FlexArray.java, V2.0 files

Add on to your two different classes named `FlexArrayPrimitive` and `FlexArrayRectangle` to enhance their capabilities. Use the same files as before (with corrections, if needed), with the addition of the methods below. The OLD methods are repeated below the new requirements. The descriptions below apply to both FlexArrays. Pay careful attention to what it means for one `Rectangle` object to be equal to another `Rectangle` object, and how `compareTo()` should be defined for the `Rectangle` class. As part of the success of this assignment, make sure you have added these two methods to the `Rectangle` class in your FlexArray 2.0 folder. The `equals()` and `compareTo()` method descriptions in <u>this</u> assignment are for the `FlexArray` objects

Coding:

Your classes should add the following methods. All methods are relevant only in the active array elements.

```
// A FlexArrayPrimitive will be compared by the sum of its active elements.
// EXAMPLE:
//  flexy1.compareTo(flexy2) returns a negative int if the sum of the elements of
//      flexy1 < the sum of the elements of flexy2
//  flexy1.compareTo(flexy2) returns a zero if the sum of the elements are the same
//  flexy1.compareTo(flexy2) returns a positive int if the sum of the elements of
//      flexy1 > the sum of the elements of flexy2
public int compareTo(FlexArrayPrimitive flexy)
```

```
// A FlexArrayRectangle will be compared by the sum of the lengths of each of its active
//    elements.
// EXAMPLE:
//  flexy1.compareTo(flexy2) returns a negative int if the sum of the lengths of
//      flexy1 < the sum of the lengths of flexy2
//  flexy1.compareTo(flexy2) returns a zero if the sum of the lengths are the same
//  flexy1.compareTo(flexy2) returns a positive int if the sum of the lengths of
//      flexy1 > the sum of the lengths of flexy2
public int compareTo(FlexArrayRectangle flexy)
```

```
// A FlexArray object will be considered equal to another FlexArray if the quantity of
//     active elements are the same.
//  EXAMPLE:  flexy1.equals(flexy2) returns true if the quantity of active elements
//     are the same; otherwise it returns false
public boolean equals(FlexArrayPrimitive flexy)
or public boolean equals(FlexArrayRectangle flexy)
```

```
//using any sort routine, sort from low to high, index 0 to mySize - 1
//  note that this will change the order of the elements in myArray permanently
public void sort() //(this method signature is correct for both classes)
```

```
// returns the contents of the FlexArray at index loc
// if loc is not in the range of active elements, return -999 for a FlexArrayPrimitive
//      object, return null for a FlexArrayRectangle object.
// precondition: loc >= 0
public int getValue(int loc)
or public Rectangle getValue(int loc)
```

```
// returns the index of the first instance of element target in the list
// if target is not in the range of active elements, return -1
public int searchFor(int target)
```
or **`public int searchFor(Rectangle target)`**

CHANGE FOR THE FlexArrayRectangle toString():
```
public String toString() //only an addition for the FlexArrayRectangle
```

The `FlexArrayRectangle` should look similar to the first version, but should also state the area of each FlexArrayRectangle (to ease confirmation that it is sorted correctly). It should still contain hard brackets at its start and end, list each Rectangle on a new line, but should also add its area like:

```
[Rectangle, width = 0 length = 1, area = 0
Rectangle, width = 4 length = 2, area = 8
Rectangle, width = 8 length = 3, area = 24
Rectangle, width = 12 length = 4, area = 48]
```

What to turn in:

Only these two files will be turned in TO THE NEW ASSIGNMENT. All of the methods in your files will be tested with my driver. All identifiers and parameter lists must be exactly as written above.

OLD METHODS (REPEATED WITHOUT CHANGE):

Private instance fields: (no additional fields should be created)

`private int mySize` represents the logical size of the array. (The quantity of active elements)

`private myArray`: an array that is appropriate to each FlexArray object type

Constructors:

a no args constructor should initialize `myArray` to hold 20 elements

a constructor that takes an integer, representing the quantity of elements `myArray` should contain

Methods: (you may add other methods as you see appropriate, but you must implement the ones shown below)

`public int getLength()`

returns value of `mySize`

`public boolean isEmpty()`

returns true if `mySize` is zero, false otherwise

`public void append(int data)`

or `public void append(Rectangle data)`

adds `data` to the end of the current list

```
//pre-condition:  index >= 0
public void insert(int index, int data)
```
or `public void insert(int index, Rectangle data)`

- inserts `data` into the list at the index value `index`
- the order of existing elements remains the same
- Example:

  if the `FlexArrayPrimitive` contains `[4, 6, -8, 2, 6, `*garbage*` . . .]`

  after the call of `insert(3, 5)`, the array will look like `[4, 6, -8, 5, 2, 6, `*garbage*` . . .]`
- any value cannot be placed at an index higher than `mySize`. To ensure this, if `index > mySize`, simply add data to the end of the active elements.

```
//pre-condition:  index >= 0
public int discard(int index)
```
or `public Rectangle discard(int index)`

- the value of the element at `index` is returned
- removes the element at the index value `index`
- the order of remaining elements remains the same
- if `index >= mySize`, return `-999` for the integer array, `null` for the `Rectangle` array
- Example:
  if the `FlexArrayPrimitive` contains `[4, 6, -8, 2, 6,` *garbage* `. . .]`
  after the call of `discard(1)`, the array will look like `[4, -8, 2, 6,` *garbage* `. . .]` and a 6 will be returned

`private` `void resize()`
This private method will change the physical size of the array when needed. Make a local array that is larger than `myArray`'s current length. Copy its elements into the new, larger array. You may decide how much to enlarge the array. Outcome: `myArray` is larger than before the call to this method and contains the same values in the new array from [0, `mySize` – 1] as before. This method is private because the driver will never need to see the inner workings of how the FlexArray enlarges as needed. The FlexArray objects will control it internally.

`public String toString()`
output for the values for each `FlexArray` must show the list as demonstrated below.

The contents of a `FlexArrayPrimitive` should be listed in order from the beginning to the end of the active elements, all on one line with one comma and a space between each element, with hard brackets showing the beginning and the end of the list. For example, a `FlexArrayPrimitive` object filled with only 4 elements should look like:

```
[43, 6, 298, 7]
```

It is okay for a long list to wrap to the next line.