## الحل:

import java.util.Iterator;

public class ArrayList<E> implements Iterable<E> {

private Object[] elements;

private int size;

private static final int DEFAULT\_CAPACITY = 10;

public ArrayList() {

elements = new Object[DEFAULT\_CAPACITY];

size = 0;

}

private void ensureCapacity() {

if (size == elements.length) {

Object[] newElements = new Object[elements.length \* 2];

System.arraycopy(elements, 0, newElements, 0, size);

elements = newElements;

}

}

public void add(E e) {

ensureCapacity();

elements[size++] = e;

}

public E get(int index) {

if (index < 0 || index >= size) throw new IndexOutOfBoundsException("Index out of bounds");

return (E) elements[index];

}

public void set(int index, E e) {

if (index < 0 || index >= size) throw new IndexOutOfBoundsException("Index out of bounds");

elements[index] = e;

}

public void remove(int index) {

if (index < 0 || index >= size) throw new IndexOutOfBoundsException("Index out of bounds");

for (int i = index; i < size - 1; i++) {

elements[i] = elements[i + 1];

}

elements[--size] = null;

}

public int size() {

return size;

}

@Override

public Iterator<E> iterator() {

return new Iterator<E>() {

private int index = 0;

@Override

public boolean hasNext() {

return index < size;

}

@Override

public E next() {

if (!hasNext()) throw new IllegalStateException("No more elements");

return (E) elements[index++];

}

};

}

public static void main(String[] args) {

ArrayList<Integer> list = new ArrayList<>();

list.add(10);

list.add(20);

list.add(30);

System.out.println("List elements:");

for (Integer num : list) {

System.out.println(num);

}

list.remove(1);

System.out.println("After removal:");

for (Integer num : list) {

System.out.println(num);

}

}

}

**Features Implemented**

1. **add(E e)** – Adds an element to the list.
2. **get(int index)** – Retrieves an element at the given index.
3. **set(int index, E e)** – Updates an element at a given index.
4. **remove(int index)** – Removes an element and shifts elements left.
5. **size()** – Returns the number of elements.
6. **iterator()** – Implements an **iterator** for easy traversal

## 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.