## Topics

1. Create Position Interface
2. Create Positional List interface
3. Create Positional Linked List Using Linked List structure (Nodes)
4. Implement Basic Methods of Positional Linked List

* addBefore(Position<E> p ,E e)
* addAfter(Position<E> p ,E e)
* remove(Position<E> p)

1. Implement Iterator and Iterable pattern design in Positional Linked Lists

## Homework

1. Implement the ArrayList Data structure as it is described in chapter 7.
2. Implement the iterator idea in your ArrayList.

import java.util.Arrays;

import java.util.Iterator;

class ArrayList<T> implements Iterable<T> {

private static final int DEFAULT\_CAPACITY = 10;

private T[] data;

private int size;

@SuppressWarnings("unchecked")

public ArrayList() {

data = (T[]) new Object[DEFAULT\_CAPACITY];

size = 0;

}

public void add(T element) {

ensureCapacity();

data[size++] = element;

}

public T get(int index) {

checkIndex(index);

return data[index];

}

public void set(int index, T element) {

checkIndex(index);

data[index] = element;

}

public void remove(int index) {

checkIndex(index);

System.arraycopy(data, index + 1, data, index, size - index - 1);

data[--size] = null;

}

public int size() {

return size;

}

private void ensureCapacity() {

if (size == data.length) {

data = Arrays.copyOf(data, data.length \* 2);

}

}

private void checkIndex(int index) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Index out of range: " + index);

}

}

@Override

public Iterator<T> iterator() {

return new ArrayIterator();

}

private class ArrayIterator implements Iterator<T> {

private int currentIndex = 0;

@Override

public boolean hasNext() {

return currentIndex < size;

}

@Override

public T next() {

if (!hasNext()) {

throw new IllegalStateException("No more elements");

}

return data[currentIndex++];

}

}

}