ADS Exam 1 - Part B

Total Marks: 70 Total Time: 2 hrs

Instructions:

1. This is a closed book exam
2. This is Part B of the exam

Introduction

In this assignment you will be creating a basic class that acts as a data structure store integers. It is backed with a simple array. One nice feature of your new data structure is that it can rearrange the data in some useful ways and also be enlarged if you need to add more items than the underlying array can hold.

Download the zip file containing the exam files for Canvas. You will make all of your changes inside of MyArray.java.

MyArray

Inside the file MyArray.java you will find method declarations with empty bodies. You need to implement each incomplete method and write code to test them inside the main method.

You are free to add new methods to the file (such as helper methods), but do not change the names of any of the provided methods.

The methods you are writing are as follows:

`public MyArray()`

- Default constructor. You probably shouldn't change this.

`public MyArray(int[] arr)`

- Constructs a MyArray object with a capacity of 20 items and then stores only the unique values from arr into it.e.g., if the argument contained [1,3,5,1,8,2,42,5,1], the MyArray object would contain [1,3,5,8,2,42]. arr could contain more items than this object can store.
- Parameters: arr — The array of values to add to the MyArray. arr must remain unchanged after the call to this constructor. (So this method is non-destructive)

`public MyArray(int numRandom, int low, int high)`

- Constructs a MyArray object with a capacity of 20 items and then puts the specified number of random integers in the range [low…high) into it.
- Parameters:
    - numRandom — The number of random numbers to generate and include in the MyArray object. If numRandom is larger than the maximum size of the array, then stop adding items once it is full
    - low — The lower end of the range of possible values for the random numbers (inclusive)
    - high — The upper end of the range of possible values for the random numbers (not inclusive)

`public boolean isEmpty()`

- Determines if the MyArray object is empty or not
- Returns: true if there are no items in the array, and false otherwise

`public boolean isDecreasing()`

- Checks if all the items are in sorted, decreasing order. i.e., a[i] >= a[i+1] for all valid array elements
- Returns: true if all elements are in decreasing order, false otherwise

`public int maximum()`

- Find the maximum integer value currently stored
- Returns: the maximum value stored or Integer.MIN_VALUE if there are no items stored

`public void insertFront(int i)`

- Adds a new integer to the beginning of the array. e.g., If the arrays contained 12, 4, 5 and this method was called to insert 13, the array would contain 13, 12, 4, 5. If the array is full, it should NOT add the new integer. Don't forget to update the instance variable that tracks how many items are stored in the array.
- Parameters: i — The new value to add to the array

`public void rotateRight()`

- Rotates all the elements in the array one position to the right, placing the last element in position 0. Example: if the array before the call to rotateRight is [4, 5, 6, 1] it is [1, 4, 5, 6] after the call

`public void rotateLeft()`

- Rotates all the elements in the array one position to the left, placing the first element at the last position. Example: if the array before the call to rotateRight is [4, 5, 6, 1] it is [5, 6, 1, 4] after the call

`public void reverse()`

- Reverses the elements stored in the array. Example: if the array before the call to reverse is [1, 4, 5, 6] it is [6, 5, 4, 1] after the call

`public void makeDups()`

- Makes a duplicate of each value in the array e.g., if the arrays contained [1,7,5] before calling this method, it would contain [1,1,7,7,5,5] after. Don't forget to ensure that the array never becomes larger than its maximum size.

`public boolean isFull()`

- Determine if the array is full.
- Returns: true if array is full and false otherwise

`public void enlarge()`

- Make the array able to hold twice as many items. To do this, you'll need to create a new, larger array, and copy the existing items into it.

`public String toString()`

- Returns a string representing the array, e.g., if the intArray contained [1,7,5] then this should return the String "[1,7,5]"
- Returns: String representing the array

Grading and Submission

Each method is worth 5 points. Complete all methods as per the instructions given above and submit the completed MyArray.java in Canvas.

Restrictions

You may only use the following java libraries to complete this exam

- java.util.Random