## Bonus exercises:

This document contains a number of simple bonus exercises that can be used to practice. The goal of these exercises is to consider what would be the cleanest, most maintainable code you could write for this. The examples are simplified use-cases of real life problems, and practice will help you understand the complexity of larger systems and build up good habits.

### Airplane reservation system

In this exercise you will create a simple airplane reservation system. The goal of the exercise is to consider how the various functions needed for such a system need to be split across different classes. Seperating the responsibilities properly will result in an easy to maintain and easy to understand codebase. The system will be build in a number of phases, with each phase building upon the last. If done correctly it will be easy to add the new functionality to the existing code-base.

You do not need to build a fancy GUI. It is sufficient if users can operate the system via a CMD. However, adding a GUI is definitely a good exercise.

#### Phase 1)

A customer should be able to reserve a specific seat on an airplane. The system should indicate which seats are still available and give an error message if an occupied seat is selected. It should be clear what seat is a window seat and what is an aisle seat. After the initial selection the system should ask for a confirmation.

#### Phase 2)

Add functionality to store and load planes.

#### Phase 3)

An employee should be able to create “trips”, with a origin and destination, and assign a specific plane to this trip. A customer should be able to then choose between trips, and select a seat on that trip. A customer should not be able to create trips, an employee should not be able to select seats, keep their roles separate.

#### Phase 4)

The company has 4 types of airplanes. Airplane A has 4 seats in a row (2,2), and 15 rows. Airplane B has 6 seats in a row with 2 aisles (2, 2, 2) and 20 rows. Airplane C has 6 seats with 1 aisle (3,3) and 15 rows. Airplane D has 8 seats with 2 aisles (2, 4, 2) and 20 rows. Implement the different airplanes and allow an employee to select the type of plane for a given trip.

#### Phase 5)

The company has 3 type A planes, 5 type B planes, 4 type C planes, and 3 type D planes. Ensure that planes cannot be double booked to go onto two trips at once. Employees should be able to see what trips a plane is taking on a given date and check its occupancy.

#### Phase 6)

The company is buying and selling planes. Implement functionality to update the number of planes of each type.

#### Phase 7)

A new plane is released, it has a first class section with 4 seats in a row (2,2) and 5 rows. It has a second class section with 6 seats in a row (2,2,2) and 10 rows, and a third class section with 8 seats in a row (2,4,2) and 15 rows. The company buys three such planes. Add these new planes to the system.