```
public interface Stack {
        public int size();
        public boolean isEmpty();
        public Object top()  throws StackEmptyException;
        public void push (Object element);
         public Object pop() throws StackEmptyException; }
```

```
public interface Queue {
  public int size();
  public boolean isEmpty();
  public Object front() throws QueueEmptyException;
  public void enqueue(Object element);
  public Object dequeue() throws QueueEmptyException;
}
```

```
public interface Deque {
  public void insertFirst(Object o);
  public void insertLast(Object o);
  public Object removeFirst() throws DequeEmptyException;
  public Object removeLast() throws DequeEmptyException;
  public Object first() throws DequeEmptyException;
  public Object last() throws DequeEmptyException;
  public int size();
}
```

```
public interface Vector {
  public int size();
  public boolean isEmpty();
  public Object elemAtRank(int r) throws OutOfBoundaryException;
  public void replaceAtRank(int r, Object e) throws OutOfBoundaryException;
  public Object removeAtRank(int r) throws OutOfBoundaryException;
  public void insertAtRank(int r, Object e) throws OutOfBoundaryException;
}
```

```
public interface Position {
    public Object element();
}
```

```
public interface List {
  public int size();
  public boolean isEmpty();
  public boolean isFirst(Position p) throws InvalidPositionException;
  public boolean isLast(Position p) throws InvalidPositionException;
  public Position first() throws EmptyContainerException;
  public Position last() throws EmptyContainerException;
  public Position before(Position p) throws InvalidPositionException,
        BoundaryViolationException;
  public Position after(Position p) throws InvalidPositionException,
        BoundaryViolationException;
  public Position insertBefore(Position p, Object element) throws InvalidPositionException;
  public Position insertAfter(Position p, Object element) throws InvalidPositionException;
  public Position insertFirst(Object element);
  public Position insertLast(Object element);
  public Object remove(Position p) throws InvalidPositionException;
  public void replaceElement(Position p, Object element) throws InvalidPositionException;
  public void swapElements(Position a, Position b) throws  InvalidPositionException;
  public Iterator positions();
  public Iterator elements();
}

public interface Sequence extends List, Vector {
  public Position atRank(int rank) throws BoundaryViolationException;
  public int rankOf(Position position) throws InvalidPositionException;
}
```