プログラミング応用 自由課題

東 政澄 (33114001)　　苅谷楓 (33114037)　　仲田樹 (33114087)

2023 年 5 月 15 日

1　初めに

ここでは作成物の内容や課題制約の確認といった，コードの内容以外の要素について説明する．

1.1　作成物の説明

私たちが今回作成したものは作品リストデータベース,「worksmanager」である．ユーザは本プログラムに名前とパスワードを設定することによりアカウント登録を行う．そして，ユーザは視聴リストに「作品名，ジャンル，状態」を記入することでその情報がデータベースで管理される．ここでの「状態」とは,「見た，見ていない」などを想定している．ユーザの視聴リストには閲覧権限が設定されており，パスワードが一致しないユーザは閲覧できないようになっている．

1.2　課題制約の確認

それぞれの課題制約について，本プログラムでどのように組み込まれたかを説明する．

データベースの使用

今回，worksmanager.db にテーブルを動的に追加する形で使用する．まず，ユーザの情報をまとめる user というテーブルを作成する．このテーブルには「ID, 名前，パスワード」を保存する．次に，各ユーザごとに ID をテーブル名としたテーブルを用意する．このテーブルには「タイトル，ジャンル，状態」を保存する．

JUnit を用いたテストコードの作成

今回基盤となっているクラスは DB クラスとなっている．そのため DBTest クラスを実装し，JUnit を用いたテストコードを作成することにした．

デザインパターンの使用

今回，State パターン [1] と Singleton パターン [2] を使用した．

State パターンについて，前にも述べたように視聴リストにはアクセス権が存在する．よって，アクセス権があるユーザと無いユーザで処理を分ける必要があるため，この問題を State パターンを使うことで解決した．

Singleton パターンについて，通常のプログラムだとアクセス権のあるクラスのインスタンスを外部のクラスで生成できてしまう．そこで，Singleton パターンを使うことで，インスタンス生成のリソースを抑えつつセキュリティを強化することとした．

Java17 新機能の使用

今回，外部からの不正アクセス防止に重点を置いているため Java17 の機能 [3] として sealed クラスを採用する．

プログラム実行方法と実行例

プログラムを実行する際は worksmanager ディレクトリと同階層で make コマンドを実行すれば実行できる．以下の図 1 に実行例を示す．

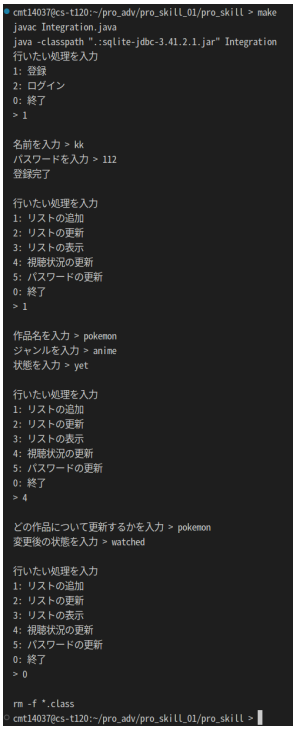

図 1　実行方法と実行例

2　ソースコードの説明

ここでは具体的なソースコードの説明を省き，各クラス・インタフェース・メソッドの役割を説明していく．なお，本ソースコードは GitHub[4] に上がっている．

DB クラス

sqlite3 によるデータベースの操作をプログラマ目線から容易に行えるようにするために実装したユーティリティクラスである．ユーティリティクラスのため，すべてのメソッドが static final かつ，コンストラクタが private である．

Connection getCon()

JDBC ドライバをロードしデータベースへの接続を行い，その Connection を返す．今後，DB クラス内のすべてのメソッドは実行前に getCon() メソッドを呼び出す．

void createTable

sqlite3 での”create table”コマンドに該当する．今回のコードでは，user テーブルとユーザ名が名前のテーブル (以降，単に name テーブルと呼ぶ) を用いる．以下の表 1,2 に各テーブルのデータ型とカラム，および具体例を示す．

表 1　user テーブルの構造と具体例

id (text)	name (unique text)	pass (text)
874488fa-040c-41c9	hoge	1234
39e99708-8588-40f9	fuga	1111

表 2　name テーブルの構造と具体例

title (unique text)	genre (text)	state (text)
cats fight	movie	未視聴
人間失格	book	視聴済み

user テーブルはユーザの ID，名前，パスワードからなる．名前は本プログラム上では重複を許さないものとして unique 制約を付けている．

name テーブルは user テーブルの name の名前が付けられ，作品名，ジャンル，状態からなる．このテーブルが各ユーザに与えられる視聴リストに該当する．作品名は重複しないため unique 制約を付けている．

int insertRow(String tableName, ArrayList<String>data)

sqlite3 の insert コマンドに該当する．与えられた data の長さが不適であったり，data の要素に重複が見られる場合はエラーとして-1 を返す．

data の長さが 2 の場合は user テーブルの要素挿入を行う．id は java.util.UUID クラスを用いて設定し，残りのカラムはデータ内の要素を挿入する．

data の長さが 3 の場合は id テーブルの要素挿入を行う．data の要素を name テーブルのカラムに挿入する．

void updateData(String tableName, String key, String after)

sqlite3 の update コマンドに該当する．今回のプログラムでは user テーブルの pass カラム，id テーブルの state カラムの 2 つの変更しか用意する必要が無いため，本来 5 引数必要とするところを 3 引数まで落とし，プログラマの負担を軽減した．

void deleteTable(String tableName)

sqlite3 の drop コマンドに該当する．tableName というテーブルを削除する．

void deleteRow(String tableName, String column, String value)

sqlite3 の delete コマンドに該当する．tableName というテーブルの column が value である行を削除する．

ArrayList<String[]>getData(String tableName, String column, String value)

sqlite3 の select コマンドに該当する．tableName というテーブルの column が value である全ての行を取得する．二重の ArrayList を返してもよかったが，user テーブルも name テーブルも 3 要素と固定であるため，要素を String 型配列とした．

DBTest クラス

DB クラスの各メソッドについて，単体テストを行うクラスである．ソースコードの上から順次実行するテストコードとなっている．ソースコード自体の説明は冗長となるため省略する．

Process インタフェース

先に述べたように本プログラムではアクセス権のあるユーザと無いユーザで処理の振る舞いを変える．そのため State パターンを採用する．このインタフェースは State パターンのためのものである．今後のクラスで各メソッドの実装を行うため，ここではメソッドの説明を省略する．

ProcessPattern クラス

State パターンのためのクラスである．アクセス権がある場合と無い場合で，インスタンスもアクセス権のあるものと無いものが生成される．

メソッドがいくつかあるが，State パターンに関係のあるもののみ説明し，残りは後に説明する．

void changeToAccess()

アクセス権のあるインスタンスを生成する．コード中の HasAccessProcess がアクセス権のある場合の処理をまとめたクラスである．

void changeToNoAccess()

アクセス権のないインスタンスを生成する．コード中の NoAccessProcess がアクセス権のない場合の処理をまとめたクラスである．

HasAccessProcess クラス

アクセス権のある処理をまとめたクラスである．他のクラスでインスタンス生成させないように Singleton パターンを採用し，セキュリティを高めている．

HasAccessProcess getProcess()

Singleton パターンを使用したインスタンス生成メソッドである．外部でこのメソッドが実行されても HasAccessProcess クラス内でしか生成されない．

void update(String table, String after, String column)

DB クラスの updateData メソッドを用いたデータ変更メソッドである．

int insertRpw(String tableName, ArrayList<String>data)

DB クラスの insertRow メソッドを用いたデータ挿入メソッドである．

deleteRow(String tableName, String column, String value)

DB クラスの deleteRow メソッドを用いた行削除メソッドである．

deleteTable(String table)

DB クラスの deleteTable メソッドを用いたテーブル削除メソッドである．

ArrayList<String[]>getData(String tableName, String column, String value)

DB クラスの getData メソッドを用いたデータ取得メソッドである．

void setPass(String NewPass, String ID)
DB クラスの updateData メソッドを用いたパスワード変更メソッドである．

NoAccessProcess クラス

アクセス権のない処理をまとめたクラスである．他のクラスでインスタンス生成させないように Singleton パターンを採用し，セキュリティを高めている．メソッドは HasAccessProcess クラスにあるものとほぼ同じであるが，アクセス権がない旨を伝える printMSG メソッドのみが実行される．

Integration クラス

これまでに作成したクラスを使って処理の本体を実装するクラスである．

void process()
実行することで本体処理を行うメソッドである．実行後，登録かログインを行い，希望の処理を行う．

void registrationUser()
ユーザの情報登録を行うメソッドである．名前とパスワードを入力し登録する．

void login()
登録済みユーザのログインを行うメソッドである．registrationUser() と同様に名前とパスワードを入力し登録する．

boolean check()
本人か否かを真理値で返すメソッドである．本人の名前と登録済みのパスワードを照合して一致すれば true，しなければ false を返す．それぞれの場合で適した ProcessPattern インスタンスを生成する．

void add()
作品・ジャンル・状態を入力し，視聴リストにその情報を追加するメソッドである．

void update()
視聴リストの状態の部分を変更するメソッドである．

void printList()

ジャンルまたは状態の値を入力し，それに一致する視聴リストを表示する．

参考文献

[1] 19. State パターン，techscore，https://www.techscore.com/tech/DesignPattern/State，(2023.4.18 閲覧)
[2] 5. Singleton パターン，techscore，https://www.techscore.com/tech/DesignPattern/Singleton，(2023.4.18 閲覧)
[3] Java 17 新機能まとめ，Qiita，https://qiita.com/nowokay/items/ec58bf8f30d236a12acb，(2023.4.25 閲覧)
[4] GitHub，https://github.com/kakao6062/pro_skill