KATA: Refactor legacy code

This is a bit of a different kata, it is designed for us to practice some SOLID principles. You will be given a scenario and some code to work from.

What is expected is that you look through the given classes and identify what SOLID principles they are breaking. The line and the principle. You are then to fix the classes to be more SOLID. There will be one or two violations – so not major refactors just a line or and whatever additions you need to make (other classes, interfaces, …).

You do not have to demonstrate their use – nothing needs to be written in main(). You are just changing how these few classes interact to be more SOLID (loose coupling, high cohesion).

Scenario: Tire pressure monitoring system

This is a simple system where there is an Alarm class designed to monitor the tire pressure and set an alarm if the pressure falls outside of the expected range. The sensor class provided simulates the behavior of a real tire sensor, providing random but realistic values.

The code for the scenario can be found here:

.\ Exercises\csharp-kata06-refactorlegacycode-starting-point

Note: There are two folders:

The first is “Starting”, where you will find the classes. You are to comment where you think a SOLID principle is broken and explain why and what the principle was.

The second is where your solution will go, in “Refactored”. The classes will be copy pasted there with your changes and additions included. There should be comments showing what you changed where possible. NB: You are allowed to add new classes if you want, but the functionality must still be the same, i.e. CheckPressure () is called to find out the pressure by a client (program.cs in this case)