Lab 04 Specification – A practical exercise to get started in Data Structures and integrating data structures in object-oriented programming.
50 points

Lab Goals

- Do a simple exercise on Multidimensional Arrays.
- Practice developing an Array of objects inside a program.
- Learn practically the limitations of Arrays.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

If you have not done so already, please read all of the relevant "GitHub Guides", available at the following website: `https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- **GT ch3: 3.1 & ch5: 5.1, 5.3.1, 5.4**

Assignment Details

Now that we have completed discussing important concepts related to object-oriented progamming and some basics on data structures together in the last few lectures, it is your turn. In this lab, you will practice a variety of programs to retain the knowledge of data structures and how to integrate data structures within object-oriented programming. This includes modifying one or more code files to implement a series of functionalities.

The lab has two parts. The first part is required to be completed by the end of the lab session. The second part is required to be completed within a one week time period. Please make sure to push your code file(s) and commit by the deadline provided in each of the following section(s).

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.

2. Submitting a copy of the other's program(s) is strictly not allowed. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as skill test, exams, etc · · ·

At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL's to clarify if there is any confusion related to the lab and/or class materials.

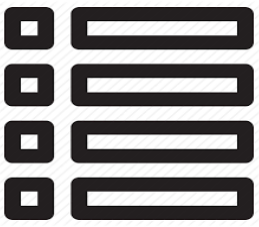
Section 1: In-Lab Exercise (25 points)

Attendance is in general a required component in computer science class(es) and laboratory session(s). By attending these sessions regularly, students get an opportunity to interact with the Professor and the Technical Leaders(s). This interaction helps maximize the student's learning experience. Attendance is taken in the form of Mastery Quizze(s) and by checking the commit log of the student's In-Lab exercise submission. No student is allowed to take the Mastery Quizze(s) outside the laboratory room unless prior arrangements are done with the Professor. It is against the class honor code to violate the policy outlined above.

Part 01 - Attendance Mastery Quiz (5 points)

The Mastery Quiz is graded based on a Credit/No-credit basis. Students are required to complete the Mastery Quiz only in the laboratory room, which is in **Alden 101** before 3:30 PM. Submissions after 3:30 PM is not acceptable. Students who had completed the Mastery Quiz according to the policy outlined above will automatically receive (5) points towards their lab grade. The link to the Mastery Quiz is provided below:
`https://forms.gle/rarPndEvBGxG1mDh8`

Part 02 - A Simple Exercise on Arrays (15 points)



Students are required to complete this part only in the laboratory room, which is in **Alden 101**. It is required to complete the final commit by the end of the laboratory session, which is before **4:20 PM** on the same day that the lab is given to the students. There is late submission accepted for this part, according to the late policy outlined in the course syllabus.

A two-dimensional array shares similar properties as a one-dimensional array in the context of homogeneous nature (all cells in the array need to be of the same data type!). However, an interesting addition to a two-dimensional array is that the array structure can contain more than one row and more than one column. This interesting addition allows a programmer to represent multidimensional data inside the structure.

The major goal in this part is to practice working with a two-dimensional array and also combine your understanding of one-dimensional array to implement a multiple-choice test grading tool.

In a recent class, we had discussed setting up a two-dimensional array, and looked at how an iterative block of code can be set up using a nested for loop in order to do the data processing. The grand idea behind the data processing is

to set up the outer for loop to iterate through each row and the inner for loop to iterate through each column in the array structure.

- **ADD** the following statement to all your code files including the file named `Grader.java`.

  This work is mine unless otherwise cited - Student Name

- The starter code is provided inside the lab repository in a file named, `Grader.java`. The starter code has detailed implementation already in place. It is only required to fill in the code inside the doGrading() method. Once the doGrading method is completed, the code will print the expected result.
- The detailed documentation of the starter code, that grades multiple choice tests, is provided below:

  1. eight students
  2. ten questions
  3. the answers are stored in a two-dimensional array
  4. each row in the two-dimensional array records a student's answers to the questions
  5. The key [correct answers] is stored in a one-dimensional array.

- For example, the array shown below stores the test:

Students' Answers to the Questions:

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

- The one-dimensional array shown below represent the key that includes the correct answers to the questions outlined above:

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

- Data Processing Outline: A critical component of implementing this part is to process both the 2D [test] and 1D [key] arrays and perform a series of comparisons to evaluate the number of correct answers, incorrect answers, computing the final score, and finally computing the class average. In your implementation, you may assume that student id is the same as the **index** value of the row representation in the test array.
- In order to process the data as required, the outline of the core logic is stated below:

1. The input arrays for both the 2D and 1D arrays by using an appropriate data type is already setup in the starter code.
2. Add the required logic in the doArray() method to iterate through each student's answer and compare the student answer with that of the correct answer provided in the key array.
3. If the answer matches, then increment the correct count
4. Else increment the incorrect count
5. Compute the class average by summing up the total score of all the students and by finding the average.
6. Finally output the total number of correct answers, incorrect answers, the total score of the student, and the class average based on user-provided student id. The output print logic is already provided in the starter code. Implementing the doGrading() method correctly is supposed to print the correct result.

- An expected sample output from your program execution should look similar to the output shown below:

  "Hello student, what is your student id? 4"
  "Congrats on taking the multiple choice test. You had received 8/10 with 8 correct, 2 incorrect answers and the class average is 63.75%"

- Input requirements:

  1. It is acceptable to make the assumption that both the 2D and 1D input arrays strictly contains char values.
  2. It is acceptable to directly hard code the content in your implementation from the arrays shown in the figure above. I don't expect you to perform any additional data validations in your code, as the input is hard coded in your program.

- To Think: There is no deliverable expected for this part of the lab. This is an opportunity for you to think through and come up with a design solution that involves identifying a suitable data structure using the **"Array of Objects"**. In the actual lab to-do part, you had used a two-dimensional array for the tests and a one-dimensional array for the keys. Additionally, I expect you to use one or more arrays internally in your code, to set up the requirements and the logic part of the lab. So now, maintaining a number of distinct arrays like that is not an efficient way of doing the coding. Instead, the efficient way is to use one solid data structure that internally holds different members including arrays and nicely encapsulates all the details from an external world. Think of a data structure (along with the lines of Student data structure we had developed in class) to meet all the requirements of the lab in a more elegant and efficient way.

Part 03 - Pair Discussion (5 points)

Students are required to complete this part only in the laboratory room, which is in **Alden 101**. It is required to complete the final commit by the end of the laboratory session, which is before **4:20 PM** on the same day that the lab is given to the students. There is no late submission accepted for this part unless prior arrangements are done with the Professor.

Pair up with at least one of your peers. Talk to your peer and discuss the following questions:

1. Let us suppose that you are creating an Array of objects? The object values can be assigned using a constructor approach or by using the setter approach. Explain the key difference between the constructor approach and the setter approach?
2. We discussed in class that Arrays.toString() is a utility function that lets you print the Array contents in the console. How is this approach different from setting up an iterative code to print the array contents by yourself? What are some limitations and advantages of using the utility function such as Arrays.toString()?

Each person in the pair should individually write a short summary to answer the questions above. Name the summary file as **pd-summary.pdf**. Here **pd** refers to pair discussion. In order to pass the submission requirements, the file should be generated using a PDF format.

ADD the following statement to all your code files including the file named `pd-summary.pdf`.

This work is mine unless otherwise cited - Student Name

After completing all the parts in Section 01, it is acceptable for students to start working on Section 2 remotely. Although it is strongly recommended to be in the laboratory room till the end of the lab session, and work on the lab along with other colleagues.

Section 2: Take Home Exercise (25 points)

Students are recommended to get started with this part in the laboratory room, by discussing ideas and clarifying with the Professor and the Technical Leader(s). All the part(s) in this section is due in **one week** time. It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. The deadline for submitting the part(s) in this section is **26th Feb 2020, 8:00 AM**. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

Part 01 - Developing a Shopping Cart application. (25 points)

In this part, you will develop a Shopping Cart application. The application mainly uses an array in conjunction with an object-oriented technique to store and process the items shopped by customers. To simplify the development process, a series of starter-code is provided in the repository, with the file named `Item.java`, `ShoppingCart.java`, and `Checkout.java`. Please make sure to follow the requirements outlined below:

1. **ADD** the following statement to all your code files including the file named `Item.java`, `ShoppingCart.java`, and `Checkout.java`.

   This work is mine unless otherwise cited - Student Name

2. Complete the Item class file. Add a constructor to initialize all the members of the class. Add the respective getters for all the three members in the class. The toString() method is provided in the stater-code to stringify the results to be displayed to the user based on the item properties.

3. Complete the ShoppingCart class by doing the following:

   - Add the constructor to the class and initialize all the members of the class. The capacity should be set to 5. Initialize the `cart` array to an array of Item of size = capacity. The idea behind this initialization is that the array of items is set to be of size 5 and will have its size increased once the capacity is reached.
   - The method `getTotalPrice()` is already implemented in the starter code. This method simply is a getter for the `totalPrice` member variable.
   - The method `displayItemsDuringCheckout` displays all the items in the cart and their respective information such as price, quantity, and name. This method makes a call to the toString() method in the Item class.

- The method `addItemToCart` is incomplete in the starter-code. Please add the required logic to add an item to the cart. In this process, it is first required to create an object for each individual item shopped. Once the object is created the current item should be printed on the console by invoking the toString() method in the Item class. Recall that an array of objects of type Item is already created in the earlier step using the variable named `cart`. After display the item details on the console, the variable `itemCount` and `totalPrice` should be incremented. This increment is done so to compute the `totalPrice` and the `itemCount` for every item shopped. If the `itemCount` reaches the capacity, then a call should be to the increaseCartSize() method.
- The method `increaseCartSize` method is incomplete in the starter-code. Please add the required logic in this method to take the `cart` array and make it bigger by increasing the size by 5 every time. The capacity was originally 5 and every time this method is called the `cart` size should increase by 5. How to do this? Create a new array called `temp` with its size as (capacity + 5). Move all the elements from the `cart` array to the `temp` array. Then reinitialize the `cart` array to the increased size. Note: at this point the `cart` array will be empty (because it is reinitialized) and now assign `cart = temp`. Finally, make sure to increment the capacity by adding 5 to it. In this way, the new capacity is 5 more than the old capacity value. This is a critical part of the lab. It is important to understand how to change the size of an array. An array is fixed in size. Hence, we have a huge limitation to add new elements to the array. Completing this part will ensure that a student will fully understand the concept of Arrays and an array of Objects.

4. The Checkout class is complete in the starter-code. The details related to the `Checkout` class is purposefully left out from this sheet. It is required to analyze the Checkout class in order to fully implement the other parts of this application. There are no code changes required to be done in the Checkout class. If all the incomplete code is implemented correctly, then the program should print the results.

5. A sample output is displayed below for your reference:

```
amohan@ALDENV8075 classes % java Checkout
Welcome to the online shopping portal
------------------------------------------------
Tell us what did you shop?
Enter item name:
mic
Enter item price:
10
Enter item quantity:
2
------------------------------------------------
Total price so far:20.0
------------------------------------------------
Do you want to checkout? (y/n)
y
------------------------------------------------
mic     $10.00  2       $20.00
------------------------------------------------
Please pay:20.0
------------------------------------------------
```

Submission Details

For this assignment, please submit the following to your GitHub repository before the **deadline** by using the link shared to you by the Professor:

1. Commented source code from the “Item”, “ShoppingCart”, and “Checkout” program.
2. Commented source code from the “Grader” program.
3. A document containing the peer discussion points, named `pd-summary.pdf`.
4. It is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllbus. Make sure to add the statement ”This work is mine, unless otherwise cited.” in all your deliverables such as source code and PDF files. In your peer discussion summary file, please make sure to include your name and your partner’s name.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits will be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your git repo will lead you to receive no points given for the lab submission. In this case, there is no solid base to grade the work.
3. There will be no partial credit awarded if your code doesn’t compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student’s responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student’s submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.
4. If you need any clarification on your lab grade, talk to the Professor. The lab grade may be changed if deemed appropriate.