**Кухня**

**Общ преглед**

Във вашата фирма постъпва проект за създаване на приложение, обслужващо кухня в ресторант.

Вашият софтуер трябва да описва **ястия** (Dish) и **готвачи (**Chef**).**

Tрябва да реализирате функционалност, която да позволява добавяне на **ястия** и **готвачи**, а в последствие приготвяне и издаване на тези ястия – всичко това ще работи чрез **команди**, които вие ще получавате. Поредицата от команди приключва с „**End**”. За ваше удобство ще получите готов Program.cs файл, а вие ще трябва да реализирате само необходимите класове в Dish.cs и Chef.cs

**Подзадача 1: Структура на Ястие – 15 точки**

**Dish**

Всички ястия имат име и цена:

name – низ, съставен от малки и големи латински букви, без други специални знаци

                Уникална стойност, която различава всички ястия

foodGroup – низ, съставен от малки и големи латински букви, без други специални знаци

                Стойност, която показва типа на яденето. Пример – „vegan”

price – число с плаваща запетая

Реализирайте конструктор:

* public Dish(string name, string foodGroup, double price)– този конструктор трябва да приема името, групата и цената на ястието.

За справка вижте следната схема за Dish.cs:

|  |
| --- |
| Dish.cs |
| public Dish(string name, string foodGroup, double price)  {      //TODO: Добавете вашия код тук …  } |

**Команди за добавяне на ястия**

Трябва да довършите следната команда за добавяне на продукти:

* **CreateDish** <**име**> <**група**> <**цена**> - тази команда има за цел да добави ястие с неговите име, група и цена. Командата е описана в Main метода.

За да сработи се нуждае от горепосочения конструктор в класа Dish!

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за ястие. Командата за това е следната:

* **PrintDishInfoByName** <**име**> - отпечатва информация за ястие във формат:  
  **Dish: <име> of type <foodGroup>. Price <цена>**
  + Цената, следва да бъде форматирана до втория знак след десетичния разделител

Тази команда ще получава винаги валидни и съществуващи имена на ястия. За успешна реализация трябва да реализирате ваша версия на **ToString()** метода за класа **Dish.**

**Подзадача 2: Структура на Готвач – 15 точки**

**Chef**

Всеки готвач има: име, департамент, списък от ястия, които може да приготвя и флаг дали е в почивка

name – низ, съставен от малки и големи латински букви, без други специални знаци.

Уникална стойност, която различава всички коли

department– низ, съставен само от големи латински букви, но без други специални знаци

dishes – списък от **Dish**

price – число с плаваща запетая

isOnABreak – булев флаг, показваш дали готвачът е в почивка или не

Трябва да реализирате конструктор:

* public Chef(string name, string department)– този конструктор трябва да приема име и департамент

За справка вижте следната схема за  Chef.cs:

|  |
| --- |
| Chef.cs |
| public Chef(string name, string department)  {        //TODO: Добавете вашия код тук …    } |

**Имайте предвид, че в момента на добавянето на нов готвач, той следва да НЕ бъде в почивка. Погрижете се за валидните състояния на всички полета в класа!**

**Команда за добавяне на готвач**

* **AddChef** <**име**> <**департамент**>- тази команда има за цел да добави нов готвач.

Командата е описана в Main метода. За да сработи се нуждае от горепосочения конструктор в класа Chef!

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за готвач. Командата за това е следната:

* **PrintChefInfoByName <име> -** отпечатва информация за готвач във формат:  
  With total price of: <цена> lv.

Chef **<име>** from department **<департамент>** is able to cook the following dishes:

<ястие>

<ястие>...

Тази команда ще получава винаги валидни и съществуващи имена на готвачи. За успешна реализация трябва да реализирате ваша версия на **ToString()** метода за класа **Chef.** Очаква се да заместите стойностите подадени в диамантени скоби с реални такива. За всяко ястие използвайте ToString() метода от Dish.cs!

**Подзадача 3: Логика – 40 точки**

**Kоманди**

Вашето приложение трябва да реализира следните команди:

* **AddDishToSpecificChef <име на ястие> <име на готвач> -** Тази команда добавя даденото ястие към списъка от ястия на готвача. *Командата използва методът* **void AddDish(Dish dish)** *от класа Chef.cs*

* **RemoveDishFromSpecificChef <име на ястие> <име на готвач>.** Тази команда премахва ястието от списъка с ястия на готвача. *Командата използва методът* **bool RemoveDish(Dish dish)** *от класа Chef.cs.* **Методът трябва да връща true стойност, ако успее да изтрие ястие и false, ако не е същестувало такова ястие в списъка с ястия на готвача.**
* **RemoveAllDishesByFoodGroupFromSpecificChef <foodGroup> <име на готвач>.** Тази команда премахва всички ястия, които са от посочената foodGroup от списъка с ястия на готвача.

*Командата използва методът* bool RemoveAllByFoodGroup(string foodGroup) *от класа Chef.cs.*

**Методът трябва да връща true стойност, ако успее да изтрие дори едно ястие и false, ако не са същестували ястия с тази група в списъка с ястия на готвача.**

* **CountExpensiveDishesOfFoodGroupFromSpecificChef <foodGroup> <priceLevel> <име на готвач>.** Тази команда следва да преброи всички ястия, които са от посочената foodGroup от списъка с ястия на готвача и имат цена по-голяма или равна на посочената.

*Командата използва методът* int CountExpensiveDishesOfFoodGroup(string foodGroup, double priceLevel) *от класа Chef.cs.*

**Методът трябва да връща целочислена стойност, което представлява броя на всички ястия в списъка от ястия, които отговарят на посочените критерии за група и цена.**

* **StartCookingChef <име на готвач>.** Тази команда следва да промени статуса на флага isOnAbreak на false и условно да прекъсне почивката на готвача. *Командата използва методът*

void StartCooking() *от класа Chef.cs.*

**Подзадача 4: Бонус логика – 20 точки**

Трябва да реализирате още няколко команди:

* **DeliverDish** <**име на ястие**> <**име на готвач**> **- *Командата използва методът***

**Dish DeliverDish(string dishName) *от класа Chef.cs.*** Методът трябва да провери дали такова ястие е налично в списъка от ястия И дали готвачът е на **разположение** (тоест не е в почивка ). Ако ДА, то следва да върнете като стойност обект от тип Dish, който отговаря на подаденото име от списъка с ястия. В обратен случай **върнете null стойност**.

* **GiveChefABreak** <**име на готвач**> **– *Командата използва методът*** void GiveChefABreak() ***от класа Chef.cs.*** Методът трябва да променя стойността на флага isOnABreak да бъде **true.**

* **IsChefAvailable** <**име на готвач**> **- *Командата използва методът*** bool IsChefAvailable() ***от класа Chef.cs.*** Методът трябва върне стойност **true**, ако флагът **isOnABreak е false** и обратното.

**За безпроблемната работа на всички изброени команди от 3 и 4 подзадача трябва да реализирате Dish.cs и Chef.cs, по аналогичен начин на показаното по-долу:**

|  |
| --- |
| Chef.cs |
| public Chef(string name, string department)  {      //TODO: Добавете вашия код тук …  }    public string Name  {      //TODO: Добавете вашия код тук …  }    public string Department  {      //TODO: Добавете вашия код тук …  }    public bool IsOnABreak  {       //TODO: Добавете вашия код тук …  }    public void AddDish(Dish dish)  {       //TODO: Добавете вашия код тук …  }    public bool RemoveDish(Dish dish)  {       //TODO: Добавете вашия код тук …  }    public bool RemoveAllByFoodGroup(string foodGroup)  {       //TODO: Добавете вашия код тук …  }    public int CountExpensiveDishesOfFoodGroup(string foodGroup, double priceLevel)  {       //TODO: Добавете вашия код тук …  }    public void StartCooking()  {       //TODO: Добавете вашия код тук …  }    public Dish DeliverDish(string dishName)  {       //TODO: Добавете вашия код тук …  }    public void GiveChefABreak()  {       //TODO: Добавете вашия код тук …  }      public bool IsChefAvailable()  {       //TODO: Добавете вашия код тук …  }    public override string ToString()  {       //TODO: Добавете вашия код тук …  } |

|  |
| --- |
| Part.cs |
| public Dish(string name, string foodGroup, double price)  {       //TODO: Добавете вашия код тук …  }    public string Name  {       //TODO: Добавете вашия код тук …  }    public string FoodGroup  {       //TODO: Добавете вашия код тук …  }    public double Price  {       //TODO: Добавете вашия код тук …  }    public override string ToString()  {       //TODO: Добавете вашия код тук …  } |

**Забележка:** Освен горепосочените методи трябва да реализирате и необходимите свойства за всеки от класовете. Възможно е да е удачно да реализирате допълнителни полета, свойства и методи, по ваша преценка.

**Подзадача 5: Валидация – 10 точки**

Освен всичко останало вие трябва да направите и валидация!

Не допускайте създаването на:

* Ястие с цена по-малка или равна на 0.00, message => **"Invalid dish price!"**
* Готвач с име, с дължина по-малка или равна на 2 символа , message => **"Invalid chef name!"**
* Департаментът в класа Chef трябва да бъде дума изписана само с ГЛАВНИ букви, message => **"Department name should be uppercase word!"**

При невалидни данни хвърлете грешка от тип **ArgumentException(“message”)**

**Вход / Изход**

**Вход**

* Програмата ще получава множество редове с информация. Всеки ред представлява команда. Самият вход се обработва изцяло от примерния Program.cs.
* Всички команди приключват с въвеждането на End

**Изход**

За някои от командите не е нужно да извеждате нищо. За всички останали изпечатването ще ви бъде дадено в Program.cs, освен ToString() методите, които са оставени на вас.

**Ограничения**

* Всички цели числа ще бъдат в диапазона **–10000** до **+10000**
* Всички числа с плаваща запетая ще бъдат въвеждани с до **15** знака след запетаята.
* Имената няма да съдържат интервал

**Примери**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| CreateDish kebapche mesni 1.19 |  |
| PrintDishInfoByName kebapche | Dish: kebapche of type mesni. Price 1.19 |
| AddChef Petrov MESOGURO  AddDishToSpecificChef kebapche Petrov |  |
| PrintChefInfoByName Petrov | Chef Petrov from department MESOGURO is able to cook the following dishes:  Dish: kebapche of type mesni. Price 1.19 |
| CreateDish kufte mesni 1.199797987688  CreateDish duner vegan 5.655  AddDishToSpecificChef kufte Petrov  AddDishToSpecificChef duner Petrov |  |
| PrintChefInfoByName Petrov | Chef Petrov from department MESOGURO is able to cook the following dishes:  Dish: kebapche of type mesni. Price 1.19  Dish: kufte of type mesni. Price 1.20  Dish: duner of type vegan. Price 5.66 |
| RemoveDishFromSpecificChef kufte Petrov | Successfully removed dish: kufte from chef Petrov! |
| PrintChefInfoByName Petrov | Chef Petrov from department MESOGURO is able to cook the following dishes:  Dish: kebapche of type mesni. Price 1.19  Dish: duner of type vegan. Price 5.66 |
| RemoveAllDishesByFoodGroupFromSpecificChef vegan Petrov | Successfully removed dishes with food group: vegan from chef Petrov! |
| PrintChefInfoByName Petrov | Chef Petrov from department MESOGURO is able to cook the following dishes:  Dish: kebapche of type mesni. Price 1.19 |
| StartCookingChef Petrov |  |
| DeliverDish kebapche Petrov | Chef Petrov successfully cooked kebapche |
| DeliverDish kufte Petrov | Chef Petrov failed to cook kufte! |
| GiveChefABreak Petrov |  |
| IsChefAvailable Petrov | Chef Petrov is NOT available! |
| DeliverDish duner Petrov | Chef Petrov failed to cook duner! |
| CreateDish parjolaaa mesni -5 | Invalid dish price! |
| CreateDish parjolaaa mesni 0 | Invalid dish price! |
| AddChef P SLADKAR | Invalid chef name! |
| AddChef Pa SLADKAR | Invalid chef name! |
| End |  |

**Точки**

Разбивката по подзадачи е следната:

1. **15** точки, като трябва да имате и задължително реализиран ToString() метод
2. **15** точки, като трябва да имате и задължително реализиран ToString() метод
3. **40** точки
4. **20** точки
5. **10** точки

Общ брой точки: **100**