CS 250 - Fall 2021 - Homework 03

Due date: Oct 13, 2021, 11:59 PM

Objectives

The objectives of this assignment are:

- Testing your ability to use array data structures in problem-solving
- Testing your ability to be familiar with linear (sequential) and binary search

---

Part 1. Non-programming Questions (50 points)

Q1. (16 points) Fill in the blank?

1. _________ is the process of arranging the elements in an array into a particular order

a. searching	b. Creating
c. sorting	d. Declaration

2. One of the preconditions of the binary search method is that the array has to be ___________.

a. random	b. ordered
c. initialized	d. none of the above

3. A 2D array of int is defined as int[][] arrInt = new int[5][10]. How many elements in this array?

a. 15	b. 10
c. 50	d. 66

4. A 2D array of int is defined as int[][] arrInt = {{15, 6, 78}, {45, 22, 14}, {4, -1, 8}}; What is the output of the following Java code:

System.out.println(arrInt[1][0] + arrInt[0][1] + arrInt[2][2]);

a. 4568	b. 45
c. 59	d. Error (exeption occurs)

5. A(n) ____________ is an array of arrays

a. multidimensional array	b. one-dimensional array
c. String	d. ArrayList

6. A(n) __________ is the position of an element within the array. A(n) ____________ is an actual piece of data stored or referenced in the array

a. element, subscript

b. index, type

c. element, index

d. subscript, element

Q2. (16 points) Given the following array of integers: 4, 10, 29, 31, 16, 5, 20, 26, 14, 33. Complete the following tasks:

a. (6 pts) Draw a diagram to illustrate a linear (sequential) search of 31 in the above array, similar to slide 29 of lecture 06.
b. (10 pts) Draw a diagram to illustrate a binary search of 14 on the above array, similar to slides 32-33 of lecture 06. Remember to sort the array before doing a binary search.

Q3. (18 pts) Complete the following methods by filling the missing code:

```
/**
 * Compute the sum of elements on each row of a 2D array
 * @param arr -- a 2D integer array
 * @return None but print the sum of the elements on each row of arr
 */
public static void sumByColums(int arr[][]) {
    //TODO: add code below
}
```

Complete the above-incompleted method to compute the sum of all elements on the same row for each row of the input array (arr). For example, given the following 2D array defined as int[][] arr = {{2, 4, 3, 4}, {7, 3, 4, 3}, {3, 3, 4, 3}, {9, 3, 4, 7}}; This array can be visualized as follow

2	4	3	4
7	3	4	3
3	3	4	3
9	3	4	7

If you call the above method with this array, it should print the following numbers 13, 17, 13, 23. Note: 13, 17, 13, and 23 are the sum of elements on the first row to the last row correspondingly).

Hint: Here are some steps to help you solve this problem:

- Define a 1D array to store the result (the length of this array should be the number of rows in the input array)

- Use two for loop to iterate through elements in the array, calculate the sum of the elements on each row, and print it to console.

Part 2. Programming Question (50 points + bonus 10 points)

2.1 Problem Descriptions

Develop a Java GUI application to read a text file containing students' information into your program, including student ID, name, birth year, and email. Then, the users can search for a specific student by entering the student's name.

2.2 Problem Design and Implementation Guideline

2.2.1. Create the HW#3 Project and a Student text file:

First, you need to create a Java project HW3 on Eclipse. Then, follow the similar approach in Lab4 to create a text file named **students.txt** under the **src** folder of your project. You can do that by right-clicking on the **src** item on the project explorer window. Then, choose **New**, **File**, enter **students.txt** in the **File name** text box, and click the **Finish** button. A subwindow to edit the content of this file is opened.

This file contains the student records in separate rows. Each row stores one student's information, including **id**, **name**, **admission year**, and **email**. A comma separates the data fields of each student. Copy the following contents to this **students.txt** file and save it.

```
10
847342, Kaitlin Goyette, 2015, kaitlin.goyette@test.edu
859403, Ward Cremin, 2017, ward.cremin@test.edu
964535, Zachary Ziemann, 2011, zachary.ziemann@test.edu
295666, Ryan Hyatt, 2001, ryan.hyatt@test.edu
211686, Arjun Price, 2009, arjun.price@test.edu
593397, Fabiola Nikolaus, 2011, fabiola.nikolaus@test.edu
552643, Mittie Tremblay, 2000, mittie.tremblay@test.edu
128645, Ronaldo Bartoletti, 2003, ronaldo.bartoletti@test.edu
970343, Cristobal Marvin, 2010, cristobal.marvin@test.edu
82617, Samir Muller, 2014, samir.muller@test.edu
```

2.2.2. Student class: You need to create a Student class to store each student's information read from the above file. This Student class is similar to the Student class on Lab 4, but the birth year is replaced by admission year, and the state is replaced by email. Code this class with the following requirements:

- This class has four instance variables: **ID** (int), **name** (String), **year** (int - admission year), and **email** (String)

- This class has the following methods:

+ **Student(int id, String name, int year, String email)**: The constructor of this class initializes the instance variables with the parameters (assign parameters to corresponding instance variables)

+ Getter and Setter methods for each instance variables

+ **String toString()**: This method overrides the toString() method derived from the Object class. You should return a string including all student information, i.e., "id, name, admission year, email".

2.2.3. StudentSearchForm class:

You should create a new class named StudentSearchForm. This class **extends JFrame,** which should display the following GUI in Fig. 3.2:

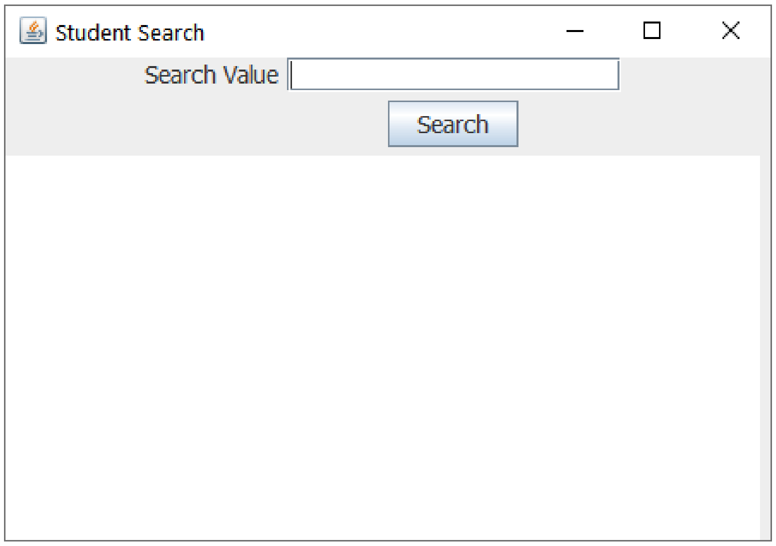


Fig 3.2: The Student Search Form

A. First, you should use **WindowsBuilder** and **SwingDesigner** to design the following GUI components: a JLabel "Search Value", a JTextField named txtSearchValue, a JButton "Search" called btnSearch, and a JTextArea named txtOutut. You should design this GUI using **GridBagLayout**.

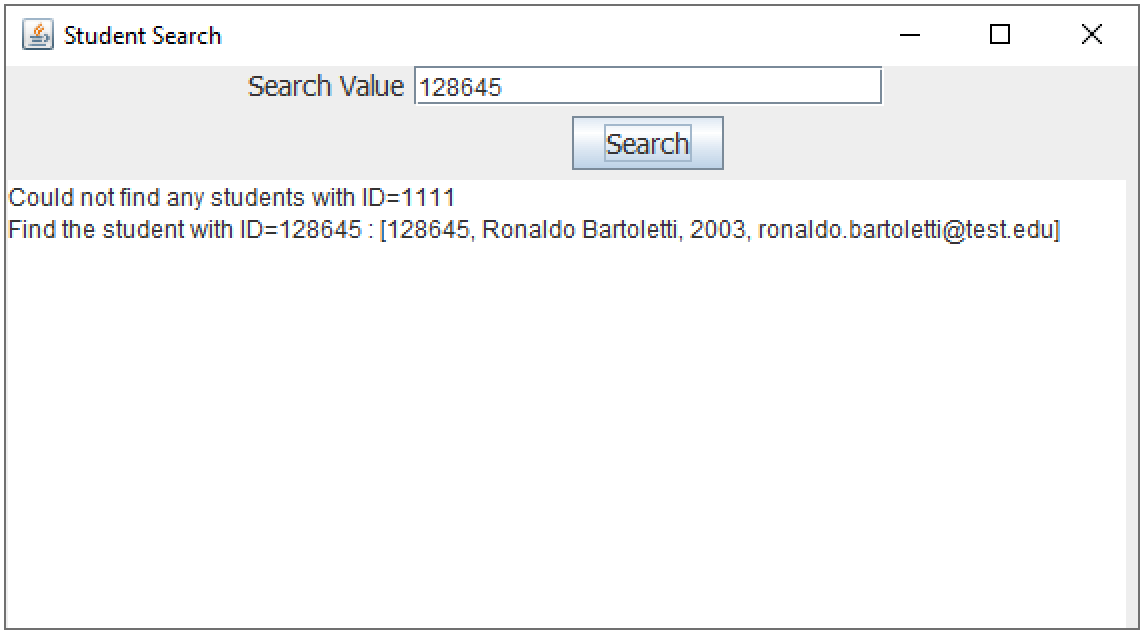
B. Second, implement additional members of this class as follows:

- Declare a **private array of students** named arrStudents
- Write a private method **void loadStudent()** that reads each line of the file **students.txt**, parse intro multiple tokens, create the **Student** object and add to the array **arrStudents.** Check lab 04 source code to implement this method. Call this method at the end of the constructor method of StudentSearchForm class.
- **Sort the array arrStudent based on student's id**: At the end of the constructor method of StudentSearchForm, sort the array **arrStudents** using Arrays.sort() with a new instance of Comparator<Student>. Check lab 6 with a similar code to sort the array of Persons.
- Write a private method **private int binarySearchID(Student[] arrStudents, int id)** which performs a binary search on **arrStudents** (first parameter) to search for the index of the Student with the given student **id** (second parameter). Implement this method similar to the method in lab 06.

3. Handle the click event on button "Search":

- On the Designer of this StudentSearchForm, double click on the button "Search" to add the handler code of this button
- Implement the method actionPerformed():
  - int **sID** = Get the input student id as integer from the text field txtSearchValue (Hint: Integer.parseInt on getText() result on text field)
  - int **index** = Call the method binarySearch to get the search index
  - If the index is -1, **append to the text area txtOutput** the text "Could not find any students with ID=" + sID. Do not forget to append a newline character at the end.
  - Else, **append to the text area txtOutput** the text "Found the student with ID=" + sID and also the founded student information (Hint: use **toString**() of the Student class). Do not forget to append a newline character at the end.
  - As **converting string (getText() from txtSearchValue) to integer can cause an exception**, you must **use try-catch to catch this exception**. Inside the catch block, append an error message to txtOutput.

Below is the sample output of this GUI application:



```
Could not find any students with ID=1111
Find the student with ID=128645 : [128645, Ronaldo Bartoletti, 2003, ronaldo.bartoletti@test.edu]
```

Fig 3.4: The search results on different ID

2.3 Submission Requirements

A. Source code: Write your **own** code

B. Program requirements: Your program can solve and meet all of the requirements described in 2.1 and 2.2. Do not write additional code that is not required.

C. Coding style: Your source code must be indented appropriately and contain adequate comments for class, methods, variables, etc.

- At the beginning of your Java class file, specify the following items:
  - Your name, the assignment number for this program, the date you create the file
  - The problem description that defines your application
  - Enumerate (list) the goals of your application
  - Enumerate (list) the inputs of your application
  - Enumerate (list) the outputs of your application
- Before each method in your class, specify the following items:
  - Specify the purpose of the corresponding method
  - Enumerate the possible inputs and outputs of the corresponding method
  - Write down the pseudocode to implement the corresponding method
- Others comments for variables, inside method implementation

2.4 Grading Rubrics

Your code in this part will be evaluated using the following grading scheme (50 points for completed functional source code + bonus 10 points for good code comments):

Points	Item
25	Correct class, variable, and method definition
10	Consistent coding style, such as indents, curly bracket positions, etc.
15	Have all required variables and methods
10	Good and helpful comments

Assignment Submission Instructions

1. Answer Part 1 questions in a WORD, PDF, or any other document application

2. Compress the whole Eclipse source code of your project in part 2 in zip format using the Export function

Submit two items above to D2L.