/\*\*

\* Creates a resizable array list

\* Eleni Stylianou ge21708

\* 26/11/2021

\*/

public class MyResizableIntArrayList

{ int size=0;

int data[];

/\*\*

\* Constracts a resizable array-list of integers with capacity=1.

\*/

public MyResizableIntArrayList()

{

data=new int[1];

}

/\*\*

\* Constracts a resizable array-list of integers with the capacity given by the user.

\* initialCapacity - The initial capacity of the resizable array-list.

\*/

public MyResizableIntArrayList(int initialCapacity)

{

data=new int[initialCapacity];

}

/\*\*

\* Changes the size of the array.

\* oldA - The array that we want to resize

\* newSize - The size of the new array

\*/

private int[] resizeArray(int oldA[], int newSize)

{

int newA[]=new int[newSize];

if(oldA.length<newSize)

for (int i=0;i<oldA.length;i++){

newA[i]=oldA[i];

}

else

for (int i=0;i<newSize;i++){

newA[i]=oldA[i];

}

return newA;

}

/\*\*

\* Appends a spesific element to the end of the resizable array-list.

\* elem - The element to be appended.

\*/

public void add(int elem)

{

if(size>=data.length){

data=resizeArray(data,2\*data.length);

}

data[size]=elem;

size+=1;

}

/\*\*

\* Inserts the specified element at the specified location in the resizable array-list.

\* index - The index of the new element after it is inserted. It MUST hold that (index >= 0 && size() >=index).

\* elem - The element to be inserted.

\*/

public void add (int index,int elem)

{

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

if(size>=data.length){

data=resizeArray(data,2\*data.length);

}

int temp=0;

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

data[i+1]=data[i];

}

data[index]=elem;

size+=1;

}

else

if(index>=0 && index<data.length){

size=index+1;

data[index]=elem;

}

}

/\*\*

\* Removes all the elements of the array-list and resets its capacity to 1.

\*/

public void clear()

{

data=new int[1];

size=0;

}

/\*\*

\* Checks whether this array-list contains a specific element.

\* elem - The element in question

\*/

public boolean contains(int elem)

{

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

if(data[i]==elem)

return true;

}

return false;

}

/\*\*

\* Returns the element at the specified position in this array-list.

\*/

public int get(int index)

{

return data[index];

}

/\*\*

\* Searches for the first occurence of the given argument. Returns the index of the first occurrence of the argument in this array-list; returns -1 if the object is not found.

\* elem - The element searched for.

\*/

public int indexOf(int elem)

{

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

if(data[i]==elem)

return i;

}

return -1;

}

/\*\*

\* Tests if this array-list is empty.

\*/

public boolean isEmpty()

{

if(size==0)

return true;

else

return false;

}

/\*\*

\* Removes the element at the specified position in this array-list.

\* index - The position of the element to be removed. It MUST hold that (index >= 0 && size() > index).

\*/

public void remove(int index)

{

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

data[i]=data[i+1];

}

data[size-1]=0;

size-=1;

if(4\*size==data.length){

System.out.println("hui");

data=resizeArray(data,(data.length/2));

}

}

/\*\*

\* Replaces the element at the specified position in this array-list with the specified element.

\* index - The index of the element to be replaced. It MUST hold that (index >= 0 && size() >index).

\* elem - The new element.

\*/

public void set(int index, int elem)

{

data[index]=elem;

if(size<=index)

size=index+1;

}

/\*\*

\* The size of this array-list.

\*/

public int size()

{

return size;

}

}