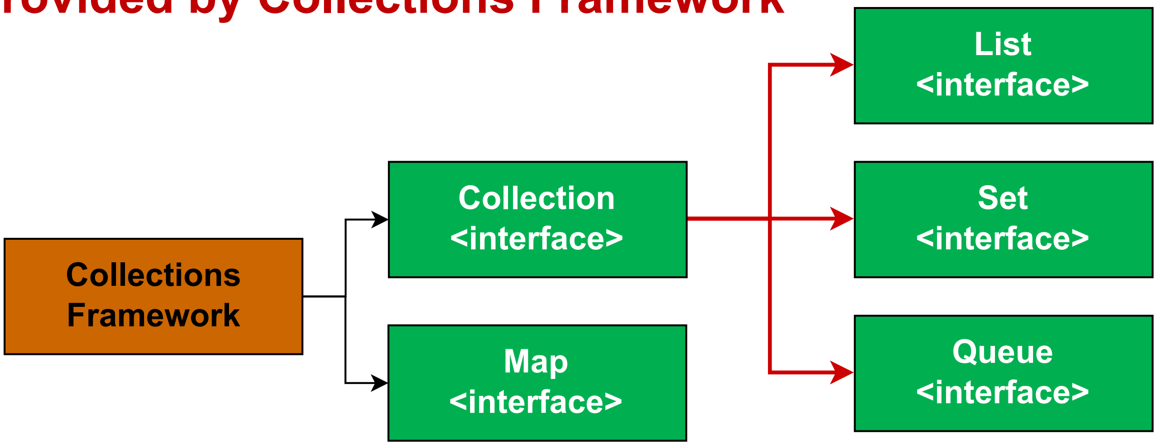
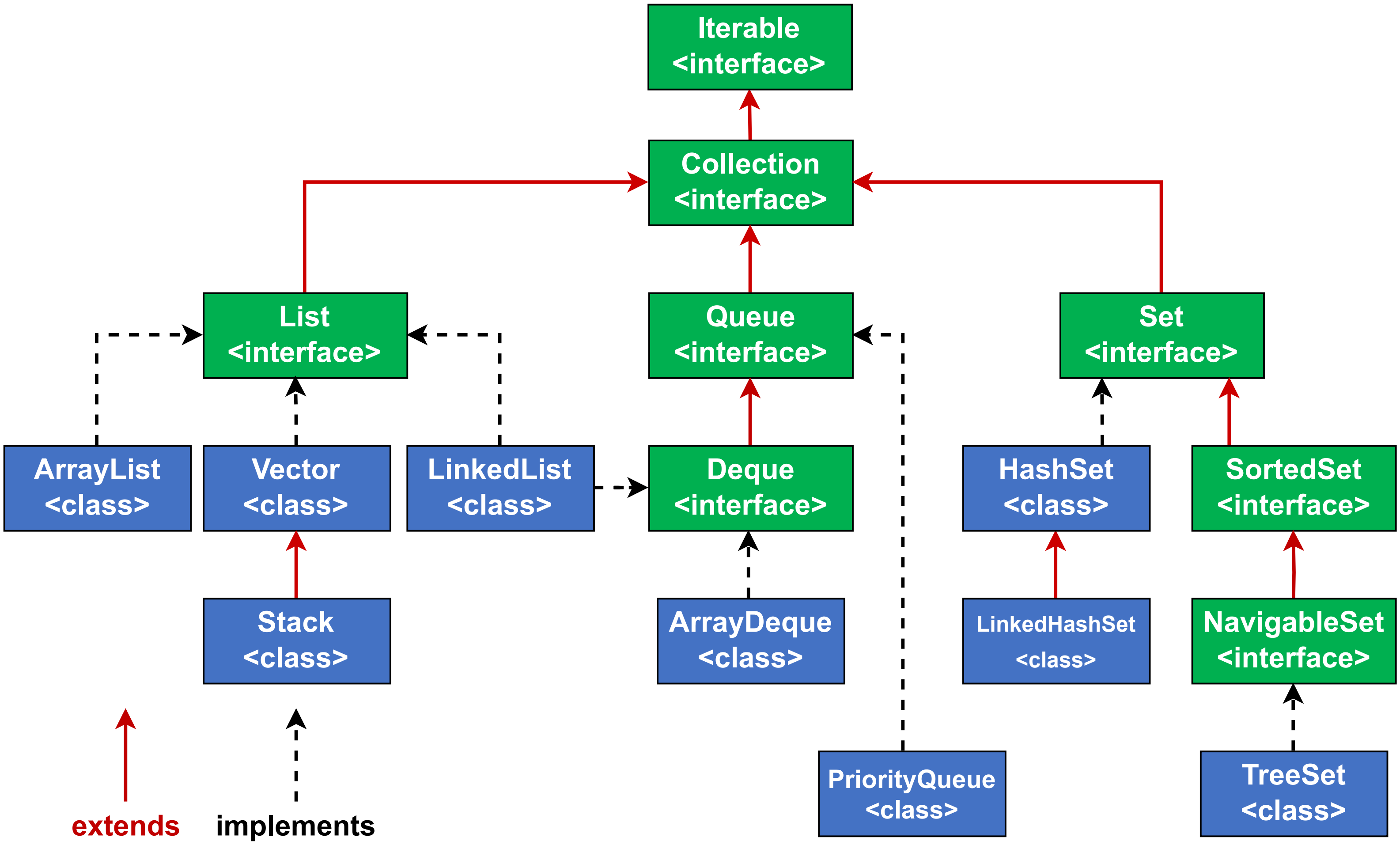
Collections Framework

Collections is a framework provided by java. This framework provides many interfaces and their implemented classes in order to store group of objects (elements) in a single entity.

Interfaces Provided by Collections Framework



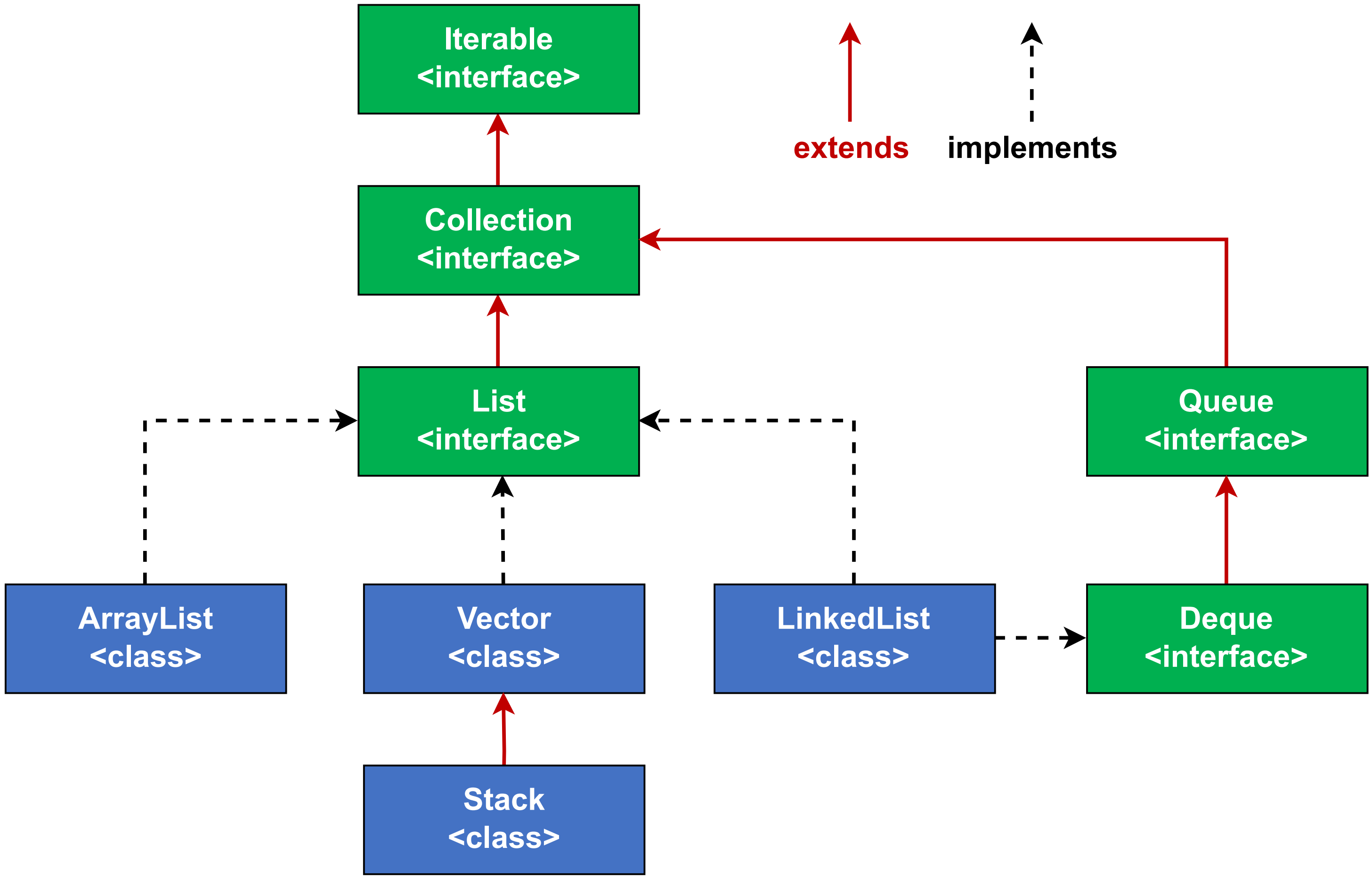
Hierarchy of Collection Interface



List Interface

The List interface is an ordered collection that allows us to store/insert and access/retrieve elements by their position in the list.

Hierarchy of List Interface



List interface is implemented by 4 classes in Java. those are:
1. ArrayList
2. Vector
3. Stack
4. LinkedList

Out of all these 4 classes LinkedList is the only class which also implements the Deque interface apart from List interface.

Hence ArrayList, Vector and Stack are called as pure implementations of List interface.

Initialization/Creation of a List

```
List<Integer> accountNumbers = new LinkedList<>();
List<String> names = new ArrayList<>();
List<String> names = new ArrayList<>(50);
List<String> petNames = new Vector<>();
List<String> carBrands = new Stack<>();
```

Inserting elements

```
accountNumbers.add(123456789);
names.add("Yadagiri Reddy");
petNames.add("Bittu");
carBrands.add("Tata");
```

Deleting elements

```
names.remove(0);
petNames.remove("Bittu");
```

Retrieving elements

```
accountNumbers.get(0);
```

Verifying elements

```
names.contains("HYR");
```

Updating elements

```
names.set(5, "HYR");
```

ArrayList vs Vector vs Stack vs LinkedList

	DC	IC	AD	AN	IO	SO	RA	SYNC
ArrayList	0	10	Yes	Yes	Yes	No	Yes	No
Vector	10	10	Yes	Yes	Yes	No	Yes	Yes
Stack	10	10	Yes	Yes	Yes	No	Yes	Yes
LinkedList	0	0	Yes	Yes	Yes	No	Yes	No

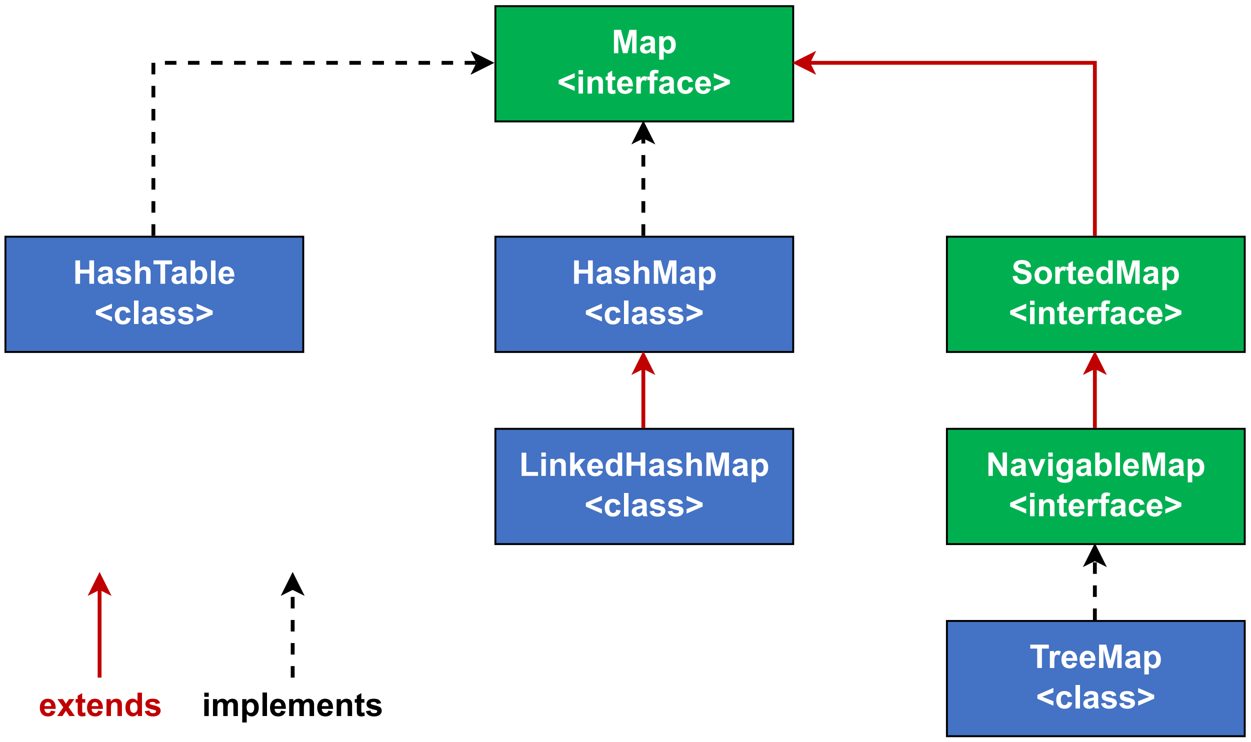
DC - Default Capacity
IC - Initial Capacity
AD - Allow Duplicates
AN - Allow Null Values

IO - Insertion Order
SO - Sorted Order
RA - Random Access
SYNC - Synchronization

Map Interface

The Map interface is an un-ordered data structure that allows us to store the data in the form of key and value pairs

Hierarchy of Map Interface



Map interface is implemented by 4 classes in Java. those are:

1. HashTable
2. HashMap
3. LinkedHashMap
4. TreeMap

Initialization/Creation of a Map

```
Map<String, Integer> population = new Hashtable<>();
Map<Integer, String> postalCodes = new HashMap<>();
Map<Integer, Employee> employees = new LinkedHashMap<>();
Map<Integer, Student> students = new TreeMap<>();
```

Addition of elements into the map

```
population.put("India", 1400000000);
postalCodes.put(500050, "Hyderabad");
employees.put(1001, new Employee(1001, "Yada", "Hyderabad"));
students.put(1001, new Student(1, "Giri", "8th Class"));
```

Retrieval of keys from the map

```
Set<String> keys = population.keySet();
```

Retrieval of values from the map

```
Collection<String> values = postalCodes.values();
```

Retrieval of value from the map based on a key

```
employees.get(1001);
```

Deletion of elements from the map

```
students.remove(1);
```

Verification of keys in the map

```
employees.containsKey(1005);
```

Verification of values in the map

```
postalCodes.containsValue("Chennai");
```

Updation of values in the map

```
postalCodes.replace(123456, "Mumbai");
```

HashTable vs HashMap vs LinkedHashMap vs TreeMap

	DC	IC	ADK	ADV	ANK	ANV	IO	SO	RA	SYNC
HashTable	11	11	No	Yes	No	No	No	No	Yes	Yes
HashMap	0	16	No	Yes	Yes	Yes	No	No	Yes	No
LinkedHashMap	0	16	No	Yes	Yes	Yes	Yes	No	Yes	No
TreeMap	0	0	No	Yes	No	Yes	No	Yes	Yes	No

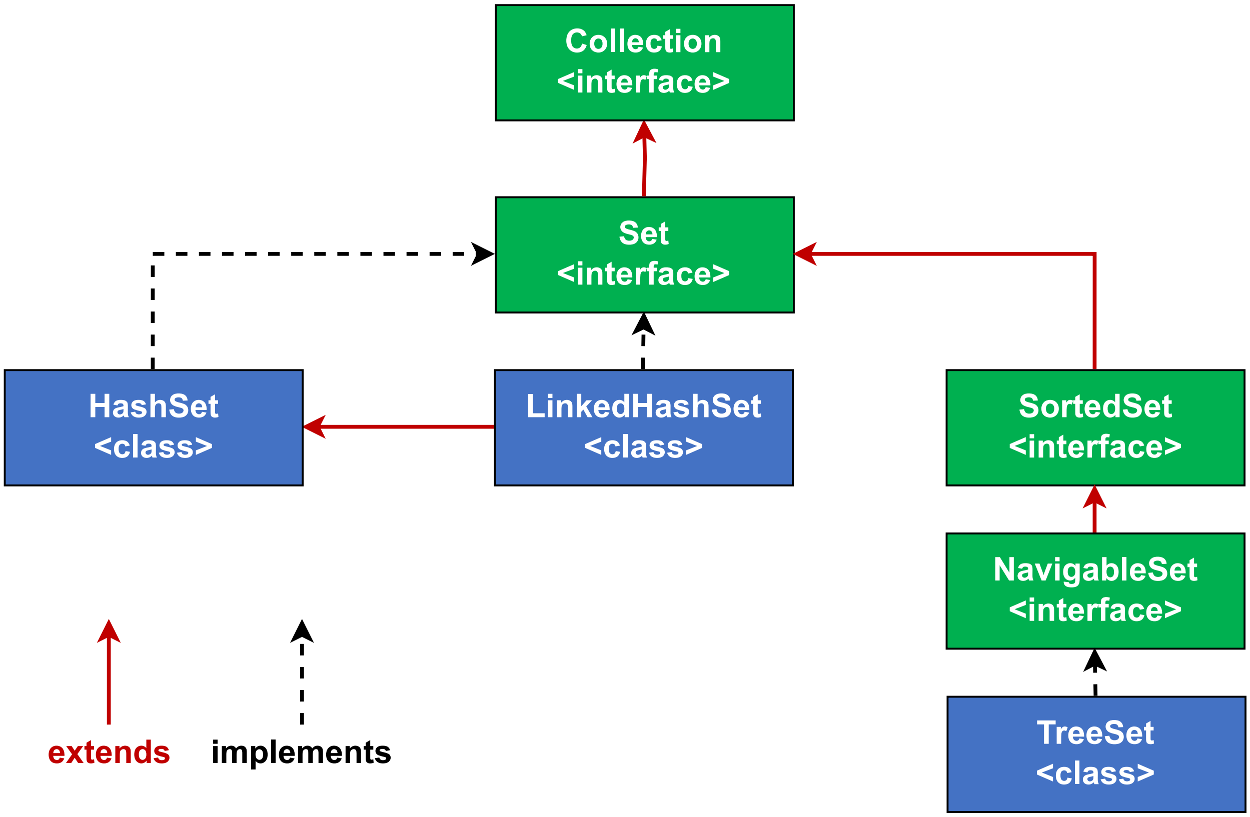
DC - Default Capacity
ADK - Allow Duplicate Keys
ADV - Allow Duplicate Values
ANK - Allow Null Keys
ANV - Allow Null Values

IC - Initial Capacity
IO - Insertion Order
SO - Sorted Order
RA - Random Access
SYNC - Synchronization

Set Interface

The Set interface is an un-ordered data structure that allows us to store the unique values into the collection object

Hierarchy of Set Interface



Set interface is implemented by 3 classes in Java. those are:

1. HashSet
2. LinkedHashSet
3. TreeSet

Initialization/Creation of a Set

```
Set<Integer> accountNumbers = new HashSet<>();
Set<String> names = new LinkedHashSet<>();
Set<String> petNames = new TreeSet<>();
```

Inserting elements

```
accountNumbers.add(123456789);
names.add("Yadagiri Reddy");
petNames.add("Bittu");
```

Deleting elements

```
names.remove(0);
petNames.remove("Bittu");
accountNumbers.clear();
```

Retrieving elements using Iterator

```
Iterator value = names.iterator();
while (value.hasNext()) {
   System.out.println(value.next());
}
```

Verifying elements

```
names.contains("HYR");
```

Retrieving elements using foreach loop

```
for (String petName : petNames) {
   System.out.println(petName);
}
```

HashSet vs LinkedHashSet vs TreeSet

	DC	IC	AD	AN	IO	SO	RA	SYNC
HashSet	0	16	No	Yes	No	No	No	No
LinkedHashSet	16	16	No	Yes	Yes	No	No	No
TreeSet	0	0	No	No	No	Yes	No	No

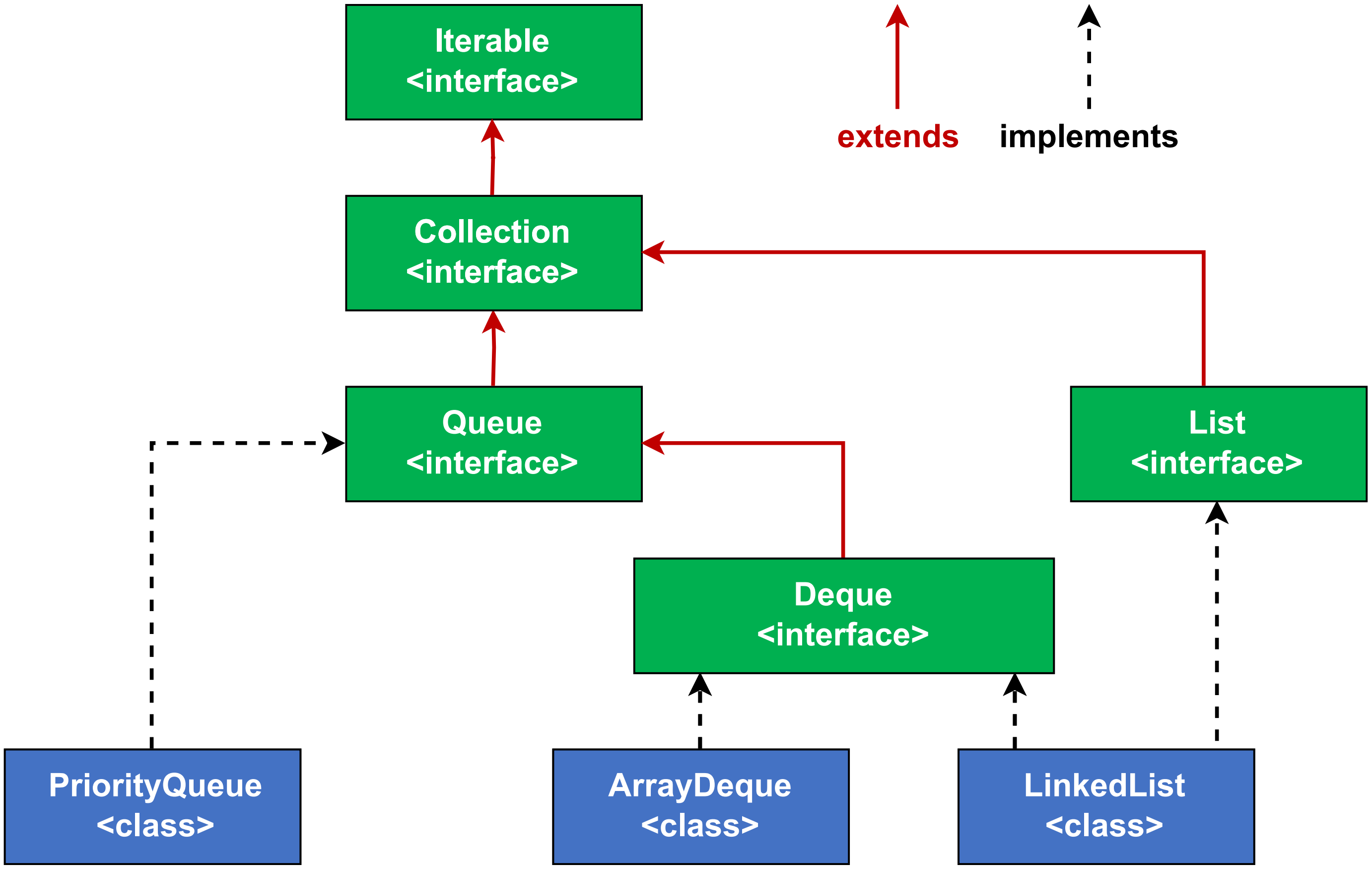
DC - Default Capacity
IC - Initial Capacity
AD - Allow Duplicates
AN - Allow Null Values

IO - Insertion Order
SO - Sorted Order
RA - Random Access
SYNC - Synchronization

Queue Interface

The Queue interface is an ordered data structure that allows us to store and retrieve the values based on FIFO (First In First Out) principle.

Hierarchy of Queue Interface



Queue interface is implemented by 3 classes in Java. those are:
1. PriorityQueue
2. ArrayDeque
3. LinkedList

Initialization/Creation of a Queue

```
Queue<Order> orders = new PriorityQueue<>();
Queue<Integer> tickets = new ArrayDeque<>();
Queue<String> tasks = new LinkedList<>();
```

Retrieving elements

```
tickets.element();
tasks.peek();
tickets.peekLast();
```

Inserting elements

```
tickets.add(74);
tasks.offer("Learn Collections");
orders.add(new Order("Dosa"));
tickets.addLast(95);
tasks.offerFirst("Read a book");
```

Deleting elements

```
orders.remove();
tasks.poll();
tickets.remove(85);
tasks.removeLast();
tickets.pollLast();
```

Retrieving elements using foreach loop

```
for (String taskName : tasks) {
   System.out.println(taskName);
}
```

Retrieving elements using Iterator

```
Iterator value = tickets.iterator();
while (value.hasNext()) {
   System.out.println(value.next());
}
```

PriorityQueue vs ArrayDeque vs LinkedList

	DC	IC	AD	AN	IO	SO	RA	SYNC
PriorityQueue	11	11	Yes	No	No	No	No	No
ArrayDeque	17	17	Yes	No	No	No	No	No
LinkedList	0	0	Yes	Yes	Yes	No	Yes	No

DC - Default Capacity
IC - Initial Capacity
AD - Allow Duplicates
AN - Allow Null Values

IO - Insertion Order
SO - Sorted Order
RA - Random Access
SYNC - Synchronization