

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

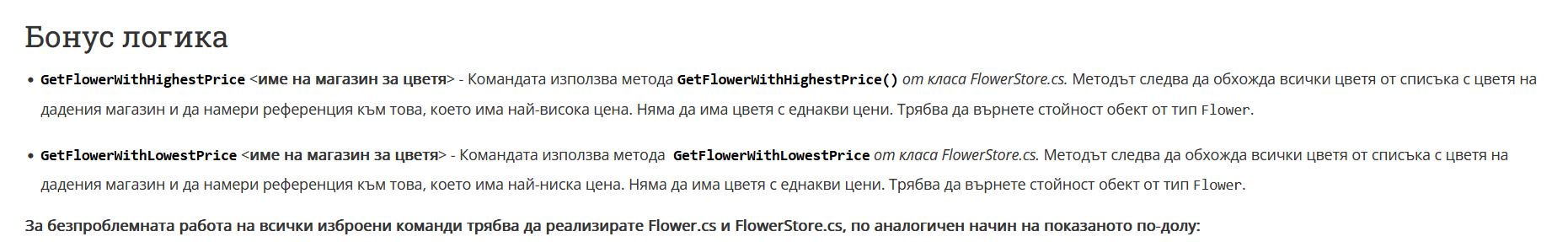
Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа маса

Описанието е генерирано автоматично



Картина, която съдържа текст

Описанието е генерирано автоматично



Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа маса

Описанието е генерирано автоматично

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);

}

}

}

}