

一つの java ファイルにつき、一つの GUI が割り振られている。それぞれの java ファイルの内容を GUI と共に説明する。また、プログラムの説明は、メソッドごとに行う。さらに、java ファイルと同じディレクトリに system.sqlite というデータベースを置き

```
connection = DriverManager.getConnection("jdbc:sqlite:system.sqlite");
```

をプログラム各所に配置して system.sqlite とデータベース接続を行った。そのため、実行の際にはクラスパスの設定をする必要があったため

```
java -classpath ".:sqlite-jdbc-3.23.1.jar" Login
```

と入力する必要がある。この jar ファイルも同じディレクトリに配置してある。また、sqlite ファイルおよび sqlite ファイル初期化 sql ファイルも作成し、同ディレクトリに設置した。また、データベースの中に users、clothes、reservations というテーブルを作成した。それぞれ、ユーザー情報、服の情報、予約の情報を格納している。

1. Login.java

まず、Login.java ファイルで、GUI は以下のように設計した。

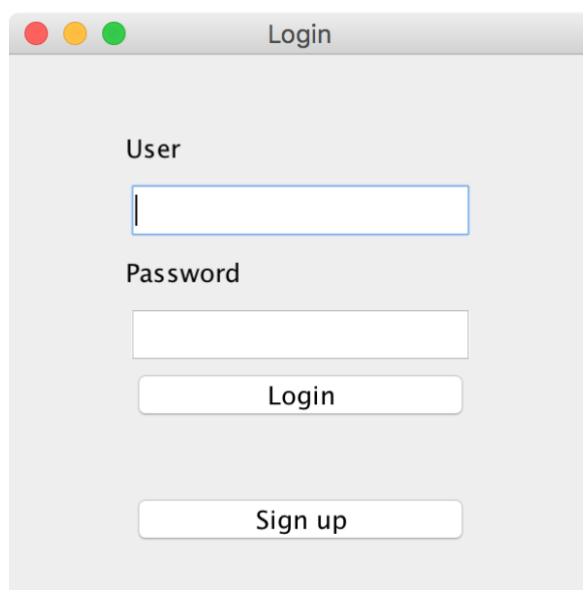


図 1 ログイン画面

さらに main 文をこのファイル内に書き、このファイルを実行の起点にした。まず、Login コンストラクタを説明する。

1.1. main 文

main 文では Login クラスの呼び出しを行い、さらに JFrame 型のグローバル変数 frame を表示することで Login 画面を表示し、コンストラクタの呼び出しを行った。

1.2. Login() コンストラクタ

コンストラクタでは GUI の設定を主に行っている。Swing の J~型の要素のインスタンスを生成し、図のように配置を行なっている。

1.3. LoginButtonAction クラス

6 のボタンが押された時の挙動を表す。これは、ユーザーネームとパスワードがそれぞれ users テーブルの user_name と password の両方に一致するものがあった場合、JFrame を親クラスに持つ MyRentedPage クラスの インスタンスを生成し、MyRentedPage を表示し、JFrame 型の frame 変数を隠している。これに当てはまらない場合、error 変数のテキストを”NOT FOUND”と設定し、下図のように表示するようにした。

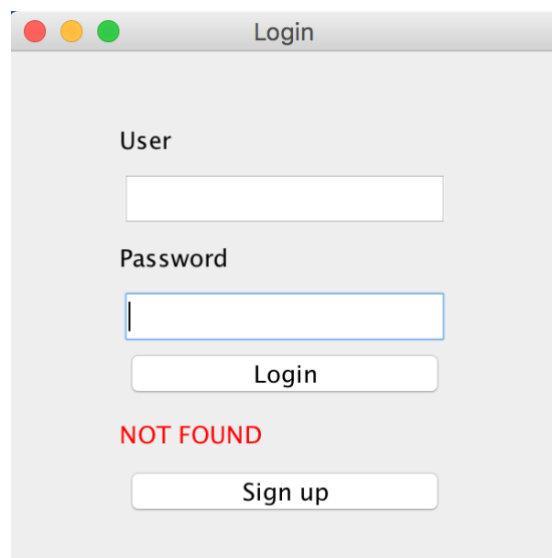


図2 ログイン画面（エラー時）

1.4. CreateAccountButtonAction クラス

7 のボタンを押した時の挙動を表していて、JFrame を親クラスに持つ SignUp クラスのインスタンス生成および、SignUp クラスを表示し、JFrame 型の frame 変数を隠す。

2. SignUp.java

2.1. SignUp() コンストラクタ

Login のコンストラクタと同様に GUI を設定している。図3のように配置した。

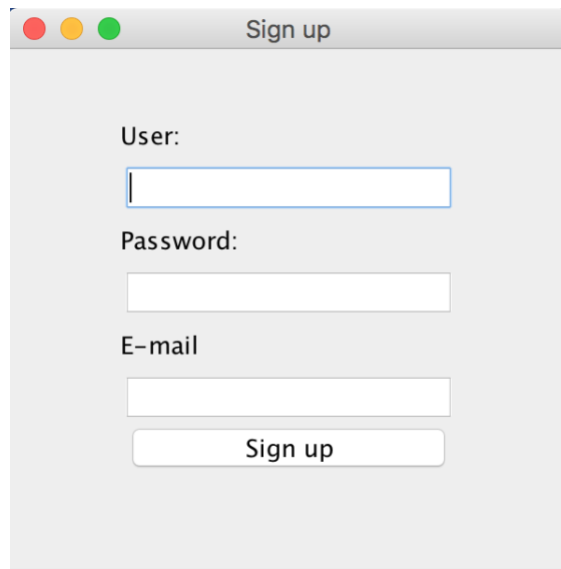
A screenshot of a 'Sign up' window. It has a title bar with three colored buttons (red, yellow, green) and the text 'Sign up'. The form contains three input fields: 'User:', 'Password:', and 'E-mail'. Below the 'E-mail' field is a 'Sign up' button.

図 3 登録画面

2.2. SignUpAction クラス

入力されたユーザーネーム、パスワード、および Email アドレスを入力し、SignUp ボタンで登録する。ユーザーネームが被っているときは下図のように表示するようになっている。これは、Login クラスの LoginButtonAction メソッドで”NOT FOUND”と表示している方法と同じである。

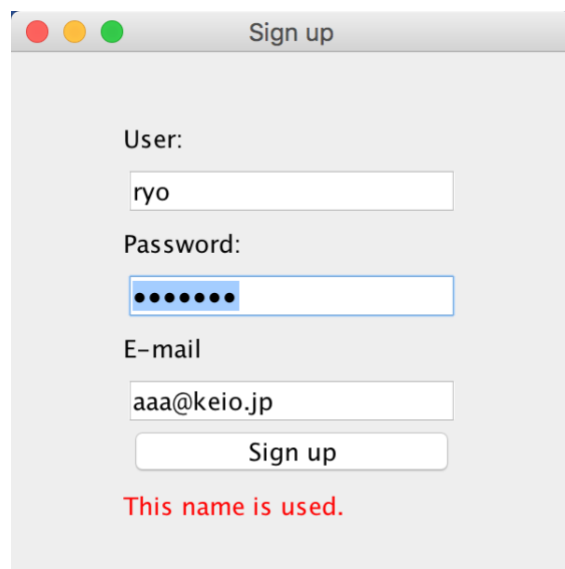
A screenshot of the 'Sign up' window showing an error state. The 'User:' field contains 'ryo', the 'Password:' field contains seven dots, and the 'E-mail' field contains 'aaa@keio.jp'. The 'Sign up' button is visible. Below the button, the text 'This name is used.' is displayed in red.

図 4 SignUp 画面（エラー時）

また、登録が無事行われると、下図のような画面になり、OK ボタンを押すと、Login クラスのインスタンス生成を行い、Login クラスを表示する。

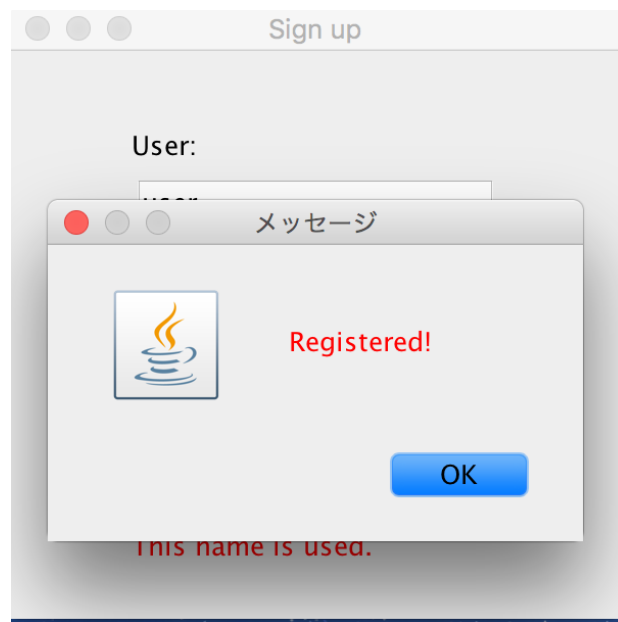


図 5 SignUp 画面（登録時）

3. MyRentedPage.java

このクラスは自分が借りたものがわかるページに関するものである。

3.1. MyRentedPage(int userId) コンストラクタ

コンストラクタでは図 6 のように配置する。

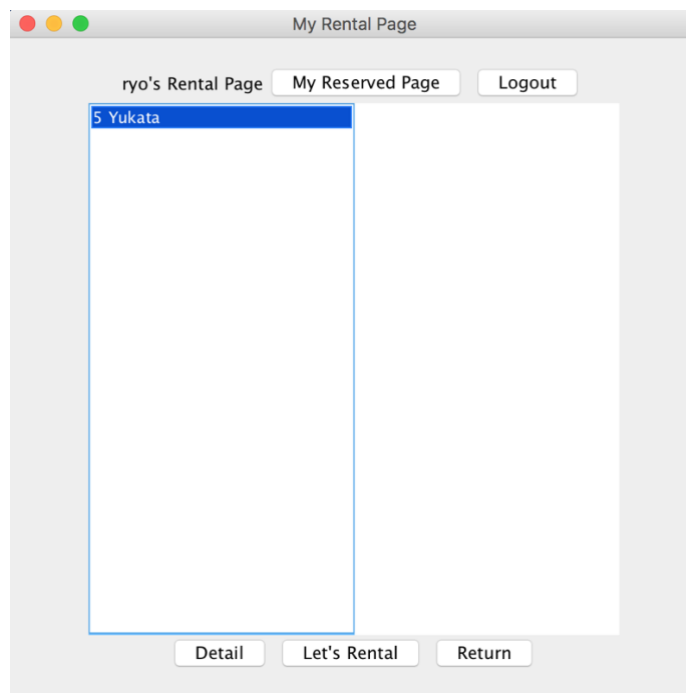


図 6 MyRentalPage 画面

真ん中のパネルに配置してある白い画面には左は JList、右は JTextArea を用いた。また、コンストラクタで user テーブルの id を受け取っている。また、コンストラクタで users テーブルとの user_name を userId から受け取り、画面上部に配置するようにしている。また、clothes テーブルと接続し、userId からそのユーザーが借りている衣装を JList 型で一つずつ表示させる。

3.2. LogoutButtonAction クラス

Login クラスのインスタンスを生成し、login の JFrame 型の frame 変数を表示する。さらに、MyRentedPage クラスを閉じる。

3.3. RentButtonAction クラス

”Let’s Rental”と書かれたボタンを押したときの挙動を示しており、ユーザ id を取得し、そのユーザー id を引数に Rental クラスのインスタンスを生成し、MyRentedPage クラスを閉じる。

3.4. ReservedButtonAction クラス

”My Reserved Page”と書かれたボタンを押したときの挙動を示しており、ユーザ id を取得し、そのユーザー id を引数に MyReservedPage クラスのインスタンスを生成し、MyReservedPage を開き、MyRentedPage クラスを閉じる。

3.5. DetailButtonAction クラス

”Detail”と書かれたボタンを押したときの挙動を示しており、まず、左の JList に選択されているものがないときは図 7 のようにエラーがでるようになっている。

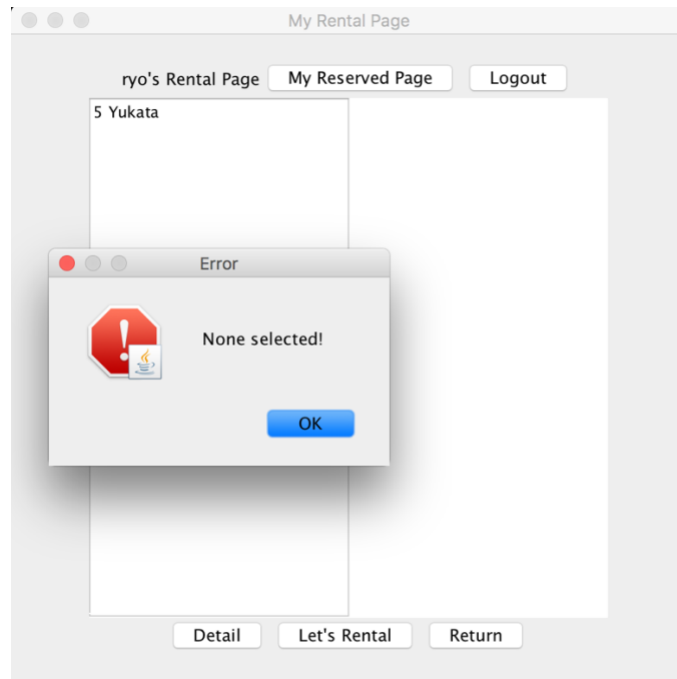


図 7 MyRentalPage 画面（選択なしエラー時）

List に選択がある場合は図 8 のようにその服の概要を右の JTextArea に表示する。これは服の id をもとに clothes テーブルにアクセスし、属性を取得していることで表示を行なっている。

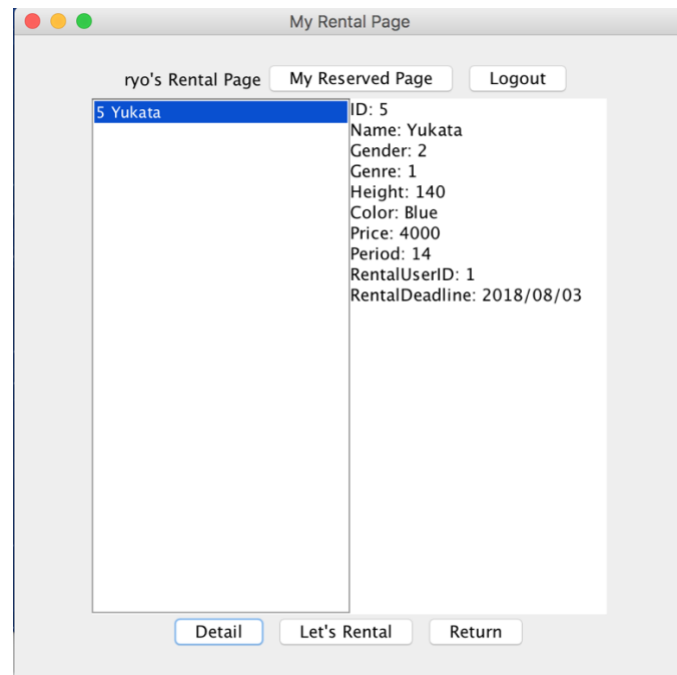


図 8 MyRentalPage 画面（Detail ボタンを押したとき）

3.6. ReturnButtonAction クラス

Return と書かれたボタンを押したときの挙動を示している。中では、返却された服の id をもとに、reservation テーブルから予約を全て取り出し、そのうちの一番上の予約を削除する。削除した予約のユーザ id を clothes テーブルの rental_user_id 属性に書き換える。さらに、時間を取得し、return_deadline 属性の日時と比べて 1 分でも過ぎていたときは図 9 上のように表示し、延滞金を徴収する。それ以外では図 9 下のように表示する。

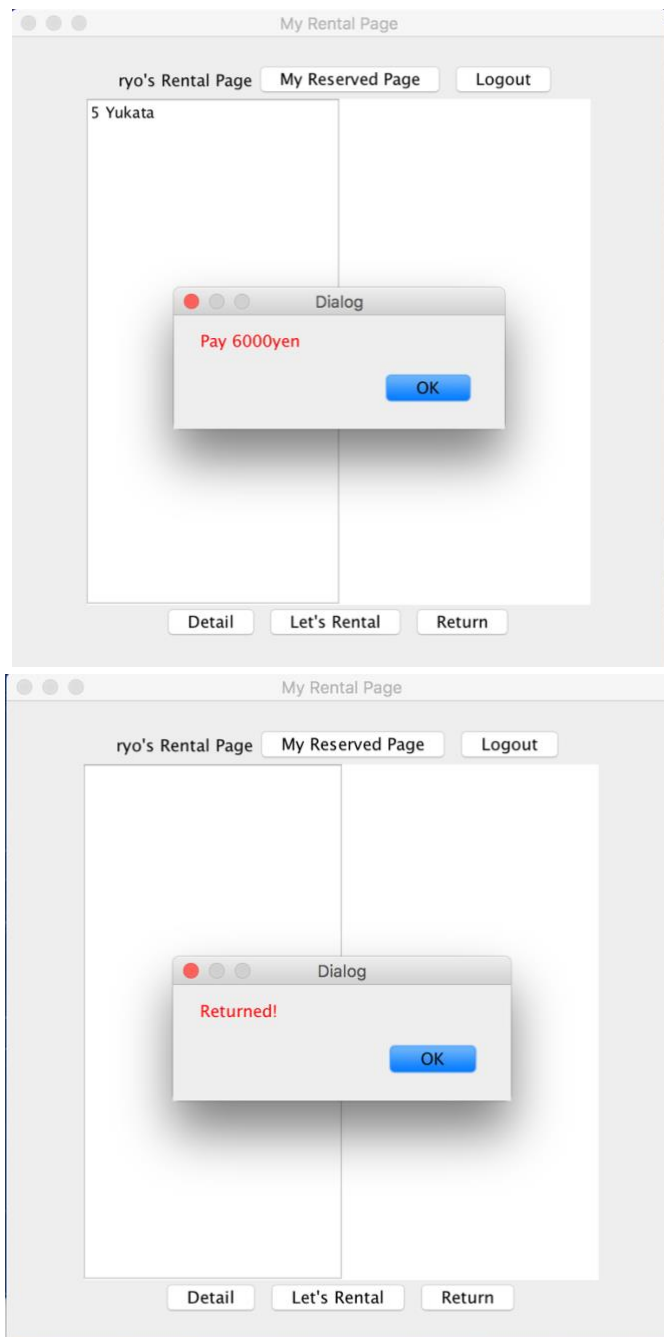


図 9 MyRentalPage 画面 (Return ボタンを押した時)

さらに、次の予約がある時は取得した時間に `period` テーブルの分だけ日を足したものを `clothes` テーブルの `return_deadline` 属性に更新している。

4. MyReservedPage.java

このページでは自分が予約したものがわかるようになっている。

4.1. MyReservedPage(int userId) コンストラクタ

コンストラクタでは図 6 のように配置している。

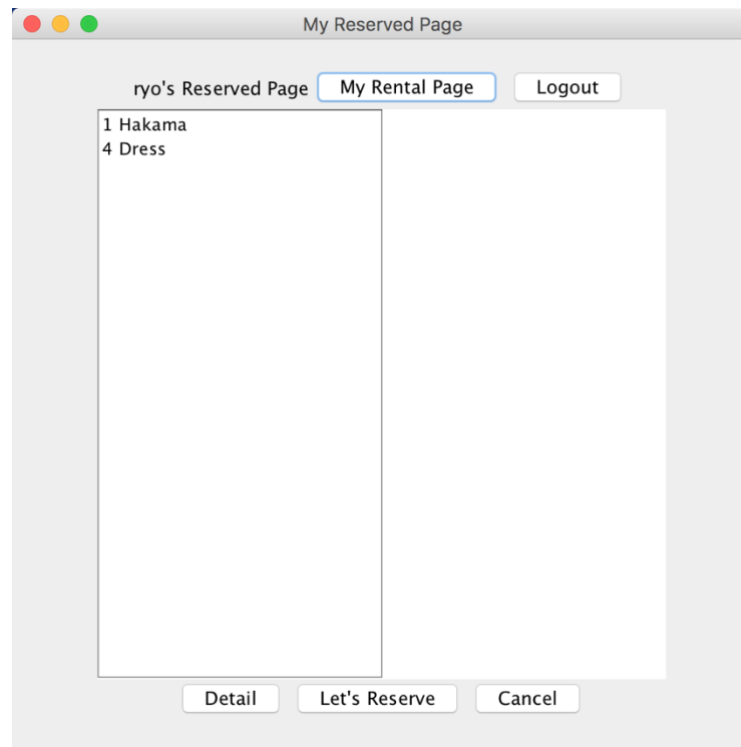


図 10 MyReservedPage 画面

真ん中のパネルに配置してある白い画面には左は `JList`、右は `JTextArea` を用いた。また、コンストラクタで `user` テーブルの `id` を受け取っている。また、コンストラクタで `users` テーブルとの `user_name` を `userId` から受け取り、画面上部に配置するようにしている。また、`clothes` テーブルと接続し、`userId` からそのユーザーが予約している衣装を `JList` 型で一つずつ表示させる。

4.2. LogoutButtonAction クラス

`Login` クラスのインスタンスを生成し、`login` の `JFrame` 型の `frame` 変数を表示する。さらに、`MyReservedPage` クラスを閉じる。

4.3. ReserveButtonAction クラス

”Let’s Reserves”と書かれたボタンを押したときの挙動を示しており、ユーザ id を取得し、そのユーザーid を引数に Reserve クラスのインスタンスを生成し、MyReservedPage クラスを閉じる。

4.4. RentedButtonAction クラス

”My Rented Page”と書かれたボタンを押したときの挙動を示しており、ユーザ id を取得し、そのユーザーid を引数に MyRentedPage クラスのインスタンスを生成し、開き、MyReservedPage クラスを閉じる。

4.5. DetailButtonAction クラス

”Detail”と書かれたボタンを押したときの挙動を示しており、まず、左の JList に選択されているものがないときは図 7 と同様のエラーがでるようになっている。

List に選択がある場合は図 8 と同様にその服の概要を右の JTextArea に表示する。これは服の id をもとに clothes テーブルにアクセスし、属性を取得していることで表示を行なっている。

4.6. CancelButtonAction クラス

Cancel と書かれたボタンを押したときの挙動を示している。中では、キャンセルされた服の id とユーザ id をもとに、reservation テーブルから予約を削除するものである。キャンセルが成功したら以下のポップアップが出る。

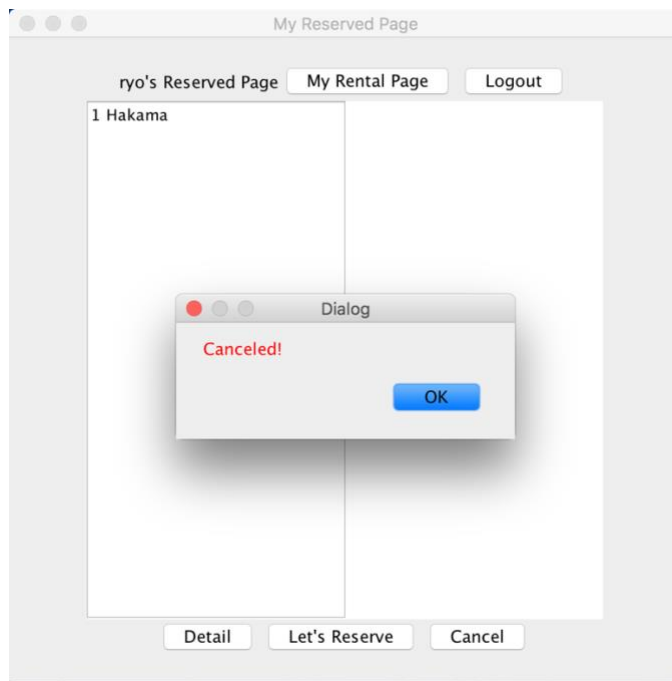


図 9 MyReservedPage 画面（Cancel ボタンを押した時）

5. Rental.java

実際に借りる動作を行うためのページである。

5.1 Rental(int id)コンストラクタ

userID を引数として起動するようになっている。画面のボタン配置を管理しているのはこの部分である。

5.2 SearchButton クラス

Search を押したときの挙動を定義している。上部の gender や genre を設定して Search すると、検索結果を絞ることができる。またテキスト入力による部分一致検索も可能。

5.3 RentalButton クラス

Rental を押したときの挙動を定義している。リストの服を選択した状態で Rental を押すと、Rental することができ、Rental 者がでたので、検索結果からは消えるようになる。選択していない状態で押すと警告がでる。

5.4 DetailButton クラス

リストの服を選択した状態で、Detail を押すと、その服に関する情報が下の欄に表示される。何も選択していない状態で押すと、警告が出る。

5.5 GenderCombo クラス

Gender に関するプルダウンを定義しているクラス。どのプルダウンが選択されているかで検索条件を変えなくてはならないため、今のインデックスを取得して、その番号から SQL で使う検索分を追加する役割。

5.6 GenreCombo クラス

基本的には 5.5 とほぼ同じである。こちらは Genre に関するプルダウンを管理している。

5.7 BackButton クラス

Back を押したときの挙動を定義している。レンタルのページを非表示にして、前のページを表示する仕組みになっている。

6. Reserve.java

実際に予約動作を行うためのページである。

6.1 Reserve(int id)コンストラクタ

userID を引数として起動するようになっている。予約の管理を行うスタック部分に userID は欠かせないので引数として受け取る。画面のボタン配置を管理しているのはこの部分である。

6.2 SearchButton クラス

Search を押したときの挙動を定義している。上部の gender や genre を設定して Search すると、検索結果を絞ることができる。またテキスト入力による部分一致検索も可能。Rental と同じである。

6.3 ReserveButton クラス

Reserve を押したときの挙動を定義している。リストの服を選択した状態で Reserve を押すと、Reserve することができる。Rental と違ってリストから消えないが、同じものを重複して予約しようとした場合は、警告が出るようになっている。選択していない状態で押すと警告がでる。

6.4 DetailButton クラス

リストの服を選択した状態で、Detail を押すと、その服に関する情報が下の欄に表示される。何も選択していない状態で押すと、警告が出る。Rental と同じである。

6.5 GenderCombo クラス

Gender に関するプルダウンを定義しているクラス。どのプルダウンが選択されているかで検索条件を変えなくてはならないため、今のインデックスを取得して、その番号から SQL で使う検索分を追加する役割。

6.6 GenreCombo クラス

基本的には 5.5 とほぼ同じである。こちらは Genre に関するプルダウンを管理している。

6.7 BackButton クラス

Back を押したときの挙動を定義している。このページを非表示にして、前のページを表示する仕組みになっている。

7. 作業分担方法

Login.java、SignUp.java、MyRentalPage.java、MyReservePage.java は私が作成し、Rental.java、Reserve.java は佃陽平くん(61612577)が作成するという形をとって作業を分担した。

8. 一緒にやって難しかったこと

まず、個人で別々に java ファイルを作成していて、お互いが可読性のあるコードになるように心がけて設計した。しかしながら、プログラムが複雑になったせいか達成できず、ファイルを統合して利用できるようにする際に自分と書き方が異なるソースコードを読んで統合しなければならなかったのが大変だった。同様にデバッグして誤っている点を見つけて、ペアのファイルに書き込むときも上記と同様に他人のソースコードを読むのが辛かった。これを考えると、データベース、GUI、機能という観点から役割分担する方がよかったのかもしれないと考えた。さらに、途中から佃くんと作業を分担することに決めたのだが、私が csv を、佃くんがデータベースを元として実装を行っていたために、途中から私が短い期間でデータベース実装に乗り換える必要があり、非常に苦労した。また、先述の通り唐突に作業を分担することになったので、お互いに考え方の相違や目標としている GUI やデータベースの統合をしなかったせいで、二人の java ファイルを統合した後に整合性を取るのがとても大変だった。これは GitHub や Slack など連絡を取り合い、ファイルや考え方を逐一共有することが必要だったかなと思った。ただ、社会に出てから共同作業で行うことが多くあると思うのでその経験になったと考える。

9. 参考文献 (()内は閲覧日時)

<https://github.com/xerial/sqlite-jdbc#usage> Github Sqlite JDBC Driver (2018/7/20)

<https://docs.oracle.com/javase/jp/8/docs/api/> Java SE8 API Document (2018/7/20)