**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>

* **Shopping Spree**

Create two classes:

* **Person**
* **Product**

Each **person** should have a **name**, **money,** and a **bag** **of products**. Each **product** should have a **name** and a **cost**. The name cannot be an **empty string**. Money cannot be a **negative number**.

Create a program where **each command** corresponds to a **person buying a product**. If the person can **afford** a product, **add** it to his bag. If a person **doesn’t have enough** money, print an **appropriate** **message** ("**{personName} can't afford {productName}**").

On the **first two lines,** you are given **all people** and **all products**. After all, purchases print **every person** in the order of **appearance** and **all products** that he has **bought** also in order of **appearance**. If **nothing was bought**, print the name of the person followed by "**Nothing bought**".

In case of **invalid input** (negative money Exception message: "**Money cannot be negative**") or an empty name (empty name Exception message: "**Name cannot be empty**") **break** the program with an appropriate message. See the examples below:

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Peter=11;George=4  Bread=10;Milk=2;  Peter Bread  George Milk  George Milk  Peter Milk  END | Peter bought Bread  George bought Milk  George bought Milk  Peter can't afford Milk  Peter - Bread  George - Milk, Milk |
| Maria=0  Coffee=2  Maria Coffee  END | Maria can't afford Coffee  Maria - Nothing bought |
| John=-3  Peppers=1;Tomatoes=2;Cheese=3  John Peppers  John Tomatoes  John Cheese  END | Money cannot be negative |