Assignment 4

Description

You will implement an array-based List with dynamic resizing.

Part 1

Implement the interface named List306.java in a Class named ArrayList306.java.

List306.java corresponds to the list interface defined in section 7.1, and ArrayList306.java corresponds to the class defined in section 7.2. The only difference is the file names ("306" is appended to each file name to avoid conflicts with the actual Java implementations with the same names as those in the text). You can use the code directly from the textbook, or write your own as long as it behaves the same and has the same public interface and is named as specified above (because unit tests will need to match the class name and public interface).

Correctly implementing the above is worth 70% of this assignment, and is the foundation upon which the next part build.

Part 2

This part is worth 30 points. The class below should be defined as a subclass of the class above, with the names specified below.

Dynamic Capacity

Implement the List306B.java interface in a subclass named ArrayList306B.java.

This portion of the assignment relates to dynamically sizing the underlying array. In addition to implementing the interface methods, the data structure's existing behavior should be modified such that the size dynamically increases when needed. One of the interface methods increases capacity, and it should be "automatically" invoked as needed.

Unit Tests

I will provide a variety of test programs that you can use to test your work. They are in files with the same names as above, but with a "Test" preceding their names. For example, TestArrayList306.java corresponds to ArrayList306.java. Your programs may be tested by other test sets, so do not assume these are exhaustive.

Requirements

1. Complete part 1. You can't complete part 2 otherwise, because part 1 is the superclass on which part 2 is based.
2. Complete part 2.
3. Name your .java files as specified above (so that unit tests will work properly).

Deliverables

You should submit a zip file named YourFullName-Assignment3 containing:

- *only* your java source code
    - no metafiles
    - no .class files
    - no interface files
    - no unit test files (e.g. Test*.java)
    - no package declarations in your source code files!

Note: any unimplemented methods in your classes should remain "stubbed" so that they compile against the interfaces, and run relative to the unit tests. Any programs that do not compile or run, or cause the test programs to fail, will earn zero points.

Stubbed means that the class will respond to the public interface even if the method is not otherwise implemented. For example:

```
public boolean isEmpty() {  return true; }
```

The method above will not work correctly (unless it's truly empty), but it will compile and it will allow unit tests to invoke it.