**Програмски језици II – Пројектни задатак**

*Никола Марковић (1230/23), 2024/25.*

## - Структура

Апликација се састоји од четири кључне цјелине – стандардних UI елемената, визуелне симулације графа, менаџера стања и алгоритама за руковање графом.

Стандардни UI елементи су имплементирани користећи искључиво Јавин уграђени Swing toolkit, а њихов распоред и презентација су написани у коду.

Визуелна симулација графа је написана користећи библиотеку *GraphStream*. Конкретан граф који се приказује кориснику је објекат типа *MultiGraph* који одговара спецификацији задатка јер омогућава бесконачан број улазних и излазних усмјерених грана у оквиру једног чвора. Сваки град је реализован као један чвор, уз идеју да су жељезничка и аутобуска станица увијек на истом мјесту. При покретању апликације, из већ обрађених улазних података се издвајају имена градова, према којима се онда креирају чворови, чији су идентификатори управо та имена, а сличан процес се одвија при креирању грана, уз обраћање пажње на могуће дупликате. Чворови и гране графа су потом визуелно приказани у произвољно дефинисаном поретку, а кориснику је дозвољено да исте превлачи мишем по потреби за бољу прегледност.

Менаџери стања представљају јавне статичке класе са једном инстанцом (singleton шаблон) које омогућавају несметан рад са свим релевантним подацима и UI елементима из било које тачке програма, што је довољно скалабилно рјешење за задатак овог обима, с обзиром да се не може десити да имамо више инстанци већине UI елемената којима рукују наведене класе.

Алгоритми за руковање графом су самостално написани, упркос чињеници да *GraphStream* већ посједује одређене стандардне алгоритме. Ово је случај јер дати алгоритми не могу да се проширују и самим тим не одговарају спецификацији задатка као такви из простог разлога што сви раде искључиво са тежинама грана, док се у задатку користи више фактора за одређивање оптималне путање у зависности од критеријума. Као рјешење се представио **модификован Јенов алгоритам за К оптималних путања** који користи стандардни Дајкстрин алгоритам као подалгоритам. Правилно имплементиран Јенов алгоритам се лако може модификовати да обраћа пажњу на разлике у временима поласка и вријеме пресједања. Све што одавде остаје јесте да „покупимо“ резултат, у нашем случају за 5 најоптималнијих путања, и даље га истичемо унутар корисничког интерфејса на одговарајућ начин.

## - Логички ток симулације оптималне путање

Сваки пут када промијенимо критеријум оптимизације, тежине грана графа се рекалкулишу у зависности од критеријума. У случају критеријума најниже цијене и најкраћег трајања пута, то су, наравно, цијена и трајање пута. У случају критеријума најмањег броја пресједања, просто свакој грани додијелимо неку константну вриједност, чиме ће сам алгоритам да тежи да у што мањем броју корака дође до циља. Притиском дугмета „Pronadji” се позива алгоритам и најоптималнија путања се постепено визуелно истиче једноставном вишенитном визуелизацијом. Остале функционалности раде као што је наведено.

## - Технички детаљи

- IDE: **IntelliJ IDEA 2024.3.7**
- Java SDK верзија: **23.0.2**
- Build system: **IntelliJ**

Репозиторијум: https://github.com/nikolacoding/JavaTransit
Build локација: \builds\JavaTransit.jar
Javadoc локација: \docs\index.html