Автокъща

Общ преглед

Във вашата фирма постъпва проект за създаване на приложение, обслужващо автокъщи. Вашият софтуер трябва да описва **кола (Car)** и **автокъща (Store).**

Трябва да реализирате функционалност, която да позволява добавяне на **коли** и **автокъщи**, а в последствие добавяне, продаване и други операции с тези коли – всичко това ще работи чрез **команди**, които вие ще получавате. Поредицата от команди приключва с команда **STOP**. За ваше удобство ще получите готов Program.cs файл, а вие ще трябва да реализирате само необходимите класове в Car.cs и Store.cs

Подзадача 1: Структура на Кола (Car) – 15 точки

Car

Всички коли имат марка, модел и цена:

number – цяло число; уникален номер за всяка кола

price – число с плаваща запетая

Реализирайте конструктор:

- public Car(int number, double price)– този конструктор трябва да приема номерът и цената на колата

За справка вижте следната схема за Car.cs:

Car.cs

public Car(int number, double price){
 	
//TODO: Добавете вашия код тук …
 	
}

Команда за добавяне на коли

Трябва да довършите следната команда за добавяне на коли:

- **AddCar** <**номер** > <**цена**> - тази команда има за цел да добави кола с номера и цената и. Командата е описана в Main метода.

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

Подзадача 2: Структура на Автокъща (Store) – 15 точки

Store

Всяка автокъща има: име и списък от коли, с които разполага.

name – низ, съставен от малки и големи латински букви; уникално име за всяка автокъща

List of Car – списък с обекти от клас Car

Трябва да реализирате конструктор:

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

За справка вижте следната схема за Store.cs:

Store.cs

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

Погрижете се за валидните състояния на всички полета в класа!

Команда за добавяне на автокъща

- **CreateStore** <**име**> - тази команда има за цел да добави нова автокъща

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

Команда за извеждане на информация

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

- **StoreInfo <име> -** отпечатва информация за автокъщата във формат:

1. Когато има поне една кола:

   Store <име> has <брой коли> car/s:

   Car number <номер> costs {цена}

   Car number <номер> costs {цена}

2. Когато няма коли:

Store <име> has no available cars.

Тази команда ще получава винаги валидни и съществуващи имена на автокъщи. За успешна реализация трябва да реализирате ваша версия на **ToString()** метода за класа **Store.** Очаква се да заместите стойностите подадени в диамантени скоби с реални такива. За всяка колаизползвайте ToString() метода от Car.cs!

Подзадача 3: Логика – 40 точки

Команди

Вашето приложение трябва да реализира следните команди:

- AddCar **<номер> <цена> <име на автокъща>** - Тази команда **добавя** дадената кола към списъка от коли на автокъщата. *Командата използва методът* **void AddCar(Car)** *от класа Store.cs.* Методът трябва да добавя в списъка с коли подадения.

- SellCar **<номер> <цена> <име на автокъща>** - Тази команда **премахва** дадена кола от списъка с коли на автокъщата. *Командата използва метода* **bool SellCar(Car)** *от класа Store.cs.* Методът трябва да трие референцията на подадената кола от списъка с коли. **Ако колата е успешно премахната (тоест такава е имало в списъка), методът трябва да връща true като стойност, в обратен случай false.**

- CalculateTotalPrice **<име на автокъща>** - Тази команда **сумира** цените на всички коли от списъка сколи на автокъщата. *Командата използва метода* double CalculateTotalPrice() *от класа Store.cs.* **Методът следва да сумира всички цени на колите в автокъщата и да върне double стойност.**

- RenameStore **<име на автокъща> <ново име на автокъща>** - Тази команда **променя** името на **автокъщата**. *Командата използва метода* void RenameStore(string newName) *от класа Store.cs.* **Методът следва да промени**

стойността в полето name, като разбира се се подчинява на правилата за валидация, описани в по-долната секция.

- SellAllCars **<име на компания> -** Тази има за цел да продаде всички коли от една автокъща. Използва се метода void SellAllCars() *от класа Store.cs.* **Методът следва да премахне всички коли от списъка.**

- GetCarWithHighestPrice <**име на автокъща**> - Командата използва метода **Car** GetCarWithHighestPrice**()** *от класа Store.cs.* Методът следва да обхожда всички коли от списъка с коли на дадената автокъща и да намери референция към тази, която има най-висока цена. Няма да има коли с еднакви цени. Трябва да върнете стойност обект от тип Car.
- GetCarWithLowestPrice <**име на автокъща** > - Командата използва метода **Car** GetCarWithLowestPrice *от класа Store.cs.* Методът следва да обхожда всички коли от списъка с коли на дадената автокъща и да намери референция към тази, която има най-ниска цена. Няма да има коли с еднакви цени. Трябва да върнете стойност обект от тип Car.

Подзадача 4: Бонус логика – 20 точки

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

Store.cs

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

```
}

public void AddCar(Car car)
{
    //TODO: Добавете вашия код тук …
}

public bool SellCar(Car car)
{
    //TODO: Добавете вашия код тук …
}

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

public Car GetCarWithHighestPrice()
{
    //TODO: Добавете вашия код тук …
}

public Car GetCarWithLowestPrice()
{
    //TODO: Добавете вашия код тук …
```

```
}


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


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


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

Car.cs

```
public Car(int number, double price)
{
     //TODO: Добавете вашия код тук …
}


public int Number
{
```

```
    //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:

- Автокъща с име, с дължина по-малка от 5 символа , message => **"**Invalid store name!**"**
- Кола с цена по-малка от 1000, message => **"**Invalid car price!**"**

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 STOP

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
CreateStore MyStore StoreInfo MyStore TestInvalid AddCar 1 8000 MyStore AddCar 14 4000 MyStore StoreInfo MyStore SellCar 1 8000 MyStore StoreInfo MyStore AddCar 2 2800.50 MyStore AddCar 3 3500.50 MyStore AddCar 4 4000.50 MyStore StoreInfo MyStore CalculateTotalPrice MyStore	You created store MyStore. Store MyStore has no available cars. Invalid command! You added car with number 1 to store MyStore. You added car with number 14 to store MyStore. Store MyStore has 2 car/s: Car number 1 costs 8000.00 Car number 14 costs 4000.00 You sold car with number 1 from store MyStore. Store MyStore has 1 car/s: Car number 14 costs 4000.00 You added car with number 2 to store MyStore. You added car with number 3 to store MyStore.

SellAllCars MyStore CalculateTotalPrice MyStore StoreInfo MyStore RenameStore MyStore MyNewStore StoreInfo MyNewStore AddCar 6 12000.50 MyNewStore AddCar 7 22000.50 MyNewStore GetCarWithHighestPrice MyNewStore AddCar 8 1500.20 MyNewStore GetCarWithLowestPrice MyNewStore CreateStore NewestStore StoreInfo NewestStore STOP	You added car with number 4 to store MyStore. Store MyStore has 4 car/s: Car number 14 costs 4000.00 Car number 2 costs 2800.50 Car number 3 costs 3500.50 Car number 4 costs 4000.50 Total price: 14301.50 You sold all cars from store MyStore. Total price: 0.00 Store MyStore has no available cars. You renamed your store from MyStore to MyNewStore. Store MyNewStore has no available cars. You added car with number 6 to store MyNewStore. You added car with number 7 to store MyNewStore. Car from store MyNewStore has highest price: 22000.50 You added car with number 8 to store MyNewStore. Car from store MyNewStore has lowest price: 1500.20 You created store NewestStore. Store NewestStore has no available cars.
Вход	**Изход**
CreateStore a CreateStore ab CreateStore abc CreateStore abcd	Invalid store name! Invalid store name! Invalid store name! Invalid store name!

CreateStore store	You created store store.
AddCar 1 300 store	Invalid car price!
AddCar 2 999 store	Invalid car price!
AddCar 3 500 store	Invalid car price!
AddCar 1 1000 store	You added car with number 1 to store store.
AddCar 6 1000 store	You added car with number 6 to store store.
RenameStore store s	Invalid store name!
RenameStore store st	Invalid store name!
RenameStore store sto	Invalid store name!
RenameStore store stor	Invalid store name!
RenameStore store store1	You renamed your store from store to store1.
STOP	

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**

```
using System;
using System.Collections.Generic;
namespace ITCareer_OOP_Exam_1
{
    class Program
    {
        static Dictionary<int, Car> cars = new Dictionary<int, Car>();
        static Dictionary<string, Store> stores = new Dictionary<string, Store>();
        static void Main(string[] args)
        {
            string input;
            while ((input = Console.ReadLine()) != "STOP")
            {
                string[] splittedInput = input.Split(' ');
```

```
                string command = splittedInput[0];
                switch (command)
                {
                    case "AddCar":
                        AddCar(int.Parse(splittedInput[1]), double.Parse(splittedInput[2]),
splittedInput[3]);
                        break;
                    case "SellCar":
                        SellCar(int.Parse(splittedInput[1]),
double.Parse(splittedInput[2]), splittedInput[3]);
                        break;
                    case "CalculateTotalPrice":
                        CalculateTotalPrice(splittedInput[1]);
                        break;
                    case "GetCarWithHighestPrice":
                        GetCarWithHighestPrice(splittedInput[1]);
                        break;
                    case "GetCarWithLowestPrice":
                        GetCarWithLowestPrice(splittedInput[1]);
                        break;
                    case "RenameStore":
                        RenameStore(splittedInput[1], splittedInput[2]);
                        break;
                    case "SellAllCars":
                        SellAllCars(splittedInput[1]);
                        break;
                    case "StoreInfo":
                        StoreInfo(splittedInput[1]);
                        break;
                    case "CreateStore":
                        CreateStore(splittedInput[1]);
                        break;
                    default:
                        Console.WriteLine("Invalid command!");
                        break;
                }
            }
        }
        private static void AddCar(int number, double price, string name)
        {
            try
            {
                Car car = new Car(number, price);
                if (!stores.ContainsKey(name))
                {
                    Console.WriteLine("Could not add this car to your store.");
                    return;
                }
                    Store store = stores[name];
                    store.AddCar(car);
                    Console.WriteLine($"You added car with number {number} to store
{store.Name}.");
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        private static void SellCar(int number, double price, string name)
        {
            try
            {
                if (!stores.ContainsKey(name))
                {
                    Console.WriteLine("Could not sell this car from your store.");
                    return;
```

```
                }
                Car car = new Car(number, price);
                Store store = stores[name];
                if (store.SellCar(car))
                {
                    Console.WriteLine($"You sold car with number {number} from store
{name}.");
                }
                else
                {
                    Console.WriteLine($"Did not sell car  with number {number} from store
{name}.");
                }
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        private static void CalculateTotalPrice(string name)
        {
            try
            {
                if (!stores.ContainsKey(name))
                {
                    Console.WriteLine("Could not calculate total price.");
                    return;
                }
                Store store = stores[name];
                Console.WriteLine($"Total price: {store.CalculateTotalPrice():F2}");
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        private static void RenameStore(string name, string newName)
        {
            if (!stores.ContainsKey(name))
            {
                Console.WriteLine($"Could not rename the store {name}.");
                return;
            }
            Store store = stores[name];
            try
            {
                store.RenameStore(newName);
                stores.Remove(name);
                stores.Add(newName, store);
                Console.WriteLine($"You renamed your store from {name} to {newName}.");
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        private static void SellAllCars(string name)
        {
            if (!stores.ContainsKey(name))
            {
                Console.WriteLine($"Could not sell all cars store {name}.");
                return;
            }
            Store store = stores[name];
            store.SellAllCars();
            Console.WriteLine($"You sold all cars from store {name}.");
```

```
        }
        private static void StoreInfo(string name)
        {
            if (!stores.ContainsKey(name))
            {
                Console.WriteLine($"Could not get store {name}.");
                return;
            }
            Store store = stores[name];
            Console.WriteLine(store.ToString());
        }
        private static void GetCarWithLowestPrice(string name)
        {
            if (!stores.ContainsKey(name))
            {
                Console.WriteLine($"Could not get car with lowest price from store
{name}.");
                return;
            }
            Store store = stores[name];
            Console.WriteLine($"Car from store {name} has lowest price:
{store.GetCarWithLowestPrice().Price:F2}");
        }
        private static void GetCarWithHighestPrice(string name)
        {
            if (!stores.ContainsKey(name))
            {
                Console.WriteLine($"Could not get car with highest price from store
{name}.");
                return;
            }
            Store store = stores[name];
            Console.WriteLine($"Car from store {name} has highest price:
{store.GetCarWithHighestPrice().Price:F2}");
        }
        private static void CreateStore(string name)
        {
            try
            {
                Store store = new Store(name);
                stores.Add(name, store);
                Console.WriteLine($"You created store {name}.");
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

    }
}
```

Car.cs

Store.cs