## Flower Shop

**Avaliable from**: Sunday, 15 May 2022, 10:00  
**Due date**: Sunday, 15 May 2022, 14:00  
**Requested files**: Flower.cs, FlowerStore.cs, Program.cs ([Download](https://it-kariera.mon.bg/e-learning/mod/vpl/views/downloadrequiredfiles.php?id=7603))  
**Type of work**: Individual work

### Общ преглед

Във вашата фирма постъпва проект за създаване на приложение, обслужващо верига магазини за цветя. Вашият софтуер трябва да описва **цвете** **(Flower)** и **магазин за цветя (FlowerStore).**

Tрябва да реализирате функционалност, която да позволява добавяне на **цветя** и **магазини за цветя**, а в последствие добавяне, продаване и други операции с тези цветя – 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 Flower.cs и FlowerStore.cs

## ****ВАЖНО: Използвайте namespace RegularExam!****

## Структура на Цвете (Flower)

## Flower

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

**type** – текст

**color** – текст

**price** – десетично число

**Реализирайте конструктор:**

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

За справка вижте следната схема за Flower.cs:

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

### Команда за добавяне на цветя

Трябва да довършите следната команда за добавяне на цветя:

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

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

**За да сработят успешно структурните тестове, трябва да имплементирате и ToString() метод за Flower, съгласно следния формат:**

Flower <вид> with color <цвят> costs <цена>

## Структура на Магазин за цветя (FlowerStore)

### FlowerStore

Всеки магазин за цветя има: име и списък с цветя, с които разполага.

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

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

Трябва да реализирате конструктор:

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

За справка вижте следната схема за FlowerStore.cs:

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

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

### Команда за добавяне на магазин за цветя

* **CreateFlowerStore** <**име**> - тази команда има за цел да добави нов магазин за цветя

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

### Команда за извеждане на информация

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

* **FlowerStoreInfo <име> -** отпечатва информация за магазин за цветя във формат:

1. Когато има поне едно цвете:  
   **Flower store <име> has <брой цветя> flower/s:  
   Flower <вид> with color <цвят> costs <цена>  
   Flower <вид> with color <цвят> costs <цена>**
2. Когато няма налични цветя в магазина:

**Flower store <име> has no available flowers.**

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

## Логика

### Kоманди

Вашето приложение трябва да реализира следните команди:

* AddFlower **<вид> <цвят> <цена> <име на магазин за цветя>** - Тази команда **добавя** даденото цвете към списъка от цветя на магазина. *Командата използва методът* **void AddFlower(Flower flower)** *от класа FlowerStore.cs.*
* SellFlower **<вид> <цвят> <цена> <име на магазин за цветя>** - Тази команда **премахва** даденото цвете от списъка с цветя на магазина. *Командата използва метода* **bool SellFlower(Flower flower)** *от класа FlowerStore.cs.* Методът трябва да трие референцията на подаденото цвете от списъка с цветя. **Ако цветето е успешно премахнато (тоест такова е имало в списъка), методът трябва да връща true като стойност, в обратен случай false.**
* CalculateTotalPrice **<име на магазин за цветя>** - Тази команда **сумира** цените на всички цветя от списъка с цветя на магазина. *Командата използва метода* **double CalculateTotalPrice()** *от класа FlowerStore.cs.* **Методът следва да сумира всички цени на цветята в магазина и да върне double стойност.**
* RenameFlowerStore **<име на магазин за цветя> <ново име на магазин за цветя>** - Тази команда **променя** името на **магазина**. *Командата използва метода* **void RenameFlowerStore(string newName)** *от класа FlowerStore.cs.* **Методът следва да промени стойността в полето name, като следва правилата за валидация, описани в по-долната секция.**
* SellAllFlowers **<име на магазин за цветя> -** Тази има за цел да продаде всички цветя от един магазин за цветя. Използва се метода **void SellAllFlowers()** *от класа FlowerStore.cs.* **Методът****следва да премахне всички цветя от списъка.**

## Бонус логика

* **GetFlowerWithHighestPrice** <**име на магазин за цветя**> - Командата използва метода **GetFlowerWithHighestPrice()** *от класа FlowerStore.cs.* Методът следва да обхожда всички цветя от списъка с цветя на дадения магазин и да намери референция към това, което има най-висока цена. Няма да има цветя с еднакви цени. Трябва да върнете стойност обект от тип Flower.
* **GetFlowerWithLowestPrice** <**име на магазин за цветя**> - Командата използва метода **GetFlowerWithLowestPrice** *от класа FlowerStore.cs.* Методът следва да обхожда всички цветя от списъка с цветя на дадения магазин и да намери референция към това, което има най-ниска цена. Няма да има цветя с еднакви цени. Трябва да върнете стойност обект от тип Flower.

**За безпроблемната работа на всички изброени команди трябва да реализирате Flower.cs и FlowerStore.cs, по аналогичен начин на показаното по-долу:**

|  |
| --- |
| **FlowerStore.cs** |
| public FlowerStore(string name)  {  //TODO: Добавете вашия код тук …  }  public string Name  {  //TODO: Добавете вашия код тук …  }  public void AddFlower(Flower flower)  {  //TODO: Добавете вашия код тук …  }  public bool SellFlower(Flower flower)  {  //TODO: Добавете вашия код тук …  }  public double CalculateTotalPrice()  {  //TODO: Добавете вашия код тук …  }  public Flower GetFlowerWithHighestPrice()  {  //TODO: Добавете вашия код тук …  }  public Flower GetFlowerWithLowestPrice()  {  //TODO: Добавете вашия код тук …  }  public void RenameFlowerStore(string newName)  {  //TODO: Добавете вашия код тук …  }  public void SellAllFlowers()  {  //TODO: Добавете вашия код тук …  }  public override string ToString()  {  //TODO: Добавете вашия код тук …  } |
| **Flower.cs** |
| public Flower (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 flower store name!**"**
* Цвете с цена по-голяма от 100, message => **"**Invalid flower price!**"**

При невалидни данни хвърлете грешка от тип **ArgumentException**

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

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| CreateFlowerStore MyStore  FlowerStoreInfo MyStore  TestInvalid  AddFlower tulip red 3.40 MyStore  AddFlower rose yellow 8.70 MyStore  FlowerStoreInfo MyStore  SellFlower tulip red 3.40 MyStore  FlowerStoreInfo MyStore  AddFlower dahlia white 2.40 MyStore  AddFlower narcissus yellow 3.50 MyStore  FlowerStoreInfo MyStore  CalculateTotalPrice MyStore  SellAllFlowers MyStore  CalculateTotalPrice MyStore  FlowerStoreInfo MyStore  RenameFlowerStore MyStore MyNewStore  FlowerStoreInfo MyNewStore  AddFlower rose black 12.30 MyNewStore  AddFlower gladiolus pink 3.50 MyNewStore  GetFlowerWithHighestPrice MyNewStore  AddFlower tulip orange 2.20 MyNewStore  GetFlowerWithLowestPrice MyNewStore  CreateFlowerStore NewestStore  FlowerStoreInfo NewestStore  STOP | You created flower store MyStore.  Flower store MyStore has no available flowers.  Invalid command!  You added flower tulip with color red to store MyStore.  You added flower rose with color yellow to store MyStore.  Flower store MyStore has 2 flower/s:  Flower tulip with color red costs 3.40  Flower rose with color yellow costs 8.70  You sold flower tulip with color red from flower store MyStore.  Flower store MyStore has 1 flower/s:  Flower rose with color yellow costs 8.70  You added flower dahlia with color white to store MyStore.  You added flower narcissus with color yellow to store MyStore.  Flower store MyStore has 3 flower/s:  Flower rose with color yellow costs 8.70  Flower dahlia with color white costs 2.40  Flower narcissus with color yellow costs 3.50  Total price: 14.60  You sold all flowers from store MyStore.  Total price: 0.00  Flower store MyStore has no available flowers.  You renamed your store from MyStore to MyNewStore.  Flower store MyNewStore has no available flowers.  You added flower rose with color black to store MyNewStore.  You added flower gladiolus with color pink to store MyNewStore.  Flower from store MyNewStore has highest price: 12.30  You added flower tulip with color orange to store MyNewStore.  Flower from store MyNewStore has lowest price: 2.20  You created flower store NewestStore.  Flower store NewestStore has no available flowers. |
| **Вход** | **Изход** |
| CreateFlowerStore a  CreateFlowerStore ab  CreateFlowerStore abc  CreateFlowerStore abcd  CreateFlowerStore stored  AddFlower dahlia white 2.40 stored  AddFlower narcissus yellow 3.50 stored  RenameFlowerStore stored s  RenameFlowerStore stored st  RenameFlowerStore stored sto  RenameFlowerStore stored stor  RenameFlowerStore stored stored1  STOP | Invalid flower store name!  Invalid flower store name!  Invalid flower store name!  Invalid flower store name!  You created flower store stored.  You added flower dahlia with color white to store stored.  You added flower narcissus with color yellow to store stored.  Invalid flower store name!  Invalid flower store name!  Invalid flower store name!  Invalid flower store name!  You renamed your store from stored to stored1. |

## Точки

Разбивката по подзадачи е следната:

1. Налична структура за Flower и FlowerStore, ToString методи и реализирана AddFlower команда - **30** точки
2. Реализация на командите от раздел Логика - **40** точки
3. Реализация на командите от раздел Бонус Логика - **20** точки
4. Валидация на данните - **10** точки

Общ брой точки: **100**

## Requested files

#### Flower.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RegularExam

{

internal class Flower

{

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

{

//TODO: Добавете вашия код тук …

}

public string Type

{

}

public string Color

{

}

public double Price

{

}

public override string ToString()

{

throw new NotImplementedException(); //премахнете този ред като добавите вашия код

//TODO: Добавете вашия код тук …

}

}

}

#### FlowerStore.cs

#### using System;

#### using System.Collections.Generic;

#### using System.Linq;

#### using System.Text;

#### using System.Threading.Tasks;

#### namespace RegularExam

#### {

#### internal class FlowerStore

#### {

#### public FlowerStore(string name)

#### {

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public string Name

#### {

#### }

#### 

#### public void AddFlower(Flower flower)

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public bool SellFlower(Flower flower)

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public double CalculateTotalPrice()

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public Flower GetFlowerWithHighestPrice()

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public Flower GetFlowerWithLowestPrice()

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public void RenameFlowerStore(string newName)

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public void SellAllFlowers()

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### 

#### public override string ToString()

#### {

#### throw new NotImplementedException(); //премахнете този ред като добавите вашия код

#### //TODO: Добавете вашия код тук …

#### }

#### }

}

#### Program.cs

#### using System;

#### using System.Collections.Generic;

#### namespace RegularExam

#### {

#### internal class Program

#### {

#### static Dictionary<int, Flower> flowers = new Dictionary<int, Flower>();

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

#### static void Main(string[] args)

#### {

#### string input;

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

#### {

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

#### string command = splittedInput[0];

#### switch (command)

#### {

#### case "AddFlower":

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

#### break;

#### case "SellFlower":

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

#### break;

#### case "CalculateTotalPrice":

#### CalculateTotalPrice(splittedInput[1]);

#### break;

#### case "GetFlowerWithHighestPrice":

#### GetFlowerWithHighestPrice(splittedInput[1]);

#### break;

#### case "GetFlowerWithLowestPrice":

#### GetFlowerWithLowestPrice(splittedInput[1]);

#### break;

#### case "RenameFlowerStore":

#### RenameFlowerStore(splittedInput[1], splittedInput[2]);

#### break;

#### case "SellAllFlowers":

#### SellAllFlowers(splittedInput[1]);

#### break;

#### case "FlowerStoreInfo":

#### FlowerStoreInfo(splittedInput[1]);

#### break;

#### case "CreateFlowerStore":

#### CreateFlowerStore(splittedInput[1]);

#### break;

#### default:

#### Console.WriteLine("Invalid command!");

#### break;

#### }

#### }

#### }

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

#### {

#### try

#### {

#### Flower flower = new Flower(type, color, price);

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### FlowerStore store = stores[name];

#### store.AddFlower(flower);

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

#### }

#### catch (ArgumentException ex)

#### {

#### Console.WriteLine(ex.Message);

#### }

#### }

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

#### {

#### try

#### {

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### Flower flower = new Flower(type, color, price);

#### FlowerStore store = stores[name];

#### if (store.SellFlower(flower))

#### {

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

#### }

#### else

#### {

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

#### }

#### FlowerStore store = stores[name];

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

#### }

#### catch (ArgumentException ex)

#### {

#### Console.WriteLine(ex.Message);

#### }

#### }

#### private static void RenameFlowerStore(string name, string newName)

#### {

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### FlowerStore store = stores[name];

#### try

#### {

#### store.RenameFlowerStore(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 SellAllFlowers(string name)

#### {

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### FlowerStore store = stores[name];

#### store.SellAllFlowers();

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

#### }

#### private static void FlowerStoreInfo(string name)

#### {

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### FlowerStore store = stores[name];

#### Console.WriteLine(store.ToString());

#### }

#### private static void GetFlowerWithLowestPrice(string name)

#### {

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### FlowerStore store = stores[name];

#### Console.WriteLine($"Flower from store {name} has lowest price: {store.GetFlowerWithLowestPrice().Price:F2}");

#### }

#### private static void GetFlowerWithHighestPrice(string name)

#### {

#### if (!stores.ContainsKey(name))

#### {

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

#### return;

#### }

#### FlowerStore store = stores[name];

#### Console.WriteLine($"Flower from store {name} has highest price: {store.GetFlowerWithHighestPrice().Price:F2}");

#### }

#### private static void CreateFlowerStore(string name)

#### {

#### try

#### {

#### FlowerStore store = new FlowerStore(name);

#### stores.Add(name, store);

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

#### }

#### catch (ArgumentException ex)

#### {

#### Console.WriteLine(ex.Message);

#### }

#### }

#### }

}