Project #5: Set Class

In this project you will implement a `Set` class which represents a general collection of values. For this assignment, a set is generally defined as a list of values that is **sorted** and **does not contain any duplicate values**. More specifically, a set shall contain no pair of elements `e1` and `e2` such that `e1.equals(e2)` and no `null` elements.

Requirements

To ensure consistency among all implementations there are some requirements that all implementations must maintain.

- Your implementation should reflect the definition of a set at all times.
- For simplicity, your set will be used to store `Integer` objects.
- An `ArrayList<Integer>` object must be used to represent the set.
- All methods that have an object parameter must be able to handle an input of `null`.
- Methods such as `Collections.sort` that automatically sort a list may not be used.
- The `Set` class shall reside in the default package.

Recommendations

There are deviations in program implementation that are acceptable and will not impact the overall functionality of the set class.

- A minimum size (initial capacity) may be specified for the `ArrayList` object.
- When elements are added, the `ensureCapacity`() method may be called to ensure that adding a new element can proceed.
- The `trimToSize`() method can be used to maintain a constant capacity equal to the number of elements in the set.

Set Class Methods

There are many methods that one would expect to be supported in a set class. This section will describe the interface of the set class. Unless specified, you will have to implement all of the described methods.

Constructors

- `Set()`
  - Description: constructs a set by allocating an `ArrayList<Integer>`.
- `Set(int size)`
  - Parameters: `size` - the desired `size` of the set.
  - Description: constructs a set with an initial capacity of `size`.
- `Set(int low, int high)`
  - Parameters:
    - `low` – an integer specifying the start value of a range of values.

        - `high` – an integer specifying the end value of a range of values.
    - Description: constructs a set of `Integer` objects containing all the inclusive values from the range `low`…`high`. The default size of the set must accommodate this mode of construction.

Addition

- `boolean add(Integer o)`
    - Parameters: `o` – element to be added to `this` set.
    - Description: if `o` is not in `this` set, `o` is added to `this` set.
    - Returns:
        - TRUE if the element is successfully added to `this` set.
        - FALSE if the element is not added to `this` set.
- `int add(Integer[] s)`
    - Parameters: `s` – array of elements to be added to `this` set.
    - Description: adds all elements of `s` to `this` set.
    - Returns: the number of elements successfully added to `this` set.

Removal

- `Integer remove(Integer o)`
    - Parameters: `o` – element to be deleted from `this` set.
    - Description: if `o` is in `this` set, the element is deleted.
    - Returns: the object that will be removed from `this` set. If the element is not contained in `this` set, `null` is returned.
- `int remove(Set s)`
    - Parameters: `s` – set of elements to be deleted from `this` set.
    - Description: deletes all elements of `s` from `this` set.
    - Returns: the number of elements successfully deleted from `this` set.

Miscellaneous

- `boolean contains(Integer o)`
    - Parameters: `o` – the element to be searched for in `this` set.
    - Description: determines whether the given element is in `this` set.
    - Returns: TRUE or FALSE whether `o` is in `this` set.
- `void clear()`
    - Description: Removes all the elements from `this` set.
- `boolean isEmpty()`
    - Description: determines whether `this` set contains any elements.

- Returns: TRUE or FALSE whether `this` set contains zero elements.

- `int size()`
  - Description: determines the number of elements in `this` set.
  - Returns: the number of elements in `this` set.

- `Integer get(int index)`
  - Parameters: `index` – the integer index of the desired element in `this` set.
  - Description: returns the `Object` at the specified `index` if the index is valid.
  - Returns: the `Object` at the specified `index`; `null` if the specified index is out of range: `(index < 0 || index >= size())`.

Union

- `Set union(Set s)`
  - Parameters: `s` – a set of Integers.
  - Description: constructs and returns a new `Set` object that contains the objects in either `this` set ***or*** the input set `s`.
  - Returns: a newly constructed `Set` object containing the union of the sets.

Intersection

- `Set intersection(Set s)`
  - Parameters: `s` – a set of Integers.
  - Description: constructs and returns a new `Set` object that contains the objects in both `this` set ***and*** the input set `s`.
  - Returns: a newly constructed `Set` object containing the intersection of the sets.

Other Methods

- `boolean subset(Set s)`
  - Parameters: `s` – a set of Integers.
  - Description: determines if `this` set is a superset of `s`.
  - Returns: TRUE or FALSE if all the elements of `s` are contained in `this` set.
  - Notes: This method can be implemented easily using other methods described in this assignment. `null` should be considered a subset of any other set.

- `boolean superset(Set s)`
  - Parameters: `s` – a set of Integers.
  - Description: determines if `this` set is a subset of `s`.
  - Returns: TRUE or FALSE if all the elements of `this` set are contained in `s`.
  - Notes: This method can be implemented easily using other methods described in this assignment. `null` can only be considered a superset of the `null` set. However, this cannot conceivably happen when using an instance of the `Set` class.

Supplied Methods

Besides the interface specification described above, there are other methods that have been provided; feel free to use them as required in your implementation and testing. ***Do not modify these methods.***

- `String toString()`
    - Description: provides a means of viewing the values contained within the `Set` object.
    - Returns: a `String` representation of `this` set.
    - Notes: It is important that this method remain unmodified since it will be used for evaluation purposes. It has been provided for debugging purposes.
- `boolean equals(Object o)`
    - Parameters: `s` – a set of values to compare against `this` set.
    - Description: determines whether the content of `o` equals `this` set.
    - Returns: TRUE or FALSE whether `o` is an instance of class `Set` and contains all elements of `this` set.
    - Notes: It is important that this method remain unmodified since it will be used for evaluation purposes. It has been provided for debugging purposes.

Testing

- `public static void main(String args[])`
    - Parameters: `args` – unused in this implementation.
    - Description: this is a method that will be part of a test class `SetTester`.
    - Notes: A complete testing class (`SetTester`) used for grading has been provided to expedite development. This method can be modified as desired; you will not have to submit this file.

Submitting

Your program must use the following standard comment at the top of the page as well as a reasonable amount of comments throughout the program. Copy and paste this comment and modify it accordingly.

```
/**********************************************************
<Student Name>
<File Name>
<Assignment>
<Describe, in general, the code contained.>
**********************************************************/
```

Place your Java file for the `Set` class directly into a zip file. The name of the zip file **must be** `proj5.zip`. Make sure your zip file contains the correct files (`Set.java`). Submit your zip file via Sakai under `Assignments > Project 5`. Be sure to review the university policy on academic dishonesty. This is an individual project.