CPSC250L Lab 2
2D Arrays and `ArrayLists`

Spring 2018

1 Introduction

In this lab you will use two-dimensional arrays and `ArrayLists` to handle data. In the first section, you will deal with matrices. In the second section, you will perform *set operations* that can tell you information about different data sets.

2 Exercises

2.1 Two-Dimensional Arrays

In this section, you will write methods that handle 2-dimensional arrays.

Exercise 1

Create a class named `TwoDArrayMethods` and implement the following method.

1. `public static int[][] transpose(int[][] anArray)`

This method receives a 2-dimensional array of integers and returns the transpose of that matrix. Suppose we are given an n by m matrix, A. Then the *tranpose* of that matrix, denoted A^T, is the m by n matrix with entries

$$(A^T)_{i,j} = (A)_{j,i}.$$

In English, the entry in the ith row and jth column of the transpose is the entry in the jth row and ith column of the original.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{pmatrix} \qquad A^T = \begin{pmatrix} 1 & 4 & 7 & 10 & 13 \\ 2 & 5 & 8 & 11 & 14 \\ 3 & 6 & 9 & 12 & 15 \end{pmatrix}$$

Hint: Do not change the original matrix! If you want to get the number of columns in a non-empty matrix, use `matrix[0].length`.

Test your code against `TwoDArrayMethodsTest.java`.

Exercise 1 Complete

Run:

```
git add .
git commit -m "Completed exercise 1"
git push origin master
```

2.2 `ArrayList`

In this section we will practice using `ArrayList`s. Before beginning this exercise find the JavaDoc for `ArrayList` and have it open in your web browser. Reference the JavaDoc for any questions you have about `ArrayList`.

Exercise 2

Create a called `ArrayListMethods` and implement the following methods

1. `public static void doReverse(ArrayList<Integer> list)`

   This method reverses the elements of an `ArrayList`. If your `ArrayList` is `{1,2,3,4}`, then the output of this method would be `{4,3,2,1}`.

2.
   ```
   public static ArrayList<Integer> getUnion(
                               ArrayList<Integer> one, ArrayList<Integer> two)
   ```
   This method returns the *union* of the lists `one` and `two`. The union of two sets, A and B, is defined as

   $$A \cup B := \{x \text{ such that } x \text{ is in } A \text{ or } x \text{ is in } B\}.$$

   In English, the union of A and B is the set that contains each element of A and each element of B. Note that the union is a *set* and thus the order of the elements in it do not matter. **Additionally, no element should be listed in the union more than once.**

   For example, $\{1, 2, 3, 4\} \cup \{0, 4, 7\} = \{0, 1, 2, 3, 4, 7\}$ and $\{1, 2, 3\} \cup \{1, 2\} = \{1, 2, 3\}$.

3.
   ```
   public static ArrayList<Double> getIntersection(
                               ArrayList<Double> one, ArrayList<Double> two)
   ```
   This method returns the *intersection* of the lists `one` and `two`. The intersection of two sets, A and B, is defined as

   $$A \cap B := \{x \text{ such that } x \text{ is in } A \text{ and } x \text{ is in } B\}.$$

The intersection of A and B is the set of each element that is in **both** A and B. Note that the intersection is a set and thus the order of the elements in it do not matter. **Additionally, no element should be listed in the intersection more than once.**

For example,

- $\{1, 2, 3, 4\} \cap \{0, 4, 7\} = \{4\}$
- $\{1, 2, 3\} \cap \{1, 2\} = \{1, 2\}$
- $\{1, 2\} \cap \{3, 4\} = \{\}$

4.
```
public static ArrayList<String> getDifference(
                        ArrayList<String> one, ArrayList<String> two)
```

This method returns the *difference* of the lists `one` and `two`. The difference of two sets, A and B, is defined as

$$A \triangle B := \{x \text{ such that } (x \text{ is in } A \text{ and } x \text{ is not in } B) \text{ or } (x \text{ is in } B \text{ and } x \text{ is not in } A)\}.$$

The difference of A and B is the set of each element that is in either A or B but not both A and B. Note that the difference is a set and thus the order of the elements in it do not matter. **Additionally, no element should be listed in the difference more than once.**

For example,

- $\{1, 2, 3\} \triangle \{2, 3, 4\} = \{1, 4\}$
- $\{1, 2, 3\} \triangle \{1, 2\} = \{1\}$
- $\{1, 2, 3\} \triangle \{1, 2, 3\} = \{\}$, the empty set.

Test Your Code

Test your code against `ArrayListMethodsTest.java`. Correct any compilation errors and ensure that your code passes the JUnit tests.

Exercise 2 Complete

Run:

```
git add .
git commit -m "Completed exercise 2"
git push origin master
```

---

3 Common Mistakes

Some solutions to common mistakes for this lab are as follows.

1. Be sure to ensure that any array or `ArrayList` that you deal with is not `null`! If you try to do an operation (other than assignment) on a `null` reference, then your program will throw a `NullPointerException` and crash.
2. 2D arrays are indexed like `array[row][column]`.
3. You can get the number of columns of a 2D array by looking at `array[0].length`.
4. When using `ArrayList`, you still need to be cautious about going out of bounds. You can get the size of `ArrayList<T> arr`, by calling `arr.size();`.
5. When creating a list that represents a set, the order of the objects in the list does not matter. However each object should only be in there **once**!