Airline Information Management

First group project for the Algorithms and Data Structures class

Faculty

- Pedro Ribeiro (lectures)
- Ana Rocha (lectures)
- Rosaldo Rosetti (recitations)

Students (G77)

- Gonçalo Pinto (202004907)
- Guilherme Magalhães (202005285)
- Pedro Cerejeira (202007525)

Problem definition

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Goal Create a management system to support an airline in its daily operations

In order to achieve this, the project should be

- **Simple and intuitive to use** It ought to be possible to interact with the project via console/text file/CSV without too much hassle
- **Readable and well documented** Aesthetics are to be prioritized over efficiency, so in the future other engineers have an easy time deciphering the code
- **Scalable and thoroughly tested** Data abstraction should be fanatically imposed and all classes must be exhaustively tested
- **Runnable in a normal computer** There's no need to melt the airline's CPUs, the code shouldn't waste system resources for no reason

Solution description

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Fun luggage problem

Assigns luggage to carriages in a car, maximizing the capacity usage (luggage weight/carriage capacity) of carriages in front

Search luggage

Luggage is issued an ID so passengers can later find it

Interface

Connects all classes and allows user to interact with them

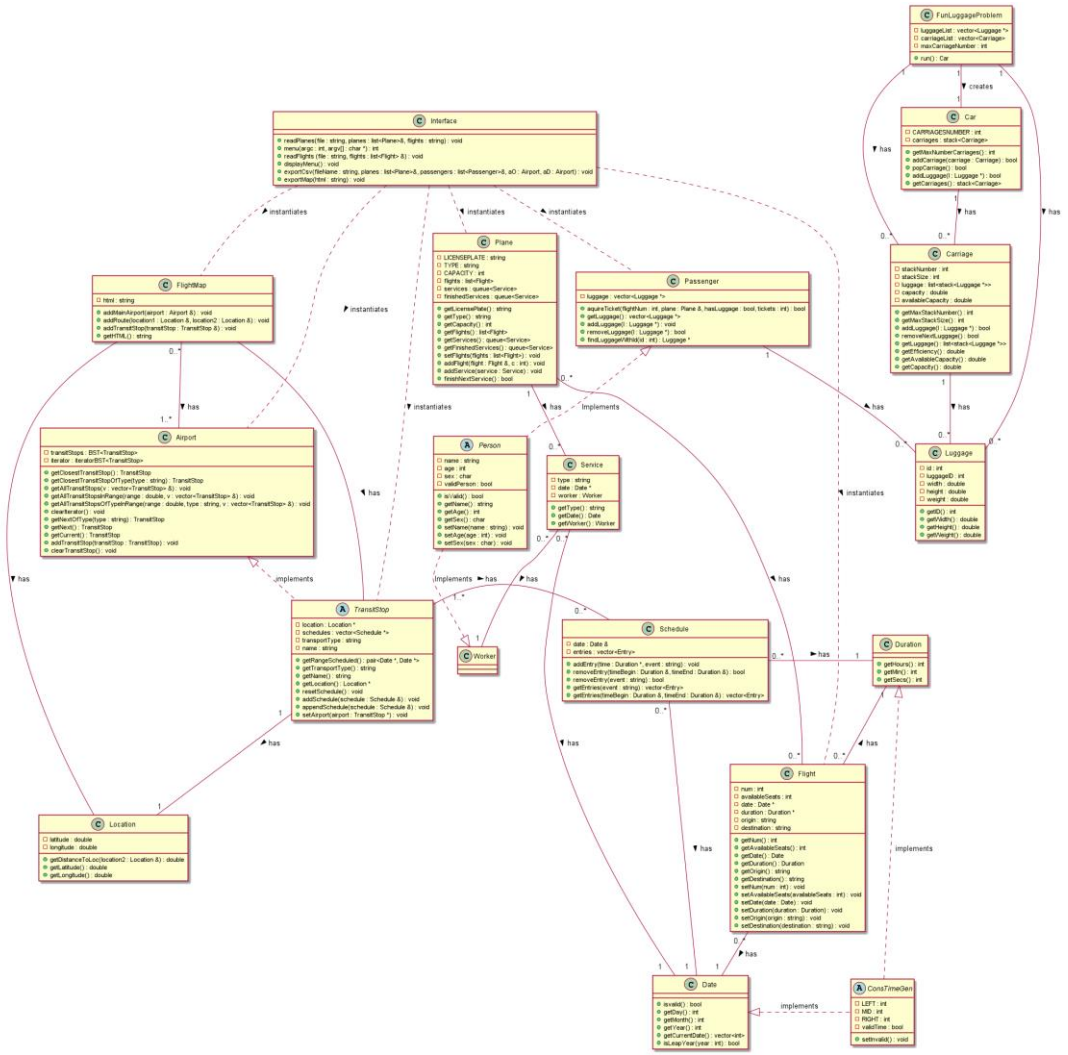
HTML Map Creation

An original html map file (exported from Python) is altered by simple string manipulation. Distances to the airport are calculated from coordinates.

Class Inheritance

Inheritance was widely used to save time, reduce the possibility of errors and, specifically, allow for airports to be included as transit stops seamlessly

Class Diagram



Download the full diagram [here]

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

File Structure

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

We only use .csv files.

Their structure is self explanatory, since columns are named

Implemented Functionalities

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

It is possible to, via the interface:

- simulate flights, from the creation of planes to buying tickets
- posibility of exporting a map with marked fligths and the surroudings of an airport
- run tests implemented for several parts of the program
- import data (in .csv files) of airoports, planes, flights and transit stops
- find a specific passenger's luggage

It's also possible to get into the fun luggage problem´s wild ride (explained next slide).

CRUD was implemented according to what seemed intuitive, for example:

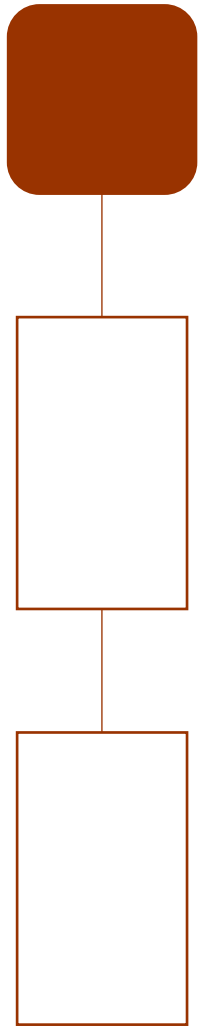
- Destructors weren't implemented, because in the cases it made sense destructors would enter in conflict with pointers
- Time classes have constant attributes, so they can't be updated

Fun Luggage Problem (Feature)

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Context



Luggage has weight, carriages have capacity: so **how do we distribute the weight throw the carriages in an efficient way**?

Used capacity of carriages in the front must be prioritized

therefore

Carriages are added or remove through the back

Solution

Inputs: luggage list, carriage list, max carriage number

Every possible layout of luggage is analyzed iteratively and the one with the most efficient carriage capacity use is chosen and added to the car

Luggage and carriage is removed and iteration is repeated until the car is built

If it is impossible to solve the problem (for example: not enough carriages), an empty car is returned

The algorithm has one drawback: **there is a situation where it can't find a solution**

Main difficulties

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Gonçalo

Difficulties: Adequate container's choice (specially in fun luggage problem) **33%**

Solution: Test different containers for their efficiency

Guilherme

Difficulties: Getting used to and understanding better the use of pointers and references. **33%**

Solution: Experience solving problems that would arise.

Pedro

Difficulties: Exception handling in tests **33%**

Solution: Expected equal of error message