## Furniture Store

**Due date**: Sunday, 22 May 2022, 14:00  
**Requested files**: Furniture.cs, FurnitureStore.cs, Program.cs ([Download](https://it-kariera.mon.bg/e-learning/mod/vpl/views/downloadrequiredfiles.php?id=7604))  
**Type of work**: Individual work

# Мебелен магазин

### Общ преглед

Във вашата фирма постъпва проект за създаване на приложение, обслужващо верига мебелни магазини. Вашият софтуер трябва да описва **мебел** **(Furniture)** и **мебелен магазин (FurnitureStore).**

Tрябва да реализирате функционалност, която да позволява добавяне на **мебели** и **мебелни магазини**, а в последствие добавяне, продаване и други операции с тези мебели – 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 Furniture.cs и FurnitureStore.cs

**ВАЖНО: Използвайте един и същ namespace при всички класове и нека той да бъде RetakeExam.**

## Структура на Мебел (Furniture)

## Furniture

Всички мебели имат вид (type), цвят (color) и цена (price):

**type** – текст

**color** – текст

**price** – десетично число

**Реализирайте конструктор:**

* public Furniture(string type, string color, double price) – този конструктор трябва да приема вида, цвета и цената на мебелта

За справка вижте следната схема за Furniture.cs:

|  |
| --- |
| **Furniture.cs** |
| public Furniture(string type, string color, double price) {  //TODO: Добавете вашия код тук …  } |

### Команда за добавяне на мебел

Трябва да довършите следната команда за добавяне на мебели:

* **AddFurniture** <**вид**> <**цвят**> <**цена**> - тази команда има за цел да добави мебел с дадения вид, цвят и цена. Командата е описана в Main метода.

**За да сработи се нуждае от горепосочения конструктор в класа Funriture!**

## Структура на мебелен магазин (FurnitureStore)

### FurnitureStore

Всеки мебелен магазин има: име и списък с мебели, с които разполага.

**name** – низ, съставен от малки и големи латински букви; всеки магазин е с уникално име

**List of Furniture** – списък с обекти от клас Furniture

Трябва да реализирате конструктор:

* public FurnitureStore (string name) – този конструктор трябва да приема само име, **но да създава валидно състояние в полето рефериращо към списъка с мебели**

За справка вижте следната схема за FurnitureStore.cs:

|  |
| --- |
| **FurnitureStore.cs** |
| public FurnitureStore (string name) {  //TODO: Добавете вашия код тук …  } |

**Погрижете се за валидните състояния на всички полета в класа!**

### Команда за добавяне на мебелен магазин

* **CreateFurnitureStore** <**име**> - тази команда има за цел да добави нов мебелен магазин

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

### Команда за извеждане на информация

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

* **FurnitureStoreInfo <име> -** отпечатва информация за мебелния магазин във формат:

1. Когато има поне една налична мебел в списъка с мебели:  
   **Furniture store <име> has <брой мебели> furniture/s:  
   Furniture <вид> with color <цвят> costs <цена>  
   Furniture <вид> with color <цвят> costs <цена>**
2. Когато няма налични мебели в мебелния магазин:

**Furniture store <име> has no available furnitures.**

Тази команда ще получава винаги валидни и съществуващи имена на мебелни магазини. За успешна реализация трябва да реализирате ваша версия на **ToString()** метода за класа **FurnitureStore.** Цената на мебелите трябва да бъде форматирана до втория знак след десетичната запетая. Очаква се да заместите стойностите подадени в диамантени скоби с реални такива. За всяко цвете използвайте ToString() метода от Furniture.cs!

## Логика

### Kоманди

Вашето приложение трябва да реализира следните команди:

* AddFurniture **<вид> <цвят> <цена> <име на мебелен магазин>** - Тази команда **добавя** дадената мебел към списъка от цмебели на мебелния магазин. *Командата използва методът* **void AddFurniture(Furniture furniture)** *от класа FurnitureStore.cs.*
* SellFurniture **<вид> <цвят> <цена> <име на мебелен магазин>** - Тази команда **премахва** дадената мебел от списъка с мебели на мебелния магазин. *Командата използва метода* **bool SellFurniture(Furniture furniture)** *от класа FurnitureStore.cs.* Методът трябва да трие референцията на подадената мебел от списъка с мебели на мебелния магазин. **Ако мебелта е успешно премахната (тоест такава е имало в списъка), методът трябва да връща true като стойност, в обратен случай false.**
* CalculateTotalPrice **<име на мебелен магазин>** - Тази команда **сумира** цените на всички мебели от списъка с мебели на мебелния магазин. *Командата използва метода* **double CalculateTotalPrice()** *от класа FurnitureStore.cs.* **Методът следва да сумира всички цени на мебелите в магазина и да върне double стойност.**
* RenameFurnitureStore **<име на мебелен магазин> <ново име на мебелен магазин>** - Тази команда **променя** името на **магазина**. *Командата използва метода* **void RenameFurnitureStore(string newName)** *от класа FurnitureStore.cs.* **Методът следва да промени стойността в полето name, като следва правилата за валидация, описани в по-долната секция.**
* SellAllFurnitures **<име на мебелен магазин> -** Тази има за цел да продаде всички мебели от един мебелен магазин. Използва се метода **void SellAllFurnitures()** *от класа FurnitureStore.cs.* **Методът****следва да премахне всички мебели от списъка.**

## Бонус логика

* **GetFurnitureWithHighestPrice** <**име на мебелен магазин**> - Командата използва метода **GetFurnitureWithHighestPrice()** *от класа FurnitureStore.cs.* Методът следва да обхожда всички мебели от списъка с мебели на дадения магазин и да намери референция към тази, която има най-висока цена. Няма да има мебели с еднакви цени. Трябва да върнете стойност обект от тип Furniture.
* **GetFurnitureWithLowestPrice** <**име на мебелен магазин**> - Командата използва метода **GetFurnitureWithLowestPrice** *от класа FurnitureStore.cs.* Методът следва да обхожда всички мебели от списъка с мебели на дадения магазин и да намери референция към тази, която има най-ниска цена. Няма да има мебели с еднакви цени. Трябва да върнете стойност обект от тип Furniture.

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

|  |
| --- |
| **FurnitureStore.cs** |
| public FurnitureStore(string name)  {  //TODO: Добавете вашия код тук …  }  public string Name  {  //TODO: Добавете вашия код тук …  }  public void AddFurniture(Furniture furniture)  {  //TODO: Добавете вашия код тук …  }  public bool SellFurniture(Furniture furniture)  {  //TODO: Добавете вашия код тук …  }  public double CalculateTotalPrice()  {  //TODO: Добавете вашия код тук …  }  public Flower GetFurnitureWithHighestPrice()  {  //TODO: Добавете вашия код тук …  }  public Flower GetFurnitureWithLowestPrice()  {  //TODO: Добавете вашия код тук …  }  public void RenameFurnitureStore(string newName)  {  //TODO: Добавете вашия код тук …  }  public void SellAllFurnitures()  {  //TODO: Добавете вашия код тук …  }  public override string ToString()  {  //TODO: Добавете вашия код тук …  } |

|  |
| --- |
| **Furniture.cs** |
| public Furniture (string type, string color, double price)  {  //TODO: Добавете вашия код тук …  }  public string Type  {  //TODO: Добавете вашия код тук …  }  public string Color  {  //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.

## Валидация

Освен всичко останало вие трябва да направите и валидация!

Не допускайте създаването на:

* Maгазин с име, с дължина по-малка от 6 символа, message => **"**Invalid furniture store name!**"**
* Мебел с цена по-голяма от 100, message => **"**Invalid furniture price!**"**

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

### 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 отпечатването 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** знакa.
* Имената няма да съдържат интервал.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| CreateFurnitureStore MyStore  FurnitureStoreInfo MyStore  TestInvalid  AddFurniture table red 3.40 MyStore  AddFurniture sofa yellow 8.70 MyStore  FurnitureStoreInfo MyStore  SellFurniture table red 3.40 MyStore  FurnitureStoreInfo MyStore  AddFurniture chair white 2.40 MyStore  AddFurniture lamp yellow 3.50 MyStore  FurnitureStoreInfo MyStore  CalculateTotalPrice MyStore  SellAllFurnitures MyStore  CalculateTotalPrice MyStore  FurnitureStoreInfo MyStore  RenameFurnitureStore MyStore MyNewStore  FurnitureStoreInfo MyNewStore  AddFurniture table black 12.30 MyNewStore  AddFurniture door pink 3.50 MyNewStore  GetFurnitureWithHighestPrice MyNewStore  AddFurniture carpet orange 2.20 MyNewStore  GetFurnitureWithLowestPrice MyNewStore  CreateFurnitureStore NewestStore  FurnitureStoreInfo NewestStore  STOP | You created furniture store MyStore.  Furniture store MyStore has no available furnitures.  Invalid command!  You added furniture table with color red to store MyStore.  You added furniture sofa with color yellow to store MyStore.  Frniture store MyStore has 2 furniture/s:  Fruniture table with color red costs 3.40  Fruniture sofa with color yellow costs 8.70  You sold furniture table with color red from furniture store MyStore.  Frniture store MyStore has 1 furniture/s:  Fruniture sofa with color yellow costs 8.70  You added furniture chair with color white to store MyStore.  You added furniture lamp with color yellow to store MyStore.  Frniture store MyStore has 3 furniture/s:  Fruniture sofa with color yellow costs 8.70  Fruniture chair with color white costs 2.40  Fruniture lamp with color yellow costs 3.50  Total price: 14.60  You sold all furnitures from store MyStore.  Total price: 0.00  Furniture store MyStore has no available furnitures.  You renamed your store from MyStore to MyNewStore.  Furniture store MyNewStore has no available furnitures.  You added furniture table with color black to store MyNewStore.  You added furniture door with color pink to store MyNewStore.  Furniture from store MyNewStore has highest price: 12.30  You added furniture carpet with color orange to store MyNewStore.  Furniture from store MyNewStore has lowest price: 2.20  You created furniture store NewestStore.  Furniture store NewestStore has no available furnitures. |
| **Вход** | **Изход** |
| CreateFurnitureStore a  CreateFurnitureStore ab  CreateFurnitureStore abc  CreateFurnitureStore abcd  CreateFurnitureStore stored  AddFurniture table white 2.40 stored  AddFurniture sofa yellow 3.50 stored  RenameFurnitureStore stored s  RenameFurnitureStore stored st  RenameFurnitureStore stored sto  RenameFurnitureStore stored stor  RenameFurnitureStore stored stored1  STOP | Invalid furniture store name!  Invalid furniture store name!  Invalid furniture store name!  Invalid furniture store name!  You created furniture store stored.  You added furniture table with color white to store stored.  You added furniture sofa with color yellow to store stored.  Invalid furniture store name!  Invalid furniture store name!  Invalid furniture store name!  Invalid furniture store name!  You renamed your store from stored to stored1. |

## Точки

Разбивката по подзадачи е следната:

1. Налична структура за Furniture и FurnitureStore, ToString методи и реализирана AddFurniture команда **- 30** точки
2. Логика - **40** точки
3. Бонус логика - **20** точки
4. Валидация - **10** точки

Общ брой точки: **100**

## Requested files

#### Furniture.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace RetakeExam

{

class Furniture

{

public Furniture(string type, string color, double price)

{

}

public string Type

{

}

public string Color

{

}

public double Price

{

}

public override string ToString()

{

}

}

}

#### FurnitureStore.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RetakeExam

{

class FurnitureStore

{

public FurnitureStore(string name)

{

}

public string Name

{

}

public void AddFurniture(Furniture furniture)

{

}

public bool SellFurniture(Furniture furniture)

{

}

public double CalculateTotalPrice()

{

}

public Furniture GetFurnitureWithHighestPrice()

{

}

public Furniture GetFurnitureWithLowestPrice()

{

}

public void RenameFurnitureStore(string newName)

{

}

public void SellAllFurnitures()

{

}

public override string ToString()

{

}

}

}

#### Program.cs

using System;

using System.Collections.Generic;

namespace RetakeExam

{

class Program

{

static Dictionary<string, FurnitureStore> stores = new Dictionary<string, FurnitureStore>();

static void Main(string[] args)

{

string input;

while ((input = Console.ReadLine()) != "STOP")

{

string[] splittedInput = input.Split(' ');

string command = splittedInput[0];

switch (command)

{

case "AddFurniture":

AddFurniture(splittedInput[1], splittedInput[2], double.Parse(splittedInput[3]), splittedInput[4]);

break;

case "SellFurniture":

SellFurniture(splittedInput[1], splittedInput[2], double.Parse(splittedInput[3]), splittedInput[4]);

break;

case "CalculateTotalPrice":

CalculateTotalPrice(splittedInput[1]);

break;

case "GetFurnitureWithHighestPrice":

GetFurnitureWithHighestPrice(splittedInput[1]);

break;

case "GetFurnitureWithLowestPrice":

GetFurnitureWithLowestPrice(splittedInput[1]);

break;

case "RenameFurnitureStore":

RenameFurnitureStore(splittedInput[1], splittedInput[2]);

break;

case "SellAllFurnitures":

SellAllFurnitures(splittedInput[1]);

break;

case "FurnitureStoreInfo":

FurnitureStoreInfo(splittedInput[1]);

break;

case "CreateFurnitureStore":

CreateFurnitureStore(splittedInput[1]);

break;

default:

Console.WriteLine("Invalid command!");

break;

}

}

}

private static void AddFurniture(string type, string color, double price, string name)

{

try

{

Furniture furniture = new Furniture(type, color, price);

if (!stores.ContainsKey(name))

{

Console.WriteLine("Could not add this furniture to your store.");

return;

}

FurnitureStore store = stores[name];

store.AddFurniture(furniture);

Console.WriteLine($"You added furniture {type} with color {color} to store {store.Name}.");

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

}

private static void SellFurniture(string type, string color, double price, string name)

{

try

{

if (!stores.ContainsKey(name))

{

Console.WriteLine("Could not sell this furniture from your store.");

return;

}

Furniture furniture = new Furniture(type, color, price);

FurnitureStore store = stores[name];

if (store.SellFurniture(furniture))

{

Console.WriteLine($"You sold furniture {type} with color {color} from furniture store {name}.");

}

else

{

Console.WriteLine($"Did not sell furniture {type} with color {color} from furniture 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;

}

FurnitureStore store = stores[name];

Console.WriteLine($"Total price: {store.CalculateTotalPrice():F2}");

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

}

private static void RenameFurnitureStore(string name, string newName)

{

if (!stores.ContainsKey(name))

{

Console.WriteLine($"Could not rename the store {name}.");

return;

}

FurnitureStore store = stores[name];

try

{

store.RenameFurnitureStore(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 SellAllFurnitures(string name)

{

if (!stores.ContainsKey(name))

{

Console.WriteLine($"Could not sell all furnitures from store {name}.");

return;

}

FurnitureStore store = stores[name];

store.SellAllFurnitures();

Console.WriteLine($"You sold all furnitures from store {name}.");

}

private static void FurnitureStoreInfo(string name)

{

if (!stores.ContainsKey(name))

{

Console.WriteLine($"Could not get store {name}.");

return;

}

FurnitureStore store = stores[name];

Console.WriteLine(store.ToString());

}

private static void GetFurnitureWithLowestPrice(string name)

{

if (!stores.ContainsKey(name))

{

Console.WriteLine($"Could not get furniture with lowest price from store {name}.");

return;

}

FurnitureStore store = stores[name];

Console.WriteLine($"Furniture from store {name} has lowest price: {store.GetFurnitureWithLowestPrice().Price:F2}");

}

private static void GetFurnitureWithHighestPrice(string name)

{

if (!stores.ContainsKey(name))

{

Console.WriteLine($"Could not get furniture with highest price from store {name}.");

return;

}

FurnitureStore store = stores[name];

Console.WriteLine($"Furniture from store {name} has highest price: {store.GetFurnitureWithHighestPrice().Price:F2}");

}

private static void CreateFurnitureStore(string name)

{

try

{

FurnitureStore store = new FurnitureStore(name);

stores.Add(name, store);

Console.WriteLine($"You created furniture store {name}.");

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

}

}

}