Lab 3

[valid 2020-2021]

The Tourist Trip Planning Problem
A tourist is about to visit a city. In this city, there are various locations such as tourism sites, hotels and restaurants, parks, etc. Locations have *names* and may have other common properties (such as a description, an image, coordinates, etc). Depending on its type, each location has various specific properties. However, if the location is a *visitable*, it must have opening hours. If the location is *payable*, it must have the entry fee (the price of the ticket). If the location is *classifiable*, it must have a classification (a rank).
The time (in minutes) required to go from one location to another is known. The tourist has also preferences regarding the order in which he (or she) would like to visit some locations.

Example: locations are: v1 (Hotel) v2 (Museum A) v3 (Museum B) v4 (Church A) v5 (Church B) v6 (Restaurant).

From-To	**Cost**
v1→v2	10
v1→v3	50
v2↔v3	20
v2→v4	20
v2→v5	10
v3→v4	20
v4↔v5	30
v4→v6	10
v5→v6	20

The main specifications of the application are:

Compulsory (1p)

- Create an object-oriented model of the problem. You should have at least the following classes *City, Hotel, Museum, Church, Restaurant*. The natural ordering of their objects is given by their names.
- Create the interfaces *Visitable*, *Payable*, *Classifiable*. The classes above must implement these interfaces accordingly.
- The *City* class will contain a *List* of locations.
- Each location will contain a *Map* representing the times required to go from this location to others.
- Create all the objects given in the example.

Optional (2p)

- In the *City* class, create a method to display the locations that are *visitable* and are not *payable*, sorted by their opening hour.
- Create *default* methods in the interface *Visitable*, with the opening hour *09:30* and the closing hour *20:00*.
- Create a *static* method *getVisitingDuration*,in the interface *Visitable*, that returns a *Duration* object, representing how long a location is opened during a day.
- Create the class *TravelPlan*. An instance of this class will contain a city and the preferences regarding the visiting order.
- Implement an efficient agorithm to determine the shortest path between two given locations, conforming to the preferences.

Bonus (2p)

- Suppose that the tourist has a specific number of days available to visit the city and every day he (or she) has the same number of minutes available for visiting.
- Suppose that there is a special start location (the hotel) - the tourist must start and end a daily *trip* in this location.
- Implement an algorithm that will create a plan (a trip for each day), such that the tourist visits as many locations as possible.
- Test your algorithm using JUnit or other framework.

Resources

- The Java Tutorials: Interfaces
- The Java Tutorials: Collections (Know Thy Complexities!)
- The Java Tutorials: Generics
- The Date-Time package

Objectives

- Create interfaces to describe specifications.
- Create multiple implementations of an interface.
- Understand the differences between abstract classes and interfaces.
- Use packages to organize the classes and intefaces of the application.
- Use collections and generics.
- Use lambda-expressions and streams.
- Understand the space-time tradeoff of various types of collections.