Maria Molloy

Professor Juan Arias

CMPT 220

4 May 2018

Final Write Up

Abstract: This project sets up a database for a library and allows an administrator to add books to the library and system or users and members of the library to the system. Furthermore, the users can check out and return books. The main class in this project is the Database class. It handles all of the interactions with the database, from adding a user, to checking out a book. All of this information is stored in a database built using MariaDB. The database has three tables: Books, Users and UserBooks. The Books table keeps track of all of the books in the library and if they are available or not, the Users table keeps track of all the members of the library (or anyone who can check out and return books), and the UserBooks table keeps track of all of the books checked out and returned and by whom. The database is accessed by a user by interacting in the console through a scanner in the class Project2_Run. This system is meant to be used by a small scale library or even a personal library. An ideal use of this system would be by an individual with a large personal library that often loans books to friends. Because this is a fairly simple system in the grand scheme of things, it would probably not be the best option for a large library such as the New York Public Library.

Introduction: The motivation of this project is to explore the organization and complex classification of books and libraries. My project will be explained with first, a detailed system

description of UML diagrams, them the requirements of the system, then a description of similar systems, a description of how users should use the system, and then a bibliography.

Detailed System Description: The system manages a database for a library. It is made for a library administrator to use. All interactions are handled in the class Project2_Run, using methods that access the database from the Database class. The database was created using MariaDB in the DataGrip IDE. Most methods in the Database class use SQL syntax in order to access, update, and add to the database. The database is called library and there are three tables in it. The first table is a Books table, that contains each book's unique ID, the title, the author, the ISBN number, and if it is available to check out. The second table is a User table that has the user's ID, their first name, last name, and the time their account was created. The third table interacts with the books and users table and keeps track of which users have which books. This table has the book's ID, the user's ID, when the book was checked out and when it was returned. If the book has not been returned yet, that data field is empty until the book is returned.

Database
+con: Connection
+Database() +addBook(t: String, a: String, i: int): void +addUser(first: String, last: String): void +checkOut(bookId: int, userId: int): void +returnBook(bookId: int): void +getUserId(last: String): ArrayList<String> -resultToArrayList(rs: ResultSet): ArrayList<String> +getBookTable(): void

Project2_Run

+data1: Database
+main(String[]): void

DBTablePrinter
-DEFAULT_MAX_ROWS: int -DEFAULT_MAX_TEXT_COL_WIDTH: int +CATEGORY_STRING: int +CATEGORY_INTEGER: int +CATEGORY_DOUBLE: int +CATEGORY_DATETIME: int +CATEGORY_BOOLEAN: int +CATEGORY_OTHER: int
-Column(): void +printTable(conn: Connection, tableName: String): void +printTable(conn: Connection, tableName: String, maxRows: int): void +printTable(conn: Connection, tableName: String, maxRows: int, maxStringColWidth: int): void +printResultSet(rs: ResultSet): void +printResultSet(rs: ResultSet, maxStringColWidth: int): void -whichCategory(type: int): int

Requirements: The system is addressing the needs of a library and providing a system for returning, and checking out books. It creates a database to store books, users, and users interactions with the library, along with access to the database tables in Java classes. The main specific uses of this system would be to add books, add users, checkout books, and return books. This system addresses an organizational problem. It organizes data into simple tables and gives a user an easy way to access and update these tables. This system takes a lot of data that could easily be confusing and organizes it in an easy to understand and easily accessible way.

Literature Survey: Many iterations of this system exist, both professional ones used by real libraries and other amateur Java projects made by students. The New York Public Library, which is the fourth largest library system in the world, uses two online catalogue systems that allow users to search the collection and request books online. There are various open source versions of this project that other amateur Java users have created, however most of these existing projects are very complicated and require the user to download lots of various software. Because of the other, more complicated iterations of this project, my system is best used on a small scale, by someone who wants a simple, easy to use library system. If a large library with a high level of interactions needed a system, one of the others existing would probably be better for them to use. This system is ideal for a user who wants a straightforward way to access their data and is dealing with a smaller number of interactions.

User Manual: This system should be used by small scale or personal libraries to track the books they have and the users that have library cards. The system should be used by running the class Project2_Run. This class calls methods in the Database class that handle interactions with the library database. When the class Project2_Run is compiled and run, it will prompt a user to enter a number for what they want to do. After entering the number corresponding to what they want to do, the user should follow the instructions prompted by the program.

If a user wanted to add another user to the system, they would start by running the class and then entering the number 0. Then the program prompts the user to enter their first name. After typing in their first name and pressing enter, the user will be prompted to enter their last name. After doing so, the program will add the user to the database (in the User table) and tell

the user their ID number (which would presumably be their library card number as well) so that they can check out and return books.

If a user wanted to check out a book, they would start by running the class and entering the number 1. Then, the program will print a list of available books. The program will prompt the user to enter the ID of the book they want to check out (which is listed next to the book's title in a table) and the ID of the user who wants to check out a book. So, for example, if you wanted to check out the book "A Tale of Two Cities" and your ID number was 23, you would type 29 23 and then press enter. After, the program will call a method from the Database class that adds this interaction to the UserBooks table and updates the entry of the book in the Books table to say that is in not available.

In order to return a book, a user would run the Project2_Run class and enter the number 2. Then, the program will prompt the user to enter the ID of the book they want to return. So, if a user wanted to return "A Tale of Two Cities", they would enter the number 29, since that is the book's ID number. Then the program will call a method in the Database class that updates both the UserBooks and Books tables. It updates the dateReturned data field in the UserBooks table to the current date and time and updates the entry of that book in the Books table to say that it is now available to check out.

If a user wanted to add a book to the database, they would run the Project2_Run class and enter the number 3. Then, the program prompts the user to enter the book's ISBN. After entering the ISBN, the program will prompt the user to enter the author of the book. After entering the author's name, the program will ask the user to enter the number of words in the title of the book. For example, if a user wanted to add "To Kill a Mockingbird" they would enter 4. Then,

the user is asked to enter the title of the book. After doing so, the program calls a method in the Database class that uses SQL syntax to add a book to the Books table in the library database. When a book is added, the availability status is automatically set to available.

While it is not a part of the system or the project, it is possible to access the database and browse the data. However, a username and password may be required. If a user wanted access to the database, the username is root and the password is password.

Conclusion: This system successfully allows a user to create a database for a library and handle interactions with this database, such as checking out books, returning books, and adding books and users. I am considering using this database myself, to keep track of the books I own and where they are. While my system does successfully solve the problem of organizing and keeping track of books and loans, there are some steps I would have taken to make it better if I had more time and resources. One of these would be creating a GUI. The system would be much more user friendly if there was a front-end aspect, however I did not have time to complete this. Another aspect I had hoped to add but ran out of time for was checking for overdue books and fining users for overdue books.

References/Bibliography:

Torun, Hami (2014) Database Table Printer. https://github.com/htorun/dbtableprinter.

I did not create the class DBTablePrinter, this was made by Hami Torun, or htorun on GitHub. In the description for the class on GitHub, Torun states that he made it in order to learn about publishing open source software and that it is published in the hope that it will be useful to other

Java users. I would have created this class on my own, however I did not have the time or skill set, since it deals with many topics that were not covered or focused on in class and is a very in depth program.

Additionally, while we did not create a group project, I worked with my peers Tori Spychalski, Damion Neth, and William Kluge because we were all working on solving similar problems, or had knowledge about databases. We all have different, unique projects but collaborated on solving similar problems that we were all facing.