Introduction in .Net Core -> Lab 2 (Florin Olariu && Dan Nastasa)

Prerequisites:

a) Create a Blank solution
b) Add a Class Library (.NET Core)
c) Add a xUnit Test Project (.NET Core)
d) Add dependency between Test Project and Class Library
e) Discussion about domain models: rich vs anemic domain models(basic DDD functionality – ubiquitous language, aggregate root, value object)

Note: The exercise is meant to learn how to build domain models and it we help us to:

Learn how to design classes
How to apply aggregation
How to use/apply encapsulation
How to use/apply inheritance
How to identify and write meaningful unit tests for a domain model

Exercise:

Build a rich domain model that allows us to:

- Manage table reservations at a restaurant
- Throw business exceptions when needed (create a BusinessException class derived from Exception)

Sample:

```
using System;

public class BusinessException: Exception
{
    public BusinessException()
    {
    }

    public BusinessException(string message)
        : base(message)
    {
    }

    public BusinessException(string message, Exception inner)
        : base(message, inner)
    {
    }
}
```

The domain has the following constraints:

- A table can be reserved between 10:00 and 22:00
- You cannot double-book a table in a specific period
- You should be able to add a new reservation to the table
- The staff should be able to view all the reservations for a table
- The staff should be able to see the reservations in a specific period for a table
- You cannot book more people than the table capacity
- Create unit tests to cover the entire functionality described by the previous constraints.

Observations:

1. Use TimeSpan to work with time periods
2. Use FluentAssertions when unit testing (NuGet package)
3. Throw exceptions with meaningful messages

Note:

1. All exercises are mandatory.
2. You will receive your points at the end of the lab.