CS 101 - Algorithms & Programming I

Fall 2024 - Lab 7

Due: Week of November 25, 2024

*Remember the **honor code** for your programming assignments.*
*For all labs, your solutions must conform to the CS101 style **guidelines**!*
All data and results should be stored in variables (or constants where appropriate) with meaningful names.

The objective of this lab is to learn how to use Arrays and ArrayLists. An array is a container object that holds a **fixed** number of values of a single type. When you need an array-like data structure that offers **dynamic resizing**, you would usually use an ArrayList. An ArrayList is an array that resizes itself as needed. **You must carefully choose one or the other based on these criteria in each case for this lab.**

Remember that analyzing your problems and designing them on a piece of paper *before* starting implementation/coding is always a best practice.

0. Setup Workspace

Start VSC and open the previously created workspace named `labs_ws`. Now, under the `labs` folder, create a new folder named `lab7`.

In this lab, you are to have four Java classes/files (under `labs/lab7` folder) as described below. A fifth Java file containing the revision should go under this folder as well. We expect you to submit 5 files, including the revision, *without* compressing them. Do *not* upload other/previous lab solutions in your submission. Outputs of sample runs are shown in **brown** whereas the user inputs are shown in **blue**.

1. Matrix Operations

Create a Java file named `Lab07_Q1.java` with the following methods. In this question, you will implement a program with static methods that will compute the transpose of a matrix, whether a matrix is symmetric or not, and whether the sum of the main and secondary diagonal of a matrix are equal or not. Please refer to this page for the definition of the concepts here.

Implement the following methods:

- **`int [][] multiplyMatrices(int [][] matrixOne, int [][] matrixTwo)`:** Takes two matrices and returns the multiplication of `matrixOne` and `matrixTwo`. Ensure that matrix dimensions are suitable for multiplication (the number of columns of the first matrix must equal the number of rows of the second matrix). This means that if `matrixOne` and `matrixTwo` have the dimensionalities of $i \times j$ and $j \times k$, respectively, the resulting matrix's dimensionality should be $i \times k$.

- **`String checkRectangular(int [][] matrix)`:** If the matrix is square, check if it's either a lower triangular matrix (all elements above the diagonal are zero) or an upper triangular matrix (all elements below the diagonal are zero). Return "Lower Triangular", "Upper Triangular", or "Neither" based on the matrix structure.

- **`boolean isSubset(int [][] parent, int [][] child):`** Takes two matrices and returns a boolean. It returns true if the child matrix is a subset of the parent matrix.

- **`void printMatrix(int [][] matrix):`** Takes an integer matrix as input and displays it so that each row is printed on a separate line, and columns should be aligned. Example output can be seen in the sample run below.

Sample run:

```
int[][] matrixOne = {
    {1, 2, 3},
    {4, 5, 6}
};

int[][] matrixTwo = {
    {7, 8},
    {9, 10},
    {11, 12}
};

int[][] matrixSquare = {
    {1, 0, 0},
    {2, 1, 0},
    {3, 4, 1}
};

int[][] childMatrix = {
    {1, 5},
    {8, 9}
};

int[][] parentMatrix = {
    {1, 0, 2, 3},
    {4, 1, 5, 6},
    {7, 8, 9, 10}
};

System.out.println("Matrix One:");
printMatrix(matrixOne);
System.out.println("Matrix Two:");
printMatrix(matrixTwo);

System.out.println("Multiplication Result:");
int[][] result = multiplyMatrices(matrixOne, matrixTwo);
printMatrix(result);

System.out.println("Checking if matrix is rectangular:");
```

```
printMatrix(matrixSquare);
String rectangularResult = checkRectangular(matrixSquare);
System.out.println("Result: " + rectangularResult);

System.out.println("Checking if child matrix is a subset of parent matrix.");
System.out.println("Parent:");
printMatrix(parentMatrix);
System.out.println("Child:");
printMatrix(childMatrix);
boolean isSubsetResult = isSubset(parentMatrix, childMatrix);
System.out.println("Is child matrix a subset of parent matrix? " + isSubsetResult);
```

```
Matrix One:
1 2 3
4 5 6
Matrix Two:
 7  8
 9 10
11 12
Multiplication Result:
 58  64
139 154
Checking if matrix is rectangular:
1 0 0
2 1 0
3 4 1
Result: Lower Triangular
Checking if child matrix is a subset of parent matrix.
Parent:
 1  0  2  3
 4  1  5  6
 7  8  9 10
Child:
1 5
8 9
Is child matrix a subset of parent matrix? true
```

2. The Bakery

Your next task is to create a new/empty file `Lab7_Q2.java` and implement the required functionality for a bakery. A popular bakery in Ankara that makes bread deals with a large number of customers daily. To reduce the crowd in the bakery, the owners completed a survey, which revealed that most customers want just one loaf of bread. However, these customers must wait in a queue because people purchase more than one loaf. This causes inconvenience for everyone at the store.

The bakery owners came up with an intuitive idea; they decided to take customers into the store in batches of 20 and serve one customer, who ordered a single loaf of bread, after serving one customer who purchased more than one loaf of bread. In addition, when the bakery is ready to bake, to save its customers' time, the shopkeepers dismiss the people who are not expected to get any bread.

To fulfill their idea, you are asked to implement a program by following the instructions below:

You should have an array to save the state of the current queue. As the available space in the store is limited, they can only host 20 people at a time.

Implement the following methods:

- **`void joinQueue(int orderedLoafs, int[] queue)`:** Takes the ordered number of loaves and the current queue. It checks the queue, making sure there is an available slot. If there is no available slot, it rejects the order and displays a message (check the sample output). Otherwise, it shows the current status of the queue using the number of orders in the queue.

- **`void bake(int expectedLoaves, int[] queue)`:** This function takes the number of loaves expected from the current batch of dough and the current queue. Using this number, it displays the dismissed people who will not get any bread and the current status of the queue.

Sample run:

```
int [] queue = new int[20];

System.out.println("---- Testing joinQueue ----");

joinQueue(3, queue);
joinQueue(2, queue);
joinQueue(4, queue);
joinQueue(1, queue);
joinQueue(2, queue);
joinQueue(2, queue);
joinQueue(1, queue);
joinQueue(1, queue);

System.out.println("\n---- Testing full queue ----");
for (int i = 0; i < 12; i++) {
    joinQueue(2, queue);
}
joinQueue(1, queue);

System.out.println("\n---- Testing Bake First ----");
bake(8, queue);

System.out.println("\n---- Testing Bake Second ----");
queue = new int[20];
queueSize = 0;
bake(8, queue);
```

```
---- Testing joinQueue ----
Current Queue: 3
Current Queue: 3 2
Current Queue: 3 2 4
Current Queue: 3 1 2 4
Current Queue: 3 1 2 4 2
Current Queue: 3 1 2 4 2 2
Current Queue: 3 1 2 1 4 2 2
Current Queue: 3 1 2 1 4 1 2 2

---- Testing full queue ----
```

```
Current Queue: 3 1 2 1 4 1 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2 2 2 2 2 2
Current Queue: 3 1 2 1 4 1 2 2 2 2 2 2 2 2 2 2 2 2 2
Queue is full. Order rejected.

---- Testing Bake First ----
Not enough bread. Dismissing customers.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 2 loaves dismissed.
Customer with order 1 loaves dismissed.
Customer with order 4 loaves dismissed.
Current Queue: 3 1 2 1
All customers will receive their bread.


---- Testing Bake Second ----
Current Queue: Empty
```

3. Movie overview and rating system

Your next task is to create a new/empty file `Lab7_Q3.java` and implement the required functionalities for a movie rating and overview system, as explained below. This program should allow users to add and remove movies, submit reviews and ratings, and view a well-formatted list of all movies with reviews and the average rating.

This program should have the following features:

1. **Menu of Options:** At the start of the program, display the following menu:

```
Movie Review System:
1. Add a new movie
2. Remove a movie
3. Submit a review and rating for a movie
4. View all movies, reviews, and average ratings
5. Exit
Enter your choice:
```

2. **Add a New Movie:**
    - The user can add a new movie title. The review list should be initialized to store multiple reviews for the movie.
    - Ratings should be initialized to zero.

3. **Remove a Movie**: The user can remove a movie by selecting it from a list of titles. The program should display the list of movies before selecting which movie to remove.

4. **Submit a Review and Rating**:
    - The user can submit a review and rating for a movie. The program will display the list of movies and ask for a review and rating. Multiple reviews can be stored for each movie.

5. **View All Movies**:
    - The program will display the list of all movies with their reviews and the average rating in a well-formatted manner. If a movie has no reviews or ratings yet, display "No reviews yet" and "Rating: N/A" respectively.

6. **Exit**: The user can exit the program at any time.

Hint: You will use multiple arrays or array lists (choose whichever is more appropriate) to manage different properties of the movie review system. One array list will store the movie titles, while an array list of array lists will store multiple reviews for each movie. Additionally, you will need an array list to store the total sum of ratings for each movie, and another array to keep track of the number of reviews/ratings submitted for each movie, which will help you calculate the average rating.

Sample run:

```
Movie Review System:
1. Add a new movie
2. Remove a movie
3. Submit a review and rating for a movie
4. View all movies, reviews, and average ratings
5. Exit
Enter your choice: 1

Enter the movie title: Inception
Movie added!

Movie Review System:
1. Add a new movie
2. Remove a movie
3. Submit a review and rating for a movie
4. View all movies, reviews, and average ratings
5. Exit
Enter your choice: 3

Select a movie to review:
1. Inception
Enter movie number: 1
Enter your review: Amazing visuals and a complex plot.
Enter your rating (1-5): 5
Review and rating submitted!

Movie Review System:
1. Add a new movie
```

```
2. Remove a movie
3. Submit a review and rating for a movie
4. View all movies, reviews, and average ratings
5. Exit
Enter your choice: 3

Select a movie to review:
1. Inception
Enter movie number: 1
Enter your review: It is trash. Don't watch.
Enter your rating (1-5): 1
Review and rating submitted!

Movie Review System:
1. Add a new movie
2. Remove a movie
3. Submit a review and rating for a movie
4. View all movies, reviews, and average ratings
5. Exit
Enter your choice: 4

Movie List:
1. Inception
   Average Rating: 3.0
   Reviews:
   - Amazing visuals and a complex plot.
   - It is trash. Don't watch.

Movie Review System:
1. Add a new movie
2. Remove a movie
3. Submit a review and rating for a movie
4. View all movies, reviews, and average ratings
5. Exit
Enter your choice: 2

Select a movie to remove:
1. Inception
Enter movie number: 6
The entered number is wrong.

Select a movie to remove:
1. Inception
Enter movie number: 1
Inception has been removed.
```

You could use the values provided in sample runs to test your class. Keep in mind that your implementation must be compatible with the provided interfaces. For the desired outputs, you need to stay consistent with the details described in this document.