Laborator04

Laborator04
- Utilizarea librăriilor externe în aplicațiile Visual Studio
- JSon (Javascript Object Notation)
  - DataManagement
  - Basket
  - Product
  - Order
  - Program
  - GetAllProducts
- Temă

Utilizarea librăriilor externe în aplicațiile Visual Studio

Exemplu:

- serializarea / deserializarea obiectelor C# în format `Json` și salvarea acestora pe disk / recuperarea acestora de pe disk

JSon (Javascript Object Notation)

Format de tip text ce permite reprezentarea obiectelor ca `string`, în structuri ierarhice.

Aplicație: Să se realizeze o clasă `Product`, o clasă `Order` și o clasă `Basket` (coș de cumpărături) ce poate să conțină mai multe comenzi (obiecte `Order`).

Să se implementeze posibilitatea salvării / recuperării pe / de pe disk în forma `JSon`, utilizând o librărie externă (de exemplu `Newtonsoft`).

DataManagement

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace BasketModels {
  // Salveaza / Citeste obiecte de diverse tipuri (Product, Order, Basket) de pe
disk
  public class DataManagement {

    public static void Save(Product product) {
      if (product == null || string.IsNullOrEmpty(product.BarCode))
        return;
```

```
      // Salvez in format JSon fisierul pe disk, cu numele identic cu
product.BarCode serializez ca json obiectul si salvez pe disk
      var jsonString = JsonConvert.SerializeObject(product);
      File.WriteAllText($ "Product_{product.BarCode}.json", jsonString);
    }

    public static void Save(Order order) {
      if (order == null || order.Id == 0)
        return;
      // Salvez in format JSon fisierul pe disk, cu numele identic cu order.Id
serializez ca json obiectul si salvez pe disk
      var jsonString = JsonConvert.SerializeObject(order);
      File.WriteAllText($ "Order_{order.Id}.json", jsonString);
    }

    public static void Save(Basket basket) {
      if (basket == null)
        return;
      // Salvez in format JSon fisierul pe disk, serializez ca json obiectul si
salvez pe disk
      var jsonString = JsonConvert.SerializeObject(basket);
      File.AppendAllText($ "Basket.json", jsonString);
    }

    public static Product GetProduct(string barCode) {
      if (string.IsNullOrEmpty(barCode))
        return null;
      // Verificam daca exista pe disk fisierul cu acest barcode!
      var fileName = $ "Product_{barCode}.json";
      if (!new FileInfo(fileName).Exists)
        return null;
      var jsonValue = File.ReadAllText(fileName);
      return JsonConvert.DeserializeObject < Product > (jsonValue);
    }

    public static Order GetOrder(int orderId) {
      // Verificam daca exista pe disk fisierul cu acest orderId!
      var fileName = $ "Order_{orderId}.json";
      if (!new FileInfo(fileName).Exists)
        return null;
      var jsonValue = File.ReadAllText(fileName);
      return JsonConvert.DeserializeObject < Order > (jsonValue);
    }

    public static Basket GetBasket() {
      // Verificam daca exista pe disk fisierul cu acest orderId!
      var fileName = $ "Basket.json";
      if (!new FileInfo(fileName).Exists)
        return null;
      var jsonValue = File.ReadAllText(fileName);
      return JsonConvert.DeserializeObject < Basket > (jsonValue);
    }
  }
}
```

Basket

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BasketModels {
  public class Basket {
    public List < Order > Orders { get; private set; }
    public Basket() {
      Orders = new List < Order > ();
    }
    public void AddOrder(Order order) {
      Orders.Add(order);
    }
    public void Remove(Order order) {
      Orders.Remove(order);
    }
    public double GetTotal() {
      double result = 0;
      foreach(Order order in Orders) {
        var product = DataManagement.GetProduct(order.ProductBarcode);
        if (product != null)
          result += product.Price * order.Count;
      }
      return result;
    }
  }
}
```

Product

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BasketModels {
  public class Product {
    public string BarCode { get; set; }
    public double Price { get; set; }
    public string Name { get; set; }
    public string Category { get; set; }
    public DateTime CreateDate { get; set; }
    public DateTime ExpirationDate { get; set; }
  }
}
```

Order

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BasketModels {
  public class Order {
    public int Id { get; set; }
    public DateTime Date { get; set; }
    public string ProductBarcode { get; set; }
    public int Count { get; set; }
    public override bool Equals(object obj) {
      if (obj == null)
        return false;
      if (! (obj is Order))
        return false;
      return Id == (obj as Order).Id;
    }
  }
}
```

Program

```
using BasketModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BasketApplication {
  class Program {
    static void Main(string[] args) {
      Basket basket = new Basket();
      // Product p1
      Product p1 = new Product() {
        BarCode = "1001a",
        CreateDate = DateTime.Now,
        ExpirationDate = DateTime.Now.AddYears(3),
        Category = "electronics",
        Name = "iphone",
        Price = 2000
      };
      // Product p2
      Product p2 = new Product() {
        BarCode = "1002b",
        CreateDate = DateTime.Now,
        ExpirationDate = DateTime.Now.AddYears(10),
        Category = "electronics",
        Name = "tv",
        Price = 2500
      };
```

```
      DataManagement.Save(p1);
      DataManagement.Save(p2);

      // Order o1
      Order o1 = new Order() {
        Id = 1,
        ProductBarcode = "1001a",
        Date = DateTime.Now,
        Count = 3
      };
      // Order o2
      Order o2 = new Order() {
        Id = 2,
        ProductBarcode = "1002b",
        Date = DateTime.Now,
        Count = 5
      };

      DataManagement.Save(o1);
      DataManagement.Save(o2);
      basket.AddOrder(o1);
      basket.AddOrder(o2);

      Console.WriteLine("Totals: {0}", basket.GetTotal());
      DataManagement.Save(basket);
      Console.ReadKey();
    }
  }
}
```

GetAllProducts

```
public static List < Product > GetAllProducts() {
  var result = new List < Product > ();
  // Cautam pe disk toate fisierele care incep cu "Product_";
  var files = Directory.GetFiles(".", "Product_*.json");
  foreach(var file in files) {
    var jsonValue = File.ReadAllText(file);
    result.Add(JsonConvert.DeserializeObject < Product > (jsonValue));
  }
  return result;
}
```

Temă

Să se adauge o interfață grafică (UI) de tip `Windows Forms` la aplicația anterioară care să permită următoarele:

1. Adăugare produse (`Product`);
2. Creare comenzi (`Order`) și adăugare în coș de cumparaturi (`Basket`)

Observație: interesant ar fi să permiteți utilizatorului să selecteze doar din lista de produse deja create (existente pe disk, în urma activității de la 1.)

- Cum s-ar implementa?

Utilizați metoda `GetAllProducts` și aflați toate codurile de bare pe care le puteți afișa într-o listă de timp `combobox`; de acolo poate selecta user-ul codul de bare dorit.

3. Salvare produse, comenzi, basket pe disk
4. Modificarea / editarea datelor despre produse / comenzi (salvarea pe disk a modificărilor)
5. Vizualizarea conținutului coșului de cumpărături (se vor include inclusiv informații despre produsele comandate, nu doar codul de bare!)
6. Eliminarea comenzii din coș
7. Calcul cost total comenzi din coș