Implementing ArrayLists

In Lab 2 you will implement the following class:

```
public class MyArrayList<E> extends AbstractList<E> {
        private E[ ] data;
        private int size;
        ….
```

This represents a list structure that stores its data in an array.

Question: After you create the class

public class MyArrayList<E>

how will you **declare** and **construct** a new MyArrayList of Strings?

A. L = new MyArrayList<String>;
B. MyArrayList L = new MyArrayList(String);
C. MyArrayList L = new MyArrayList<String>();
D. MyArrayList<String> L = new MyArrayList<String>();

Answer:

D. MyArrayList<String> L = new MyArrayList<String>();

Of course, arrays are fixed-size and lists aren't, so an essential part of your implementation is a method

```
private void resize( )
```

that makes a new array A twice as long as the current data array, copies the elements of data into it, and then makes the data variable be this array with the assignment

```
data = A
```

Any method that adds data into your MyArrayList structure needs to check that there is room for the new data, and if there isn't to make room for it by calling resize( ).

You need to implement the abstract methods in the AbstractList abstract class. These are

- **constructors** new MyArrayList(int initialSize) and new MyArrayList( ) which uses initial size 2.
- int size( ) which returns the number of elements currently in the list
- void add(E element) which adds the new element to the end of the list.
- void add(int i, E element) which adds the new element at position i, shifting everything from there on back one slot to make room for the new element.

- E get(int i) which returns the ith element of the list.
- E set(int i, E element) which changes the data stored at position i to the new element. It returns the value previously stored at position i.
- E remove(int i) which removes the ith element from the list and then returns it. Remember to decrement the size. Also, shift everything above index i down 1 entry.
- boolean isEmpty( ) which just says if the size of the list is 0.
- void clear( ) which empties out the list. This needs to allow for garbage collection, so set all of the entries of data to null. Remember to ensure that size=0.

Many of these methods are supposed to throw exceptions, so your code needs to do that. For example, get(i) throws an IndexOutOfBoundsException unless i is between 0 and size() -1.