElements = new PriorityQueue<String>();
12
13         // add elements to a Queue (they are added to the end of a queue)
14         queueElements.add("5");
15         queueElements.add("4");
16         queueElements.add("3");
17
18
19         // however, the PriorityQueue is different in a way
20         // when removing elements from a PriorityQueue - elements are sorted
21         // right before removing
22         System.out.println("Removing: " + queueElements.remove());
23         System.out.println("Removing: " + queueElements.remove());
24
25         queueElements.add("8");
26
27         for (String element : queueElements) {
28             System.out.println("Element: " + element);
29         }
30
31     }
32 }
```

Set

- A set is unordered collection of objects, and it cannot contains duplicates
- It acts like a mathematical set
- Methods:
 - `add()` - adds an element to a set
 - `remove()` - removes an element from a set
 - `size()` - returns number of set elements
 - `contains()` - check if the set contains a certain element
 - `clear()` - removes all the elements from the set
- Implementations:
 - `HashSet`, `LinkedHashSet`, `TreeSet`

```
7
8 public static void main(String[] args) {
9
10     // Creating a set using the HashSet class
11     Set<Integer> set1 = new HashSet<Integer>();
12
13     // Add elements to the set1
14     set1.add(2);
15     set1.add(3);
16     System.out.println("Set1: " + set1);
17
18     // Creating another set using the HashSet class
19     Set<Integer> set2 = new HashSet<>();
20
21     // Add elements
22     set2.add(1);
23     set2.add(2);
24     System.out.println("Set2: " + set2);
25
26     // Union of two sets - no duplicates!
27     set2.addAll(set1);
28     System.out.println("Union is: " + set2);
29
30     // trying to add duplicated value to set2:
31     set2.add(2);
32     System.out.println("Set2: " + set2);
33
34     System.out.println("Number of Set2 elements: " + set2.size());
35
36     for (Integer element : set2) {
37         System.out.println("Element: " + element);
38     }
39
40 }
```

Map

- Map interface provides functionality of the map data structure
- Elements of Map are stored in a key/value pairs
- Keys are unique - no duplicate keys
- Each key is associated with a single value


```
6 public class MapExample {
7
8     public static void main(String[] args) {
9
10         // create a hashmap
11         HashMap<String, Integer> numbers = new HashMap<>();
12
13         System.out.println("Initial HashMap: " + numbers);
14
15         // put() method to add elements
16         numbers.put("One", 1);
17         numbers.put("Two", 2);
18         numbers.put("Three", 3);
19         System.out.println("HashMap after put(): " + numbers);
20
21
22         // Example with languages and their indexes
23
24         HashMap<Integer, String> languages = new HashMap<>();
25         languages.put(1, "Java");
26         languages.put(2, "Python");
27         languages.put(3, "JavaScript");
28         System.out.println("HashMap: " + languages);
29
30         // get() method to get value
31         String value = languages.get(1);
32         System.out.println("Value at index 1: " + value);
33
34
35         // return set view of keys
36         // using keySet()
37         System.out.println("Keys: " + languages.keySet());
38
39         // return set view of values
```