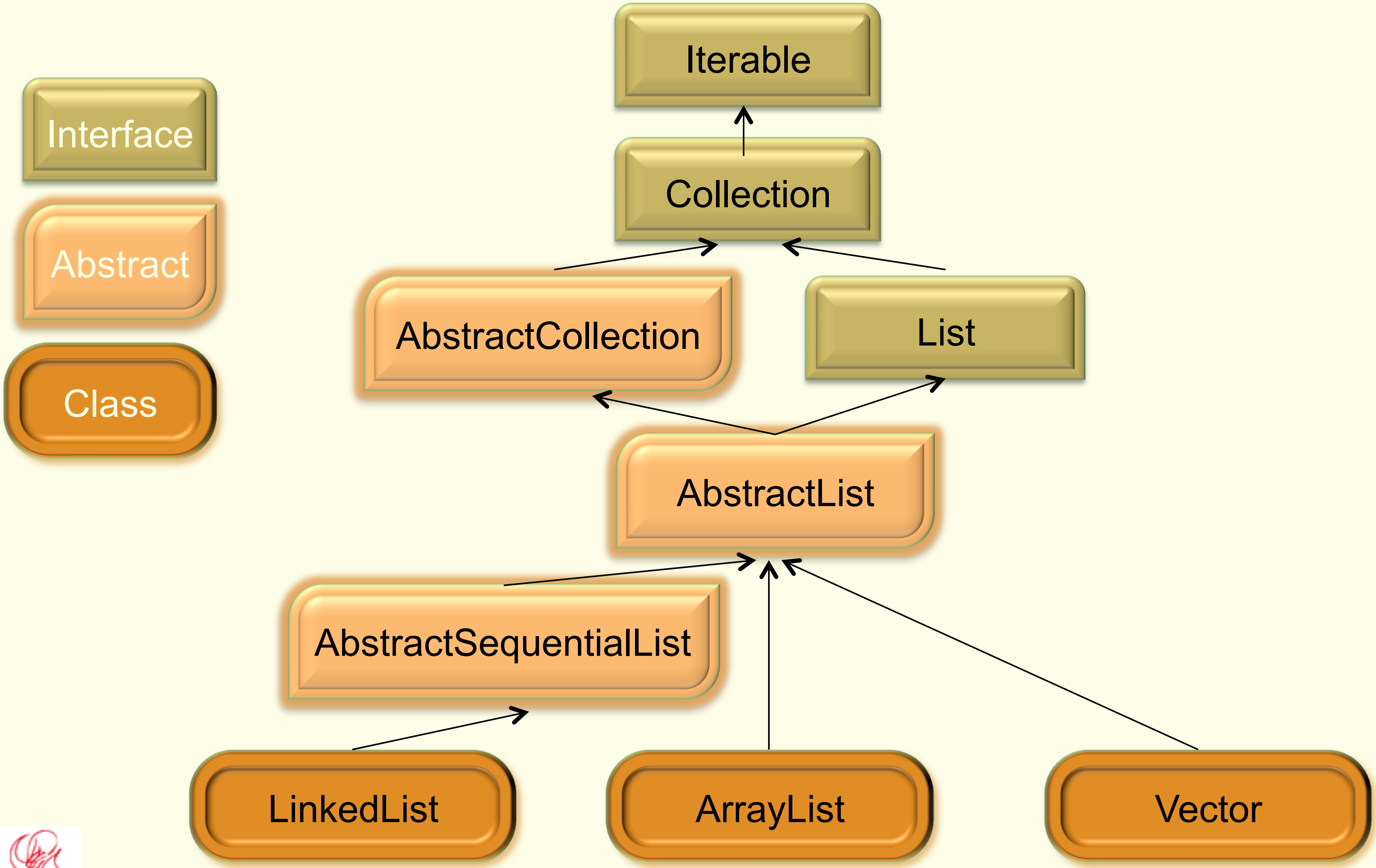
JAC444 - BTP400 Course Object-Oriented Software Development II - Java

Collections

Segment 3

The List Interface

Order Data Types Hierarchy



The List<E> Interface

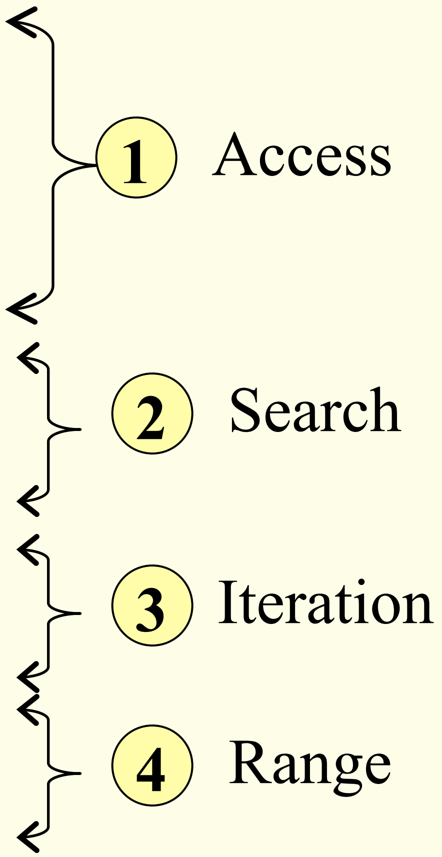
A **List** is an ordered **Collection** (called a sequence)

```
public interface List extends Collection {
      // Positional Access
      Object get(int index);
      Object set(int index, Object element);
      void add(int index, Object element);
      Object remove(int index);
      boolean addAll(int index, Collection c);

      // Search
      int indexOf(Object o);
      int lastIndexOf(Object o);

      // Iteration
      ListIterator listIterator();
      ListIterator listIterator(int index);

      // Range-view
      List subList(int from, int to);
  }
```



ListIterator<E>

```
public interface ListIterator extends Iterator {
        boolean hasNext();
        Object next();

        boolean hasPrevious();
        Object previous();

        int nextIndex();
        int previousIndex();

        void remove();           // Optional
        void set(Object o);      // Optional
        void add(Object o);      // Optional
   }
```

Standard idiom for iterating backwards through a list:

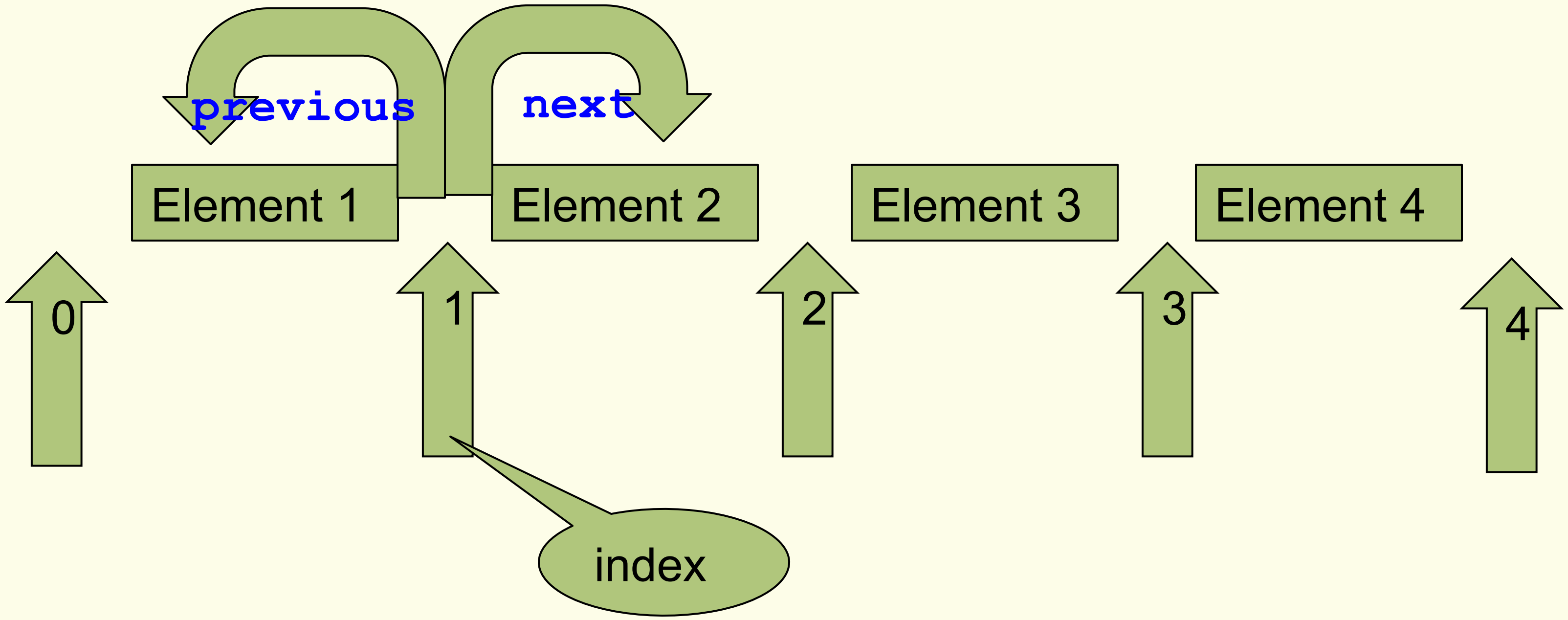
```
for ( ListIterator i  =  list.listIterator(list.size());i.hasPrevious();){
                 Object o =  i.previous();
```

Operations on List

- Positional access — manipulates elements based on their numerical position in the list.
- Search — a specified object in the list and returns its numerical position.
- Iteration — extends Iterator semantics to take advantage of the list's sequential nature.
- Range-view — The `sublist` method performs arbitrary range operations on the list.

Cursor Positions in a List

- The cursor is always between two elements of a list



In a list of length n, there are n+1 valid values for index, from 0 to n, inclusive.

LinkedList

- The `LinkedList<E>` class extends `AbstractSequentialList<E>` and implements the `List<E>` interface

- It has two constructors: the default and `LinkedList(Collection<? extends E> c)`
- Multithreaded

```
Collections.synchronizedList(new LinkedList(...));
```

- Important method

```
public <T> T[] toArray(T[] a)
```

Returns an array containing all of the elements in this list in proper sequence