Dept. of Computer Science & Software Eng., Concordia University
COMP 6481 --- Fall 2024

Programming and Problem Solving

Assignment 2 --- Due Saturday, November 09, 2024

Part I

Please read carefully: You must submit the answers to all the questions below. However, this part will not be marked. Nonetheless, failing to submit this part fully will result in you missing 50% of the total mark of the assignment.

Question 1

You are designing a simple music playlist application where users can add, remove, and play songs. The application needs to handle operations like adding songs at the beginning and removing songs from random positions in the playlist.

- Discuss the advantages and disadvantages of using an ArrayList versus a LinkedList for implementing this playlist.
- Provide pseudocode to explain how to add and remove songs in both data structures.

Question 2

You are developing a music player that allows users to create a playlist using a circular linked list. The user can set songs to repeat after playing through the entire list.

- Design a method playMusic(int repeatCount) that plays each song in the playlist the specified number of times. After playing all songs, it should loop back to the start for the specified number of repeats.
- Provide pseudocode for this method.
- Analyze the time complexity of your method based on the number of songs and repeats and discuss the space complexity.
- Twist: If a user decides to remove a song while it's playing (e.g., during a repeat), how would you handle this? Discuss the complexity implications of handling dynamic changes during playback.

Question 3

Imagine you have two sorted lists, and you need to merge them into a single sorted list.

a) Write pseudocode for merging two ArrayLists and two LinkedLists.

b) Determine the time complexity for the merge operation for both the data structures.

c) Discuss the space complexity and whether any additional memory is required during the merging process.

Part II

Purpose: The purpose of this assignment is to allow you practice Array Lists and Linked Lists, as well as other previous object-oriented concepts.

"Your present circumstances don't determine where you can go; they merely determine where you start."

--- Nido Qubein

Diwali, known as the Festival of Lights, is a mesmerizing celebration that radiates joy, hope, and community spirit. As the families come together to illuminate their homes with vibrant diyas and colorful rangoli designs, the festival signifies the victory of light over darkness and good over evil. The air is filled with the aroma of delectable sweets, the sound of laughter, and the spectacle of fireworks, making Diwali a cherished occasion for millions around the world..

Picture an enchanting evening where creativity knows no bounds. Participants of all ages gather for the Diyas Collection Contest, each eager to showcase their artistic talents. The atmosphere buzzes with excitement as colorful diyas—crafted with intricate designs and vibrant colors—fill the venue, each telling its own story. This year, the contest introduces a delightful twist: participants can use a variety of materials, from eco-friendly paints to shimmering sequins, allowing their imaginations to run wild.

The competition is structured to recognize not just the visual appeal of the diyas but also the thoughtful integration of traditional designs. Each participant's diya collection is evaluated based on two criteria:

1. ***Decorative Value:*** The artistic merit of each diya, considering creativity, color, and design intricacy.
2. ***Cost of Materials:*** The monetary value associated with materials used, encouraging participants to balance creativity with resourcefulness.

At the end of the contest, the top three participants will be honored with coveted awards. In a departure from traditional accolades, the contest will award uniquely themed prizes that embody the spirit of Diwali:

- ***Luminary Legend:*** Awarded to the top participant whose collection sparkles with unmatched creativity, receiving a handcrafted lamp that symbolizes wisdom & light.
- ***Artisan's Heart:*** For the second-place, this prize includes a beautifully crafted clay pot, signifying the depth of artistry and culture in every diya made.
- ***Candlelight Dreamer:*** The third-place participant will get a set of scented candles, each representing a different flavor of Diwali, illuminating their home with fragrance and warmth.

Participants will submit their diya collections, detailing the number of each type of diya and the materials used. A detailed list of the five type of Diyas is mentioned below.

1. ***Starry Night Diyas:*** Decorative value of 8, material cost $3.00 each.
2. ***Lotus Blossom Diyas:*** Decorative value of 7, material cost $2.50 each.
3. ***Ocean Wave Diyas:*** Decorative value of 9, material cost $4.00 each.
4. ***Sunset Glow Diyas:*** Decorative value of 6, material cost $2.00 each.
5. ***Mystic Forest Diyas:*** Decorative value of 7, material cost $3.50 each.

In this assignment, you will design and implement a tool which will determine the top three places of the competition. You are given few samples, containing information about participants, their Diya collection and a few requests. Each request contains one/more participant names. Your program will have to get the input through scanner and must work for inputs other than the sample input as well. You will input this information and will produce an outcome for each participant present in request(s). The outcome for each request should be one of the following responses where X signifies the participant name.

a) Participant X wins the ***Luminary Legend award*** with a collection having highest decorative value
b) Participant X wins the ***Luminary Legend award*** with a collection having highest decorative value and a lower material cost
c) Participant X wins the ***Artisan's Heart award*** with a collection having second highest decorative value
d) Participant X wins the ***Artisan's Heart award*** with a collection having second highest decorative value and a lower material cost
e) Participant X wins the ***Candlelight Dreamer award*** with a collection having third highest decorative value
f) Participant X wins the ***Candlelight Dreamer award*** with a collection having third highest decorative value and a lower material cost
g) Participant X is ***not in top three*** owing to collection with low decorative value
h) Participant X is ***not in top three*** owing to collection with higher material cost

You can assume that the competition has finished and all relevant information is made available. The information given to you may not be in any specific order (information structure for each participant will be same but order in which participants are entered will not be same). A sample of participant information input data is depicted in Figure 1 (ParticipantName and number of each type of Diya created), and a sample of request data is depicted in Figure 2. A detailed description of all the details that you have to implement for this assignment is made avaialble after these two figures.

MysticMoonbeam	12	08	05	10	06
GlimmeringGem	07	15	03	12	09
RadiantRani	10	10	10	00	02
TwinklingTulip	05	05	15	08	07
DazzlingDiva	09	12	04	06	11
ShimmeringSage	04	03	20	07	10
LuminousLotus	08	10	06	14	05
BrilliantBreeze	03	02	18	11	04
EnchantingEmber	11	06	07	09	08
CelestialCharm	15	05	05	04	02

Figure 1. Illustration of participant information input

```
TwinklingTulip
DazzlingDiva
ShimmeringSage
LuminousLotus
```

Figure 2. Illustration of Requests

I) Create an interface named **Lightable** which has a single parameter boolean method called isInTheTopThree (Participant P) where P is an object of type **Participant** described below.

II) The **Participant** class, has the below mentioned attributes: a participantID (String), a participantName (String), a diyaCollection (int[]). It is assumed that participant name is always recorded as a single word (_ is used to combine multiple words). It is also assumed that no two participants can have the same participantID. You are required to write implementation of the **Participant** class. Besides the usual mutator and accessor methods (*i.e.* getParticipantID(), setParticipantName(), *etc.*) class must also have the following:

a) Parameterized constructor that accepts three values and initializes participantID, participantName, diyaCollection, and points to these passed values;
b) Copy constructor, that takes in two parameters, a Participant object and a String value. The newly created object will be assigned all the attributes of the passed object, with the exception of the participantID. participantID is assigned value passed as the second parameter to the constructor. It is always assumed that this value will correspond to the unique participantID rule;
c) clone() method, that will prompt a user to enter a new participantID, then creates and returns a clone of the calling object with the exception of the participantID, which is assigned the value entered by the user;
d) Additionally, class will have a toString() and an equals() method. Two participants are equal if they have the same attributes, with the exception of the participantID, which could be different.

III) The **ParticipantList** class has the following:

(a) An inner class named **ParticipantNode**. This class has the following:
   i. Two private attributes: a participant object and a pointer to a ParticipantNode object.
   ii. A default constructor, which assigns both attributes to null.
   iii. A parameterized constructor that accepts two parameters, a Participant object and a ParticipantNode object, then initializes the attributes accordingly.
   iv. A copy constructor.
   v. A clone () method
   vi. Other mutator and accessor methods.

(b) A private attribute called head, which points to the first node in this ParticipantList.

(c) A private attribute called size, which always indicates the current size of the list (how many nodes are in the list).

(d) A default constructor, which creates an empty list.

(e) A copy constructor, which accepts a **ParticipantList** object and creates a copy of it.

(f) A method called addToStart(), which accepts only one parameter, an object from Participant class, creates a node with that passed object, and inserts this node at the head of the list.

(g) A method called insertAtIndex(), which accepts two parameters, an object from the Participant class, and an integer representing an index. If the index is not valid (a valid index must have a value between zero and size-1), then the method must throw a **NoSuchElementException** and terminates the program. If index is valid, then the method creates a node with passed Participant object and inserts this node at the given index. The method must properly handle all special cases.

(h) A method called deleteFromIndex(), which accepts one int parameter representing an index. If index is not valid, method must throw a **NoSuchElementException** and terminate the program. Otherwise, node pointed by that index is deleted from the list. The method must properly handle all special cases.

(i) A method called deleteFromStart(), which deletes the first node in the list (the one pointed by head). All special cases must be properly handled.

(j) A method called replaceAtIndex(), which accepts two parameters, an object from Participant class, and an integer representing an index. If index is not valid, the method simply returns; otherwise, object in list at passed index must be replaced with the object passed.

(k) A method called find(), which accepts one parameter of type String representing a participantID. This method then searches the list for a Participant node with that participantID. If such an object is found, then method returns a deep copy of that Participant; otherwise, method returns null. The method must keep track of the number of iterations made before the search finally finds a Participant or concludes that it is not in the list.

(l) A method called contains (), which accepts a parameter of type String representing a participantID. Method returns true if a participant with that participantID is in the list; otherwise, the method returns false.

(m) A method called equals (), which accepts one parameter of type ParticipantList. Method returns true if the two lists contain similar participants; otherwise, the method returns false. Recall that two Participant objects are equal if they have the same values except for the participantID, which is expected to be, different.

Finally, here are some general rules that you must consider while implementing above methods:

- Whenever a node is added or deleted, the list size must be adjusted accordingly.
- All special cases must be handled, whether the method description explicitly states that or not.
- All clone() and copy constructors must perform a deep copy; and not shallow copies.
- If any of your methods allows a privacy leak, you must clearly place a comment at the beginning of the method 1) indicating that this method may result in a privacy leak 2) explaining reason behind the privacy leak. Please keep in mind that you are not required to implement these proposals.

IV) Now, you are required to write a public class called **CompetitionResults** which must implement **Lightable** interface. In main() method, you must do the following:

(a) Create at least two empty lists of ParticipantList class (for copy constructor III (e)).

(b) Input participant information as depicted in Figure 1 to initialize one of the ParticipantList objects you created above. You can use the addToStart() method to insert Participant objects into the list. However, the list should not have any duplicate records, so if the input has duplicate entries, your code must handle this case so that each record is inserted in the list only once.

(c) Input request information as depicted in figure 2 and create **ArrayList** from the contents. Iterate over each participant. Process each of the participants and print the outcome whether the participant will be in top three or not. A sample output for a given input is mentioned below. Again, your program must ask for the input information through console as your program will be tested against similar input information.

(d) Prompt the user to enter a few participantIDs and search the list that you created for these values. Make sure to display number of iterations performed.

(e) Following that, you must create enough objects to test each of the constructors/ methods of your classes. The details of this part are left as open to you. You can do whatever you wish if your methods are being tested including some of the special cases. You must also test the isInTheTopThree() method.

(f) This class needs to implement interface from part I. The method isInTheTopThree that takes in another Participant object P and should return true if P is from the same group as the current participant object, or vise versa; otherwise it returns false. There will be two groups; one with top three and the other with the rest.

```
TwinklingTulip is not in  the top three owing to collection with higher material cost.
DazzlingDiva is not in top three owing to collection with lower decoration value.
ShimmeringSage wins the Luminary Legend award with a collection having highest decorative value.
LuminousLotus wins the Candlelight Dreamer award with a collection having third highest decorative
value and a lower material cost.
```

Figure 3. A sample outcome of the enrolment request

Some general information:

a. Do not use any external libraries or existing software to produce what is needed; that will directly result in a 0 mark!

b. How about a Candy class. Focus on the optimal design.

c. Again, your program must work for any input data. The sample input provided with this assignment is only a possible version, and must not be considered as the general case when writing your code.

General Guidelines When Writing Programs:

- Include the following comments at the top of your source codes.
  ```
  // Assignment (include number)
  // Question: (include question/part number, if applicable)
  // Written by: (include your name and student id)
  // -----------------------------------------------------
  ```
- In a comment, provide a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output must be displayed with clear messages and in an easy-to-read format.
- End your program with a closing message so that the user knows that the program has terminated.

JavaDoc Documentation:

Documentation for your program must be written in **javaDoc**.
In addition, the following information must appear at the top of each file:

Name and ID	(include full name and ID)
Assignment #	(include the assignment number)
Due Date	(include the due date for this assignment)

SUBMISSION INSTRUCTIONS

Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must contain your name(s) and student ID(s). Use "official" name only – no abbreviations/nick names; capitalize the "last" name. Inappropriate submissions will be heavily penalized.

IMPORTANT: For Part II of the assignment, a demo for about 5 to 10 minutes will take place with the marker. You **must** attend the demo and be able to explain their program to the marker. The demo schedule will be determined and announced by the markers, and students must reserve a time slot for the demo.

Now, please read very carefully:

- **If you fail to demo, a zero mark is assigned regardless of your submission.**
- **If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.**
- **Failing to demo at second appointment will result in zero marks and no more chances will be given under any conditions.**

EVALUATION CRITERIA

IMPORTANT: **Part I** must fully be submitted. Failure to submit that part will cost 50% of the total marks of the assignment!

Total	**10 pts**
Documentations	**1 pt**
JavaDoc documentations	1 pt
Tasks	**9 pts**
Task#1	1 pts
Task#2	2 pts
Task#3	3 pts
Task#4	3 pts