JAC444 - Lecture 9

Java Collections

Segment 1- Basics

Collections

In this lesson you will be learning about:

- Java Collections Framework
- The Collection Interface
- Set, List, Map Interfaces
- General Purpose Implementations
- Algorithms
- Compatibility with Vector, Hashtable
- Streams

Collection Basics

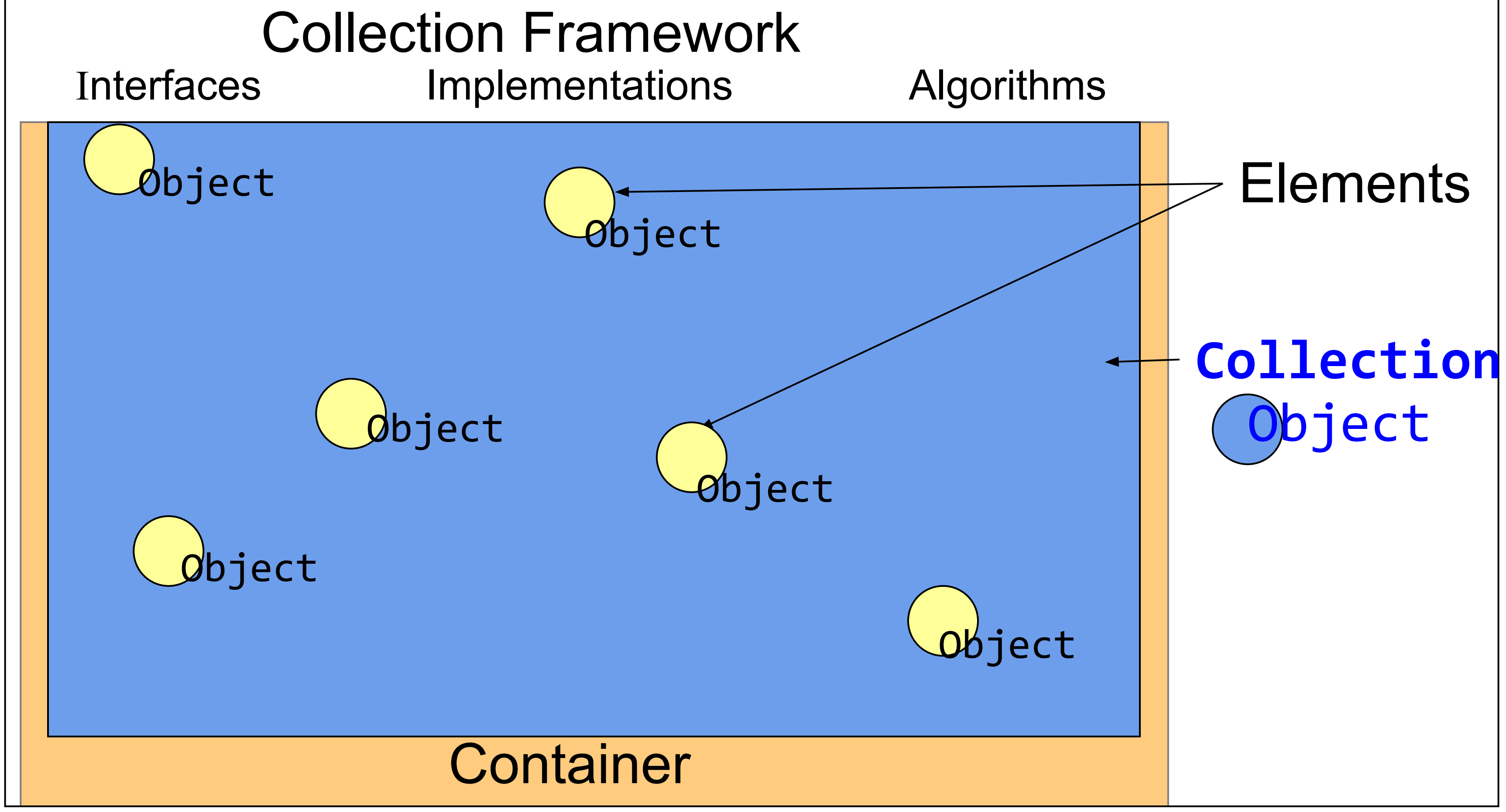
In this first segment you will be learning about:

- Java Collections Framework
- The Collections Interface
- The Collections Implementation

Reference:
http://docs.oracle.com/javase/tutorial/collections/index.html

The Collection

A *collection* represents a group of objects, known as its elements.



Using Collections

The collection classes and interfaces are defined in the package `java.util`

The interface `Collection<E>` from `java.util` is the root interface in the collection hierarchy

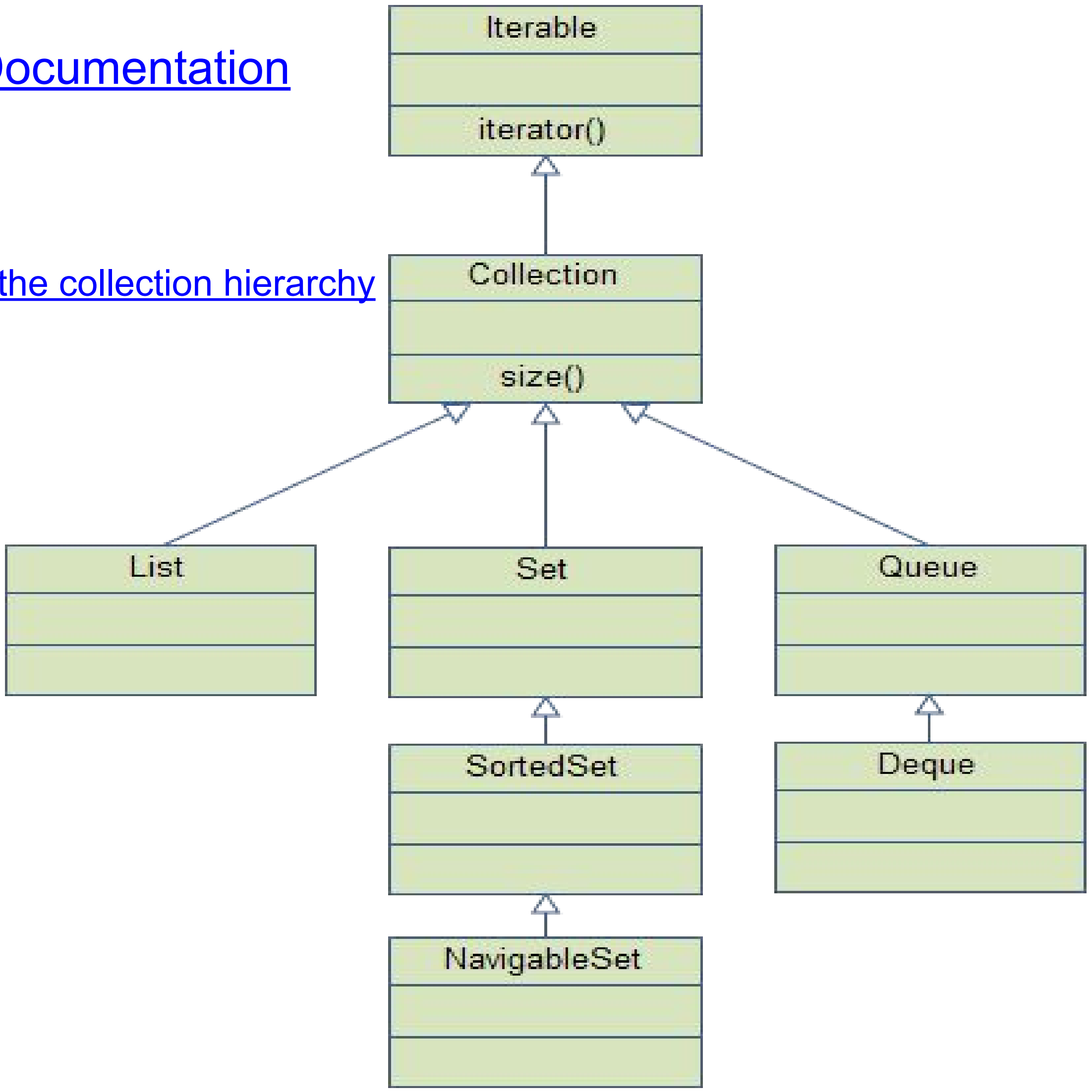
Important note:

There is also `java.util.Collections` but this is a class that consists exclusively of static methods that operate on or return collections

Collections Hierarchy

Iterable Documentation

The root interface in the collection hierarchy



The Collection<E> Interface

```
public interface Collection<E> {
            // Group 1
            int size();
            boolean isEmpty();
            boolean contains(Object element);
            boolean add(Object element);    // Optional
            boolean remove(Object element); // Optional
            Iterator iterator();

            // Group 2
            boolean containsAll(Collection c);
            boolean addAll(Collection c);    // Optional
            boolean removeAll(Collection c); // Optional
            boolean retainAll(Collection c); // Optional
            void clear();                    // Optional

            // Group 3
            Object[] toArray();
            Object[] toArray(Object a[]);
}
```

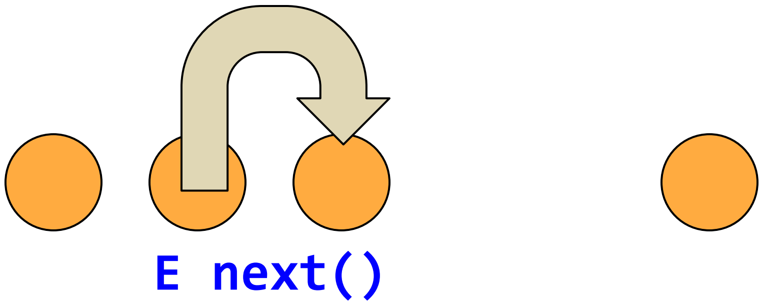
1 Basic Operations

2 Bulk Operations

3 Array Operations

Iterator<E> Interface

```
public interface Iterator<E> {
    boolean hasNext();
    E next();
    default void remove();
}
```



1. Returns the current element (initially the first element)
2. Steps to the next element and makes it the current element.

`Iterator` Interface

The **`Iterator`** interface allows to obtain the collection's elements

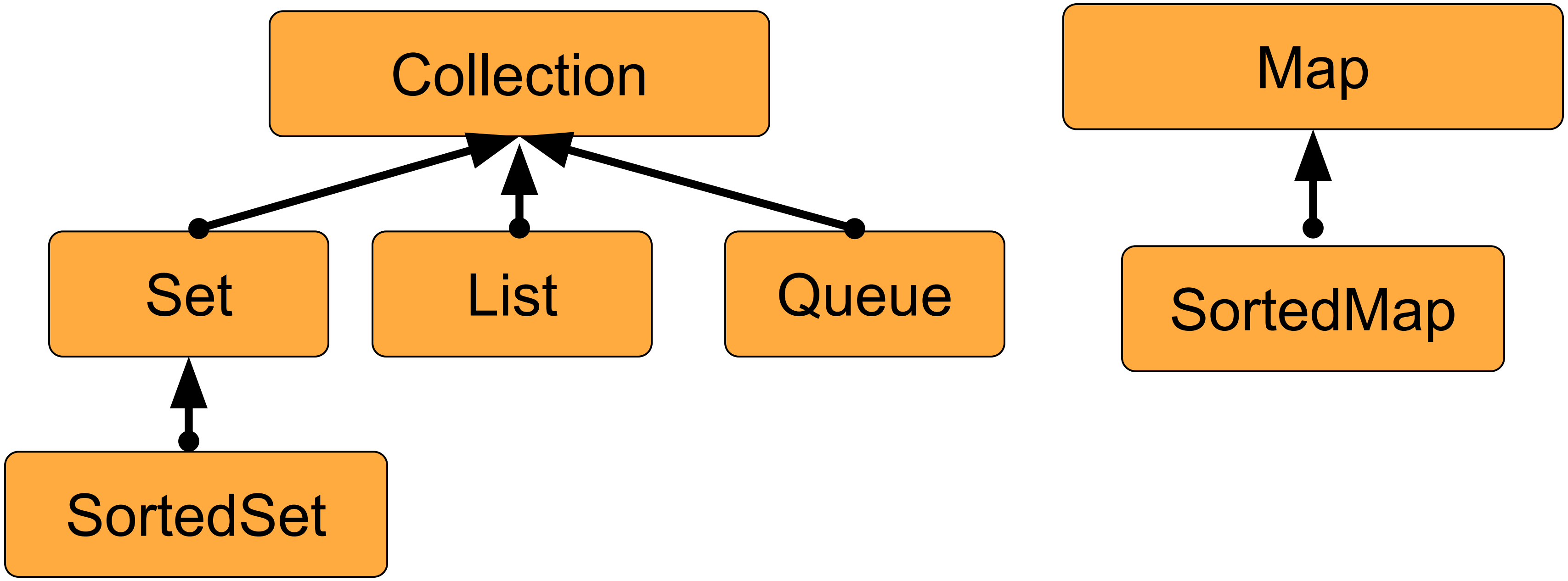
Example of filtering a collection using the **`Iterator`** interface:

```
public void filter(Collection c)   {

   for ( Iterator i  =  c.iterator();  i.hasNext();)

      if  ( !cond(i.next()) )
          i.remove();
}
```

Core Collection Interfaces


Collection
Set
List
Queue
SortedSet
Map
SortedMap