ITCS 2214-002 Program 1.2 - Lists Spring 2016

Due Thursday March 24th, 2016 at 11:55PM

This assignment is the third and final portion of the List project for this course. The primary goals are to read and parse a datafile, construct extended Singly Linked Lists using Bridges SLelements, and generate a series of visualizations highlighting features of the data.

Overview

The primary goals of this program are as follows:

(1) Parse the IGN Video Game Reviews dataset. Fields are as follows:
Game, Platform, Score, Genre

(2) Create three new classes, **GameStats**, **Genre**, and **Platform,** to store the data from the given dataset

(3) Create two classes with Singly Linked Lists extensions; **GenreList** parameterized to store **Genre** objects and **PlatformList** to store **Platform** objects

(4) Write a few methods to apply visual properties to the elements in each list

The **List** interface and **Singly Linked List** implementation can be taken from Program 1.1

The **GameStats** class will be a parent class for both the **Genre** and the **Platform** classes. It will contain attributes and methods that both these subclasses require.

The **Genre** class will organize the game data based on each unique genre. It must extend the **GameStats** class and implement the **Comparable** interface. It must keep track of the Genre name.

The **Platform** class will organize the game data based on each unique platform. It must extend the **GameStats** class and implement the **Comparable** interface. It must keep track of the platform name.

The **GenreList** and **PlatformList** classes should both extend **SLList** and parameterize the list for the respective **Genre** and **Platform** classes. Both must override the *insert* method to insert in alphabetical order.

The **Driver** will read the datafile line by line, adding each record into both the **GenreList** and **PlatformList** structures. Next, it should allow the user to select from among a few options to modify and visualize the data structures using Bridges.

Tasks

Create a new package for Program 1.2. Copy the List interface and your Singly Linked List implementation from Program 1.1.

Add the **Bridges JAR** to the build path for this project.

Documentation for Bridges classes can be found at the following link:
http://bridgesuncc.github.io/doc/java-api/current/

GameStats class:

Private Member variables –

- The number of games
- The average rating of the games
- The standard deviation of the ratings
- The maximum rating
- The minimum rating
- An alphabetized list of game titles

Methods –

- *Get* methods for each member variable
- An *Add* method to add a game. This method should update all the member variables appropriately. (Increment count, re-compute average and standard deviation, insert game title in order, etc.)
  - See the **equations** section below for further details on average and stddev computations

Genre class (extends **GameStats,** implements **Comparable)**

Private Member variables –

- The Genre name

Methods –

- A *compareTo* method to compare Genre names

Platform class (extends **GameStats,** implements **Comparable)**

Private Member variables –

- The Platform name

Methods –

- A *compareTo* method to compare Platform names

GenreList class (extends **SLList<Genre>**)

Methods –

- *Insert* (overridden from SLList) to insert Genres into the list in alphabetical order. Check if the genre game list already includes the current game, ignore if so.
    - *Action -> Adventure -> Platformer -> RPG -> (etc)*
- A method to set the labels of each Bridges element in the GenreList. Each label should display the list of all games in each Genre
- A method to highlight the Genre with the highest average rating
- A method to highlight the Genre with the lowest average
- A method to reset the visual properties of all elements to some default settings

PlatformList class (extends **SLList<Platform>**)

Methods –

- *Insert* (overridden from SLList) to insert Platforms into the list in alphabetical order
    - *Android -> PC -> PlayStation 1-> Xbox-> (etc)*
- A method to set the labels of each Bridges element in the PlatformList. Each label should display the following rating stats: count, max, min, average, and standard deviation for each platform
- A method to change the visual properties of each platform (shape, opacity, color, and/or size) based on the average ratings
- A method to change the visual properties of each platform (shape, opacity, color, and/or size) based on the number of games
- A method to reset the visual properties of all elements to some default settings

Driver –

- Create instances of the GenreList and PlatformList data structures
- Open the *games* data file and read it line by line. Each line has the following structure:
    - Game \t Platform \t Score \t Genre
    - Use the *String.split(arg)* method to parse the attributes from the line (splits the string into an array of Strings where *arg* is the String to split around)
- For each game line, insert the game into **both** lists.
    - If the list is empty, create a new Genre/Platform with the game info and directly insert it into the list
    - Otherwise, search through the list to find the place to insert the current Genre/Platform. If the Genre/Platform is already in the list, simply add the game data to it. If not, create a new Genre/Platform with the game info and insert in order
- Create a simple menu allowing users of your program to specify which visualizations to send to Bridges.
    - Reset visual properties of either list
    - Options for each method in either list
    - Use *Bridges.setDataStructure()* and *Bridges.visualize()* each time

Helpful Equations

To compute a running average of scores:

Initialize the average to 0

$average = average + (score - average) / count$

We will keep track of the power sum average in almost the exact same way as the average:

Initialize the power average to 0

$powerSumAverage = powerSumAverage + (score^2 - powerSumAverage) / count$

We use the power sum average to compute the standard deviation:

Initialize standardDeviation to 0

$\text{standardDeviation} = \sqrt{(\text{powerSumAverage} - \text{average}^2)}$

Scoring Rubric

Driver: 30 points

- Up to 5 points for appropriate documentation and comments
- Up to 10 points for properly reading and parsing the entire data file
- Up to 5 points for inserting data into both lists
- Up to 10 points for implementing a menu and visualizing the data structure after each user operation

GameStats: 10 points

- Up to 10 points for correctly computing the average, count, stddev, max, min, and game list for each game inserted

Genre/GenreList, Platform/PlatformList: 30 points each

- Up to 5 points for appropriate documentation and comments
- Up to 5 points for correctly constructing both classes according to specifications
- Up to 10 points for maintaining sorted order
- Up to 10 points for implementing specified visualization methods with unique visual properties

Total points available: 100

This will be graded as a programming assignment

Programs turned in after the due date (Thursday March 24th) will lose 10% for every late day