Object-Oriented Programming (CS F213)

Module V: Collections in Java

CS F213 RL 12.1: Introduction to Java's Collection Framework

BITS Pilani

Dr. Pankaj Vyas
Department of Computer Science, BITS-Pilani, Pilani Campus

CS F213 RL 12.1 : Topics

- Java's Collection Framework

What Are Collections?

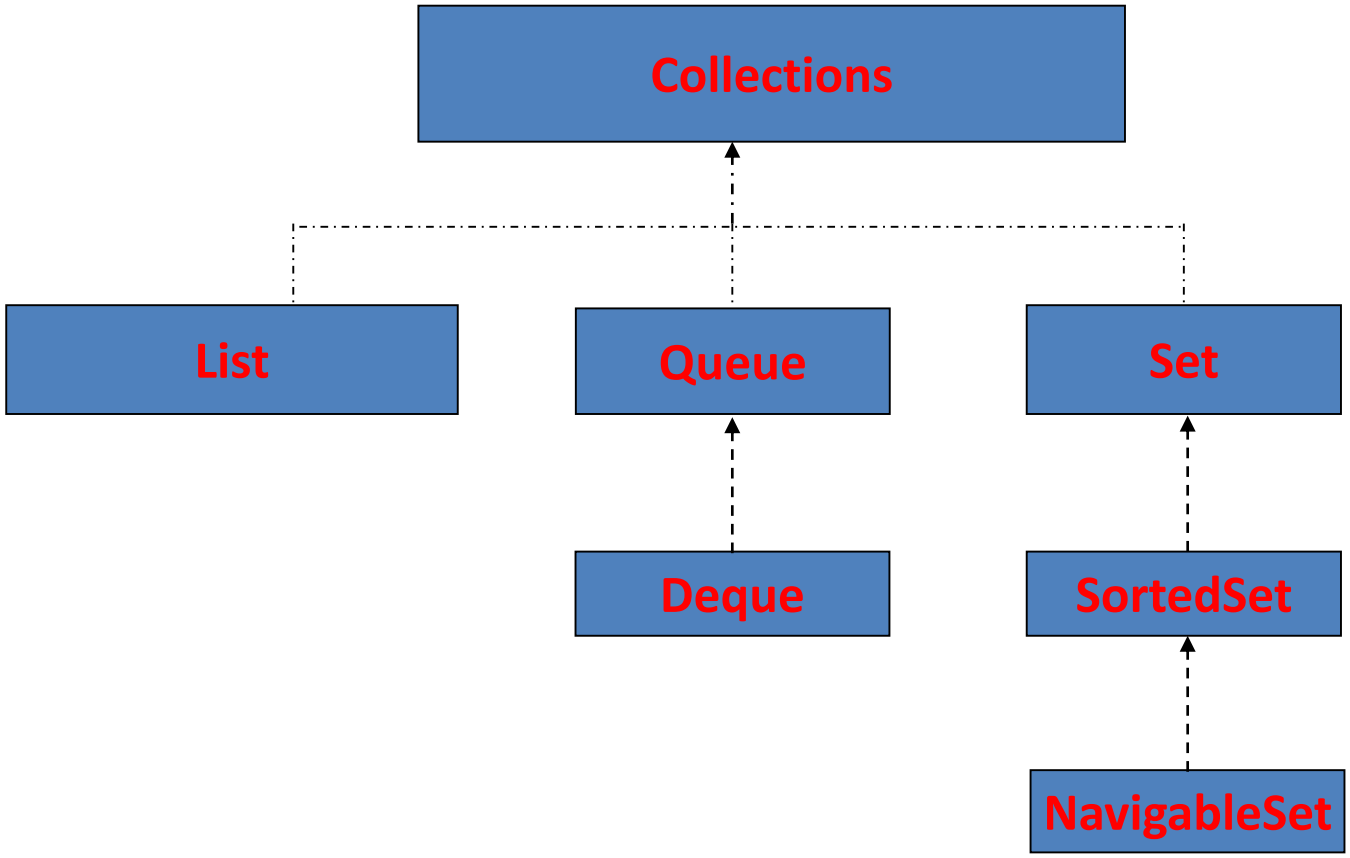
innovate achieve lead

- Group/Collection of Objects Treated as a Single Unit.
- Java Provides Support for Manipulating Collections in the form of
  - ❑ *Collection Interfaces*
  - ❑ *Collection Classes*
- Collection Interfaces→ Provides Basic Functionalities for all the Collections
- Collection Classes → Provides the Implementations of the Functionalities Offered by the Collection Interfaces

Collection Interfaces

- Collection → Enables you to work with Group of Objects.
- List → Extends Collection interface to handle sequences (list of objects)
- Queue → Extends Collection interface to handle Queues (FIFO lists)
- Set → Extends Collection interface to handle Sets. Sets Don't have duplicate elements
- Deque → Extends Queue interface to support doubly-ended queue
- SortedSet → Extends Set interface to handle sorted sets
- NavigableSet→ Extends SortedSet interface to handle retrieval of elements

Collection Interfaces

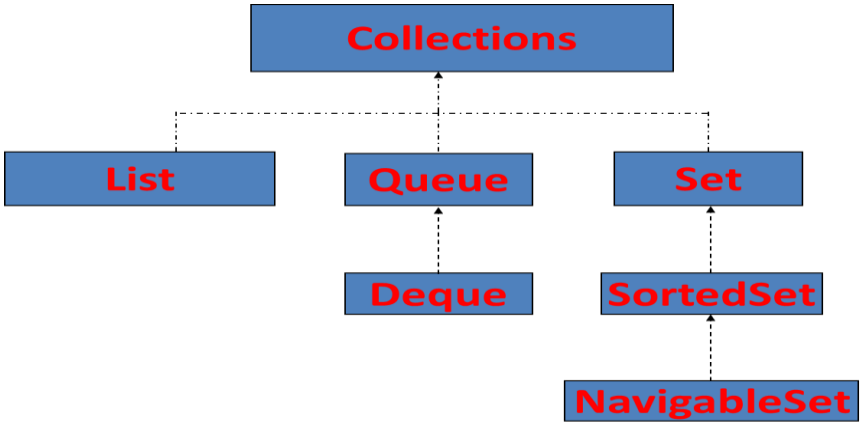


Other Interfaces Used by Collections: Comparator, RandomAccess, Iterator and ListIterator

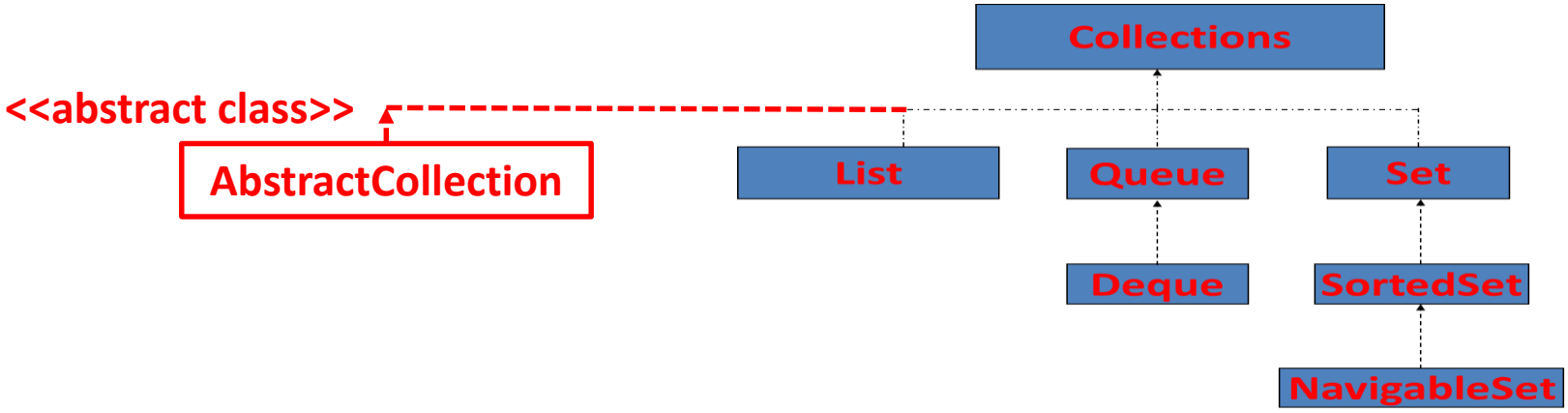
Collection Classes

- AbstractCollection → Implements most of the Collection Interface
- AbstractList → Extends AbstractCollection class and implements most of the List Interface
- AbstractQueue→ Extends AbstractCollection class and implements most of the Queue Interface
- AbstractSequentialList→ Extends AbstractList class for use by a collection that uses sequential rather than random access
- **LinkedList→** Implements the linked list data structure by extending AbstractSequentialList class
- **ArrayList →** Supports Dynamic Arrays by extending AbstractList class
- **ArrayDeque →** Implements a dynamic double-ended queue by extending AbstractCollection class and implementing Deque interface
- AbstractSet → Extends AbstractCollection class and implements most of the Set interface
- **PriorityQueue →** Extends AbstractQueue class to support priority-based queue

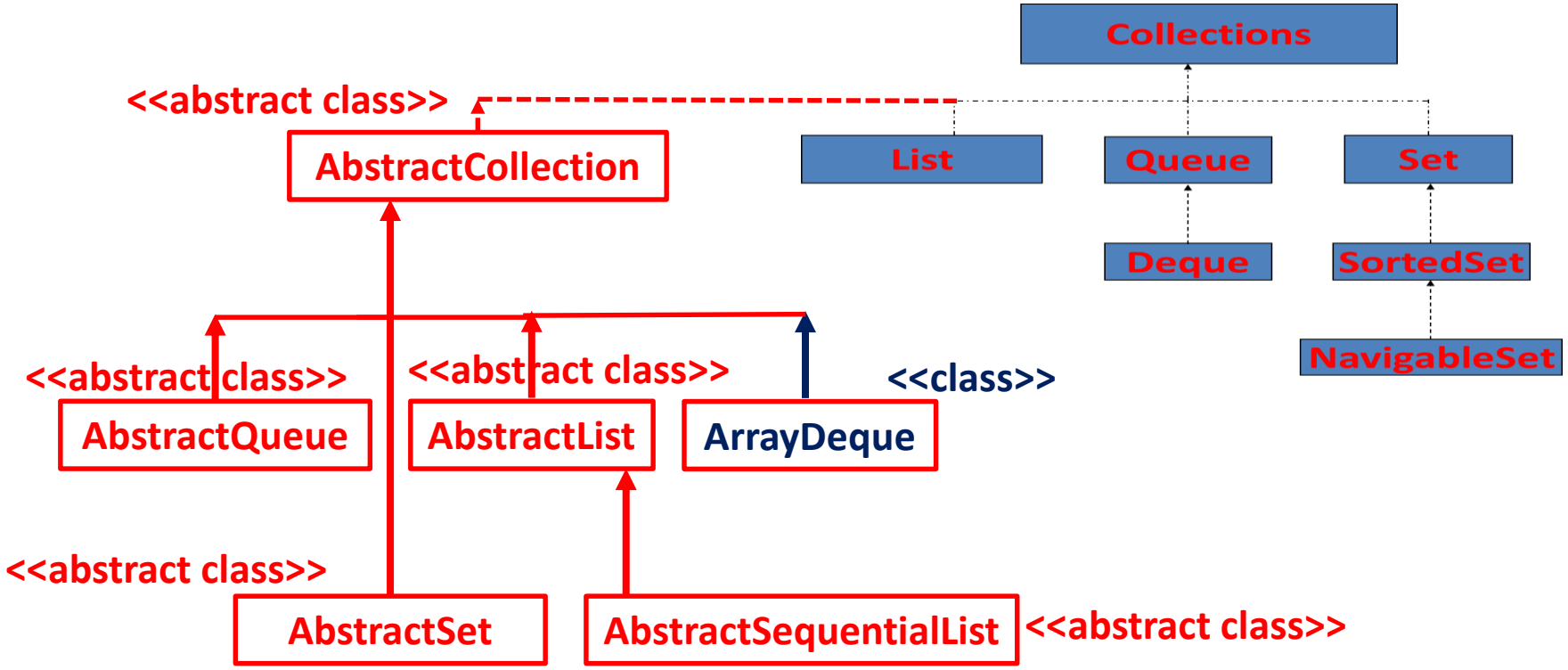
Partial View of Java's Collection Framework



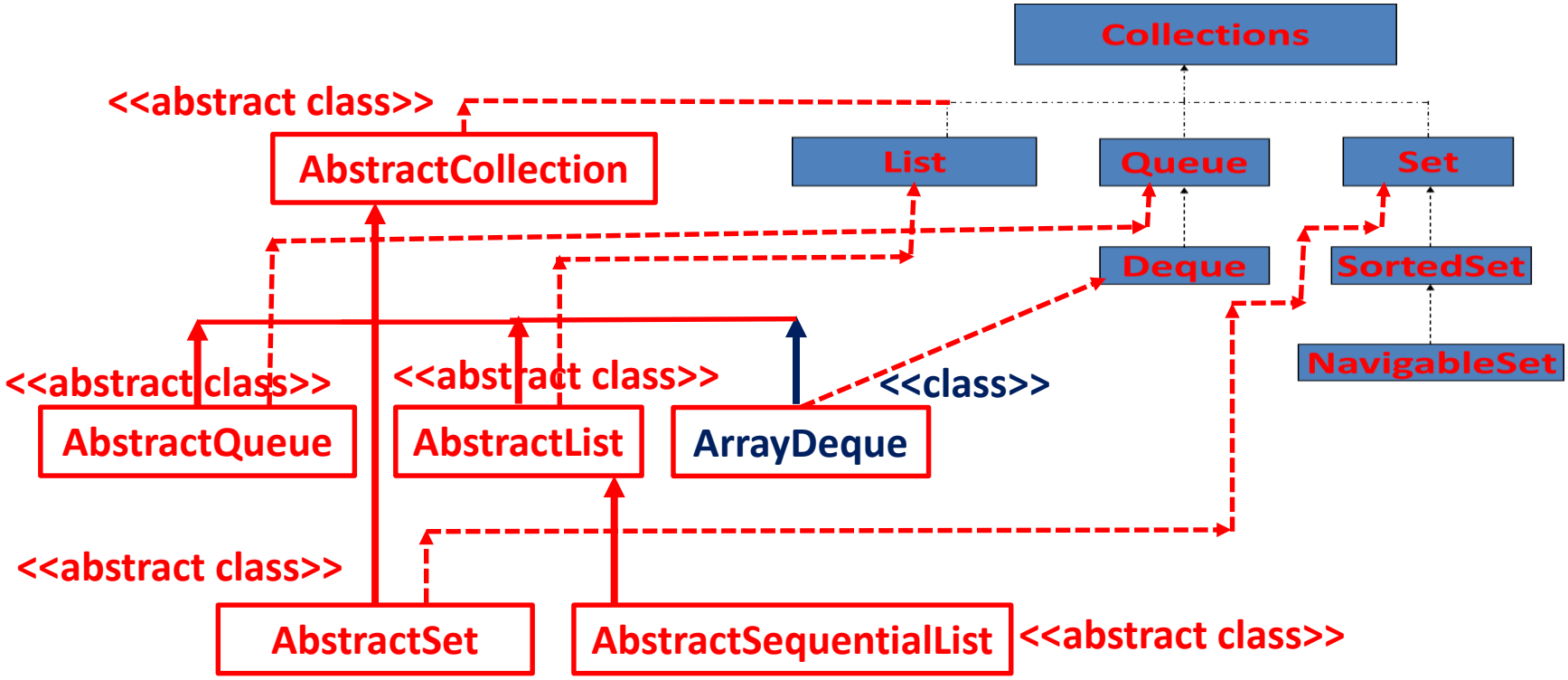
Partial View of Java's Collection Framework



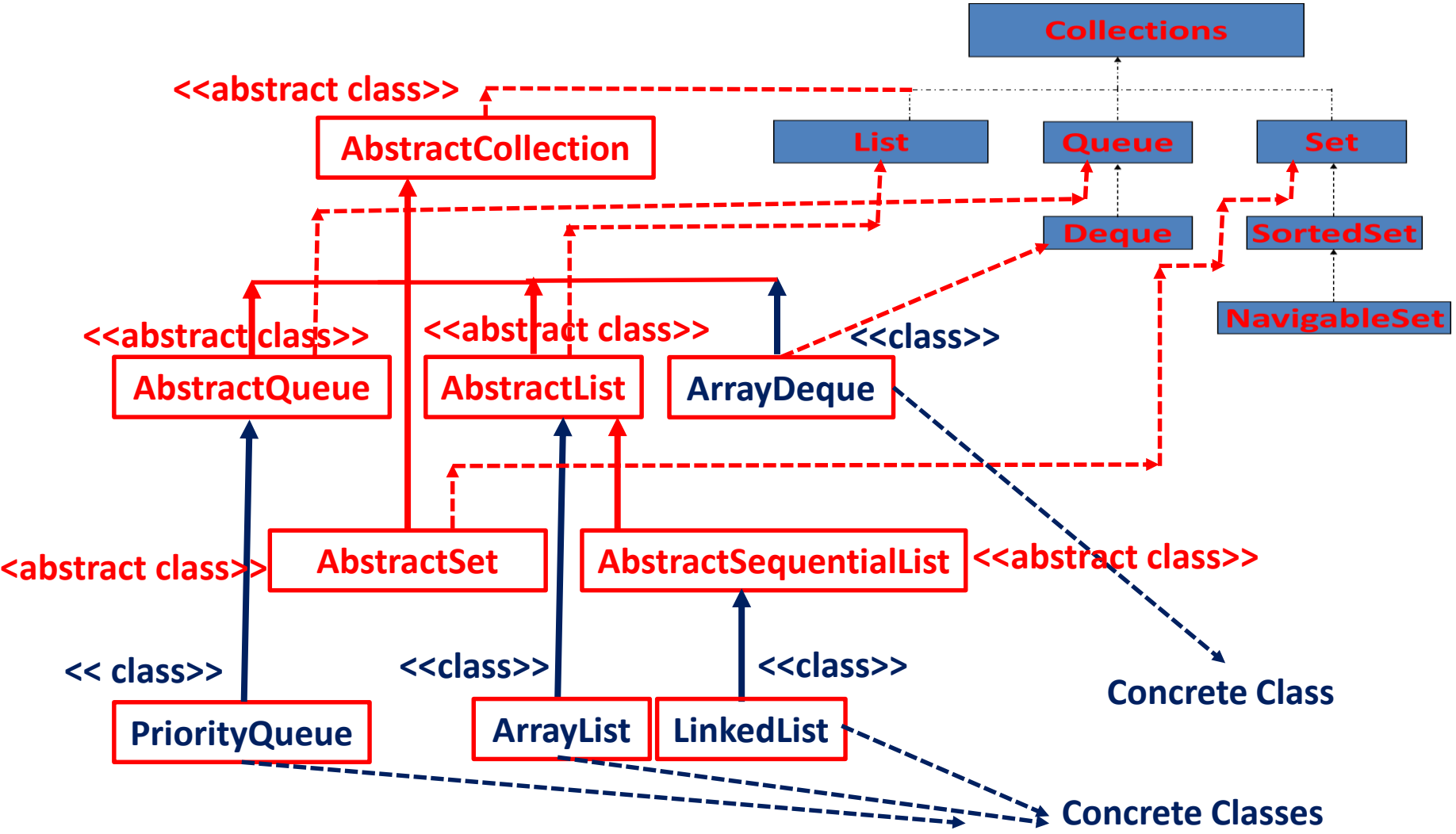
Partial View of Java's Collection Framework



Partial View of Java's Collection Framework



Partial View of Java's Collection Framework



Collection Interface

innovate achieve lead

interface Collection<E>

1. boolean add(E obj) → Adds 'obj' at the end of this collection
2. boolean addAll(Collection<? extends E> c) → Adds all elements of 'c' to the end of the invoking collection.
3. void clear() → Removes all elements of invoking collection and sets size =0
4. boolean contains(Object obj) → Searches 'obj' in invoking collection and returns true if exists otherwise false
5. boolean containsAll(Collection<?> c) → Returns true if all elements of collection 'c' are present in invoking collection
6. boolean equals(Object obj) → Returns true if 'obj' equals to the invoking collection, otherwise false
7. boolean isEmpty() → Returns true if size of the invoking collection is 0 otherwise false
8. Iterator<E> iterator() → Helps to iterate the collection
9. boolean remove(Object obj) → Removes the first occurrence of 'obj' from the invoking collection and returns true if successful otherwise false
10. boolean removeAll(Collection<?> c) → Removes all the elements (if exists) of collection 'c' from the invoking collection and returns true if successful otherwise false
11. int size() → returns the size of the collection
12. Object[] toArray() → Stores the elements of the invoking collection in Object type array and returns Object[].

List Interface

innovate achieve lead

interface List<E> extends Collection<E>

1. void add(int index, E obj) → Inserts 'obj' at location 'index' of this collection. Value of 'index' must be in the range '0<=index<=L' where 'L' is size of the collection.
2. boolean addAll(int index, Collection<? extends E> c) → Adds all elements of collection 'c' at location 'index' in the invoking collection. Value of 'index' must be in the range '0<=index<=L' where 'L' is size of the collection.
3. E get(int index) → Returns element from location 'index' of the invoking collection. Value of 'index' must be in the range '0<=index<=L-1' where 'L' is size of the collection.
4. int indexOf(Object obj) → returns the first index (>0) of 'obj' from the invoking collection if exists, otherwise -1 will be returned
5. int lastIndexOf(Object obj) → returns the last index (>0) of 'obj' from the invoking collection if exists, otherwise -1 will be returned
6. ListIterator<E> listIterator() → Helps to iterate the invoking collection in both (forward and backward) directions. Initially the iterator position is set either at the beginning or at the end of the collection
7. ListIterator<E> listIterator(int index) → Helps to iterate the invoking collection in both (forward and backward) directions. Initially the iterator position is set at 'index' location.
8. E remove(int index) → Removes the element from 'index' location. Value of 'index' must be in the range '0<=index<=L-1' where 'L' is size of the collection.
9. E set(int index, E obj) → Replaces the element at 'index' location by 'obj' value. Value of 'index' must be in the range '0<=index<=L-1' where 'L' is size of the collection.
10. List<E> subList(int start, int end) → Returns a list that includes element elements from 'start' to 'end - 1' in the invoking list.

Note : IndexOutOfBoundsException Will be Thrown if 'index' value is outside the range

Thank You