public **class ArrayList<E>** extends AbstractList<E>

implements List<E>, RandomAccess, Cloneable, java.io.Serializable {

private static final int DEFAULT\_CAPACITY = 10;

transient **Object[] elementData**;

private **int size**;

**public ArrayList(int initialCapacity) {**

**super();**

**if (initialCapacity < 0)**

**throw new IllegalArgumentException("Illegal Capacity: "+**

**initialCapacity);**

**this.elementData = new Object[initialCapacity];**

**}**

**public int size() {**

**return size;**

**}**

**public boolean isEmpty() {**

**return size == 0;**

**}**

**private void rangeCheck(int index) {**

**if (index >= size)**

**throw new IndexOutOfBoundsException(outOfBoundsMsg(index));**

**}**

**private void rangeCheckForAdd(int index) {**

**if (index > size || index < 0)**

**throw new IndexOutOfBoundsException(outOfBoundsMsg(index));**

**}**

**public E get(int index) {**

**rangeCheck(index);**

**return elementData(index);**

**}**

**public E set(int index, E element) {**

**rangeCheck(index);**

**E oldValue = elementData(index);**

**elementData[index] = element;**

**return oldValue;**

**}**

**public boolean add(E e) {**

ensureCapacityInternal(size + 1);

**elementData[size++] = e;**

**return true;**

**}**

**public void add(int index, E element) {**

**rangeCheckForAdd(index);**

**ensureCapacityInternal(size + 1);**

**System.arraycopy(elementData, index, elementData, index + 1,**

**size - index);**

**elementData[index] = element;**

**size++;**

**}**

**public E remove(int index) {**

**rangeCheck(index);**

**E oldValue = elementData(index);**

**int numMoved = size - index - 1;**

**if (numMoved > 0)**

**System.arraycopy(elementData, index+1, elementData, index,**

**numMoved);**

**elementData[--size] = null;**

**return oldValue;**

**}**

**public int indexOf(Object o) {**

**if (o == null) {**

**for (int i = 0; i < size; i++)**

**if (elementData[i]==null)**

**return i;**

**} else {**

**for (int i = 0; i < size; i++)**

**if (o.equals(elementData[i]))**

**return i;**

**}**

**return -1;**

**}**

**public boolean contains(Object o) {**

**return indexOf(o) >= 0;**

**}**

**}**