FA21 CSE 8B Assignment 3: Arrays and Classes

Due Date: Wednesday, October 20th, 11:59 PM Pacific time

Learning goals:

- Java 1-D array
- Java 2-D array
- Java classes
- Object-Oriented Programming (OOP)
- Unit testing

In this assignment, you will implement two classes that support multiple list manipulations using both 1-dimensional and 2-dimensional arrays.

NOTE: This assignment should be completed INDIVIDUALLY. Pair programming is NOT allowed for this assignment.

IMPORTANT: You cannot explicitly import any Java packages (e.g. "import java.util.Arraylist;"). Keep in mind that the Math package is automatically imported.

Coding Style (10 points)

For this programming assignment, we will be enforcing the CSE 8B Coding Style Guidelines. These guidelines can also be found on Canvas. Please ensure to have *COMPLETE* file headers, class headers, and method headers, to use descriptive variable names and proper indentation, and to avoid using magic numbers.

Part 0: Getting started with the starter code (0 points)

1. Make sure there is no problem with your Java coding environment. If there is any, then review Assignment 1, or come to the office/lab hours before you start Assignment 3.
2. Download the starter code.
   If you are working on your local machine, then you can download the starter code from Piazza → Resources → Homework → `Assignment3.zip` and unzip it.
   If you are working via UCSD Linux Cloud through your CSE 8B account, then use the commands below to copy the starter code to your home directory:

```
$ cp -r ../public/assignments/PA3 ~
$ cd ~/PA3/starter
$ ls
```

The starter code should only be three files called `MyList.java`, `My2DList.java`, `Assignment3.java`.

3. Try to compile your Java files using the following terminal command: `javac MyList.java My2DList.java Assignment3.java`. You will see a compile error when you try to compile for the first time. Complete the methods with TODO by following the instructions below.

Part 1: Implement `MyList.java` (40 points)

In class `MyList` of the starter code, 5 methods are already implemented for you: `MyList()`, `getArray()`, `getSize()`, `setArray()`, and `setSize()`. **Those methods are used by the Gradescope Autograder. Therefore, you should NOT modify any of these methods!**

There are 5 methods that you need to implement yourself, including: `MyList(int capacity)`, `append(int x)`, `pop()`, `get(int i)`, `index(int x)`. You should remove any `TODO` comments and implement those methods.

There are two member variables: `array` and `size`. The variable `array` is an integer array storing the data. The variable `size` indicates the number of data points in the array. Notice that the `size` is not the length of the `array`. Instead, there is another concept called capacity that represents the length of the array. The following example shows a possible scenario where only the first two data points are considered as our data (`size` is 2, in red) even though the capacity is 6.

```
array: {1, 2, 3, 4, 5, 6}
size: 2
```

`public MyList(int capacity)` (8 points):

This method is called a constructor. This constructor takes an integer and initializes the `array` member variable with an array containing integers. The initial length of the array is `capacity`. If `capacity` is less than 1, then initialize your array with length 1. An empty `MyList` object should have `size` of 0 at the beginning.

`public void append(int x)` (8 points):

This method appends a data point `x` to the array. If `x` is less than 0, then the method will add the absolute value of `x` instead. This makes sure that all elements inside our `array` are

non-negative. When your array capacity is full, you should create a new `array` with double the capacity, copy all original values, append the new element `x`, and increment `size`. The following example shows before and after adding two elements.

```
array: {1, 2, 6}
size: 2

// After calling append(-1)
array: {1, 2, 1}
size: 3

// After calling append(1)
array: {1, 2, 1, 1, 0, 0}
size: 4
```

`public int pop()` (8 points):

This method pops the last element as shown below. If MyList is empty, then return -1. Otherwise return the value of the last element. Notice that the value of the popped element in the `array` is not changed. Simply, you only need to decrease `size`.

```
array: {1, 2, 1, 1, 0, 0}
size: 4

// After calling pop()
array: {1, 2, 1, 1, 0, 0}
size: 3
pop returns 1, corresponding to the element at index 3.
```

`public int get(int i)` (8 points):

This method returns the element at position `i`. If input `i` is out of range, then return -1.

```
array: {1, 2, 1, 1, 0, 0}
size: 3

get(-5) → -1
get(0) → 1
get(3) → -1
```

`public int index(int x)` (8 points):

This method returns the index of the first element that matches input `x`. If the element is not found, then return -1.

```
array: {1, 2, 1, 3, 0, 0}
size: 3

index(1) → 0
index(3) → -1
```

Part 2: Implement `My2DList.java` (40 points)

In class `My2DList` of the starter code, 3 methods are already implemented for you: `My2DList()`, `getArray()`, `setArray()`. **Those methods are used by the Gradescope Autograder. Therefore, you should NOT modify any of these methods!**

There are 5 methods you need to implement yourself, including: `My2DList(int nRow, int nColumn)`, `horizontalFlip()`, `verticalFlip()`, `transpose()`, `rotate()`. You should remove any `TODO` comments and implement those methods.

There is only one member variable: `array`. The variable `array` is a 2-dimensional integer array storing the data.

`public My2DList(int nRow, int nColumn)` (8 points):

This constructor initializes the `array` member variable by creating a new 2-dimensional array with `nRow` and `nColumn`. For example, if `nRow` is 5 and `nColumn` is 3, then the `array` is a 5 by 3 matrix.

`public void horizontalFlip()` (8 points):

This method flips the `array` horizontally. You may assume the `array` object is not `null`. An example is shown below:

```
{{1, 2, 3},          {{3, 2, 1},
 {4, 5, 6},     →     {6, 5, 4},
 {7, 8, 9}}           {9, 8, 7}}
```

`public void verticalFlip()` (8 points):

This method flips the `array` vertically. You may assume the `array` object is not `null`. An example is shown below:

```
{{1, 2, 3},          {{7, 8, 9},
 {4, 5, 6},     →     {4, 5, 6},
 {7, 8, 9}}           {1, 2, 3}}
```

`public void transpose()` (8 points):

This method transposes the `array`. You may assume the `array` object is not `null`. An example is shown below:

```
{{1, 2, 3},           {{1, 4},
 {4, 5, 6}}     →      {2, 5},
                       {3, 6}}
```

`public void rotate()` (8 points):

This method rotates the `array` 90-degrees clockwise. You may assume the `array` object is not `null`. An example is shown below:

```
{{1, 2, 3},           {{4, 1},
 {4, 5, 6}}     →      {5, 2},
                       {6, 3}}
```

Part 3: Complete `unitTests` (10 points)

Typically, A good programmer will write two testers for each method he/she writes to ensure correctness. The more thorough the testers you write, the better scores you will likely achieve. To receive full points in this part, you need to add at least two testers for checking two different methods. You are allowed to use magic numbers inside your testers ONLY.

Submission

VERY IMPORTANT: Please follow the instructions below carefully and make the exact submission format.

1. Go to Gradescope and click on `PA3`.
2. Click the DRAG & DROP section and directly select the required files (`MyList.java, My2DList.java, Assignment3.java`). Drag & drop is fine. **Please make sure you don't submit a zip. Make sure the filenames are correct.**
3. **You can resubmit an unlimited number of times before the due date.** Your score will depend on your final submission, even if your former submissions have a higher score.
4. The autograder is for grading your uploaded files automatically. Make sure your code can compile on Gradescope.

NOTE: The Gradescope Autograder you see is a minimal autograder. For this particular assignment, it will only show the compilation results. After the assignment deadline, a thorough Autograder will be used to determine the final grade of the assignment. **Thus, to ensure that you would receive full points from the thorough Autograder, it is your job to extensively test your code for correctness via `unitTests`.**

5. Your submission should look like the screenshot below. **If you have any questions, then feel free to post on Piazza!**