Exercise: Encapsulation and Polymorphism

This document defines an in-class exercise from the "OOP" Course @ Software University.

Problem 1. Encapsulation

You should be familiar with encapsulation already. For this problem, you'll be working with the **Encapsulation.sln** solution. It contains an abstract class Animal and a concrete class Chicken. Animal contains several fields, a constructor, several properties and several methods. Your task is to encapsulate or hide anything that is not intended to be viewed or modified 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 Animal class private.

In case the value inside a field is needed elsewhere, use **properties** to reveal it.

Step 2. Ensure Classes Have a Correct State

Having **properties** is useless if you don't actually use them. The Animal constructor modifies the fields directly which is wrong when there are suitable properties available. Modify the constructor to fix this issue.

Step 3. Validate Data Properly

Assume an Animal is immutable (cannot be changed after being created). This means all setters need to be kept **private**. Any other access modifier leaves room for another class to change an animal, which shouldn't happen.

Validate the animal's **name** (it cannot be null, empty or whitespace). Validate the **age** properly, minimum and maximum age are provided, make use of them.

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 Animal constructor should only be accessed by child classes, so the appropriate modifier is **protected**. The **CalculateProductPerDay()** method is used by the **ProductPerDay** public property. This means the method can safely be hidden inside the Animal class by declaring it **private**.