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);

}

}

}

}

using System;

namespace RegularExam

{

internal class Flower

{

private string type;

private string color;

private double price;

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

{

Type = type;

Color = color;

Price = price;

}

public string Type

{

get { return type; }

private set { type = value; }

}

public string Color

{

get { return color; }

private set { color = value; }

}

public double Price

{

get { return price; }

private set {

if (value > 100)

throw new ArgumentException("Invalid flower price!");

price = value; }

}

public override string ToString()

{

return $"Flower {type} with color {color} costs {price:f2}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

namespace RegularExam

{

internal class FlowerStore

{

private string name;

private List<Flower> flowers;

public FlowerStore(string name)

{

Name = name;

flowers = new List<Flower>();

}

public string Name

{

get { return name; }

private set

{

if (value.Length < 6)

throw new ArgumentException("Invalid flower store name!");

name = value;

}

}

public void AddFlower(Flower flower)

{

flowers.Add(flower);

}

public bool SellFlower(Flower flower)

{

Flower flowerToRemove = flowers.FirstOrDefault(x => x.Type == flower.Type);

return flowers.Remove(flowerToRemove);

}

public double CalculateTotalPrice()

{

return flowers.Sum(flower => flower.Price);

}

public Flower GetFlowerWithHighestPrice()

{

return flowers.OrderByDescending(flower => flower.Price).First();

}

public Flower GetFlowerWithLowestPrice()

{

return flowers.OrderBy(flower => flower.Price).First();

}

public void RenameFlowerStore(string newName)

{

Name = newName;

}

public void SellAllFlowers()

{

flowers.Clear();

}

public override string ToString()

{

if (flowers.Count == 0)

return $"Flower store {name} has no available flowers.";

string end = $"Flower store {name} has {flowers.Count} flower/s:\n";

end += String.Join("\n", flowers);

return end;

}

}

}