Exercise 2

Briefly describe in your own words;

- What the *event sourcing* pattern is, and when **not** to use it.
- The difference between the *adapter* and the *façade* coding patterns.
- The main idea behind the *actor model* pattern.

Consider the following requirements:

A customer asks for a system to calculate salaries.

The customer's employees get salaries based on **their position** and **the number of years they have been employed**, with an **added bonus based on a satisfaction score** decided on by their manager.

Currently, they only have *sales representatives* and *technicians*. But they might add more positions later on.

Every year, each manager is responsible for deciding on the satisfaction score for each employee they manage, for the previous year. The satisfaction score ranges from 0 (very unsatisfied) and 5 (very satisfied), and the salary for that employee is then adjusted for the following year.

The bonus is calculated based on the average satisfaction score for the employee over the past 3 years, with double weight put on the last year's performance, rounded upwards.

bonus = RoundUp(Average(satisfaction score[last], Average(satisfaction score[last-1], satisfaction score[last-2])))

The salary is then computed as follows:

monthly salary = base salary[position, years employed] + bonus

The customer has provided the following lists printed out on paper:

Position	**Years**	**Base salary**
Technician	0-2	€ 1 000
Technician	3-5	€ 1 200
Technician	6-8	€ 1 520
Sales	0-2	€ 1 100
Sales	3-6	€ 1 400
Sales	7-9	€ 1 650

Minimum wage = € 800 / month.

Every few years, the base salaries are reviewed and increased by a few per cents.

Satisfaction score vs. bonus

0	0 %
1	0 %
2	2%
3	7%
4	15%
5	20%

And gives the following example, handwritten on a piece of napkin:

> Joe is a technician and has worked for 4 years in the company.
> The last three satisfaction scores given were:
> 2017: 5
> 2016: 3
> 2015: 1
>
> The current salary for that employee is then:
> Technician, 4 years = € 1 200 / month
> Averaged satisfaction score with double weight on 2017 (last score made) = 3.5 (rounded up to 4) = 15%
> Final salary for year 2018 is then € 1 200 + 15% = € 1 380 / month, in addition to the minimum wage which is € 800 = € 2 180 / month.

Write and unit test code to calculate salaries for employees given the above rules.

Model *the bare minimum* database required to store the necessary information to calculate salaries, including employee list, who is a manager and who manages what employees, positions, base salaries and the last three satisfaction scores. Do include a name field. Do not include fields such as phone numbers, addresses or anything beyond what is absolutely needed. This should not be more than 3-5 tables. This can be a SQL Server project within the solution, going against (localdb)\MSSqlLocalDB for easy deployment and validation of your solution.

Write a data access layer using any technology of your choice to retrieve and update the information in the database. Use a *seed* mechanism to populate the data with some test data.

Create a very simple user interface in ASP.NET Core 2 with Angular 6, with just one single view, showing the list of employees and their current salary. Allow the user to select one of them, and click a button to show a pop-up dialog where you can enter a new satisfaction score to update the selected employee's salary. No authentication/authorization, login or permission handling.