Exercises: Unit Testing

Problems for exercises and homework for the "C# OOP" course @ SoftUni".

Problem 1. Database

You are provided with a simple class - **Database**. It should **store integers**. **The initial integers should be set by constructor**. They are stored **in array**. **Database** have a functionality to **add**, **remove** and **fetch all stored items**. Your task is to **test the class**. In other words **write tests**, so you are sure its methods are working as intended.

Constraints

- Storing array's **capacity** must be **exactly 16 integers**
  - If the size of the array is not 16 integers long, **InvalidOperationException** is thrown.
- **Add** operation, should **add an element at the next free cell** (just like a stack)
  - If there are 16 elements in the Database and try to add 17th, **InvalidOperationException** is thrown.
- **Remove** operation, should support only removing an element **at the last index** (just like a stack)
  - If you try to remove element from empty Database, **InvalidOperationException** is thrown.
- **Constructors** should take integers only, and store them in **array**.
- **Fetch method** should return the elements as **array**.

Hint

Do not forget to **test the constructor(s)**. They are methods too!

Problem 2. Fighting Arena

You are provided with a project named "**FightingArena**" containing **two classes** - "**Warrior**" and "**Arena**". **Your task** here is simple - **you need to write tests** on the project **covering the whole functionality**. But before start writing tests, you need to **get know** with the project's **structure** and **bussiness logic**. Each **Arena** has a **collection of Warriors** enrolled for the fights. On the **Arena**, **Warriors** should be able to **Enroll for the fights** and **fight each other**. Each Warrior has **unique name**, **damage** and **HP**. **Warriors** can **attack** other **Warriors**. Of course there is some kind of validations:

- **Name** cannot be **null**, **empty** or **whitespace**.
- **Damage** cannot be **zero or negative**.
- **HP** cannot be **negative**.
- **Warrior** cannot **attack** if his **HP** are **below 30**.
- **Warrior** cannot **attack Warriors** which **HP** are **below 30**.
- **Warrior** cannot **attack stronger enemies**.

On the **Arena** there should be performed **some validations** too:

- **Already enrolled Warriors** should not be able to **enroll again**.
- **There cannot be fight** if **one of the Warriors** is not **enrolled** for the fights.

In the skeleton you are provided **Test Project** named "**FightingArena.Tests**". There you **should place all the unit tests** you write. The **Test Project** have **two classes** inside:

- "**WarriorTests**" - here you should place **all code** testing the "**Warrior**" and **it's functionality**.
- "**ArenaTests**" - here you should place **all code** testing the "**Arena**" 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 "**Warrior**" and "**Arena**" classes:

- 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 "**FightingArena.Tests**" project **holding the two classes 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 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**.