Route 66 Trip Planner

Dokumentácia

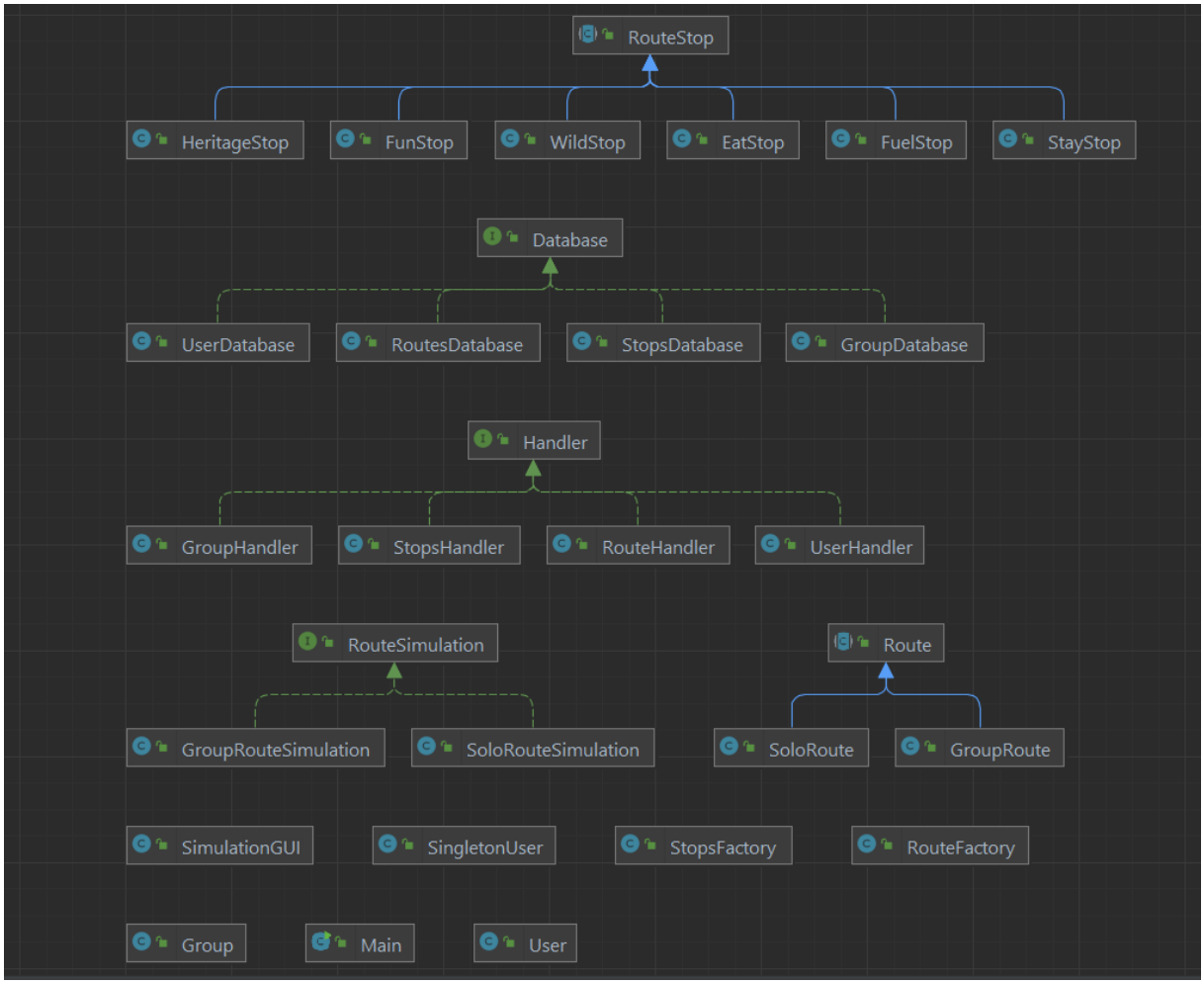
Obsah

Zámer projektu 3
Štruktúra najdôležitejších tried 3
Diagram s popisom vzťahov najdôležitejších tried 4
Splnenie kritérií 4

Zámer projektu

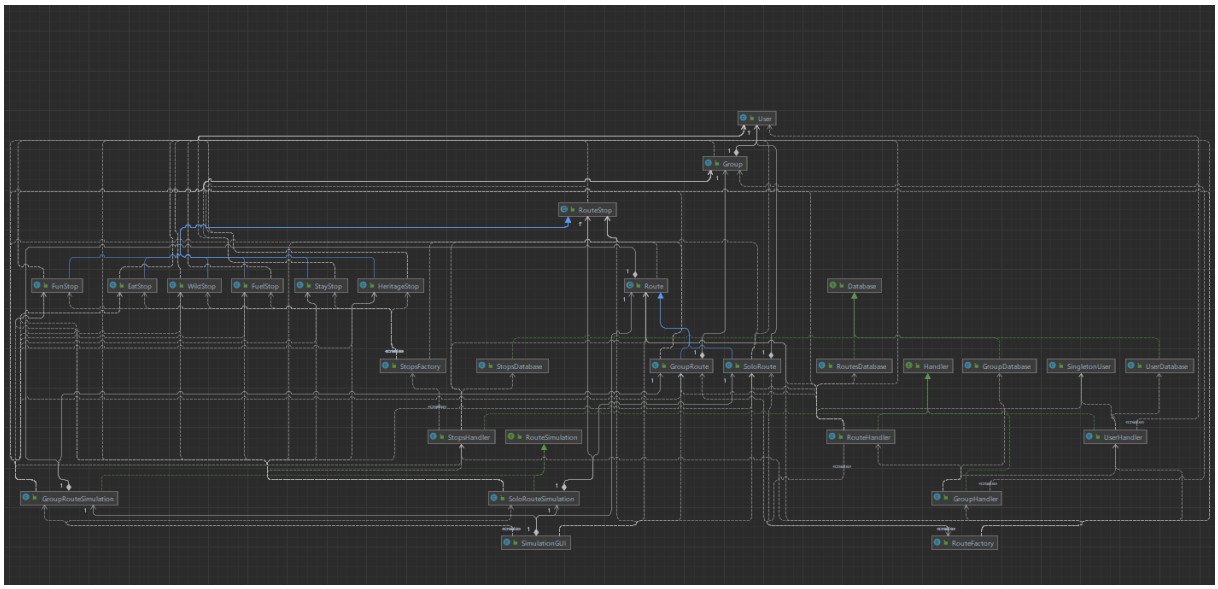
Aplikácia umožňuje jednotlivým používateľom zažiť neopakovateľné dobrodružstvo v rámci legendárnej americkej cesty "Route 66". Po prihlásení si užívateľ môže naplánovať individuálny plán cesty s jednotlivými zastávkami, ale môže sa pridať aj do skupiny iných používateľov a zdieľať svoj cestovateľský zážitok spolu s nimi. Aplikácia umožňuje prihláseným jedincom zvoliť si jednotlivé zastávky, či už sa jedná o občerstvenie, rôzne typy atrakcií alebo nocľah. Po naplánovaní cesty si môže užívateľ vyskúšať jej simuláciu, a oboznámiť sa tak s potenciálnymi prekážkami, ktoré ho môžu počas jeho dobrodružstva sprevádzať. Užívateľ si môže pridať do aplikácie aj parametre svojho auta a na základe toho si vypočítať, koľko ho daná cesta bude stáť, prípadne, kde bude musieť zastaviť, aby doplnil palivo. S touto aplikáciou sa užívateľ na "Route 66" jednoducho nemôže stratiť !

Štruktúra najdôležitejších tried



Toto je štruktúra najdôležitejších tried (bez GUI), ktoré sú stavebnými kameňmi môjho projektu, každá trieda ja podrobne popísaná v JavaDoc.

Diagram reprezentujúci vzťahy najdôležitejších tried



Vzhľadom na to, že diagram v danom rozlíšení nie je možné dobre zobraziť, prikladám ho ako prílohu na môj GitHub repozitár vo forme obrázku.

Splnenie kritérií

Aby program a dokumentácia mohli byť hodnotený nenulovým počtom bodov, musia byť splnené nasledujúce podmienky:

1. Program musí byť funkčný a zodpovedať zadaniu a zámeru projektu, ktoré schválil vyučujúci, a zásadným požiadavkám vyučujúceho, ktoré vznikli počas realizácie projektu.
2. Odovzdaný zdrojový kód musí zahŕňať všetky potrebné súbory a musí sa dať preložiť v prostredí Eclipse inštalovanom v učebni, v ktorej sa realizujú cvičenia.
3. Program musí obsahovať zmysluplné dedenie medzi vlastnými triedami s prekonávaním vlastných metód.
4. V programe musí byť použité zapuzdrenie.
5. Program musí obsahovať dostatok komentáru na pochopenie kódu.
6. Dokumentácia musí zodpovedať programu a musí obsahovať diagram tried.
7. Pri záverečnej prezentácii študent musí vedieť zodpovedať otázky vyučujúceho v súvislosti s projektom.

Splnenie:

1. Na cvičení bola odprezentovaná finálna verzia programu, ku ktorej cvičiaci nemal námietky.
2. Program je možné spustiť a preložiť v Eclipse.
3. Program obsahuje dedenie hneď niekoľko druhov dedenia. Ako príklad uvediem abstraktnú triedu RouteStop, od ktorej dedí rovno až 6 tried – EatStop, FunStop, StayStop, FuelStop, HeritageStop, WildStop. Všetky prekonávajú metódu increase.
4. Zapúzdrenie využíva okrem iného každá z tried, vytvorená podľa Singleton design patternu – GroupDatabase, GroupHandler, UserDatabase, UserHandler....inak povedané, všetky handlery a všetky databázy.

5. Každá trieda, vrátane jej metód a premenných má popísanú svoju funkciu pomocou JavaDoc.
6. Dokumentácia zodpovedá programu, je k nej aj manuál na používanie aplikácie. Diagramy tried som uviedol vyššie, avšak z dôvodu zlého rozlíšenia ich prikladám do repozitáru v podobe obrázkov.
7. Záverečná prezentácia prebehla bez problémov.

Spôsob hodnotenia je nasledujúci:

- splnenie *hlavných kritérií* – max. 15 b
    - program v súlade so zadaním a zámerom projektu, ako aj pokynmi vyučujúceho, s adekvátne použitým dedením a polymorfizmom v aspoň dvoch oddelených hierarchiách dedenia, vrátane použitia rozhraní, a s korektným zapuzdrením a vhodným použitím agregácie; dôsledné oddelenie aplikačnej logiky od používateľského rozhrania; kód vhodne organizovaný do balíkov; prehľadná dokumentácia so všetkými položkami podľa opisu vyššie – 14–15 b
    - program v súlade so zadaním a zámerom projektu, ako aj pokynmi vyučujúceho, s adekvátne použitým dedením v aspoň dvoch oddelených hierarchiách dedenia a polymorfizmom v aspoň jednej z nich, vrátane použitia rozhraní, a s korektným zapuzdrením a vhodným použitím agregácie; principiálne oddelenie aplikačnej logiky od používateľského rozhrania; kód vhodne organizovaný do balíkov; prevažne prehľadná dokumentácia so všetkými položkami podľa opisu vyššie – 11–13 b
    - program v súlade so zadaním a zámerom projektu, ako aj pokynmi vyučujúceho, s menšími akceptovateľnými odchýlkami, s adekvátne použitým dedením a polymorfizmom v aspoň jednej hierarchii dedenia, a s prevažne korektným zapuzdrením a prevažne vhodným použitím agregácie; dokumentácia so všetkými položkami podľa opisu vyššie – 8–10 b
    - program v súlade so zadaním a zámerom projektu, ako aj pokynmi vyučujúceho, s ešte stále akceptovateľnými odchýlkami, s adekvátne použitým dedením a polymorfizmom v aspoň jednej hierarchii dedenia, a s obmedzeným využitím zapuzdrenia a použitím agregácie; ešte stále akceptovateľná dokumentácia – 0–7 b
- adekvátne splnenie *ďalších kritérií* (najmä tých, ktoré sú vymenované vyššie, pričom použitie príslušného mechanizmu nezakladá automaticky nárok na hodnotenie – vhodnosť posúdi vyučujúci) v súlade so zadaním a zámerom projektu, ako aj pokynmi vyučujúceho, s adekvátnym krátkym opisom v dokumentácii: splnenie každého kritéria bude hodnotené v rozpätí 0–4 b – max. 25 b

Hlavné kritériá:

Mám za to, že program spĺňa zadanie projektu a je v súlade so zámerom odovzdaným v prvej fáze. Pri prezentácií nemal cvičiaci žiadne pripomienky, a tak mám za to, že aplikácia je v súlade s jeho pokynmi. Dedenie a polymorfizmus sa vyskytuje vo vyššie spomínanom príklade RouteStop – EatStop, FuelStop, FunStop, WildStop, HeritageStop, StayStop. Uvedených 6 tried nielen že dedí od abstraktnej triedy RouteStop, ale aj prekonáva jej metódu increase, pričom pri každej z tried plní daná metóda inú funkcionalitu, čiže sa jedná o polymorfizmus. Rovnako tak aj pri rozhraní RouteSimulation, ktorú implementujú triedy SoloRouteSimulation a GroupRouteSimulation, pričom obe implementujú funkciu simulate, ktorá vykonáva podobný, avšak rozdielny kód. Príkladov na dedenie a polymorfizmus sa dá v programe nájsť hneď viacero, a tak považujem túto podmienku za splnenú. Ako som už vyššie spomínal, zapúzdrenie je implementované v každej triede podľa návrhového vzoru singleton – takže minimálne vo všetkých handleroch a databázach. Čo sa týka agregácie, tá je použitá napríklad pri

abstraktnej triede Route, kde existuje premenná stops (protected), v ktorej je uložený ArrayList zastávok triedy RouteStop. Čo sa týka oddelenia aplikačnej logiky od používateľského rozhrania, v projekte je jasne oddelené GUI (fxml files) v priečinku FXML, ďalej databázy v priečinku database a štýly v priečinku styles. Ďalej v balíku gui sa nachádzajú všetky kontrolery. Zvyšný kód, predstavujúci backend aplikácie, bohužiaľ, nie je usporiadaný do balíkov. Čo sa týka dokumentácie, budem sa snažiť splniť všetky náležitosti.

Ďalšie kritériá zahŕňajú:

- použitie návrhových vzorov okrem návrhového vzoru Singleton – každý implementovaný návrhový vzor sa počíta ako splnenie jedného ďalšieho kritériá, ale implementácia všetkých návrhových vzorov sa posudzuje maximálne na úrovni splnenia troch ďalších kritérií
- ošetrenie mimoriadnych stavov prostredníctvom vlastných výnimiek – stačí jedna vlastná výnimka, ale musí byť skutočne vyhadzovaná a ošetrovaná
- poskytnutie grafického používateľského rozhrania oddelene od aplikačnej logiky a s aspoň časťou spracovateľov udalostí (handlers) vytvorenou manuálne – počíta sa ako splnenie dvoch ďalších kritérií
- explicitné použitie viacniťovosti (multithreading) – spustenie vlastnej nite priamo alebo prostredníctvom API vyššej úrovne (trieda **Task** a pod.)
- použitie generickosti vo vlastných triedach – implementácia a použitie vlastnej generickej triedy (ako v príklade spájaného zoznamu poskytnutého k prednáške 5)
- explicitné použitie RTTI – napr. na zistenie typu objektu alebo vytvorenie objektu príslušného typu (ako v hre s obrami a rytiermi pri zisťovaní počtu bytostí)
- použitie vhniezdených tried a rozhraní – počíta sa iba použitie v aplikačnej logike, nie v GUI, pričom rozhrania musia byť vlastné (jedna možnosť je v príklade vnútorných tried k prednáške 4)
- použitie lambda výrazov alebo referencií na metódy (method references) – počíta sa iba použitie v aplikačnej logike, nie v GUI (jedna možnosť je v príklade referencií na metódy a lambda výrazov k prednáške 4)
- použitie implicitnej implementácie metód v rozhraniach (default method implementation)
- použitie aspektovo-orientovaného programovania (AspectJ)
- použitie serializácie

Mám za to, že z ďalších kritérií spĺňam nasledovné:

- V projekte sa využívajú dva návrhové vzory – Singleton, Factory (StopsFactory, RouteFactory)
- V prípade, že chcem ukončiť simuláciu, zámerne vyhodím nasledujúcu výnimku.

```
throw new RuntimeException("The user has died.");
```

- Na simuláciu používam handlery v triedach GroupRouteSimulation a SoloRouteSimulation, ktoré komunikujú s triedou SimulationGUI.
- V rámci tried GroupRouteSimulation a SoloRouteSimulation vytváram vždy pri spustení simulácie nový thread.
- Použitie RTII – napríklad v triede RouteHandler v metóde getMine, na zistenie typu cesty.
- Pri prezeraní kódu môjho projektu nie je ťažké natrafiť na lamba výraz. Používa ho takmer každá trieda, ktorá má v názve „Handler“.

- Serializáciu implementovanú v Jave som priamo nepoužil, avšak použil som vlastnú na tom istom princípe. Všetky údaje sú prepísané do jedného reťazca znakov, ktorý je následne uložený do databázy. Príklad je v takmer každej triede, ktorá má v názve „Database“.