**Exercise: Encapsulation**

Problems for exercise and homework for the ["C# OOP" course @ SoftUni"](https://softuni.bg/trainings/3585/csharp-oop-february-2022).

You can check your solutions here: <https://judge.softuni.org/Contests/1498/Encapsulation-Exercise>

* **Class Box Data**

Create a class **Box**, which has the following properties:

* **Length -** double, should not be zero or negative number
* **Width -** double, should not be zero or negative number
* **Height -** double, should not be zero or negative number

If one of the properties is a zero or negative number throw **ArgumentException** with the message: "{propertyName} cannot be zero or negative." Use try-catch block to process the error. All properties are set by the constructor and when set, they cannot be modified.

**Behavior**

**double SurfaceArea()**

Calculate and return the **surface area** of the **Box**.

**double LateralSurfaceArea()**

Calculate and return the **lateral surface area** of the **Box**.

**double Volume()**

Calculate and return the **volume** of the **Box**.

**NOTE: You can find all formulas** [here](http://www.mathwords.com/r/rectangular_parallelepiped.htm).

**Input**

* On the **first three lines,** you will get the **length**, **width,** and **height**.

**Output**

* On the **next three lines** print the **surface area**, **lateral surface area,** and the **volume** of the box:

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  3  4 | Surface Area - 52.00  Lateral Surface Area - 40.00  Volume - 24.00 |
| 1.3  1  6 | Surface Area - 30.20  Lateral Surface Area - 27.60  Volume - 7.80 |
| 2  -3  4 | Width cannot be zero or negative. |

* **Animal Farm**

For this problem, you have to **download** the provided **skeleton**.

You should be familiar with **encapsulation** already. For this problem, you’ll be working with the **AnimalFarm project**. It contains a class **Chicken**. **Chicken** contains several **fields**, a **constructor**, several **properties**, and **methods**. Your task is to **encapsulate**or **hide** anything **unintended for viewing** or **modification** from **outside** the class.

**Step 1. Encapsulate Fields**

**Fields** should be **private**. Leaving fields open for modification from outside the class is potentially **dangerous**. Make **all fields** in the **Chicken** class **private**. In case the value inside the field is needed elsewhere, use **getters** to reveal it.

**Step 2. Ensure Classes Have a Correct State**

Having **getters and setters is useless if you don’t use them. The Chicken constructor modifies the fields directly, which is wrong when there are suitable setters available. Modify the constructor to fix this issue.**

**Step 3. Validate Data Properly**

Validate the chicken’s **name** (it cannot be **null**, **empty**, or **whitespace**). In case of an **invalid name**, print the Exception message: "Name cannot be empty.".

Validate the **age** properly, **minimum** and **maximum** **age** are provided, make use of them. In case of an **invalid age**, print Exception message: "Age should be between 0 and 15.". Don’t forget to **handle properly** the possibly **thrown Exceptions**.

**Step 4. Hide Internal Logic**

If a **method** is intended to be used only by **descendant** classes or **internally** to perform some action, there is no point in keeping them **public**. The **CalculateProductPerDay()** method is used by the **ProductPerDay** public getter. This means the method can safely be **hidden** inside the **Chicken** class by declaring it **private**.

**Step 5. Submit Code to Judge**

Submit your code as a **zip file** in Judge. Zip everything **except** the **bin** and **obj** **folders** within the project and submit the **single zip file** in judge.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Molly  10 | Chicken Molly (age 10) can produce 1 eggs per day. |
| Molly  17 | Age should be between 0 and 15. |