using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using RestaurantManager.Interfaces;

namespace RestaurantManager.Models

{

public class Restaurant: IRestaurant

{

private string name;

private string location;

//private IList<IRecipe> Recipes = new List<IRecipe>();

public Restaurant(string name, string location)

{

this.name = name;

this.location = location;

this.Recipes = new List<IRecipe>();

}

public string Name

{

get { return this.name; }

set

{

Validator.CheckIfStringIsNullOrEmpty(

value,

string.Format(

ErrorMessages.IsRequired, "The restaurant name "));

this.name = value;

}

}

public string Location

{

get { return this.location; }

set

{

Validator.CheckIfStringIsNullOrEmpty(

value,

string.Format(

ErrorMessages.IsRequired, "The restaurant location "));

this.location = value;

}

}

public IList<IRecipe> Recipes { get; private set; }

public void AddRecipe(IRecipe recipe)

{

this.Recipes.Add(recipe);

}

public void RemoveRecipe(IRecipe recipe)

{

this.Recipes.Remove(recipe);

}

public string PrintMenu()

{

StringBuilder output = new StringBuilder();

output.AppendFormat("\*\*\*\*\* {0} - {1} \*\*\*\*\*", this.Name, this.Location).AppendLine();

if (this.Recipes.Count == 0)

{

output.AppendLine("No recipies... yet");

return output.ToString();

}

//output.AppendFormat(this.)

}

// EBojilova

private string GetMenuGroup(string groupName, string title)

{

var menuGroup = this.Recipes

.Where(r => r.GetType().Name == groupName)

.OrderBy(x => x.Name);

if (menuGroup.Any())

{

var sb = new StringBuilder();

sb.AppendFormat("~~~~~~ {0} ~~~~~", title).AppendLine();

sb.Append(string.Join(String.Empty, menuGroup.Select(x => x.ToString())));

return sb.ToString();

}

return string.Empty;

}

}

}

public override string ToString()

{

StringBuilder toPrint = new StringBuilder();

var measurement = this.Unit == MetricUnit.Grams ? "g" : "ml";

toPrint.AppendFormat("== {0} == ${1:F2}", this.Name, this.Price).AppendLine();

toPrint.AppendFormat("Per serving: {0} {1}, {2} kcal", this.QuantityPerServing, measurement, this.Calories)

.AppendLine();

toPrint.AppendFormat("Ready in {0} minutes", this.TimeToPrepare);

return toPrint.ToString();

}

•••+++•••+++•••+++

В абстрактния клас (Recipie) пропъртито е virtual a set-ъра е protected

public override int Calories

{

get { return base.Calories; }

protected set

{

if (value > MaxCalories)

{

throw new ArgumentException("The calories in the drinks must not be greater than 100.");

}

base.Calories = value;

}

}

---++++ парсване на enum за да бъде в create метод

public IMainCourse CreateMainCourse(string name, decimal price, int calories, int quantityPerServing, int timeToPrepare, bool isVegan, **string type**)

{

MainCourseType **typeEnum** = (MainCourseType) Enum.Parse(typeof (MainCourseType), type);

return new MainCourse(name, price, calories, quantityPerServing, timeToPrepare, isVegan, **typeEnum**);

}

+++ FACTORY

public class RecipeFactory : IRecipeFactory

{

public IDrink CreateDrink(string name, decimal price, int calories, int quantityPerServing, int timeToPrepare, bool isCarbonated)

{

return new Drink(name, price, calories, quantityPerServing, MetricUnit.Milliliters, timeToPrepare, isCarbonated);

}

public ISalad CreateSalad(string name, decimal price, int calories, int quantityPerServing, int timeToPrepare, bool containsPasta)

{

return new Salad(name, price, calories, quantityPerServing, timeToPrepare, containsPasta);

}

public IMainCourse CreateMainCourse(string name, decimal price, int calories, int quantityPerServing, int timeToPrepare, bool isVegan, string type)

{

MainCourseType typeEnum = (MainCourseType) Enum.Parse(typeof (MainCourseType), type);

return new MainCourse(name, price, calories, quantityPerServing, timeToPrepare, isVegan, typeEnum);

}

public IDessert CreateDessert(string name, decimal price, int calories, int quantityPerServing, int timeToPrepare, bool isVegan)

{

return new Dessert(name, price, calories, quantityPerServing, timeToPrepare, isVegan);

}

}