Problem 1 – Restaurant Management

You are assigned to create a restaurant management system. The system keeps track of **restaurants** and **recipes**.

Restaurants have **name**, **location**, and a set of **recipes**. They can **add** and **remove** recipes from their menus, and **print** their menus in a customer-friendly way. A recipe may belong to more than one restaurant at the same time.

Recipes have **name**, **price**, **calories per serving**, **quantity per serving**, and **time to prepare**. There are two main types of recipes – **meals** and **drinks**. A **meal** can be **vegan** or not, and a **drink** can be **carbonated** or not. A **meal** can be one of the following: **salad** (always **vegan**, can optionally **contain pasta**), **main course** (has **type**), and **dessert** (may be prepared with sugar or with some other sweetener). It is possible for a dessert to be both vegan and sugar-free – because, you know vegans are always whining about something in their food. If not explicitly specified, assume the desserts contain sugar.

Design the Class Hierarchy

Your task is **to design an object-oriented class hierarchy** following the best practices in object-oriented programming (OOP) and object-oriented design (OOD). **Avoid duplicated code** through **abstraction**, **inheritance** and **polymorphism** and **encapsulate all fields** correctly.

You are given a few C# **interfaces** that you should **obligatorily** implement and use as the basis for your code:

```
public interface IRestaurant
{
    string Name { get; }
    string Location { get; }
    IList<IRecipe> Recipes { get; }
    void AddRecipe(IRecipe recipe);
    void RemoveRecipe(IRecipe recipe);
    string PrintMenu();
}

public interface IRecipe
{
    string Name { get; }
    decimal Price { get; }
    int Calories { get; }
    int QuantityPerServing { get; }
    MetricUnit Unit { get; }
    int TimeToPrepare { get; }
}

public interface IMeal : IRecipe
{
    bool IsVegan { get; }
    void ToggleVegan(); // Turns "vegan" to "not vegan" and vice versa
}

public interface IDrink : IRecipe
{
    bool IsCarbonated { get; }
}

public interface ISalad : IMeal
{
    bool ContainsPasta { get; }
```

```
}
public interface IMainCourse : IMeal
{
    MainCourseType Type { get; }
}
public interface IDessert : IMeal
{
    bool WithSugar { get; }
    void ToggleSugar(); // Turns "with sugar" to "without sugar" and vice versa
}
```

All **restaurants** should implement **IRestaurant**. All **recipes** should implement **IRecipe**. All **meals** should implement **IMeal**, and all drinks should implement **IDrink**. All **salads** should implement **ISalad**. All **main courses** should implement **IMainCourse**. All desserts should implement **IDessert**.

You should follow these validity rules very strictly in order to ensure the data integrity within the system:

Restaurant validity rules:

- Name and location are required (cannot be null or empty).

Recipe validity rules:

- Name is required (cannot be null or empty).
- Price, calories, quantity per serving, and time to prepare must be positive.

Meal validity rules:

- The measuring unit for all meals is g (grams).

Drink validity rules:

- The calories in a drink must not be greater than 100.
- The time to prepare a drink must not be greater than 20 minutes.
- The measuring unit for all drinks is ml (milliliters).

Salad validity rules:

- A salad must always be vegan

Throw appropriate exception when data validation fails. When there is **an error in a parameter** (such as a missing required parameter), throw an **ArgumentException**, and when there is a **forbidden or meaningless method**, throw an **InvalidOperationException**.

The messages for the exceptions should be as follows:

- For required parameters: **The <parameter> is required.**
- For parameters with only positive values: **The <parameter> must be positive.**

Always provide the lowest possible visibility for properties and methods.

A programmer using the system **must not be able to create instances of classes implementing IMeal and IRecipe directly**.

All restaurants can be created only through **IRestaurantFactory** implemented by the class **RestaurantFactory**. All recipes can be created only through **IRecipeFactory** implemented by the class **RecipeFactory**. Both classes **RestaurantFactory** and **RecipeFactory** are located in the namespace **RestaurantManager.Engine.Factories**.

The **restaurant menu** should return information in the following form:

```
***** <name> - <location> *****
<recipes>
```

<recipes> can be one of the following:

```
No recipes... yet
```

if the restaurant still does not have any recipe added to it, or a list of recipes, ordered by type. **If there are no recipes of a certain type, it must not be present in the menu**:

```
~~~~~ DRINKS ~~~~~
~~~~~ SALADS ~~~~~
~~~~~ MAIN COURSES ~~~~~
~~~~~ DESSERTS ~~~~~
```

All recipes in a category should be **ordered alphabetically by name**. Refer to the example to gain a deeper understanding how the menu should work.

Drinks are presented in the following form:

```
==  <name> == $<price>
Per serving: <quantity> <unit>, <calories> kcal
Ready in <time_to_prepare> minutes
Carbonated: <yes / no>
```

Main courses are presented in the following form:

```
<[VEGAN] >==  <name> == $<price>
Per serving: <quantity> <unit>, <calories> kcal
Ready in <time_to_prepare> minutes
Type: <type>
```

Salads are presented in the following form:

```
<[VEGAN] >==  <name> == $<price>
Per serving: <quantity> <unit>, <calories> kcal
Ready in <time_to_prepare> minutes
Contains pasta: <yes / no>
```

Desserts are presented in the following form:

```
<[NO SUGAR] ><[VEGAN] >==  <name> == $<price>
Per serving: <quantity> <unit>, <calories> kcal
Ready in <time_to_prepare> minutes
```

Round all prices to 2 digits after the decimal separator. The **[VEGAN]** and **[NO SUGAR]** labels should only be printed when needed.

Additional Notes

You are given a **command execution engine** to simplify your work. Please put all your classes in the **RestaurantManager.Models** namespace and all factories in the **RestaurantManager.Engine.Factories** namespace.

You are not allowed to change anything else in the project. The evaluating team will replace all other files with their initial content, so do not try to cheat :).

The engine accepts the following commands:

- **CreateRestaurant(name=<name>;location=<location>)**
- **CreateDrink(name=<name>;price=<price>;calories=<calories>;quantity=<quantity>;time=<time>;carbonated=<yes / no>)**
- **CreateSalad(name=<name>;price=<price>;calories=<calories>;quantity=<quantity>;time=<time>;pasta=<yes / no>)**
- **CreateMainCourse(name=<name>;price=<price>;calories=<calories>;quantity=<quantity>;time=<time>;vegan=<yes / no>;type=<type>)**
- **CreateDessert(name=<name>;price=<price>;calories=<calories>;quantity=<quantity>;time=<time>;vegan=<yes / no>)**
- **AddRecipeToRestaurant(restaurant=<restaurant>;recipe=<recipe>)**
- **RemoveRecipeFromRestaurant(restaurant=<restaurant>;recipe=<recipe>)**
- **PrintRestaurantMenu(name=<name>)**
- **ToggleVegan(name=<name>)**
- **ToggleSugar(name=<name>)**

The **engine is already implemented** correctly and you do not need to touch it. The parameters may be provided in any sequence. The engine returns appropriate messages for each command. Duplicate restaurant and recipe names are not allowed. You may refer to the sample input and output for more details.

Sample Input

```
CreateRestaurant(name=New Restaurant;location=Sofia)
CreateRestaurant(location=Silicon Valley;name=SoftUni Restaurant)
PrintRestaurantMenu(name=New Restaurant)
CreateMainCourse(name=Grilled
Chicken;price=5.88;calories=320;quantity=370;time=15;vegan=no;type=Meat)
CreateMainCourse(name=Spaghetti
Carbonara;time=25;price=7.39;type=Pasta;calories=455;quantity=450;vegan=no)
CreateSalad(price=7.99;name=Mexican Bean Salad;pasta=no;quantity=300;time=14;calories=150)
CreateDessert(calories=450;name=Black Magic Cake;quantity=150;price=1.500001;vegan=no;time=2)
CreateDrink(name=Home-made Lemonade;price=2.41;carbonated=no;calories=10;time=5;quantity=200)
PrintRestaurantMenu(name=SoftUni Restaurant)
AddRecipeToRestaurant(recipe=Black Magic Cake;restaurant=SoftUni Restaurant)
AddRecipeToRestaurant(restaurant=SoftUni Restaurant;recipe=Grilled Chicken)
AddRecipeToRestaurant(restaurant=SoftUni Restaurant;recipe=Mexican Bean Salad)
AddRecipeToRestaurant(recipe=Home-made Lemonade;restaurant=SoftUni Restaurant)
AddRecipeToRestaurant(restaurant=SoftUni Restaurant;recipe=Spaghetti Carbonara)
PrintRestaurantMenu(name=SoftUni Restaurant)
AddRecipeToRestaurant(restaurant=New Restaurant;recipe=Spaghetti Carbonara)
PrintRestaurantMenu(name=New Restaurant)
RemoveRecipeFromRestaurant(restaurant=New Restaurant;recipe=Spaghetti Carbonara)
PrintRestaurantMenu(name=New Restaurant)
RemoveRecipeFromRestaurant(recipe=Spaghetti Carbonara;restaurant=SoftUni Restaurant)
RemoveRecipeFromRestaurant(recipe=Grilled Chicken;restaurant=SoftUni Restaurant)
PrintRestaurantMenu(name=SoftUni Restaurant)
CreateMainCourse(name=Vegan Red Lentil
Soup;vegan=yes;price=5.99;quantity=250;time=15;calories=150;type=Soup)
AddRecipeToRestaurant(recipe=Vegan Red Lentil Soup;restaurant=New Restaurant)
PrintRestaurantMenu(name=New Restaurant)
ToggleVegan(name=Vegan Red Lentil Soup)
```

```
PrintRestaurantMenu(name=New Restaurant)
CreateDessert(name=Black Chocolate
Cake;quantity=120;price=2.32;vegan=yes;time=6;calories=300)
AddRecipeToRestaurant(recipe=Black Chocolate Cake;restaurant=New Restaurant)
PrintRestaurantMenu(name=New Restaurant)
ToggleSugar(name=Black Chocolate Cake)
PrintRestaurantMenu(name=New Restaurant)
PrintRestaurantMenu(name=No Such Restaurant)
AddRecipeToRestaurant(restaurant=No Such Recipe;recipe=No Such Recipe)
AddRecipeToRestaurant(restaurant=New Restaurant;recipe=No Such Recipe)
ToggleSugar(name=Grilled Chicken)
ToggleVegan(name=Home-made Lemonade)
End
```

Sample Output

```
Restaurant New Restaurant created
Restaurant SoftUni Restaurant created
***** New Restaurant - Sofia *****
No recipes... yet
Recipe Grilled Chicken created
Recipe Spaghetti Carbonara created
Recipe Mexican Bean Salad created
Recipe Black Magic Cake created
Recipe Home-made Lemonade created
***** SoftUni Restaurant - Silicon Valley *****
No recipes... yet
Recipe Black Magic Cake successfully added to restaurant SoftUni Restaurant
Recipe Grilled Chicken successfully added to restaurant SoftUni Restaurant
Recipe Mexican Bean Salad successfully added to restaurant SoftUni Restaurant
Recipe Home-made Lemonade successfully added to restaurant SoftUni Restaurant
Recipe Spaghetti Carbonara successfully added to restaurant SoftUni Restaurant
***** SoftUni Restaurant - Silicon Valley *****
~~~~~ DRINKS ~~~~~
==  Home-made Lemonade == $2.41
Per serving: 200 ml, 10 kcal
Ready in 5 minutes
Carbonated: no
~~~~~ SALADS ~~~~~
[VEGAN] ==  Mexican Bean Salad == $7.99
Per serving: 300 g, 150 kcal
Ready in 14 minutes
Contains pasta: no
~~~~~ MAIN COURSES ~~~~~
==  Grilled Chicken == $5.88
Per serving: 370 g, 320 kcal
Ready in 15 minutes
Type: Meat
==  Spaghetti Carbonara == $7.39
Per serving: 450 g, 455 kcal
Ready in 25 minutes
Type: Pasta
~~~~~ DESSERTS ~~~~~
==  Black Magic Cake == $1.50
Per serving: 150 g, 450 kcal
Ready in 2 minutes
Recipe Spaghetti Carbonara successfully added to restaurant New Restaurant
***** New Restaurant - Sofia *****
~~~~~ MAIN COURSES ~~~~~
```

```
==  Spaghetti Carbonara == $7.39
Per serving: 450 g, 455 kcal
Ready in 25 minutes
Type: Pasta
Recipe Spaghetti Carbonara successfully removed from restaurant New Restaurant
***** New Restaurant - Sofia *****
No recipes... yet
Recipe Spaghetti Carbonara successfully removed from restaurant SoftUni Restaurant
Recipe Grilled Chicken successfully removed from restaurant SoftUni Restaurant
***** SoftUni Restaurant - Silicon Valley *****
~~~~~ DRINKS ~~~~~
==  Home-made Lemonade == $2.41
Per serving: 200 ml, 10 kcal
Ready in 5 minutes
Carbonated: no
~~~~~ SALADS ~~~~~
[VEGAN] ==  Mexican Bean Salad == $7.99
Per serving: 300 g, 150 kcal
Ready in 14 minutes
Contains pasta: no
~~~~~ DESSERTS ~~~~~
==  Black Magic Cake == $1.50
Per serving: 150 g, 450 kcal
Ready in 2 minutes
Recipe Vegan Red Lentil Soup created
Recipe Vegan Red Lentil Soup successfully added to restaurant New Restaurant
***** New Restaurant - Sofia *****
~~~~~ MAIN COURSES ~~~~~
[VEGAN] ==  Vegan Red Lentil Soup == $5.99
Per serving: 250 g, 150 kcal
Ready in 15 minutes
Type: Soup
Command ToggleVegan executed successfully. New value: false
***** New Restaurant - Sofia *****
~~~~~ MAIN COURSES ~~~~~
==  Vegan Red Lentil Soup == $5.99
Per serving: 250 g, 150 kcal
Ready in 15 minutes
Type: Soup
Recipe Black Chocolate Cake created
Recipe Black Chocolate Cake successfully added to restaurant New Restaurant
***** New Restaurant - Sofia *****
~~~~~ MAIN COURSES ~~~~~
==  Vegan Red Lentil Soup == $5.99
Per serving: 250 g, 150 kcal
Ready in 15 minutes
Type: Soup
~~~~~ DESSERTS ~~~~~
[VEGAN] ==  Black Chocolate Cake == $2.32
Per serving: 120 g, 300 kcal
Ready in 6 minutes
Command ToggleSugar executed successfully. New value: false
***** New Restaurant - Sofia *****
~~~~~ MAIN COURSES ~~~~~
==  Vegan Red Lentil Soup == $5.99
Per serving: 250 g, 150 kcal
Ready in 15 minutes
Type: Soup
~~~~~ DESSERTS ~~~~~
[NO SUGAR] [VEGAN] ==  Black Chocolate Cake == $2.32
```

```
Per serving: 120 g, 300 kcal
Ready in 6 minutes
The restaurant No Such Restaurant does not exist
The restaurant No Such Recipe does not exist
The recipe No Such Recipe does not exist
The command ToggleSugar is not applicable to recipe Grilled Chicken
The command ToggleVegan is not applicable to recipe Home-made Lemonade
```