# Programming 2 December Coursework Specification (2019-2020)

An app is required to manage a collection of restaurant reviews.

Restaurant & review data are stored in a file; either delimited text format or object file.

A restaurant is identified by a unique id and has name & location also stored. For each review, the following data is stored: reviewer and a star rating between 1 & 5.

## Model Class Diagram

See Appendix 1

## Requirements

A starter project is supplied for a console menu-driven application. The starter project contains a controller class with stubs for methods implementing the required functionality listed below:

1. Load restaurant data information from a specified file into a restaurant collection implemented using a repository and list in id order
2. Add a new restaurant to the restaurant collection
3. Add a review to a specified restaurant in the restaurant collection
4. List, for a specified location, restaurant & review data in restaurant name order
5. List, for each restaurant, the restaurant name and average star rating
6. Persist the collection to a specified file

The starter project contains starter model, repository and controller classes which require to be completed. A sample delimited text file is additionally supplied.

## Architecture

The architectural pattern in the labs is to be used with classes as detailed in Appendix 2.

## Development Approach

An incremental development approach should be used with the following increments:

### Load restaurant data from a file and list in id order

This fulfils Requirement 1. To complete this increment you should:

1. Complete the **Restaurant**, and **Review** classes with required fields and methods as depicted in the class diagram in Appendix 1.
2. Decide on which type of collection you want the repository to define and make the necessary adjustments to the **Repository** class and **RepositoryInterface**.
3. Partially complete the **RestaurantController** class by completing the constructor which creates a new **Repository** object - from a file if specified by the user.
4. Partially complete the **RestaurantController** class to implement the *listRestaurantDataInIdOrder()* method.
5. Create a **DAOImpl** class to realize the **DAOInterface** using either text file or object file format.
6. Modify the **Repository** class to use the **DAOImpl** class to load restaurant data from a specified file.

Document the results of your testing using screen snippets copied into a Word document. Take a copy of the **RestaurantController** class and name it as *RestaurantController\_Increment1.java*.

### Add a new restaurant

This fulfils Requirement 2. To complete this increment you should:

1. Fully implement the *addRestaurant()* method in the **RestaurantController** class; this should ask the user for required details, create a new **Restaurant** object and add it to the **Repository** object.

Document the results of your testing using screen snippets copied into a Word document. Take a copy of the **RestaurantController** class and name it as *RestaurantController\_Increment2.java*.

### Add a review to a specified restaurant

This fulfils Requirement 3. To complete this increment you should:

1. Complete the *addReview()* method in the **RestaurantController** class; use a user supplied id value to retrieve the **Restaurant** to update and then add the new **Review** with data supplied by the user.

Document the results of your testing using screen snippets copied into a Word document. Take a copy of the **RestaurantController** class and name it as *RestaurantController\_Increment3.java*.

### List, for a specified location, restaurant data in order of restaurant name

This fulfils Requirement 4. To complete this increment you should:

1. Complete the *listLocationRestaurantDataInNameOrder()* method in the **RestaurantController** class using a user supplied location value.
2. Ensure the **Restaurant** class defines a method to compare restaurants on the *name* attribute.

Document the results of your testing using screen snippets copied into a Word document. Take a copy of the **RestaurantController** class and name it as *RestaurantController\_Increment4.java*.

### List, for each restaurant, the restaurant name and average star rating

This fulfils Requirement 5. To complete this increment you should:

1. Complete the *listRestaurantRatings()* method in the **RestaurantController** class.

Document the results of your testing using screen snippets copied into a Word document. Take a copy of the **RestaurantController** class and name it as *RestaurantController\_Increment5.java*.

### Persist restaurant collection to a file

This fulfils Requirement 6. To complete this increment you should:

1. Implement the *store()* method of the **DAOImpl** class if you have not already done so.
2. Implement any **necessary** *toString(DELIMITER)* methods.

Document the results of your testing using screen snippets copied into a Word document.

## Submission Requirements

### Submission Deadline: 9th December 2019 9:00am

A zip file should be uploaded to GCULearn with a NetBeans project – which runs in the university labs; and, a Word document including:

* a brief narrative summary of the (attempted) implementation of each of the increments explaining methods and objects used and providing screen snippets of test output – you can use this section to explain any unresolved problems and how you have attempted to fix them

## Marking Scheme

|  |  |
| --- | --- |
|  | **Marks** |
| Restaurant class | 7 |
| Review class | 3 |
| Repository class | 4 |
| DAOImpl class | 5 |
| Increment1 RestaurantController class | 7 |
| Increment2 RestaurantController class | 3 |
| Increment3 RestaurantController class | 4 |
| Increment4 RestaurantController class | 5 |
| Increment5 RestaurantController class | 3 |
| Increment6 RestaurantController class | 4 |
| Use of nested classes (inner/local/anonymous) | 5 |
| Use of abstract & generic classes |
| Use of Enums |
| Use of lambda expressions/aggregate operations |
| **Total** | 50 |

## Sample Delimited Text File Format

1,"Le Bistro","Glasgow",5

“Kris Kristofferson”,3

”Tori Amos”,4

”Herbie Hancock”,4

”Diana Krall”,2

”k. d. Lang”,3

2,”Court and Spark”,”Edinburgh”,2

”Norah Jones”,4

”Herbie Hancock”,5

3,"Food Inc","Glasgow",1

“Kris Kristofferson”,3

## Starter Project Test Run

run:

Restaurant Reviews App

=====================

Restaurant Id Order

===================

A. Add Restaurant B. Add Restaurant Review C. List Location Restaurant Data In Name Order D. List Restaurant Ratings Q. Quit

Enter choice:

A

Add Restaurant

==============

Restaurant Id Order

===================

A. Add Restaurant B. Add Restaurant Review C. List Location Restaurant Data In Name Order D. List Restaurant Ratings Q. Quit

Enter choice:

B

Add Restaurant Review

=====================

Restaurant Id Order

===================

A. Add Restaurant B. Add Restaurant Review C. List Location Restaurant Data In Name Order D. List Restaurant Ratings Q. Quit

Enter choice:

C

Name Order

==========

Restaurant Id Order

===================

A. Add Restaurant B. Add Restaurant Review C. List Location Restaurant Data In Name Order D. List Restaurant Ratings Q. Quit

Enter choice:

D

Restaurant Ratings

==================

Restaurant Id Order

===================

A. Add Restaurant B. Add Restaurant Review C. List Location Restaurant Data In Name Order D. List Restaurant Ratings Q. Quit

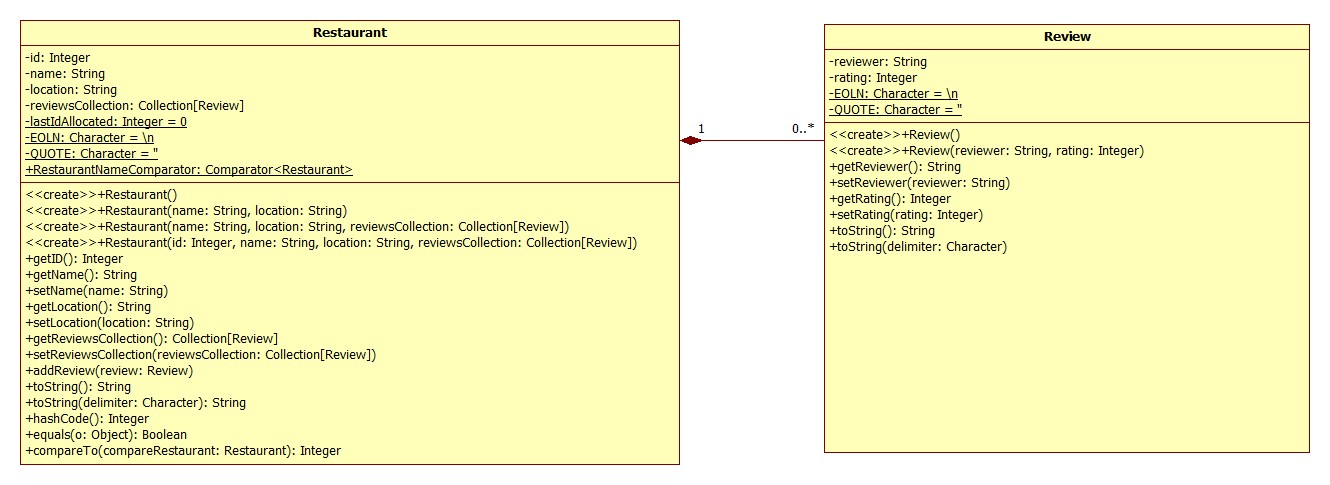
Enter choice:

Q

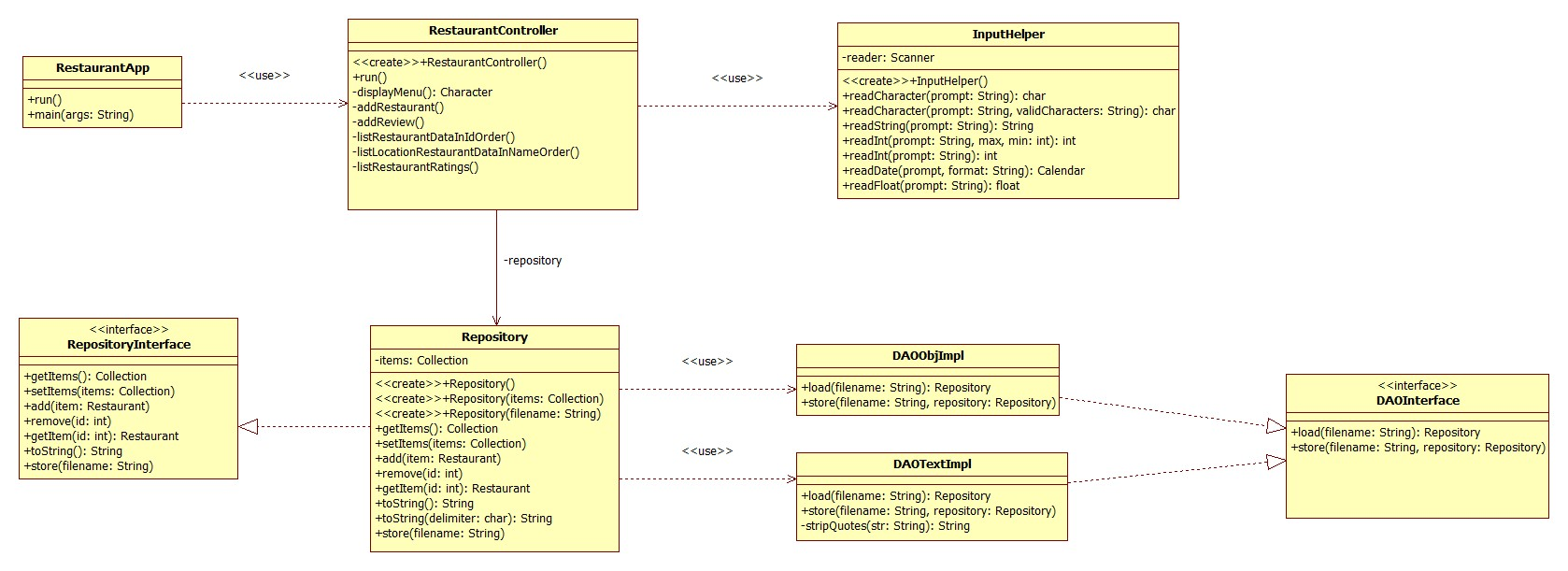
Thank you for using Restaurant Reviews App. Good bye.

BUILD SUCCESSFUL (total time: 32 seconds)

Appendix 1



Appendix 2

****