CS272 Lab #2: Collection class

Learning objectives: Objective 1 (collection), Objective 5, Objective 6, Objective 7

Note:

- **Specifications** for all your classes and methods:
  Please properly explain (1) the functionality of the methods, (2) the parameters, (3) the return values, (4) the pre-conditions if there is any;
  Please use inline comments, meaningful variable names, indentation, formatting, and whitespace throughout your program to improve its readability.
- You can (but are not required to) design and implement other facilitating methods (E.g., other get and set methods, toString method) to finish the implementation of the required methods.

1. Requirements

In the previous lab, we use a fixed-length array to store all the authors for a book.
This approach has one disadvantage: when the author array is full, we cannot add more authors for the book.

In this lab, based on what we have learned in class, you are asked to design a data structure `StringSet`, which can be used to store all the authors of a book.
You are required to design the data structure `StringSet` to store `String` values and support basic operations.

This `StringSet` data structure should implement the functionality of a collection whose space can grow automatically.
This collection data structure does NOT allow you to store the same String value multiple times.
I.e., if the collection already contains a String value `a`, you cannot add another given String value `a` to the collection.

The class `StringSet` should define proper instance variables and implement the following methods.

1. (5 pts) This class should include proper instance variables to keep all the distinct string values and the actual number of string values.
2. (5 pts) A no-argument constructor, which initializes a `StringSet` instance, sets its capacity to 2, and creates an array to store 2 String instances.

   ```
   public StringSet()
   ```

3. (10 pts) The following constructor, which initializes a `StringSet` instance, sets its capacity to the input parameter `_capacity`, and creates an array to store `_capacity` string values.

   The precondition is that `_capacity` is a positive value.

   ```
   public StringSet(int _capacity)
   ```

4. (10 pts) A copy constructor, which creates a new `StringSet` instance and copies the content of the given object to the new instance.

   The precondition is that `obj` should NOT be a null object and should be an instance of `StringSet`.

   ```
   public StringSet(Object obj)
   ```

5. (5 pts) The following method which returns the actual number of elements in this collection.

   ```
   public int size()
   ```

6. (5 pts) The following method which returns the capacity of this collection instance.

   ```
   public int capacity()
   ```

7. (12 pts) A method which adds a given String value to the first available space of the string array in this `StringSet` instance.
   When the collection space is sufficient to hold the new String, this String value can be directly added to the collection.
   Otherwise, you need to double the space of the instance array by implementing a method `ensureCapacity` (defined below).

   The precondition is that the String value `a` should NOT be null.

   ```
   public void add(String a)
   ```

8. (10 pts) A method to check whether this collection contains the input parameter.
   If `a` is a NULL object, directly return `false`.
   If `a` is not a NULL object,
   - if this collection contains a String value which equals to `a`, then return `true`;
   - Otherwise, return `false`.

   ```
   public boolean contains(String a)
   ```

9. (10 pts) A method to remove from the collection the String which has the same value as the given parameter.
   If `a` is a NULL object, directly return `false`.
   If `a` is not a NULL object,
   - if this collection contains one `String` value `a1` which equals to `a`, then remove `a1` from this collection and return `true`;
   - Otherwise, do nothing and return `false`.

   ```
   public boolean remove(String a)
   ```

10. (13 pts) The following method which guarantees the capacity of the collection.
    If this collection's capacity is smaller than the input parameter, this method sets the capacity to `minimumCapacity` and enlarges the array to hold `minimumCapacity` Strings;
    Otherwise, this collection is left unchanged.

    The precondition is that the input parameter `minimumCapacity` should be positive.

    ```
    private void ensureCapacity(int minimumCapacity)
    ```

11. (15 pts) A method which adds one String value to this `StringSet` instance such that the values in the string array are ordered ascending.
    When the collection space is sufficient to hold the new String, this String value can be directly added to the collection.
    Otherwise, you need to double the space of the instance array by implementing a method `ensureCapacity`.

    The preconditions are that

    - the String value `a` should NOT be null, and
    - the values in the collection's string array are already ordered ascendingly.

    ```
    public void addOrdered(String a)
    ```

Please implement a `main` function inside `StringSet.java` with test cases to thoroughly (by considering all possible test cases) **test** your code.

2. Grading Criteria

- The score allocation has already been put beside the questions.
- Please make sure that you test your code thoroughly.
  Your code may be testd using more test cases.

3. Submission

A zip file *bannerid-lab2.zip* containing `StringSet.java`.