# Healthy Heaven

## Preparation

Download the skeleton provided in Judge. **Do not** change the **StartUp** class or its **namespace**.

## Problem description

Your task is to create a repository (a restaurant), which stores salads by creating the **classes** described below.

### Vegetable

First, write a C# class Vegetable with the following properties:

* **Name: string**
* **Calories: int**

The class **constructor** should receive **name** and **calories**.

The class also should have the methods:

* Override the **ToString()** method in the following format:

**" - {name} have {calories} calories"**

### Salad

**Next**, write a **C#** class Salad that has **products** (a collection field, which stores the entity **Vegetable**). All entities inside the repository have the **same properties**. Also, the Salad class should have those properties:

* **Name: string**

The class **constructor** should receive **name**, also it should initialize the **products** with a new instance of the collection**.**

The class also should have the methods:

* **int G**etTotalCalories() – returns the sum of all vegetable calories in the salad
* **int** **G**etProductCount() - **returns** the **number** of products
* **void A**dd(Vegetable product) – adds an entity to the products
* Override ToString() – by the format bellow:

**"\* Salad {name} is {calories} calories and have {product count} products:  
{Vegetable 1}  
{Vegetable 2}  
{Vegetable 3}  
{…}"**

### Restaurant

**Next**, write a **C#** class Restaurant that has **data** (a collection field, which stores the entity **Salad**). All entities inside the repository have the **same properties**. Also, the Restaurant class should have those properties:

* **Name: string**

The class **constructor** should receive **name**, also it should initialize the **data** with a new instance of the collection.

Implement the following features:

* Field data – **collection** that holds added salads
* Method Add(Salad salad) – adds an entity to the data
* Method Buy(string name) – removes a salad by given name, if such exists, and returns boolean
* Mehod GetHealthiestSalad() – returns the healthiest salad
* Method GenerateMenu() - **returns** a **string** in the following **format**:

"**{name} have {salad count} salads:  
{Salad 1}  
{Salad 2}  
{…}**"

## Constraints

* The **names** of the vegetables and salads will be **always unique**.
* The **calories** of the vegetables will always be with **positive values**.

## Examples

This is an example how the **Restaurant** class is **intended to be used**.

|  |
| --- |
| Sample code usage |
| // Initialize the repository  Restaurant restaurant = new Restaurant("Casa Domingo");  // Initialize the entities  Vegetable tomato = new Vegetable("Tomato", 20);  Vegetable cucumber = new Vegetable("Cucumber", 15);  Salad salad = new Salad("Tomatoes with cucumbers");  salad.Add(tomato);  salad.Add(cucumber);  Console.WriteLine(salad.GetTotalCalories()); // 35  Console.WriteLine(salad.GetProductCount()); // 2  Console.WriteLine(salad.ToString());  // \* Salad Tomatoes with cucumbers is 35 calories and have 2 products:  // - Tomato have 20 calories  // - Cucumber have 15 calories  restaurant.Add(salad);  Console.WriteLine(restaurant.Buy("Invalid salad")); // False  // Initialize the second entities  Vegetable corn = new Vegetable("Corn", 90);  Salad casaDomingo = new Salad("Casa Domingo");  casaDomingo.Add(tomato);  casaDomingo.Add(cucumber);  casaDomingo.Add(corn);  restaurant.Add(casaDomingo);  Console.WriteLine(restaurant.GetHealthiestSalad()); // Tomatoes with cucumbers  Console.WriteLine(restaurant.GenerateMenu());  // Casa Domingo have 2 salads:  // \* Salad Tomatoes with cucumbers is 35 calories and have 2 products:  // - Tomato have 20 calories  // - Cucumber have 15 calories  // \* Salad Casa Domingo is 125 calories and have 3 products:  // - Tomato have 20 calories  // - Cucumber have 15 calories  // - Corn have 90 calories |

## Submission

Zip all the files in the project folder except **bin** and **obj** folders