Plano East Senior High School

Final database project

An App Store

William Deprez

May 17, 2019

1 Introduction

My program will implement a database of Users, Apps, and Reviews to make up a social app store. It tracks a user's owned apps, existing apps, and reviews of that app. This creates a useful store and essentially it's own digital marketplace where anyone can create their own applications. My list of apps will be contained within a store object, while my users are contained within my community object.

2 Application Architecture

The application contains several components which are depicted in Figure 1. I will use three plain java objects for my program. My first object App will contain the name of the app, the creator (username), the appId, and the date of creation. The second object User will contain the username, userId, ownedApps, and usersReviews. The third object Review will store a rating — an integer of one through ten — and some review related text.

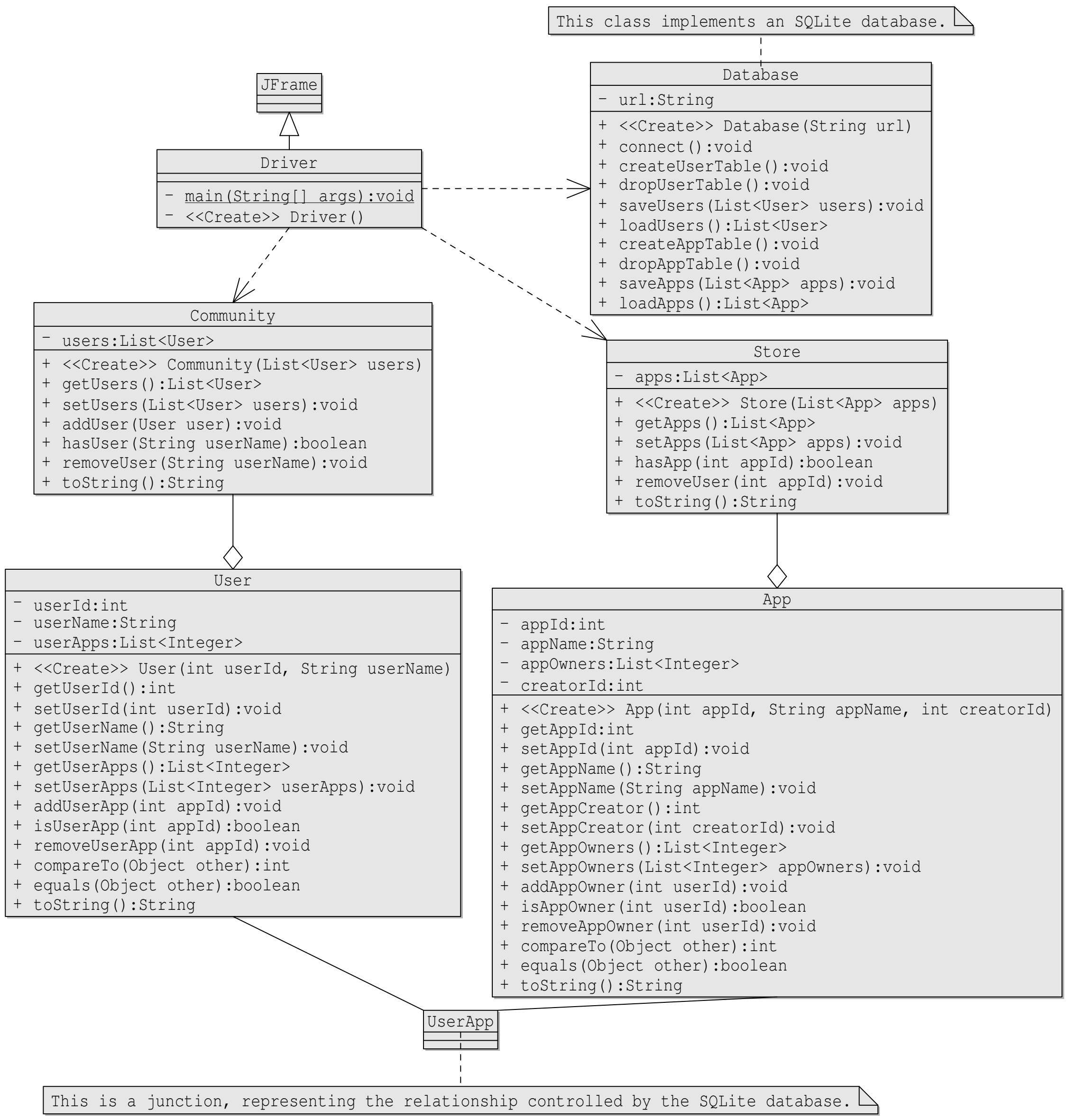
This class implements an SQLite database.
Database
- url:String
+ <<Create>> Database(String url)
+ connect():void
+ createUserTable():void
+ dropUserTable():void
+ saveUsers(List<User> users):void
+ loadUsers():List<User>
+ createAppTable():void
+ dropAppTable():void
+ saveApps(List<App> apps):void
+ loadApps():List<App>
JFrame
Driver
- main(String[] args):void
- <<Create>> Driver()
Community
- users:List<User>
+ <<Create>> Community(List<User> users)
+ getUsers():List<User>
+ setUsers(List<User> users):void
+ addUser(User user):void
+ hasUser(String userName):boolean
+ removeUser(String userName):void
+ toString():String
Store
- apps:List<App>
+ <<Create>> Store(List<App> apps)
+ getApps():List<App>
+ setApps(List<App> apps):void
+ hasApp(int appId):boolean
+ removeUser(int appId):void
+ toString():String
User
- userId:int
- userName:String
- userApps:List<Integer>
+ <<Create>> User(int userId, String userName)
+ getUserId():int
+ setUserId(int userId):void
+ getUserName():String
+ setUserName(String userName):void
+ getUserApps():List<Integer>
+ setUserApps(List<Integer> userApps):void
+ addUserApp(int appId):void
+ isUserApp(int appId):boolean
+ removeUserApp(int appId):void
+ compareTo(Object other):int
+ equals(Object other):boolean
+ toString():String
App
- appId:int
- appName:String
- appOwners:List<Integer>
- creatorId:int
+ <<Create>> App(int appId, String appName, int creatorId)
+ getAppId:int
+ setAppId(int appId):void
+ getAppName():String
+ setAppName(String appName):void
+ getAppCreator():int
+ setAppCreator(int creatorId):void
+ getAppOwners():List<Integer>
+ setAppOwners(List<Integer> appOwners):void
+ addAppOwner(int userId):void
+ isAppOwner(int userId):boolean
+ removeAppOwner(int userId):void
+ compareTo(Object other):int
+ equals(Object other):boolean
+ toString():String
UserApp
This is a junction, representing the relationship controlled by the SQLite database.

Figure 1: UML Architecture Diagram

3 Application Views

See Figure 2 for application view. The application will have the following pages:

- Login Splash
- Main Menu
- Community
- Store

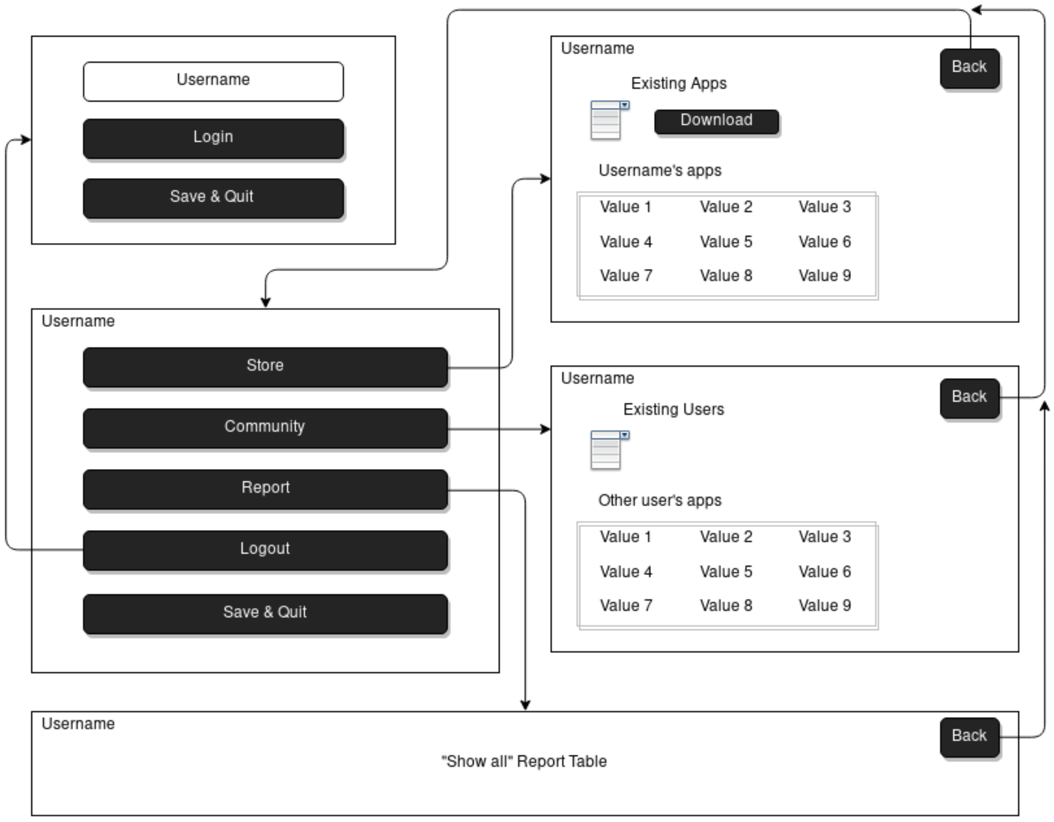

Figure 2: Views Diagram

3.1 Login Splash

The login splash page will have a username field and a login button. Additionally this page will have a quit button.

3.2 Main Menu

The main menu view will have the following buttons:

- Store — Clicking this button will change the view to the store view.
- Community — Clicking this button will change the view to the community view.
- Report — Clicking this button will open a dialogue box with the report view.
- Logout — Clicking this button will return the view to the login splash view.
- Save & Quit — Clicking this button will store all data in the SQLite database and exit the application.

3.3 Store

The store view allows the current user to add or remove apps from their inventory. This view will also provide a list of the current inventory.
The store view consists of the following components:

- Label — name of current user.
- Combo box — contains available apps.
- App add button — adds the currently selected item in combo box to inventory if not already therein.
- Inventory list box — contains a list of current apps associated with the current user.
- Delete selected app button
- Make review button — opens option pane with the following:
  - Combo box
  - Save review button
- Back button — returns to main menu view.

3.4 Community

This view will also a user to view the apps owned other users and the potential reviews of those apps.

- Label — name of current user.
- Combo box — contains list of all other users.
- App inventory list box — contains a list of the selected user's apps.
- Back button — returns to main menu view.

3.5 Report

Table 1 contains an example "show all" report of the apps owned by each user. Table 2 contains an example "show all" report of the apps owned by each user. The report view will have two sections showing (1) the apps owned by each user, and (2) the users that own each app.

userId	userName	appId	appName	review
1	"Amelia47"	1	"Flappy Bird"	"This game is so addicting. . . guh!"
1	"Amelia47"	5	"Chess"	null
1	"Amelia47"	23	"Angry Birds"	"Old but gold!"
4	"William123"	3	"Battleship"	null
5	"Ashley"	3	"Battleship"	null
5	"Ashley"	5	"Chess"	"Way too hard!"

Table 1: User database information

4 Class Descriptions

4.1 Driver

4.1.1 Description

The Driver class creates an object of itself for JFrame. Additionally, it creates an instance of Community, Store, and Database. It passes the data from the database to the Community and Store on startup.

appId	appName	userId	userName	review
1	“Flappy Bird”	1	“Amelia47”	“This game is so addicting. . . guh!”
3	“Battleship”	4	“William123”	null
3	“Battleship”	5	“Ashley”	null
5	“Chess”	1	“Amelia47	null
5	“Chess”	5	“Ashley”	“Way too hard!”
23	“Angry Birds”	1	“Amelia47”	“Old but gold!”

Table 2: App database information

4.1.2 Methods

- public static void main(String[] args) — The main method for the project.
- private Driver() — Creates an instance of a JFrame inside of main

4.2 Database

4.2.1 Data

The class will only store a String that contains the url of the database.

4.2.2 Methods

- public Database() — creates a database object.
- public void connect() — connects to the database.
- public void createUserTable() — creates the user table.
- public void dropUserTable() — drops the user table from the database.
- public void saveUsers(List<User> users) — overwrites the existing table with the new arraylist.
- public List<User> loadUsers() — returns the user table as an arraylist.
- public void createAppTable() — creates the app table.
- public void dropAppTable() — drops the app table from the database.
- public void saveApps(List<App> apps) — overwrites the existing table with the new arraylist.
- public List<App> loadApps() — returns the app table as an arraylist.

4.3 Community

4.3.1 Data

The Community class’ sole instance field contains a list of User objects

4.3.2 Methods

- public Community(List<User> users) — creates a community object.
- public List<User> getUsers() — returns an arraylist of users, primarily for use in conjunction with the ’saveApps()’ method of the database class.
- public void setUsers(List<User> users) — replaces the existing class attribute ‘users’.
- public void addUser(User user) — adds a user to the community list.

- public void sort() — sorts the list using a merge sort.
- public boolean hasUser(String userName) — returns a boolean for if the username was found within the ‘users’ list.
- public User getUser(String userName) — returns the ‘User’ object which matches contains the implicit parameter ‘userName’ which matches the parameter.
- public void removeUser(String userName) — removes the ‘User’ object which username matches.
- public String toString() — returns the data as a String

4.4 Store

4.4.1 Data

Stashes a list of ‘App’ objects.

4.4.2 Methods

- public Store(List<App> apps) — creates a Store object
- public List<App> getUsers() — returns the arraylist of apps
- public void setApps(List<App> apps) — replaces ‘apps’ with passed list.
- public void addApp(App app) — adds an app object to the end of the list.
- public void sort() — sorts the list using a merge sort.
- public boolean hasApp(String appName) — returns a boolean for if the appName is found within the ‘apps’ list
- public User getApp(String appName) — returns the ‘App’ with the matching appName.
- public void removeApp(String appName) — removes ‘App’ with matching appName.
- public String toString() — returns the list of apps as a String.

4.5 User

4.5.1 Data

- int userId — the userId used for sorting within the database.
- String userName — the userName of the ‘User’ object.
- List<Integer> userApps — the list of apps the user owns.

4.5.2 Methods

- public User(int userId, String userName) — creates a ‘User’ object, with a userId based on size of ‘users’ list within community and a userName based on the users input passed from main.
- public int getUserId() — returns the userId of the ‘User’ object.
- public void setUserId(int userId) — replaces the userId with parameter.
- public int getUserName() — returns the username of the ‘User’ object.
- public void setUserName(String userName) — replaces the implicit userName with the parameter.
- public List<Integer> getUserApps() — returns the ‘userApps’ list

- public void setUserApps(List<Integer> userApps) — replaces the existing userApps list with the parameter.
- public void addUserApp(int userId) — adds the userId to the 'userApps' list.
- public boolean isUserApp(int appId) — returns true if appId is found within 'userApps'.
- public void removeUserApp(int appId) — removes appId from 'userApps' list.
- public int compareTo(Object other) — returns the difference of the implicit userId by other's userId.
- public boolean equals(Object other) — returns true if userId and userName Strings match.
- public String toString() — returns the 'User' object as a String, containing its instance fields.

4.6 App

4.6.1 Data

- int appId — the appId used for sorting within the database.
- String appName — the appName of the 'App' object.
- List<Integer> appOwners — the userIds of the 'User' objects that own the 'App' object.

4.6.2 Methods

- public App(int appId, String appName, int creatorId) — creates an 'App' object.
- public void int getAppId() — returns the implicit appId.
- public void setAppId(int appId) — sets the implicit appId to the parameter.
- public String getAppName() — returns the implicit appName.
- public void setAppName(String appName) — sets the implicit appName to the parameter.
- public List<Integer> getAppOwners() — returns the 'appOwners'.
- public void setAppOwners(List<Integer> appOwners) — sets the implicit 'appOwners' to the parameter.
- public void addAppOwner(int appId) — adds the appId to the 'appOwners' list.
- public boolean isAppOwner(int userId) — returns true if userId is found within 'appOwners'.
- public void removeAppOwner(int userId) — removes userId from 'appOwners' list if isAppOwner(userId) returns true.
- public int compareTo(Object other) — returns the difference of the implicit appId by other's appId.
- public boolean equals(Object other) — returns true if appId matches.
- public String toString() — Returns the object as a String.

5 Application Storage

The storage for this application will be SQLite. The following tables are required for basic functionality:

- user_tb — stores the userId and userName for the community.
- app_tb — stores the appId, appName, and the creator's userId for the store.
- userapp_tb — stores the relationship between each appId and userId along with an optional review.

The tables used within the database are depicted in Table 3, Table 4, and Table 5 below.

userId (int, primary key, & autoincrement)	userName (varchar (32), not null, & unique)
1	“Amelia47”
2	“1John”
3	“Samantha99”
4	“William123”
5	“Ashley”

Table 3: the user_tb table of the program's SQLite database

appId (int, primary key, & autoincrement)	appName (varchar (64), not null, & unique)	creatorId (int)
1	“Flappy Bird”	3
2	“Chess”	5
3	“Battleship”	23
4	“Learn Languages”	4

Table 4: the app_tb table of the program's SQLite database

appId (int)	userId (int)	review (varchar (2000))
23	1	“I like this game!”
1101	43	null
50	2643	null
23	2	“This game was alright. . . ”
4	703	null
2	4	“I love this game”

Table 5: the userapp_tb table of the program's SQLite database