CSE8B - Final Exam (Part 2) - WI22

Due: March 15th, 2022 (Tuesday) at 8am

Hello everyone! Welcome to Part 2 for the Final Exam for CSE 8B WI22.

This document details a take-home exam that you will complete over the next few days. You can NOT communicate with anyone about the content of the assignment until you receive your grade. You can message us privately on Piazza, but the course staff will not give programming advice or answer most questions about the problems. If you have technical trouble creating a screencast (detailed below), then please reach out for assistance as soon as possible.

Do not use any online service other than Piazza to ask questions about the assignment. Do not search for, solicit, or use solutions to the problems that you find elsewhere for the exam. These are all violations of academic integrity that students have committed on exams like this in the past.

You can make use of any course notes, online resources about Java and its libraries, Java tools, and so on to complete the exam, including reusing code from class notes.

Outline of Part 2

Part 2 will be worth 100 points and will be split up into 3 tasks as shown below:

Task 1 - Coding (30 points)
Task 2 - Coding (40 points)
Task 3 - Video (30 points)

Submission Checklist

- `Matrix.java` containing all methods for Task 1
- `Hotel.java` containing all methods for Task 2
- `Room.java` containing all methods for Task 2
- `explanation.mp4` (or another compatible video extension) that has your recording for Task 3

Tasks

For Exam 2, you will have 3 tasks to complete.

If you haven't already, please go through the Autograder guide before you start your exam.

NOTE 1: You do not need to worry about style guidelines for Exam 2.

NOTE 2: You should not have to use any extra imports than what is already provided. If you have a problem with the Gradescope Autograder, then you should check your import statements (as well as your own compiler errors).

Task 1

For Task 1, you will be working with matrices represented using **2-dimensional integer arrays** alongside constructor overloading and file parsing. Matrices are rectangular arrays of numbers arranged in a fixed number of rows and columns. Since these arrays will be rectangular, that implies that the number of rows and columns are **not** guaranteed to be equal.

Please download the starter code from here. Ensure that there are two files: **`Matrix.java`** and **`MatrixTester.java`**. **You will be implementing several methods within `Matrix.java` and you will be testing those methods in `MatrixTester.java`.** For Task 1, you ONLY need to submit `Matrix.java`.

`MatrixTester.java`

Within `MatrixTester.java` are a few test cases for the Matrix class. **As you are implementing `Matrix.java`, you should test thoroughly to make sure that your program is functioning as expected.** We will not be collecting `MatrixTester.java`.

`Matrix.java`

Within `Matrix.java`, there is 1 *private* member variable given to you:

- `private int[][] myMatrix` - a 2-dimensional rectangular integer array.

NOTE: You can assume that `myMatrix` (and its inner rows) will never be `null`, and `myMatrix` will only contain integers between 0 and 1000 (inclusive).

Take note of the declared type of `myMatrix`. Do NOT change the type.

We also provide you with a single getter method. Please read and understand the method.

Finally, we provide you with a method called `findTransposeWrong`. You can use `findTransposeWrong` as inspiration for one of the methods in Task 1, but (more importantly) you will also be using `findTransposeWrong` in Task 3 (which will be discussed later). Do NOT modify `findTransposeWrong`.

For the rest of `Matrix.java`, you will need to implement the 3 following methods. DO NOT MODIFY ANY METHOD DECLARATIONS:

1. `public Matrix(int[][] matrixIn)`

   Within this constructor, you need to perform a **deep copy** of `matrixIn` into the member variable `myMatrix`. Use what you have learned in this course to perform a **deep copy**.

   NOTE: For this constructor (and all other constructors that involve deep copies), we will be testing for shallow copies and deducting points for not performing a deep copy.

2. `public Matrix(String inputPath) throws IOException`

   Within this constructor, you will be reading in the text from the file `inputPath` to initialize `myMatrix`. **We provide two input files for you (`input1` and `input2`), but you should create more for further testing.**

   The input file will have the following format:

   1. The first line of the file will always contain two numbers that are separated by a single space.
      a. The first number will represent the number of rows in the array, while the second number will represent the number of columns in the array. These numbers will not necessarily be equal (*i.e.* the array could be a rectangle)**.** You can also assume that these numbers will always be positive and will always exist.

   2. The rest of the lines of the file is the array:
      a. **Each non-space character will always be a valid integer.** No other characters will be in the file.
      b. Between each column, there will be a single space. Every row is separated with a `'\n'` (a new-line character).
      c. There are no leftover blank lines in the input file (*i.e.* no trailing whitespace). There are no extra spaces between rows or between columns. There are no extra spaces before nor after each line.

   Using what you have learned in this course, read the input file, and convert the input file into a 2-dimensional integer array to initialize `myMatrix`. If you use a Scanner object, then DO NOT forget to close the Scanner for the input file after you finish reading the file.

   NOTE: You can assume that the file always exists, and that the file is properly formatted (*i.e.* no corrupt/bad inputs).

   HINT: This process of file processing is similar to previous programming assignments.

3. `public int findTranspose()`

This method should return the transpose of the matrix `myMatrix`. The transpose of a matrix is found by interchanging its rows into columns or columns into rows.

Consider an example, if a matrix is of size 2×3, it means it has 2 rows and 3 columns. While finding the transpose of a matrix, the elements in the first row of the given matrix are written in the first column of the new matrix. Similarly, the elements in the second row of the given matrix are written in the second column of the new matrix. Hence, the order of the new matrix becomes 3×2, as it has 3 rows and 2 columns. Let A be the input matrix. On interchanging the rows and columns of the given matrix, the transpose of matrix A is given as AT which is as follows:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}. \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

Complete the method to compute the transpose of `myMatrix` and return the transpose. **HINT:** You can use `findTransposeWrong` as inspiration (although there are errors (more than 1) with that code).

Task 2

For Task 2, you will be working on two classes that represent a hotel with a list of rooms.

Please download the starter code from here. Ensure that there are three files: `Room.java`, `Hotel.java`, and `HotelTester.java`. **You will be implementing several methods within** `Room.java` and `Hotel.java`**, and you will be testing those methods in** **`HotelTester.java`.** For Task 2, you need to submit `Hotel.java` and `Room.java`.

`HotelTester.java`

Within `HotelTester.java` are a few test cases for the Hotel and Room class. **As you are implementing** `Room.java` and `Hotel.java`**, you should test thoroughly to make sure that your program is functioning as expected.** We will not be collecting `HotelTester.java`.

`Room.java`

Within `Room.java`, there are 3 *private* member variables given to you:

- `private int roomNumber` - represents a unique room number

- `private String[] guestNames` - list of guest names residing in the room if booked, should be an empty list if room is available
- `private boolean isAvailable` - parameter set to true if room is available (not booked) else false (booked)

NOTE: You can assume that `roomNumber` will always be a positive number, but `guestNames` could be an empty array.

Take note of their declared types. Do NOT change any of their types.

We also provide you with three getter methods and two setter methods. Please read and understand the methods. The setter methods will come to use in implementing the `bookRoom()` and `bookRoom(int roomNumber)` methods.

For the rest of `Room.java`, you will need to implement the 3 following constructors. DO NOT MODIFY ANY CONSTRUCTOR/METHOD DECLARATIONS:

1. `public Room(int roomNumber)`
   Within this constructor, you need to create a free room with the given `roomNumber`. Since the room should be free, the list of `guestNames` should be initialized to be an empty array. Set `isAvailable` accordingly. (When creating a new Room object, since the room is empty, `isAvailable` should be default to True)

2. `public Room(int roomNumber, String[] guestNames, boolean isAvailable)`
   Within this constructor, you need to perform a **deep copy** of the parameters into `this` (current instance's) respective member variables. Use what you have learned in this course to perform a **deep copy**.

3. `public Room(Room room)`
   Within this *copy* constructor, you need to perform a **deep copy** of the member variables from the parameter `room` into `this` (current instance's) respective member variables. Use what you have learned in this course to perform a **deep copy**.

`Hotel.java`

Within `Hotel.java`, there are 2 *private* member variables given to you:

- **`private String name`** - the name of the hotel
- **`private Room[] rooms`** - an array of rooms that belong to the hotel

NOTE: You can assume that `name` will always be a valid String.

Take note of their declared types. Do NOT change any of their types.

We also provide you with two getter methods. Please read and understand the methods.

For the rest of `Hotel.java`, you will need to implement the 3 following methods. DO NOT MODIFY ANY METHOD DECLARATIONS:

1. `public Hotel(String name, Room[] rooms)`
   Within this constructor, you need to perform a **deep copy** of the parameter's `name` and `rooms` into their respective member variables. Use what you have learned in this course to perform a **deep copy**.

2. `public int bookRoom(String[] guestNames)`
   This method should book the first available room in the list of rooms, returning the `roomNumber` of the room that was booked. If no room is available for booking, then the method should return `-1`. The `guestNames` argument should be used while booking to set the names of guests the room is being booked for . *Note that if the private member variable* `rooms` *array is empty, then this method should just return* `-1`.

3. `public boolean bookRoom(int roomNumber,String[] guestNames)`
   This method should book the room with the room number specified by the input parameter `roomNumber` if it's available and return `true`. The `guestNames` argument should be used while booking to set the names of guests the room is being booked for.

   If the room has been booked already, or the `roomNumber` is invalid (i.e. not a positive number or the room number does not exist in the array), then this method should return `false`. *Note that the* `roomNumber` ***does not*** *correspond to the index of the room in the* `rooms` *array.*

Task 3

For Task 3, you will record a short video of *no more than 5 minutes* that will answer the following three questions. Please read ALL of the instructions and guidelines below before recording your video.

1. Back in Task 1, we talked about how `findTransposeWrong()` has some errors. However, this function works for some matrices. **For this question, we want you to write, show, and run a test case that `findTransposeWrong()` would produce the CORRECT transpose value of a matrix.**
   a. To be specific, we are looking for the following:
      i. Within `MatrixTester.java`, *show* and describe your test case that will produce the correct output, then run your tester file.
      ii. *Show* the result that is shown in your command prompt or terminal.

iii. Show where and discuss ***why*** this test case works (*i.e.* we want you to *show* where the errors are in `findTransposeWrong()` and discuss ***why*** the errors didn't cause the code to break for this test case?).

2. Consider a test case with the following block of code inside `HotelTester.java`:

```
Room r1 = new Room(101, new String[] {"Benson", "Jack"},
false);
Room r2 = new Room(121, new String[] {"Prajwala",
"Sasya"},true);
Room r3 = new Room(90);
Room[] roomList = new Room[] {r1, r2, r3};
Hotel h1 = new Hotel("Marriott", roomList);
String[] guestList = new String[] {"Bao", "James"};
```

For this question, we want you to show AND describe what happens if you were to call `h1.bookRoom(guestList).`

a. To be specific, we are looking for the following:
   i. Within `HotelTester.java`, *show the lines of code* that are given to you above, alongside `h1.bookRoom(guestList).`
   ii. Inside `Hotel.java`, *show* and discuss **what** happens inside `bookRoom()`, explain **how** the output is produced. Does this output match what you expected? If not then **what is the correct expected output and how would you change the given test case to get the correct output**?
      1. If you do not have a properly implemented method, then you can describe what would happen to the best of your ability. However, we want to see your code no matter what.
   iii. *Show* the returned value of `h1.bookRoom(guestList)` before and after changing the given test case.

3. Consider the following 3 lines of code inside `HotelTester.java`:

```
Room r1 = new Room(101, new String[] {"Benson", "Jack"},
false);
Room r2 = new Room(121, new String[] {"Prajwala",
"Sasya"},true);
Room r3 = new Room(90);
```

For this question, we want you to *show* a memory model diagram that depicts *everything* in memory *AFTER* the 3 lines of code have finished running. We do not need to see any constructor stack frame(s), but we expect those 3 lines of code to exist within the `main` method stack frame.

a. To be specific, we are looking for the following:

i. Where is the memory allocated in the stack and in the heap for `r1`?
ii. Where is the memory allocated in the stack and in the heap for `r2`?
iii. Where is the memory allocated in the stack and in the heap for `r3`?

An example of a memory model diagram can be found at the 30:30 timestamp of A00's Lecture 5 Recording or on slide 6 of B00's Lecture 4. **You can draw the memory model diagram in whatever fashion you would like** - you can draw the diagram on a tablet, draw the diagram on paper then take a picture of the drawing, draw the diagram with a (free) drawing program like Google Drawings or GIMP, or you could just use Google Slides to create shapes and texts. **Whatever you do, we just need to see it in the video.**

Here are the required items to include in your video:

1. **For the first few seconds of the video, show ONLY your face and a picture ID** (your UCSD Student ID is highly preferred, but any picture ID with your full name will suffice). Afterwards, you do not have to be on camera, although it is fine to leave it on. This is simply to confirm that it is you recording your video and doing your work. If you do NOT have a webcam, then take a picture of yourself and your picture ID with your phone, and display that picture at the start of your recording.
2. For the rest of the video, answer the 3 questions above *in that exact order*.

Here are some *important* notes in regards to Task 3:

1. **If you do not provide a picture ID, then you will get a 0 on Exam 2 until you prove to us that it was actually you who recorded the video**.
2. Please be sure to stand as still as possible when showing yourself and your picture ID. If we cannot verify that it is you, then you will get a 0 on the exam until you prove to us that it was actually you who recorded the video.
3. Alongside your picture ID, if you do not show your code for Q1 and Q2, then you will also get a 0 on the exam until you prove to us that it was actually you who wrote your code.
4. Video must have sound! Furthermore, while explaining your memory model diagram, make sure to enunciate to the best of your ability. We must hear you clearly explain your diagram! Also, try to NOT use your webcam to show the memory model diagram - that makes your diagram too small to see.

Here is a short tutorial demonstrating how to make a screencast with Zoom: Screencast Tutorial

Submission Instructions

You will be submitting your answers directly to Gradescope under the assignment **"Final Exam - Part 2"**. As a final reminder, **YOU MUST PROVIDE YOUR FACE + PICTURE ID, OR ELSE YOU WILL NOT GET CREDIT FOR THIS EXAM.** The 4 files required for submission are:

- `Matrix.java` containing all methods for Task 1
- `Hotel.java` containing all methods for Task 2

- `Room.java` containing all methods for Task 2
- `explanation.mp4` (or another compatible video extension) that has your recording for Task 3

If you cannot upload your video to Gradescope:

Gradescope has a max file size of 100MB. Any files larger than that will be rejected. All files must be submitted to Gradescope; we will **not** accept videos (or code) through email or Piazza.

If your video is larger than 100MB, then here are some tips to ensure that your video can be uploaded to Gradescope:

- Use Zoom:
    - A 100MB video is about a 20 minute video on Zoom (1620x1080 video resolution).
    - If you are not using Zoom, then convert your file to an MP4. There are web tools that can do this. MP4 videos have smaller video sizes than other video formats.
- Use a smaller screen size for your computer, or only record a smaller portion of the screen.
- Reduce background clutter (*i.e.* desktop icons). Background clutter reduces compression, making larger file sizes. You can hide the background by maximizing your text editor's window.
- Keep your video short (*i.e.* under 5 minutes):
    - 5 minutes is the max, and it is certainly possible to cover everything in less time.
    - We recommend creating a script of what you are going to say.
    - Pause the video as you switch to the code for the next question (make sure to tell us which question you are showing the code for when you switch).
- Use a 3rd party video editor to reduce the width/height of the video, but make sure your code is not blurry when you play it on Gradescope as we cannot grade blurry videos.