CS234 Computer Science II
Lab 5
Total points: 100

Read the instructions carefully.

You will practice the use of **2D arrays**. By using basic 2D arrays you can simulate a database for your programs.

For example, you can use a 2D arrays to store information about students in two different "tables".

Personal Information

001	Eduardo	06/25/1977
002	Emma	12/12/2013
003	John	07/15/1983

Academic Information

001	CS234	4.0
002	CS123	3.5
003	ESL101	3.8

And then, use them to show the student's information:
001, Eduardo, 06/25/1977, CS234, 4.0

First, I'll explain the general functionality, then I will show the required methods, and finally I will show some examples of how the program must work.

Write a **Java program to manage** (add, display, update, and delete) the information for students using 2D arrays.

The program needs to present a menu to the user. The menu must have five options:

Option 1 is for adding a new student's information in the database (2 tables). The program needs to check if the student ID does not exist already in the dataset. If it exists, then the program shows a message to the user. Else, it needs to ask for the student's information.

Option 2 is for showing the user information. The program asks for the student ID and then check if the student ID exists. If it does not exist, then the program shows a message to the user. Else, it needs to show the student's information.

Option 3 is for updating a student's information. The ID of the student is needed as input. If does not exist, then the program shows a message to the user. Else, it needs to ask for the new student's information to be updated.

Option 4 is for deleting a student from the database. The program needs to check if the student ID to remove exists. Then, it needs to remove that row from the database. The program simulates the deletion *by creating an empty array* (remember, a 2D array is an array of arrays) in the position of the previous content.
If the ID does not exist, your program should send a message to the user.

Option 5 is used to exit the program.
The ***only way*** for the program to finish is by using this option. Otherwise, the ***program continues*** *with its execution* by showing the menu to the user for an option.

If the user inputs an *incorrect* option for the menu, the program must show a message for the invalid option.

Your program **must implement** the following:

- A 2D array of ***Strings*** to store the student Information
- A 2D array of ***String*** to store the academic Information
- The number of **rows** for **both** arrays is **100**
- The number of **columns** for **both** arrays is **3**
- The 2D arrays should be defined as **static**. This will help you to access them anywhere. I used a static variable in my 02/23 lecture.

Do not use ArrayLists. They must be basic 2D arrays (arrays of arrays).

Your program **must implement** the following methods (besides the main method):
No additional methods are allowed.

- The **main** method handles the Menu

```
/**
 * This is the method that intiates the addition of a new student.
 * It avoids duplicate entries.
 * The entered information is stored in the 'studentInfo' and 'academicRecords' arrays at the first
 * available empty slot.
 * The method needs to check for duplicate student IDs.
 * It also checks if the arrays are full or not.
 *
 * The 'findStudentIndex' and 'findEmptySlot' methods are called internally for checking existing
 *   student IDs and finding available slots, respectively.
 * If the student ID does not exist and there is an empty slot, then the method
 * gets the information for a student.
 * After getting all the information for the new student, the method calls the addStudent method to
 * add the information to the arrays.
 * It prints a success message after successfully adding the student.
 */
public static void option1() {
```

```
/**
* This method directly sets the information for a new student at the specified index in the arrays.
* It creates a new 1D array with the student info to be assigned to the corresponding array index
* It creates a new 1D array with the academic records to be assigned to the corresponding array index
*/
public static void addStudent(int index, String studentID, String studentName,
                                String dateOfBirth, String course, String gpa)
```

```java
/**
 * This method tries to locate an index in the 'studentInfo' array where the student ID is null,
 * indicating an empty slot available for storing new student information.
 * Call this method when adding a new student using the 'addStudent' method to find an
 * available slot for storing student details.
 *
 * The method returns the index of the first empty slot found. If no empty slots are found,
 * it returns -1 to indicate that the student database is full.
 */
public static int findEmptySlot() {
```

```java
/**
 * This method prompts the user to input a Student ID, searches for the corresponding
 * student in the arrays, and calls the method that displays their information.
 * This method calls the 'findStudentIndex' method to locate the index of the provided Student ID.
 * If the student id is found, then it calls the 'displayStudentInformation' method.
 * If the Student ID is not found, the program shows a message indicating that the student has
 * not been found
 */
public static void option2() {
```

```java
/**
 * This method prints the Student ID, name, date of birth, academic course, and GPA
 * for the student located at the specified index in the arrays.
 *
 * The method expects a valid index to be provided. It directly prints the information
 *     for the student at the specified index in the arrays.
 */
public static void displayStudentInformation(int index)
```

```java
/** Option 3 on the menu calls directly to this method (There is no option3() method)
 * This method gets a Student ID, searches for the corresponding
 * student, and allows the user to update the student's name,
 * date of birth, academic course, and GPA. The Srudent ID must not
 * be changed.
 *
 * This method uses the 'findStudentIndex' method to locate the index of the provided Student ID.
 * If the Student ID is found, the user can enter new information for the student.
 * To update the information, it uses the 'addStudent' method, passing the new values.
 * If the Student ID is not found, a message should be displayed.
 */
public static void updateStudentInformation() {
```

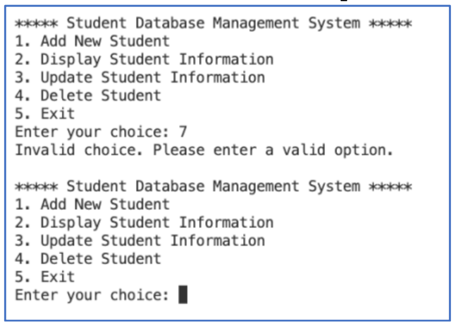
```java
/**
 * This method gets a Student ID, searches for the corresponding
 * student the arrays
 *
 * This method calls the 'findStudentIndex' to locate the index of the provided Student ID.
 * If the Student ID is found, the method 'deleteStudent' is called to
 * deleted the student information from both arrays by replacing the information in the index
 *  with new empty arrays.
 * If the Student ID is not found, a message is presented.
 */
public static void option4() {
```

```java
/**
 * This method removes the information for the student at the specified index
 * in the 'studentInfo' and 'academicRecords' arrays by replacing it with new empty arrays.
 * The method expects a valid index to be provided. It directly deletes the information
 * for the student at the specified index in the arrays by replacing it with new empty arrays.
 */
public static void deleteStudent(int index)
```

```java
/**
 * This method iterates through an array to locate the index of a student whose ID matches
 * the specified Student ID. It returns the index if found; otherwise, it returns -1.
 */
public static int findStudentIndex(String studentID) {
```

Examples

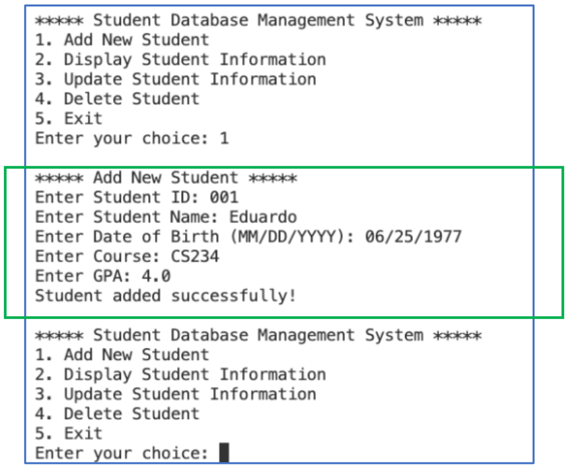
1. Menu and invalid option



```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 7
Invalid choice. Please enter a valid option.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice:
```

2. Option 1. Add New Student (non-existing ID)

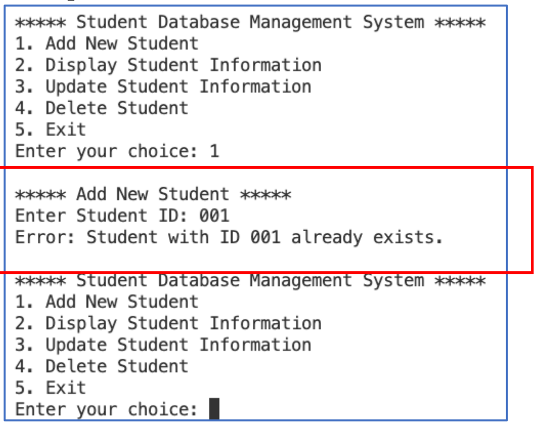


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 1

***** Add New Student *****
Enter Student ID: 001
Enter Student Name: Eduardo
Enter Date of Birth (MM/DD/YYYY): 06/25/1977
Enter Course: CS234
Enter GPA: 4.0
Student added successfully!

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice:
```

3. Option 1. Add New Student (existing ID)

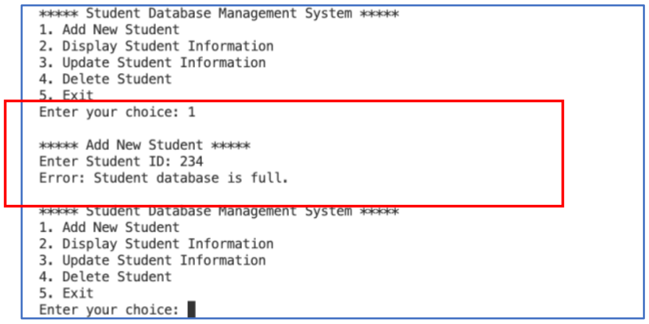


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 1

***** Add New Student *****
Enter Student ID: 001
Error: Student with ID 001 already exists.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: █
```

4. Option 1. Add New Student (array is full)

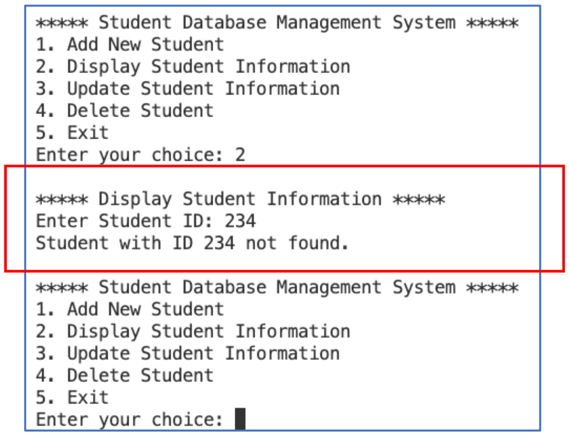


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 1

***** Add New Student *****
Enter Student ID: 234
Error: Student database is full.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: █
```

5. Option 2. Display Student Information (non-existing ID)

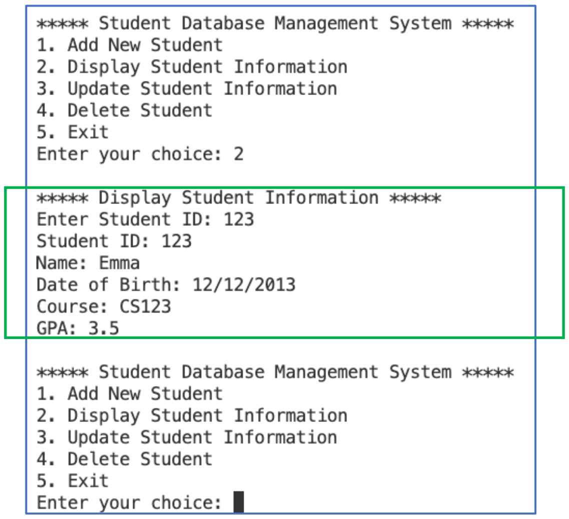


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 2

***** Display Student Information *****
Enter Student ID: 234
Student with ID 234 not found.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: █
```

6. Option 2. Display Student Information (existing ID)

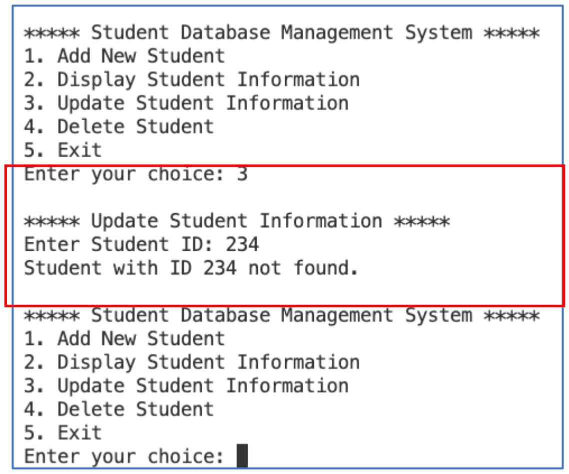


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 2

***** Display Student Information *****
Enter Student ID: 123
Student ID: 123
Name: Emma
Date of Birth: 12/12/2013
Course: CS123
GPA: 3.5

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice:
```

7. Option 3. Update Student Information (non-existing ID)

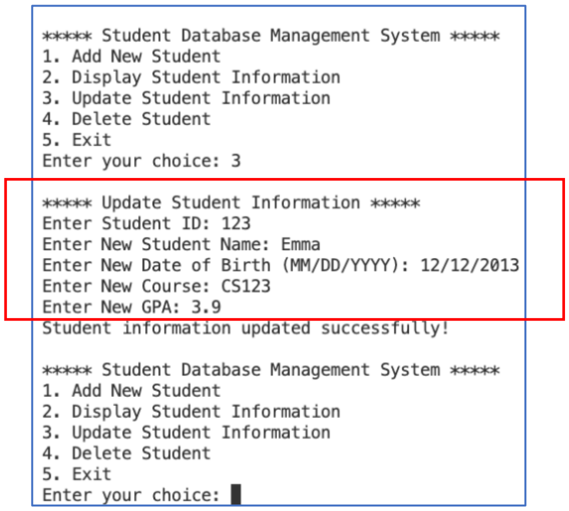


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 3

***** Update Student Information *****
Enter Student ID: 234
Student with ID 234 not found.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice:
```

8. Option 3. Update Student Information (existing ID)

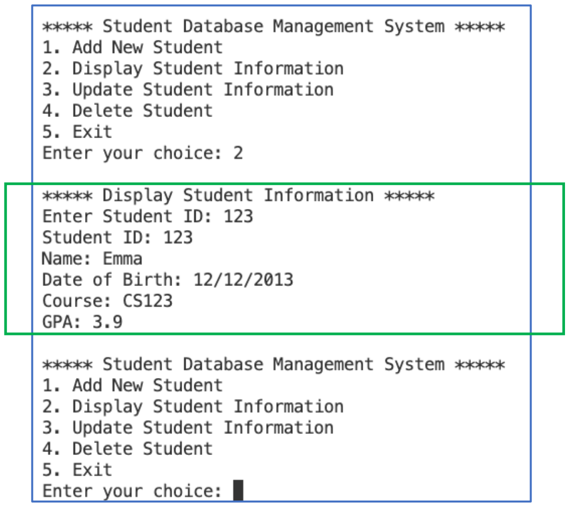


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 3

***** Update Student Information *****
Enter Student ID: 123
Enter New Student Name: Emma
Enter New Date of Birth (MM/DD/YYYY): 12/12/2013
Enter New Course: CS123
Enter New GPA: 3.9
Student information updated successfully!

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: █
```

We can see the updated information

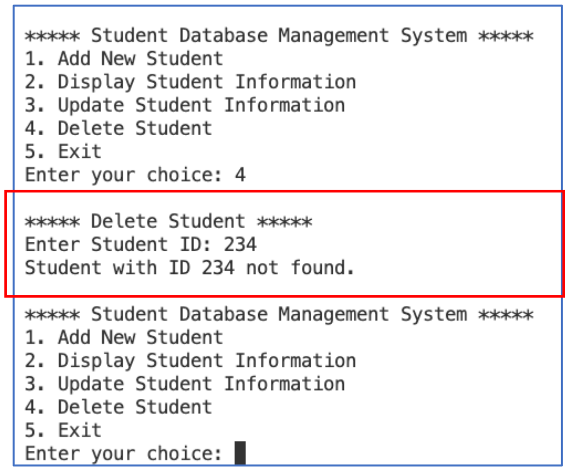


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 2

***** Display Student Information *****
Enter Student ID: 123
Student ID: 123
Name: Emma
Date of Birth: 12/12/2013
Course: CS123
GPA: 3.9

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: █
```

9. Option 4. Delete Student (non-existing ID)

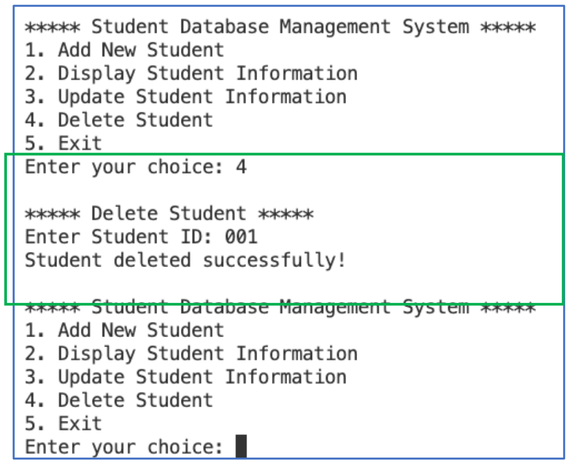


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 4

***** Delete Student *****
Enter Student ID: 234
Student with ID 234 not found.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 
```

10. Option 4. Delete Student (existing ID)

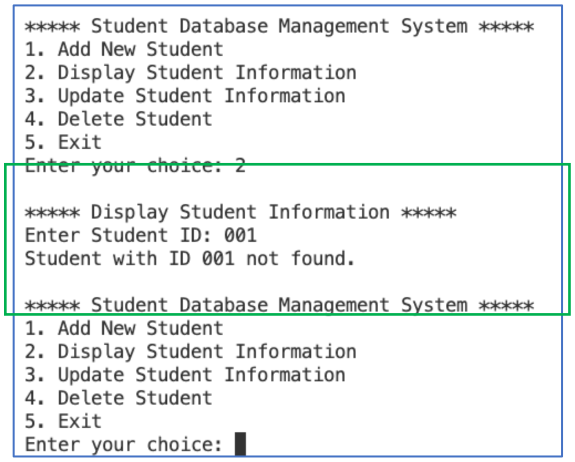


```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 4

***** Delete Student *****
Enter Student ID: 001
Student deleted successfully!

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 
```

We can see the updated information



```
***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 2

***** Display Student Information *****
Enter Student ID: 001
Student with ID 001 not found.

***** Student Database Management System *****
1. Add New Student
2. Display Student Information
3. Update Student Information
4. Delete Student
5. Exit
Enter your choice: 
```

Submission details:

Upload a **single ZIP** file.

Name your file as follows: **Lab5_Lastname_Firstname.zip**

Your **.zip** file must contain the following:

1. Your **.java** source file (no .class).
2. A .txt file (readme.txt) with simple instruction on how to **compile** and **execute** your programs (I reviewed in class how to compile your .java files)
3. A **SINGLE PDF** with **screenshots** from your program running. *Show the use of the different menu options*. Do not send .jpg files.

In each .java file, write as a *multiline* comment at the beginning of the file the following: Your name

The **zip** file must be uploaded to Canvas. I do not accept answers via email or as a comment.
I do not accept image files; it must be a PDF file with the screenshots.

Make sure to check the **due date** for this activity on Canvas. Try to submit it before the due date so you can have time to check for improvements. No late submissions.

This Lab might be challenging for some students. Do not procrastinate. Please, start working on this Lab as soon as possible.

Make sure you are **submitting the correct files**. I will grade the files uploaded to Canvas.
Use the ***javac* and *java*** commands **before** submitting your solution to test if they work outside any IDE.

Make sure to review the **grading rubric.**
Read all the instructions carefully.

Do not assume requirements, contact me if you have questions!

Ensure that your code is original and developed by you.