Министерство образования и науки Российской Федерации

федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | | | | | | | | | | |  |
|  | Информационных систем и технологий | | | | | | | | | | | | |  |
|  | (наименование кафедры) | | | | | | | | | | | | |  |
|  |  | | | | | | | | | | | | |  |
|  | Данилов Алексей Александрович | | | | | | | | | | | | |  |
|  | (фамилия, имя, отчество студента) | | | | | | | | | | | | |  |
|  |  | | | | | | | | | | | | |  |
|  | Институт | ИМИКТ | | | курс | 2 | | | группа | 28 | | |  |  |
|  |  | | | | | | | | | | | | |  |
|  | |  | | --- | | 230400.62 Информационные системы и технологии | | (код и наименование направления подготовки/специальности) | | | | | | | | | | | | | |  |
|  | **ОТЧЁТ** | | | | | | | | | | | | |  |
|  | о лабораторном практикуме | | | | | | | | | | | | |  |
|  |  | | | | | | | | | | | | |  |
|  | По дисциплине | |  | Технология программирования | | | | | | | | | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  | Отметка о зачёте | |  |  | | | | | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  | (дата) | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  | Руководитель практикума | | | | | |  |  | | |  |  | |  |
|  |  | |  |  | | | | | | |  |  | |  |
|  | Доцент | |  |  | | | | | | |  | И. С. Майоров | |  |
|  | (должность) | |  | (подпись) | | | | | | |  | (инициалы, фамилия) | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | |  |  | | |  |  | | |  |  | |  |
|  |  | | | | | | | | | | | | |  |
|  | Архангельск  2014 | | | | | | | | | | | | |  |

ЛИСТ ДЛЯ ЗАМЕЧАНИЙ

1. Задание

Создать структуру, описывающую заказ в пиццерии (ФИО, адрес, телефон и т.д.). Создать класс для работы с данными о заказах, который позволяет: хранить информацию о заказах, добавлять/удалять/изменять информацию о заказе.

дополнительно:

1. поиск и вывод на экран всех заказов сделанных с определенного адреса;
2. поиск человека сделавшего максимальное/минимальное количество заказов, и вывод информации о нем.

Модифицировать созданное в Л.р. №7 приложение, таким образом, чтобы работа с созданным списком (List) производилась с помощью индексатора. Организовать доступ к полям класса через свойства. Создать библиотеку dll, в которой будет находиться созданные класс и структура.

1. Планируемая реализация

Планировалось создать класс Pizzeria для хранения информации о списке заказов пиццерии и методов добавления заказов, удаления заказов, изменения информации о заказе и прочих методов.

Также планировалось создать формально класс Order (он создаётся в среде разработки как класс), фактически − это структура public struct Order. В этом классе будет храниться информация о заказе: имя заказчика, адрес и телефон. А также метод изменения информации о заказчике и метод ToString() для вывода информации о заказе.

Хранить информацию обо всех заказах пиццерии планировалось в поле класса Pizzeria List<Order>.

1. Итоговая реализация

В процессе реализации программы для системы хранения заказов пиццерии была создана структура Order для хранения информации о заказе. В этой структуре есть поля имя заказчика, адрес и телефон. В структуре содержатся: конструктор объекта экземпляра структуры Order(), метод для изменения значений полей экземпляра структуры ChangeOrder() и метод вывода информации об объекте ToString().

Фрагмент программного кода

public struct Order

{

public string Name;

public string Address;

public string PhoneNumber;

public Order(string name, string address, string phoneNumber)

{

this.Name = name;

this.Address = address;

this.PhoneNumber = phoneNumber;

}

public override string ToString()

{

return "Имя заказчика: " + Name + ", адресс: " + Address + ", номер телефона: " + PhoneNumber + ".";

}

public void ChangeOrder(string name, string address, string phoneNumber)

{

this.Name = name;

this.Address = address;

this.PhoneNumber = phoneNumber;

}

}

Класс Pizzeria содержит поле типа List<string> ListOfOrders для хранения всех заказов пиццерии. Конструктор класса инициализирует поле как пустой список. Класс содержит методы Add() и Remove() для добавления и удаления заказов из списка соответственно. Эти методы получают на вход имя заказчика, адрес и телефон в соответствии с ними методы производят свои действия. При удалении учитывается ситуация, если такого заказа не оказалось.

Фрагмент программного кода

public List<Order> ListofOrders;

public Pizzeria()

{

ListofOrders = new List<Order>();

}

public void AddOrder(string name, string address, string phoneNumber)

{

ListofOrders.Add(new Order(name, address, phoneNumber));

}

public void RemoveOrder(string name, string address, string phoneNumber)

{

if (ListofOrders.Contains(ListofOrders.Find(target => target.Name.Contains(name) && target.Address.Contains(address) && target.PhoneNumber.Contains(phoneNumber)))){

ListofOrders.RemoveAll(target => (target.Name.Contains(name) && target.Address.Contains(address) && target.PhoneNumber.Contains(phoneNumber)));

}

else{

Console.WriteLine("Нет такого заказа");

}

}

Также класс Pizzeria содержит методы ChangeOrderInfo(), FindOrder() и PrintOrdersFromAddress(). Метод ChangeOrderInfo() позволяет менять информацию о заказе путём удаления старой информации о заказе и заменой её новым заказом, метод FindOrder() используется этим методом для поиска заказа. Метод PrintOrdersFromAddress() позволяет напечатать все заказы, сделанные с одного адреса.

Фрагмент программного кода

public void ChangeOrderInfo(string name, string address, string phoneNumber, string newName, string newAddress, string newPhoneNumber)

{

var order = FindOrder(name, address, phoneNumber);

ListofOrders.Remove(order);

order.ChangeOrder(newName, newAddress, newPhoneNumber);

ListofOrders.Add(order);

}

private Order FindOrder(string name, string address, string phoneNumber)

{

return ListofOrders.Find(target => target.Name.Contains(name) && target.Address.Contains(address) && target.PhoneNumber.Contains(phoneNumber));

}

public void PrintOrdersFromAddress(string address)

{

List<Order> FromOneAddress = ListofOrders.FindAll(target => target.Address.Contains(address));

foreach (Order order in FromOneAddress){

Console.WriteLine(order.ToString());

}

}

Последний функционал класса Pizzeria − это вывод заказчиков с наибольшим и наименьшим количеством заказов. Это реализуют методы PrintPersonsWithMostOrdersInfo() и PrintPersonsWithLessOrdersInfo(). Оба метода выводят информацию с помощью одного метода PrintPresonsOrders(), метод ExtractNames() используется для организации работы методов вывода информации о клиентах с наибольшим и наименьшим количеством заказов. Метод OutputOrders() выводит информацию о добавлении, отмене и изменении информации о заказах.

Фрагмент программного кода

public void PrintPersonsWithMostOrdersInfo()

{

List<string> Names = ExtractNames();

uint MaxOrdersFromPerson = 0;

uint OrdersFromPerson = 0;

foreach (string name in Names){

OrdersFromPerson = (uint)Names.Count<string>(target => target == name);

if (OrdersFromPerson > MaxOrdersFromPerson){

MaxOrdersFromPerson = OrdersFromPerson;

}

else{

OrdersFromPerson = 0;

}

}

PrintPresonsOrders(Names, MaxOrdersFromPerson);

}

public void PrintPersonsWithLessOrdersInfo()

{

List<string> Names = ExtractNames();

uint LessOrdersFromPerson = uint.MaxValue;

uint OrdersFromPerson = 0;

foreach (string name in Names){

OrdersFromPerson = (uint)Names.Count<string>(target => target == name);

if (OrdersFromPerson < LessOrdersFromPerson){

LessOrdersFromPerson = OrdersFromPerson;

}

else{

OrdersFromPerson = 0;

}

}

PrintPresonsOrders(Names, LessOrdersFromPerson);

}

private void PrintPresonsOrders(List<string> Names, uint OrdersFromPerson)

{

string PersonWithNotisableOrders;

List<string> TempNames = Names;

for (int i = 0; i < TempNames.Count; i++){

if (OrdersFromPerson == (uint)Names.Count<string>(target => target == TempNames[i])){

PersonWithNotisableOrders = TempNames[i];

List<Order> FromPerson = ListofOrders.FindAll(target => target.Name.Contains(PersonWithNotisableOrders));

foreach (Order order in FromPerson){

Console.WriteLine(order.ToString());

}

TempNames.RemoveAll(target => target.Contains(PersonWithNotisableOrders));

}

}

}

private List<string> ExtractNames()

{

List<string> Names = new List<string>();

foreach (Order order in ListofOrders){

Names.Add(order.Name);

}

return Names;

}

public void OutputOrders(List<Order> ListofOrders, string msg)

{

Console.WriteLine(msg);

foreach (Order order in ListofOrders){

Console.WriteLine(order.ToString());

}

}

В классе Program в методе Main() изложен общий ход работы программы. Сначала создаётся экземпляр класса Pizzeria Pizza и в этой пиццерии делаются заказы с помощью метода FirstAdding(), затем несколько заказов отменяется с помощью метода Removing() все изменения, происходящие со списком заказов, выводятся на консоль. Далее меняется информация об одном из заказов (ChangeOrderInfo()), выводятся все заказы с одного адреса (PrintOrdersFromAddress()) и делаются новые заказы (SecondAdding()). В конце на консоль выводятся заказы клиентов, сделавших больше всего и меньшего всего заказов (PrintPersonsWithMostOrdersInfo() и PrintPersonsWithLessOrdersInfo() соответственно).

Фрагмент программного кода

static void Main()

{

Pizzeria Pizza = new Pizzeria();

FirstAdding(Pizza);

Pizza.OutputOrders(Pizza.ListofOrders, "Сделаны заказы:");

Removing(Pizza);

Pizza.OutputOrders(Pizza.ListofOrders, "Заказы отменены:");

Pizza.ChangeOrderInfo("Крюков", "Архангельск", "345678", "Крюков", "Ломоносовский район", "236578");

Pizza.OutputOrders(Pizza.ListofOrders, "Заказы изменены:");

Console.WriteLine("Заказы с одного адреса:");

Pizza.PrintOrdersFromAddress("Архангельск");

SecondAdding(Pizza);

Console.WriteLine("Больше всего заказов от:");

Pizza.PrintPersonsWithMostOrdersInfo();

Console.WriteLine("Меньше всего заказов от:");

Pizza.PrintPersonsWithLessOrdersInfo();

Console.ReadKey(true);

}

private static void FirstAdding(Pizzeria Pizza)

{

Pizza.AddOrder("Крюков", "Архангельск", "345678");

Pizza.AddOrder("Ильченко", "Северодвинск", "654587");

Pizza.AddOrder("Данилов", "Архангельск", "158356");

Pizza.AddOrder("Игорь", "Архангельск", "278905");

}

private static void Removing(Pizzeria Pizza)

{

Pizza.RemoveOrder("Ильченко", "Северодвинск", "654587");

Pizza.RemoveOrder("Юшманов", "Архангельск", "426414");

}

private static void SecondAdding(Pizzeria Pizza)

{

Pizza.AddOrder("Крюков", "10 корпус", "453254");

Pizza.AddOrder("Крюков", "Привокзальный", "564532");

Pizza.AddOrder("Кирьянов", "Соломбала", "342456");

Pizza.AddOrder("Игорь", "Соломбала", "435678");

}

После написания программы системы заказов для пиццерии, эта программа была модифицирована в соответствии с заданием.

Доступ к полям был организован через свойства класса Pizzeria, работа со списком заказов − через индексаторы. Также была создана и подключена библиотека Pizza.dll.

Фрагмент программного кода

private List<Order> listofOrders;

public List<Order> ListOfOrders

{

get

{

return listofOrders;

}

set

{

listofOrders = value;

}

}

public Order this[int index]

{

get

{

return listofOrders[index];

}

set

{

listofOrders[index] = value;

}

}

foreach (Order order in Pizza.ListOfOrders){

Console.WriteLine(order.ToString());

}

for (int i = 0; i < Pizza.ListOfOrders.Count; i++){

Console.WriteLine(Pizza[i].ToString());

}

ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы была создана структура, описывающую заказ в пиццерии. Создан класс для работы с данными о заказах, позволяющий хранить информацию о заказах, добавлять/удалять/изменять информацию о заказе.

Также был реализован поиск и вывод на экран всех заказов сделанных с определенного адреса, и поиск человека сделавшего максимальное/минимальное количество заказов, и вывод информации о нем.

После работа была модифицирована, таким образом, чтобы работа с созданным списком (List) производилась с помощью индексатора, был организован доступ к полям класса через свойства и создана библиотека Pizza.dll, в которой находиться созданные класс и структура.