Java 2 Lists

- Lists
  - Impose order on a data type
  - Types
    - By Element Value
      - Self-ordered lists (sorted)
    - By absolute position (index number)
      - Indexed lists (sequence)
    - By relative position (front, rear, after)
      - Non-indexed (“bullet” list)
    - By time of insertion
      - Temporal lists (stacks, queues)
    - By priority
      - Priority queues

- **List Implementation Choices**
  - Array-Based or Node-Based implementation
  - Array-Based
    - Keep elements left-justified (anchored at 0, no gaps)
    - Keep a size counter (can serve as a rear marker)
  - Node-Based
    - Singly-linked
    - Not circular, no dummy
    - Keep both a front and rear pointer
    - Keep a size counter

- **Array-Based Implementation**

```
public class ArrayIndexedList<T> implements IndexedList<T>
{
        private T[] element;
        private int rear;
        …

        public void add(int index, T element) {
                if (index < 0 || index > size()) {
                        throw new IndexOutOfBoundsExceptions();
                }

                if (isFull()) {
                        resize(elements.length * 2);
                }

                shiftRight(index);
                elements[index] = element;
                rear++;
        }
}
```

- **Node-Based Implementation**

```
public class LinkedIndexedList<T> implements IndexedList<T>
{
        private Node<T> head;
        private Node<T> tail;
        private int size;
```

```
	public void add(int index, T element) {
		if (index < 0 || index > size) {
			throw new IndexOutOfBoundException();
		}

		LinearNode<T> temp = new LinearNode<T>(element);
		if (isEmpty())
		{
			head = temp;
			tail = temp;
		} else if (index == 0) {
			temp.setNext(head);
			head = temp;
		} else if (index == size) {
			tail.setNext(temp);
			tail = temp;
		} else {
			Node<T> p = head;
			for (int i = 0; i < index - 1; i++) {
				p = p.getNext();
			}
			temp.setNext(p.getNext());
			p.setNext(temp);
		}
	}
	size++;
}
```