MyHikingPal*

Jakub Martinák
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

13th May 2023

*Dokumentácia k OOP projektu, 2022/23, Cvičiaci: Ing. Tomáš Frťala, PhD.

Contents

1 Project Description

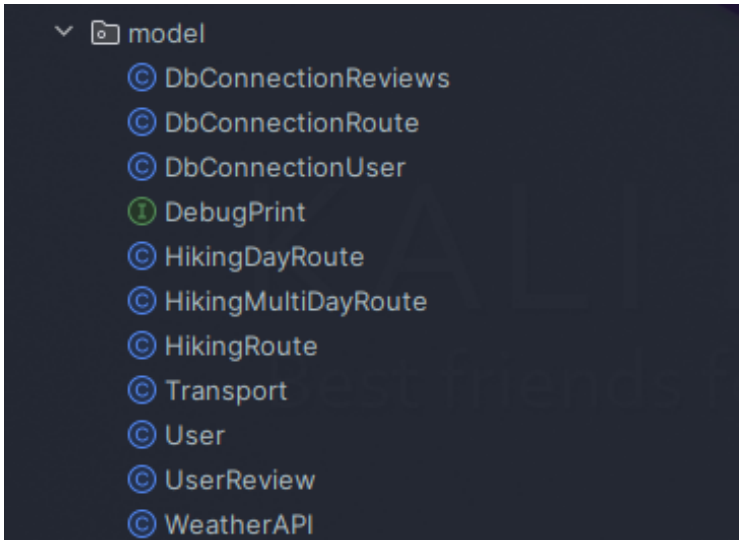
My Hiking Pal je aplikácia, ktorá poskytuje služby spájané s turistikou a samotným plánovaním cesty na turistiku. Užívateľ si môže vybrať z rôznych turistík, naplánovať/vybrať si spôsob prepravy, kde sa užívateľovi zobrazí mapa a všetky potrebné informácie o danej trase, a môže vyraziť na svoju želanú cestu. Aplikácia sleduje postup užívateľa po turistickej trase, má možnosť sa zastaviť počas turistiky aby si užívateľ oddýchol a následne pokračoval ďalej. Po dokončení turistiky môže daný užívateľ poskytnút recenziu na danú trasu pre ostatných užívateľov. Taktiež aplikácia zaznamenáva všetky podstúpené turistiky a všetky výdavky spojené s ňou.

2 Projektová Štruktúra

Nasledujúce sekcie reprezentujú najdôležitejšie časti aplikácie.

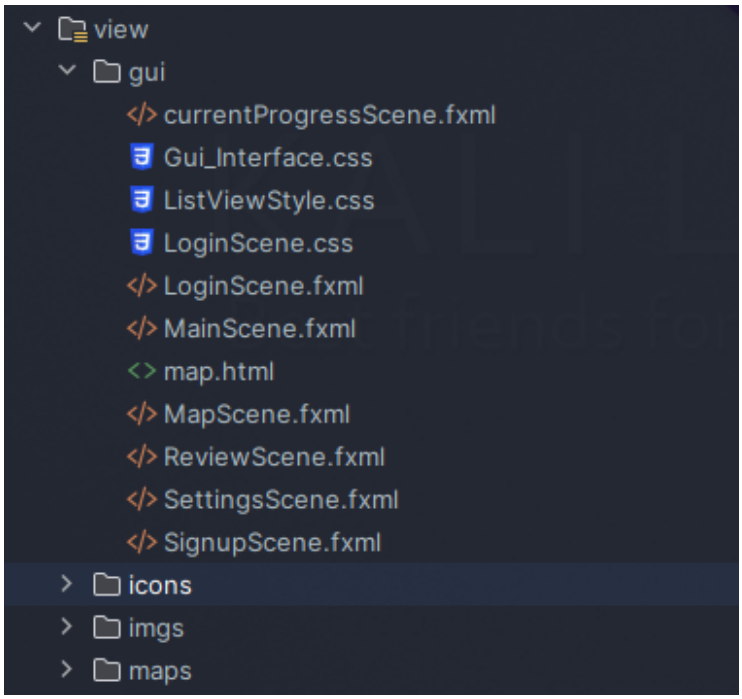
2.1 Model

Tento package obsaguje celú aplikačnú logiku, ktorá sa stará o chod aplikácie ako backend. Nachádza sa tu trieda User.java, kde je vytvorená inštancia užívateľa, ktorá sa stará o všetky informácie, ktoré sa týkajú užívateľa. Taktiež sa tu nachádzajú súbory na prípajanie k databáze, a súbory na staranie sa o dané turistické trasy.



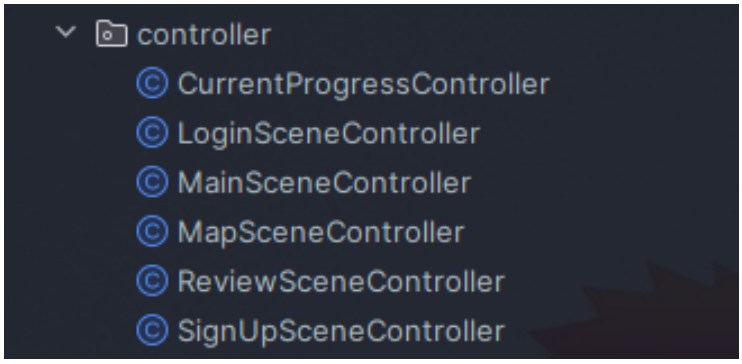
2.2 View

View package je nastavený ako zdroj(GUI, ikony, ...) pre maven infraštruktúru. Nachádzajú sa tu všetky fxml súbory, obrázky, ikony a css súbory.



2.3 Controller

Tento package obsahuje všetky triedy na kontrolovanie fxml súborov a scén.



2.4 Database

Tento package obsahuje databázy vo formáte sqlite, nachádzajú sa tu databázy pre users, routes a reviews.

2.5 Class HikeRoute

Trieda Hike Route sa stará o vrátenie dát z databázy o daných turistických trasách. V tejto triede je volaná Connection na routes databázu, a následne sa vracajú z tejto triedy všetky potrebné informácie.

2.6 Class User

Táto trieda využíva vzor Singleton, a stará sa o aktualizáciu informácií o užívateľoch a vracaní informácií o nich.

2.7 Class User Review

Táto trieda sa stará čisto iba o aktualizovanie a vracanie dát z databázy ohľadom recenzií daných užívateľov.

2.8 AspectJ

Trieda ExecutorAspect sa stará o logovanie počas toho, keď je užívateľ na turistickej trase a sleduje či sa daná niť neskončila.

2.9 CurrentProgress Controller

Táto trieda využíva viacniťovosť pre simuláciu užívateľa na turistickej trase.

3 Splenenie kritérií

3.1 Hlavné kritéria

3.1.1 Dedičnosť

Príkladom je trieda HikeDayRoute, ktorá dedí z triedy HikeRoute a následne, HikeMultiDayRoute dedí z triedy HikeDayRoute. Týmto sa dosahuje viacnásobné dedenie.

```
10 usages  1 inheritor  JackobMartinak
public class HikingDayRoute extends HikingRoute {       //! Inheritance
    4 usages
    private static HikingDayRoute instance = null;
```

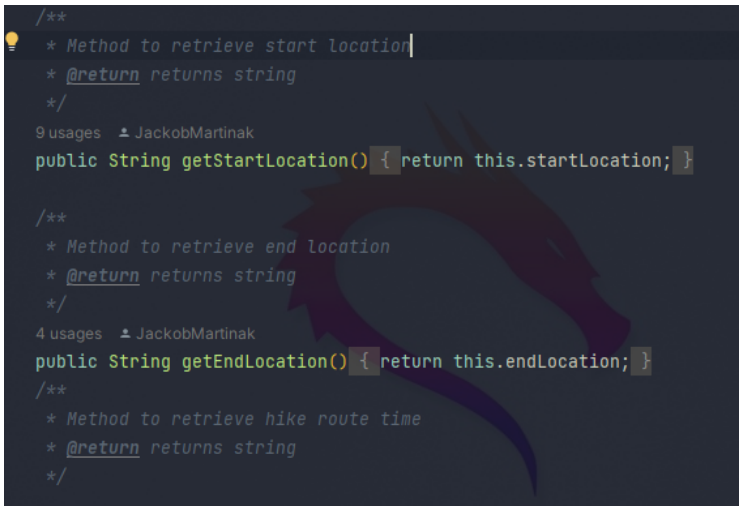
3.1.2 Polymorfizmus

Metóda getRouteInfo sa používa aj v HikeDayRoute.

```
1 usage  JackobMartinak
@Override
public HikingRoute getRouteInfo(String startLocation,  String endLocation){      //! Polymorphism
    String sql = "SELECT * FROM route where startLocation = ? AND endLocation = ?";

    try
    {
        connection = DbConnectionRoute.getInstance().getConnection();
        PreparedStatement pstmt  = connection.prepareStatement(sql);
        pstmt.setString( parameterIndex: 1, startLocation);
        pstmt.setString( parameterIndex: 2, endLocation);
        ResultSet rs  = pstmt.executeQuery();

        this.startLocation = rs.getString( columnLabel: "startLocation");
        this.endLocation = rs.getString( columnLabel: "endLocation");
        this.routeTime = rs.getString( columnLabel: "Time");
        this.distance = rs.getFloat( columnLabel: "lenght");
        this.difficulty = rs.getString( columnLabel: "difficulty");
        this.map = rs.getString( columnLabel: "map");
          this.Multiday = rs.getInt("Multiday");

        System.out.printf("[DEBUG] === GET %s ROUTE === \n", startLocation);

        return this;
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        return null;
    }
      System.out.println("The chosen location is .....")
}
```

3.1.3 Zapúzdrenie

Zapúzdrenie je používane pozdĺž celého projektu, keďže väčšina tried používa getteri a setteri. Príklad je trieda HikingRoute.



3.1.4 Agregácia

Agregácia je využitá v triede UserReview, kde na vytvorenie review objektu je potrebná HikingRoute a zároveň Inštancia User.

3.2 Vedľajšie kritéria

3.2.1 Dizajnové Vzory

Hlavný použitý vzor v projekte je MVC (Model, View, Controller), ktorý oddeľuje aplikačnú logiku od Užívateľského rozhrania. Model obsahuje aplikačnú logiku, View obashuje všetko čo užívateľ vidí, a Controller obsahuje všetky triedy, ktoré sa starajú a ovládajú fxml súbory.

3.2.2 Vlastné výnimky

Aplikácia má jednú vlastnú výnimku, NothingSelectedException je výnimka, ktorá sa stará o to, keď užívateľ nevyberie všetky potrebné údaje v MapSceneControlleri.

```java
/**
 * Exception class for when nothing is selected
 */
5 usages  JackobMartinak
static class NothingSelectedException extends Exception {
    3 usages  JackobMartinak
    public NothingSelectedException(String message) { super(message); }
}
```

3.2.3 GUI oddelené od logiky

Bola zodpovedaná vyšie pomocou MVC dizajnového vzoru.

3.2.4 Viacnitovosť

Viacnitovosť je použitá v simulácií turistickej trasy, kde sa jedna niť stará o sledovanie a inkrementovanie progressu, a druhá niť sa stará a sleduje či užívateľ si chce oddýchnuť alebo pokračovať.

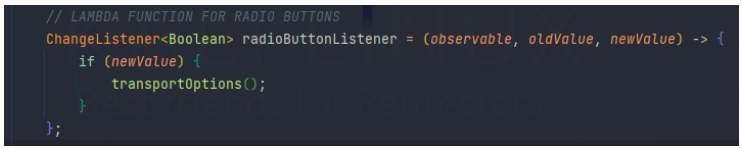
```java
public void progressThread() {
    ExecutorService executorService = Executors.newFixedThreadPool( nThreads: 5);
    JackobMartinak
    Runnable task = new Runnable() {
        JackobMartinak
        @Override
        public void run() {
            resumeButton.setDisable(true);
            if(!mapButton.isDisabled() & !mainSceneButton.isDisabled() & !submitButton.isDisablee
                mapButton.setDisable(true);
                mainSceneButton.setDisable(true);
                submitButton.setDisable(true);
                reviewArea.setDisable(true);
            }
            System.out.println("Running on a New Thread");
            for (int i = 0; i <= 100; i++) {
                try{
                    progressProperty.setValue(i / 100.0);
                    semaphore.acquire();
                    semaphore.release();
                    try {
                        Thread.sleep( millis: 500);  // Simulate work
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                    }

                      int finalI = i;
                    Platform.runLater(() -> progressPercentage.setText(String.format("%.2f", prog
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
            if(mapButton.isDisabled() & mainSceneButton.isDisabled() & submitButton.isDisabled()
                mapButton.setDisable(false);
                mainSceneButton.setDisable(false);
                submitButton.setDisable(false);
                reviewArea.setDisable(false);
            }
            if(!executorService.isShutdown()){
                System.out.println("Shutting down");
                executorService.shutdown();
            }
```

3.2.5 Vhniezdenie tried

Vhniezdenie tried je použité v triede MapSceneController kde sa náchádza aj vlastná výnimka, čo spolu tvorí vhniezdenie tried.

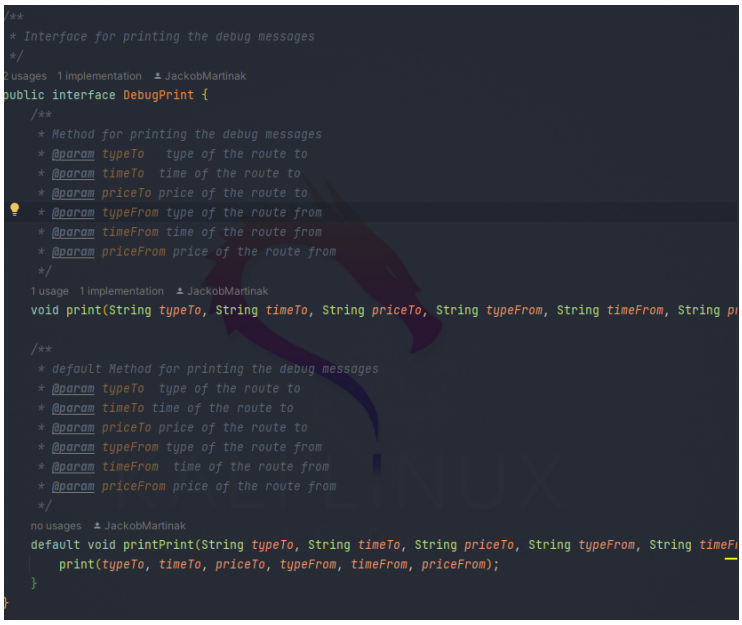
3.2.6 Lambda funkcie

Lambda funkcie sú použité na sledovanie prekliknutia radio tlačítka v MapSceneControlleri. A taktiež je ešte jedna použitá na vypisovanie debugu v triede Transport.



3.2.7 Predvolené metódy v rozhraniach

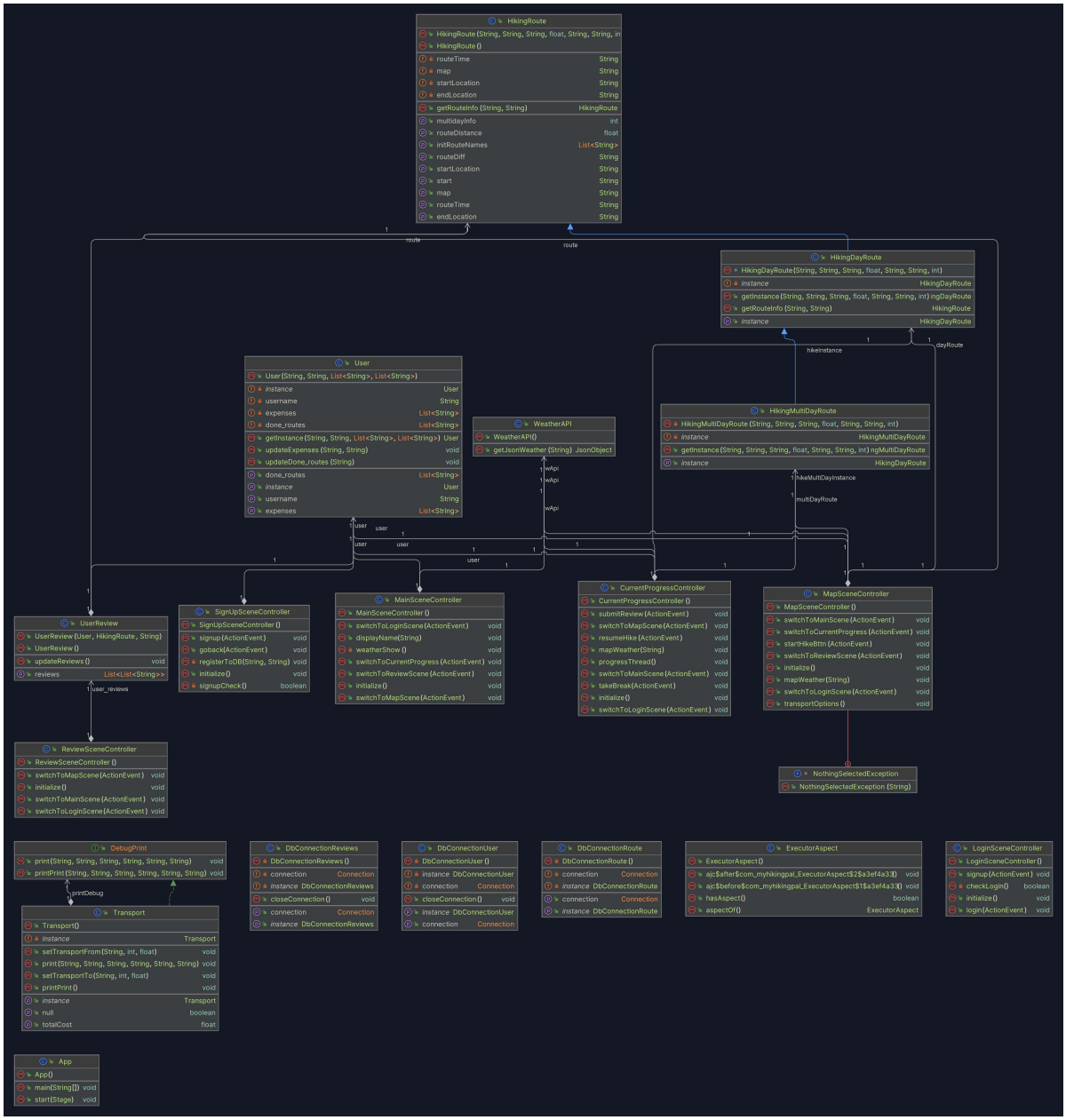
Rozhranie DebugPrint, ktorý je použitý v triede transport, má predvolenú metódu printPrint, ktorá volá metódu na vypisovanie.



3.2.8 Aspektovo-orientované programovanie (AspectJ)

AspectJ je použitý na logovanie začiatku a konca použitia novej nite pri simulácií progressu užívateľa.

4 Diagram


HikingRoute
HikingRoute (String, String, String, float, String, String, in
HikingRoute ()
routeTime String
map String
startLocation String
endLocation String
getRouteInfo (String, String) HikingRoute
multidayInfo int
routeDistance float
initRouteNames List<String>
routeDiff String
startLocation String
start String
map String
routeTime String
endLocation String
route
route
HikingDayRoute
HikingDayRoute(String, String, String, float, String, String, int)
instance HikingDayRoute
getInstance (String, String, String, float, String, String, int) ngDayRoute
getRouteInfo (String, String) HikingRoute
instance HikingDayRoute
hikeInstance
dayRoute
User
User(String, String, List<String>, List<String>)
instance User
username String
expenses List<String>
done_routes List<String>
getInstance (String, String, List<String>, List<String>) User
updateExpenses (String, String) void
updateDone_routes (String) void
done_routes List<String>
instance User
username String
expenses List<String>
WeatherAPI
WeatherAPI()
getJsonWeather (String) JsonObject
wApi
wApi
wApi
HikingMultiDayRoute
HikingMultiDayRoute (String, String, String, float, String, String, int)
instance HikingMultiDayRoute
getInstance(String, String, String, float, String, String, int) ngMultiDayRoute
instance HikingDayRoute
hikeMultiDayInstance
multiDayRoute
user
user
user
user
user
UserReview
UserReview (User , HikingRoute , String)
UserReview ()
updateReviews () void
reviews List<List<String>>
user_reviews
SignUpSceneController
SignUpSceneController ()
signup (ActionEvent) void
goback(ActionEvent) void
registerToDB(String, String) void
initialize() void
signupCheck() boolean
MainSceneController
MainSceneController ()
switchToLoginScene (ActionEvent) void
displayName(String) void
weatherShow () void
switchToCurrentProgress (ActionEvent) void
switchToReviewScene (ActionEvent) void
initialize() void
switchToMapScene (ActionEvent) void
CurrentProgressController
CurrentProgressController ()
submitReview (ActionEvent) void
switchToMapScene (ActionEvent) void
resumeHike (ActionEvent) void
mapWeather(String) void
progressThread() void
switchToMainScene (ActionEvent) void
takeBreak(ActionEvent) void
initialize() void
switchToLoginScene (ActionEvent) void
MapSceneController
MapSceneController ()
switchToMainScene (ActionEvent) void
switchToCurrentProgress (ActionEvent) void
startHikeBttn (ActionEvent) void
switchToReviewScene (ActionEvent) void
initialize() void
mapWeather(String) void
switchToLoginScene (ActionEvent) void
transportOptions () void
ReviewSceneController
ReviewSceneController ()
switchToMapScene (ActionEvent) void
initialize() void
switchToMainScene (ActionEvent) void
switchToLoginScene (ActionEvent) void
NothingSelectedException
NothingSelectedException (String)
DebugPrint
print(String, String, String, String, String, String) void
printPrint(String, String, String, String, String, String) void
printDebug
Transport
Transport()
instance Transport
setTransportFrom(String, int, float) void
print(String, String, String, String, String, String) void
setTransportTo(String, int, float) void
printPrint() void
instance Transport
null boolean
totalCost float
DbConnectionReviews
DbConnectionReviews()
connection Connection
instance DbConnectionReviews
closeConnection() void
connection Connection
instance DbConnectionReviews
DbConnectionUser
DbConnectionUser()
instance DbConnectionUser
connection Connection
closeConnection() void
instance DbConnectionUser
connection Connection
DbConnectionRoute
DbConnectionRoute()
connection Connection
instance DbConnectionRoute
connection Connection
instance DbConnectionRoute
ExecutorAspect
ExecutorAspect()
ajc$after$com_myhikingpal_ExecutorAspect2a3ef4a33() void
ajc$before$com_myhikingpal_ExecutorAspect1a3ef4a33() void
hasAspect() boolean
aspectOf() ExecutorAspect
LoginSceneController
LoginSceneController()
signup(ActionEvent) void
checkLogin() boolean
initialize() void
login(ActionEvent) void
App
App()
main(String[]) void
start(Stage) void