/\*\*

\* An implementation of the ListADT interface using arrays internally.

\*/

public class MyList<ElementType> implements ListADT<ElementType> {

// the array that represents our list

private ElementType[] \_listArray = null;

// the number of elements in the list

private int \_size;

/\*\*

\* The no-parameter constructor for the list class.

\*/

public MyList() {

// initialize the array with a capacity of 100

\_listArray = (ElementType[]) new Object[100];

// initialize this.\_size to 0

\_size = 0;

}

@Override

public void add(ElementType element) {

// check if we need to resize

if (\_listArray.length <= \_size) {

// create a new list that is double in size

Object[] newArray = new Object[\_size \* 2];

for (int i = 0; i < \_size; i++) {

// copy every element from the old to the new list

newArray[i] = \_listArray[i];

}

\_listArray = (ElementType[]) newArray;

}

// add new element at the end of the list

\_listArray[\_size] = element;

// increment element counter

\_size++;

}

@Override

@SuppressWarnings("unchecked")

public ElementType get(int index) {

// check if the index exists in our list

if (index >= 0 && index < this.\_size) {

// if it does, return element for the index

return (ElementType)\_listArray[index];

} else {

// if it does not exist, throw IndexOutOfBoundsException

throw new IndexOutOfBoundsException("index not in range (index < 0 || index > size())");

}

}

@Override

public int size() {

// return the number of elements currently in the list

return \_size;

}

@Override

public ElementType remove(int index) {

// check if the index exists in our list

if (index >= 0 && index < this.\_size) {

// if it does, delete element at this position by shifting all elements

// to the right of it one position to the left

ElementType deletedElement = this.get(index);

for (int i = index; i < \_size; i++) {

\_listArray[i] = \_listArray[i + 1];

\_listArray[i + 1] = null;

}

// decrement element counter

\_size--;

// return the deleted element

return deletedElement;

} else {

// throw an IndexOutOfBoundsException for non-existing indices

throw new IndexOutOfBoundsException("index not in range (index < 0 || index > size())");

}

}

@Override

public void clear() {

// instantiate a new empty array

\_listArray = (ElementType[]) new Object[100];

// reset element counter to 0

\_size = 0;

}

@Override

public String toString() {

// create and return a string representation for the list

StringBuffer sb = new StringBuffer();

sb.append("[ ");

for (int i = 0; i < this.size(); i++) {

if (i > 0) sb.append(", ");

sb.append(this.get(i).toString());

}

sb.append(" ]");

return sb.toString();

}

}