Homework assignment 2.
VM management system

1 Introduction

The goal of this assignment is to practice skills related to the following topics:

- OOP
- Abstract programming
- LINQ
- Databases

2 Application idea and requirements

This assignment is the second part of the vending project. You will need to implement a simple information system for a company operating multiple vending machines (also called vending terminals, VMs, VTs). The following prerequisites are given to you:

- Vending machines are installed at different locations.
- Selection of products may vary across different VMs (one may sell drinks, the other one - snacks), the information system should store information about the whole range of products.
- The price of a given product is the same for all VMs that currently sell it

Your application needs to store the following information (ultimately in a database, but you can first start with a simpler storage).

- List of products that the company offers through its vending terminals, for each product - internal 4-letter code (can be used as primary key in the DB), name, selling price and purchase price (must be less than the selling price).
- List of vending terminals, for each terminal - id (integer value, generated automatically), location (address), selection of products and current state (quantities of products and accumulated cash - both the banknotes inserted by the user and coins for returning the change)

- Daily selling statistics for each terminal (date, number of sold products of each type).

The following CRUD operations need to be implemented in logic and the UI.

- Adding, editing, deleting products. A product can only be deleted if it has not been linked to any of the vending terminals. The product code cannot be edited after the product has been inserted into the DB.
- Adding, editing vending terminals. When adding or editing a vending terminal you only need to provide UI controls to change its location. Identifiers need to be assigned automatically. Product selection, quantities, amount of banknotes and coins will be updated automatically by the VT itself. In this assignment this data can be filled using the DB seed approach.
- Viewing report results (see description of tasks)

Inside the database the VT state / selling statistics can be stored in different ways, the two main possibilities being:

1. Classic relational model: separate column for each attribute, intermediate tables for many-to-many relationships
2. Json-encoded string in one of the columns (below is just one example of how the resulting json may look like):

```
{"products": {"COLA": 10, "MARS": 4, "SNCK": 0, ...},
    "cash": {"k500": 20,"k100":103,..."m10":350,"m5"
   :98,...}}
```

The choice usually depends on the queries you make to the DB. If you need to filter data by a certain attribute of your entity, it should be present as a separate column. If certain attributes can be logically grouped together and are always processed as a group, if the set changes dynamically (one record has 8 attributes, the other - 10 attributes), then the second approach may be taken.

3 Description of tasks

1. Create a new solution consisting of two projects - a class library that will store all components connected with logic and data storage and a WPF (or any other kind of GUI) application.
2. (2 points) Taking into account the requirements stated above, design data schema for the given subject area, add entity classes and connections between them. Add storage (repository) classes. Your initial set of data can be hardcoded or read from a file.

3. (1.5 points) Design the application UI. Split functionality evenly between multiple windows / pages - don't make one window / page responsible for all of the operations.

4. (2 points) Using LINQ, implement the following reports, show their results in the UI.

   (a) List of terminals that have at least one product missing (quantity = 0), sorted in increasing order of the total available products

   (b) List of terminals sorted in descending order of the total profit for a given month and year (for a single item its profit is the difference between the selling price and the purchase price multiplied by the quantity sold, for simplicity taxes won't be considered).

   (c) 5 least sold products for a given month and year

5. (2.5 points) Using Entity Framework, ADO .NET or any other library, implement database storage for all of the application data.

6. (2 points) Add capability to change and store all changes of product prices (both purchase and selling prices). Provide a datepicker control in the UI to choose the date from which the new prices are activated (must be larger than the current date). You can assume that prices are syncronized by all vending terminals at midnight. Adjust the profit report so that it takes into account changes of prices, i.e, if the selling price changes on the 10th day of a given month, then when calculating total monthly profit, the old price needs to be taken for the first 9 days and the new price for the remaining days. The number of changes is only limited by the number of days in a month (a price cannot change twice per day).

4 Submission

Submit your solution following these steps:

1. Delete three temporary subfolders - "bin", "obj" from the project folder and "packages" (**not the packages.config file**) from the solution folder.

2. Add the whole solution folder to a ZIP **(not RAR or 7Z)** archive.

3. Upload the archive to the Canvas LMS in the corresponding section

5 Grading policy

The final grade for the assignment is determined by the number and quality of completed tasks. An instructor has the option to ask one or two additional questions in case of an unclear grade.

The grade is lowered in the following cases:

- Inefficient implementation of algorithms (-1 point)
- Program crash (-0.5 points each)
- Strong coupling from logic to the UI (-1 point)
- Poor programming style (-1 point)