Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Exercises about Error handling

- Solve them in Visual Studio.

Exercise 19.01: A Signed Contract of Behaviour

Study first the examples that were given during the course.

All the items needed to solve this exercise are already implemented. You just need to copy paste and do some minor changes.

- Create a class contract.
  - A contract has a start date.
  - A contract has a length in months.
- These 2 variables / properties are also given in the constructor.
- Create another class with custom-made exceptions.
  - Contracts with a bigger length than 24 months are not allowed and should be thrown (hint) by the program.
  - Contracts are minimum 1 month long.
  - When a wrong length in months is given, it because default 1 month.
- This error should also be logged in a logfile.
- The logfile is a text file that is placed in the same directory where the application is running.
- Create also a ToString() for the contract that shows the start date and the end date of the contract.
  - Pay attention.
    - Start date: 31/01/2023.
    - Length in months: 1.
    - Should result in an end date of: 28/02/2023.
- Create a test program that test all the functionality.

Exercise 19.02: Leave the Door open

It is possible that you have already created this exercise, but the focus here is on the error throwing and catching.

If possible do the implementation with interfaces, so you can make later subclasses of Doors. Create first an interface, and than a class on that interface.

- Create an interface that will be used to create a door.
- Create a class that simulates a door.
- There are 2 properties.
  - Locked (or not)
  - Closed (or not)
- There are 4 methods.
  - Open a door.
  - Close a door.
  - Lock a door.
  - Unlock a door.

- There are a lot of possibilities.
- Cover all actions, and the wrong ones with exceptions.
  - You can close a door that is open.
  - You can't close a door that is closed.
  - You can't open a door when it is locked.
  - You can't lock a door when it is open.
  - And so on …

I will test all possibilities.

Make sure you are covering them all.

When all succeed. Create now a new class CastlePort with the same functionality.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Exercise 19.03: We almost lift everything

We try to simulate an elevator.

An elevator is stationed at a certain level. For example floor 4.

At that location, the elevator has always open doors. At the other levels, the elevator is closed.

Use error handling to show wrong actions. You can also use events to create this functionality.

The title of this exercise was a slogan of a commercial of TVH. Look it up.

- You have a building with a certain number of floors (levels).
- The elevator can move from one floor to another.
  - This is done when you hit a button inside the elevator.
- Below you can find some of the possible situations.
  - The elevator is at floor 0. Here the door is open.
  - You are at floor 4. Here the door is closed.
  - You hit the button to call the elevator.
  - The elevator closes the doors at floor 0.
  - Is going up to floor 4.
  - There the doors are opened.
  - Calling the elevator again (at floor 4 with open doors) should be handled by error handling.
- Inside the elevator you can go to a certain floor.
  - Hit the button to go to a floor.
  - The doors closed and you are moving inside the elevator towards a certain floor.
  - The doors are opened and you can leave the elevator.
- You can from here invent a lot of situations
  - Is there a button to call the elevator or are it buttons that ask to go up or to go down?

Notes

COPY PASTE)

Variant 1

- Can you make this work when 2 persons hit the buttons?
  - Elevator is at floor 0
  - Person 1 at floor 2 (wants to go to floor 6)
  - Person 2 at floor 4 (wants to go to floor 1).

Variant 2

- Can you make this work when there are 2 elevators, next to each other?

This is a big exercise and a never-ending story.

Make clear in your comments what kind of behaviour you have implemented.

Use the agile way of working.

Only this will be tested.

Variant 3

- Can you make a visual representation of this?
  - Windows Forms or Windows Presentation Foundation.

Start small.

Let your application grow with working functionality.

Notes