**Lab: Unit Testing**

Problems for exercises and homework for the ["C# OOP" course @ SoftUni"](https://softuni.bg/trainings/3008/csharp-oop-october-2020).

* **Part I: Unit Testing Basics**

**Test Axe**

Load provided solution in Visual Studio. Add new project **Tests**

Create a class **AxeTests**

Create the following tests:

* Test if weapon loses durability after each attack
* Test attacking with a broken weapon

**Solution**



* **Test Dummy**

Create a class **DummyTests**

Create the following tests:

* Dummy loses health if attacked
* Dead Dummy throws exception if attacked
* Dead Dummy can give XP
* Alive Dummy can't give XP

**Hints**

Follow the logic of the previous problem

* **Refactor Tests**

Refactor the tests for **Axe** and **Dummy** classes

Make sure that:

* **Names** of test methods are **descriptive**
* You use **appropriate** **assertions** (assert equals vs assert true)
* You use **assertion** **messages**
* There are **no magic numbers**
* There is **no code duplication** (Don’t Repeat Yourself)

**Hints**

Extract constants and private fields for **Axe** class

Create a method that executes **before each test**

Make use of constants and private fields, as well as add assertion messages

Follow the same logic for other test methods and **TestDummy** class

* **Car Manager**

You are provided with a simple project **containing only one class** - "**Car**". The provided class is simple - its **main point is to represent some of the functionality of a car**. **Each car contains information** about its **Make**, **Model**, **Fuel Consumption**, **Fuel Amount** and **Fuel Capacity**. Also **each car can add some fuel to its tank by refueling** and **can travel distance by driving**. **In order to be driven**, **our car needs to have enough fuel**. **Everything in the provided skeleton is working perfectly fine** and **you mustn't change it**.

In the skeleton you are provided **Test Project** named "**CarManager.Tests**". There you **should place all the unit tests** you write. The **Test Project** have only **one class** inside:

* "**CarTests**" - here you should place **all code** testing the "**Car**" and **it's functionality**.

Your job now is to **write unit tests on the provided project** and **it's functionality**. You should test exactly **every part** of code inside the "**Car**" class:

* You should test **all the constructors**.
* You should test **all properties** (**getters** and **setters**).
* You should test **all the methods** and **validations inside the class**.

**Before you submit** your solution to Judge, you should **remove all the references and namespaces referencing the other project**. You should **upload only** the "**CarManager.Tests**" project **holding the class with your tests**. **Remove** the "**bin**" and "**obj**" folders **before** submission.

**Constraints**

* **Everything in the provided skeleton is working perfectly fine**.
* **You mustn't change anything in the project structure**.
* **You can test both constructors together**.
* **You shouldn't test the auto properties**.
* **Any part of validation should be tested**.
* **There is no limit on the tests you will write but keep your attention on the main functionality**.