Аптека

Общ преглед

Във вашата фирма постъпва проект за създаване на приложение, обслужващо аптеки.

Вашият софтуер трябва да описва **лекарство** (Medicine) и **аптека (**Pharmacy**).**

Трябва да реализирате функционалност, която да позволява добавяне на **лекарства** и **аптеки**, а в последствие купуване, продаване и други операции с тези лекарства – 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 Medicine.cs и Pharmacy.cs

Подзадача 1: Структура на Лекарство (Medicine) – 15 точки

Medicine

Всички лекарства имат име и цена:

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

Уникална стойност, която различава всички лекарства

price – число с плаваща запетая

Реализирайте конструктор:

- public Medicine(string name, double price)– този конструктор трябва да приема името и цената на лекарството

За справка вижте следната схема за Medicine.cs:

Medicine.cs

public Medicine(string name, double price){
 	
//TODO: Добавете вашия код тук …
 	
}

Команда за добавяне на лекарства

Трябва да довършите следната команда за добавяне на лекарства:

- **AddMedicine** <**име**> <**цена**> - тази команда има за цел да добави лекарство с неговите име и цена. Командата е описана в Main метода.

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

Команда за извеждане на информация

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

- **MedicineInfo** <**име**> - отпечатва информация за лекарство във формат:

  Medicine: <име> with price <цена>

  - Цената, следва да бъде форматирана до втория знак след десетичния разделител

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

Подзадача 2: Структура на Аптека (Pharmacy) – 15 точки

Company

Всяка аптека има: име и списък от лекарства, с които оперира

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

Уникална стойност, която различава всички аптеки

List of Medicine – списък с обекти от клас Medicine

Трябва да реализирате конструктор:

- public Pharmacy(string name)– този конструктор трябва да приема само име, **но да създава валидно състояние в полето реферирщо към списък от лекарства**

За справка вижте следната схема за Pharmacy.cs:

Pharmacy.cs

public Pharmacy(string name) {
 	
//TODO: Добавете вашия код тук …
 	
}

Погрижете се за валидните състояния на всички полета в класа!

Команда за добавяне на компания

- **CreatePharmacy** <**име**> - тази команда има за цел да добави нова аптека.

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

Команда за извеждане на информация

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

- **PharmacyInfo <име> -** отпечатва информация за аптека във формат:

1. Когато има поне едно лекарство:

   Pharmacy **<име>** has **<брой лекарства>** medicines and they are:

   Medicine: <име> with price <цена>

   Medicine: <име> with price <цена>
2. Когато няма лекарства:

Pharmacy **<име>** has **0** medicines and they are:
N/A

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

Подзадача 3: Логика – 40 точки

Команди

Вашето приложение трябва да реализира следните команди:

- **Order <име на лекарство> <име на аптека>** - Тази команда **добавя** даденото лекарство към списъка от лекарства в аптеката. *Командата използва методът* **void Order(Medicine medicine)** *от класа* **Pharmacy***.cs.* Методът трябва да добавя в списъка с лекарства подаденото.

- **Sell <име на лекарство> <име на аптека>** - Тази команда **премахва** дадено лекарство от списъка с лекарства на аптеката. *Командата използва*

метода **bool Sell(Medicine medicine)** *от класа Pharmacy.cs.* Методът трябва да трие референцията на подаденото лекарство от списъка с лекарства. **Ако лекарството е успешно премахнато (тоест такова е имало в списъка), методът трябва да връща true като стойност, в обратен случай false.**

- **CalculateTotalPrice <име на аптека>** - Тази команда **сумира** цените на всички лекарства от списъка с лекартства на аптеката. *Командата използва метода* double CalculateTotalPrice() *от класа Pharmacy.cs.* **Методът следва да сумира всички цени на лекарствата в аптеката и да върне double стойност.**

- **RenamePharmacy <име на аптека> <ново име на аптека>**- Тази команда **променя** името на **аптеката**. *Командата използва метода* void RenamePharmacy(string newName) *от класа Pharmacy.cs.* **Методът следва да промени стойността в полето name, като разбира се се подчинява на правилата за валидация, описани в по-долната секция.**

- **SellAllMedicines <име на аптека>.** Тази има за цел да продаде всички лекарства от дадена аптека. Използва се метода void SellAllMedicines() *от класа Pharmacy.cs.* **Методът следва да премахне всички лекарства от списъка.**

Подзадача 4: Бонус логика – 20 точки

Трябва да реализирате още няколко команди:

- **GetMedicineWithHighestPrice** <**име на аптека**> - Командата използва метода **Medicine GetMedicineWithHighestPrice()** *от класа Pharmacy.cs.* Методът следва да обхожда всички лекарства от списъка с лекарства на дадената аптека и да намери референция към това, което има най-висока цена. Няма да има лекарства с равни цени. Трябва да върнете стойност обект от тип Medicine.
- **GetMedicineWithLowestPrice** <**име на аптека**> - Командата използва метода **Medicine GetMedicineWithLowestPrice()** *от класа Pharmacy.cs.* Методът следва да обхожда всички лекарства от списъка с лекарства на дадената аптека и да намери референция към това, което има най-ниска цена. Няма да има лекарства с равни цени. Трябва да върнете стойност обект от тип Medicine.

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

Pharmacy.cs

```
public Pharmacy(string name)
{
   //TODO: Добавете вашия код тук …
}

public string Name
{
   //TODO: Добавете вашия код тук …
}

public void Order(Medicine medicine)
{
   //TODO: Добавете вашия код тук …
}

public bool Sell(Medicine medicine)
{
   //TODO: Добавете вашия код тук …
}

public double CalculateTotalPrice()
{
   //TODO: Добавете вашия код тук …
```

```
}

public Medicine GetMedicineWithHighestPrice()
{
   //TODO: Добавете вашия код тук …
}

public Medicine GetMedicineWithLowestPrice()
{
   //TODO: Добавете вашия код тук …
}

public void RenamePharmacy(string newName)
{
   //TODO: Добавете вашия код тук …
}

public void SellAllMedicines()
{
   //TODO: Добавете вашия код тук …
}

public override string ToString()
{
   //TODO: Добавете вашия код тук …
```

```
}
```

Medicine.cs

```
public Medicine(string name, double price)

{

    //TODO: Добавете вашия код тук …

}



public string Name

{

    //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:

- Лекарство с цена по-малка от 0, message => **"Invalid price"**
- Аптека с име, с дължина по-малка от 3 символа , message => **"Invalid name"**

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 числа с плаваща запетая ще бъдат въвеждани с до **15** знака след запетаята.
- Имената няма да съдържат интервал

Примери

Вход	Изход
AddMedicine Analgin 3.70 MedicineInfo Analgin CreatePharmacy Pharmacy1 PharmacyInfo Pharmacy1 Order Paracetamol Pharmacy1 PharmacyInfo Pharmacy1 Sell Analgin Pharmacy1	Medicine: Analgin with price 3.70 Pharmacy Pharmacy1 has 0 medicines and they are: N/A Could not order medicine Pharmacy Pharmacy1 has 0 medicines and they are: N/A

PharmacyInfo Pharmacy1 AddMedicine Degan 8.70 Order Analgin Pharmacy1 Order Paracetamol Pharmacy1 PharmacyInfo Pharmacy1 GetMedicineWithHighestPrice Pharmacy1 CalculateTotalPrice Pharmacy1 SellAllMedicines Pharmacy1 CalculateTotalPrice Pharmacy1 PharmacyInfo Pharmacy1 RenamePharmacy Pharmacy1 Pharmacy2 Order Benalgin Pharmacy2 Order NeoAngin Pharmacy2 AddMedicine Mukusolvan -1000 CreatePharmacy M End	Did not sell medicine Analgin Pharmacy Pharmacy1 has 0 medicines and they are: N/A Could not order medicine Pharmacy Pharmacy1 has 1 medicines and they are: Medicine: Analgin with price 3.70 Medicine: Analgin with price 3.70 3.7 0 Pharmacy Pharmacy1 has 0 medicines and they are: N/A Could not order medicine Could not order medicine Invalid price Invalid name

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**