The travel card problem

Based on the Helsinki public transportation system.

Level 1

# Introduction

Inhabitants of crowded capitals require a fluently working, standardized system for using the public transportation. In many cities travel cards are provided and used for handling this problem alongside the regular tickets. This Code Kata aims to solve the problems of a simple travel card system in an imaginary city.

This Kata was inspired by the Helsinki transportation system. See the references section to learn more about the system in use and extend your practices with more complex system.

# Requirements

Create a class what handles the lifecycle of a travel card. Utilize TDD during the implementation. Feel free to use the unit tests from the attached files. This Kata does not require any GUI for testing. It is strongly advised to create a class library with unit tests.

Travel cards are created with initially with a balance of 0.0 €. The balance should not be possible to modify directly. Travel cards can be used to purchase tickets. Tickets can be:

* **Value tickets** after charging the balance of the card
  + Cost of the value ticket is 2 €
  + The validity of the value ticket is one ride
* **Seasonal tickets** on the present day
  + Seasonal tickets can be purchased between intervals with a daily cost of 2 €
  + If the card has valid seasonal ticket, the balance remains unchanged
  + Seasonal tickets have higher priority than value tickets
  + Seasonal tickets can be purchased for the cards based on amount of money or number of days

Travel cards can have discount for all tickets (assuming that the owner is student or retired).

The TravelCard class should implement the following functionality:

* Buying tickets for rides (both seasonal and value – preferring seasonal, if the card has valid period)
* Purchasing seasonal tickets based on number of days and from amount of money
* Retrieving the date when the current period of season is ending.
* Extending the balance on the card (so value tickets can be purchased)
* Retrieving balance on the card
* Setting discount on the card (any, but valid percentage)

# UML

|  |
| --- |
| TravelCard |
|  |
| + buyTicket(): Bool  + buySeasonalTicket(forDays: Int)  + buySeasonalTicketForAmount (withBalance: Double)  + getSeasonalExpiryDate(): Date  + extendBalance(withAmount: double);  + getBalance() -> Double  + setDiscount(discountValue: double) |

# References

1. **Helsingin Seudun Liikenne (HSL)**, Tickets and fares - <https://www.hsl.fi/en/tickets-and-fares>