**9 Интерфейсы**

Задание 1. Интерфейсы Ix, Iy, Iz, содержат объявления методов с одной и той же сигнатурой.

Интерфейсы наследуются в классе TestClass, содержащий член w типа int и реализуются как 10\*w. В каждом методе задать вывод результата.

Рассмотреть случай:

неявной реализации интерфейсов(w–10);

явной реализации интерфейса Iz(w/10).

В программе должна выполняться:

неявная неоднозначная реализация методов интерфейсов Iy и Iz;

вызов функций с явным приведением к типу интерфейса;

вызов метода для объекта посредством интерфейсной ссылки.

Листинг программы:

namespace MySpase

{

interface Ix

{

void IxF0(int inter);

void IxF1();

}

interface Iy

{

void F0(int inter);

void F1();

}

interface Iz

{

void F0(int inter);

void F1();

}

class TestClass : Ix, Iy, Iz

{

private int w;

public TestClass(int w)

{

this.w = w;

}

// Неявная реализация интерфейсов Ix, Iy и Iz

public void IxF0(int inter)

{

Console.WriteLine("IxF0: " + (10 \* w - 10));

}

public void IxF1()

{

Console.WriteLine("IxF1: " + (10 \* w - 10));

}

void Iy.F0(int inter)

{

Console.WriteLine("Iy.F0: " + (10 \* w - 10));

}

void Iy.F1()

{

Console.WriteLine("Iy.F1: " + (10 \* w - 10));

}

// Явная реализация интерфейса Iz

void Iz.F0(int inter)

{

Console.WriteLine("Iz.F0: " + (w / 10));

}

void Iz.F1()

{

Console.WriteLine("Iz.F1: " + (w / 10));

}

}

class Program

{

static void Main(string[] args)

{

TestClass test = new TestClass(15);

// Вызов методов через интерфейсную ссылку

Ix ix = test;

ix.IxF0(15);

ix.IxF1();

Iy iy = test;

iy.F0(15);

iy.F1();

Iz iz = test;

iz.F0(15);

iz.F1();

// Вызов методов с явным приведением к типу интерфейса

((Iy)test).F0(15);

((Iy)test).F1();

((Iz)test).F0(15);

((Iz)test).F1();

Console.ReadLine();

}

}

}

Таблица 9.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 15 | 140, 1 |

Анализ результатов:

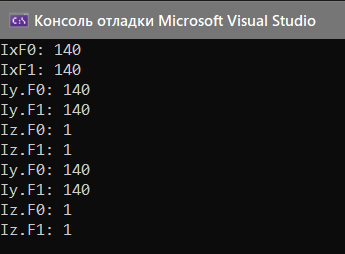


Рисунок 9.1 – Результат работы программы

Задание 2. Описать класс «записная книжка». Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по фамилии и доступа к записи по номеру. Написать программу, демонстрирующую все разработанные элементы класса.

Листинг программы:

namespace MySpase

{

class Contact

{

public string LastName { get; set; }

public string FirstName { get; set; }

public string MiddleName { get; set; }

public DateTime BirthDate { get; set; }

public string PhoneNumber { get; set; }

public Contact(string lastName, string firstName, string middleName, DateTime birthDate, string phoneNumber)

{

LastName = lastName;

FirstName = firstName;

MiddleName = middleName;

BirthDate = birthDate;

PhoneNumber = phoneNumber;

}

public override string ToString()

{

return $"{LastName} {FirstName} {MiddleName} - {BirthDate.ToShortDateString()} - {PhoneNumber}";

}

}

class Notebook

{

private List<Contact> contacts = new List<Contact>();

public void AddContact(Contact contact)

{

contacts.Add(contact);

}

public void RemoveContact(Contact contact)

{

contacts.Remove(contact);

}

public void ShowContacts()

{

foreach (var contact in contacts)

{

Console.WriteLine(contact);

}

}

public void PrintContacts()

{

foreach (Contact contact in contacts)

{

Console.WriteLine(contact);

}

}

public List<Contact> SearchContacts(string query)

{

List<Contact> result = new List<Contact>();

foreach (Contact contact in contacts)

{

if (contact.LastName.Contains(query) || contact.FirstName.Contains(query) || contact.MiddleName.Contains(query) || contact.PhoneNumber.Contains(query))

{

result.Add(contact);

}

}

return result;

}

public int GetCount()

{

return contacts.Count;

}

public int FindIndex(string lastName, string firstName, string middleName, DateTime birthDate, string phoneNumber)

{

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

{

var contact = contacts[i];

if (contact.LastName == lastName &&

contact.FirstName == firstName &&

contact.MiddleName == middleName &&

contact.BirthDate == birthDate &&

contact.PhoneNumber == phoneNumber)

{

return i;

}

}

return -1;

}

}

class Program

{

static void Main(string[] args)

{

Notebook notebook = new Notebook();

int n = 0;

while (n < 9)

{

Console.WriteLine("Введите данные контакта: ");

Console.Write("Фамилия: ");

string lastName = Console.ReadLine();

Console.Write("Имя: ");

string firstName = Console.ReadLine();

Console.Write("Отчество: ");

string middleName = Console.ReadLine();

Console.Write("Дата рождения (в формате ДД.ММ.ГГГГ): ");

DateTime birthDate = DateTime.ParseExact(Console.ReadLine(), "dd.MM.yyyy", null);

Console.Write("Номер телефона: ");

string phoneNumber = Console.ReadLine();

Contact contact = new Contact(lastName, firstName, middleName, birthDate, phoneNumber);

notebook.AddContact(contact);

Console.WriteLine();

n++;

}

Console.Write("Введите запрос: ");

string query = Console.ReadLine();

List<Contact> searchResult = notebook.SearchContacts(query);

if (searchResult.Count == 0)

{

Console.WriteLine("Контакты не найдены.");

}

else

{

Console.WriteLine($"Результаты поиска ({searchResult.Count}):");

foreach (var contact in searchResult)

{

Console.WriteLine(contact);

}

}

Console.WriteLine();

Console.Write("Введите запрос для удаления: ");

string queryDel = Console.ReadLine();

List<Contact> searchResultDel = notebook.SearchContacts(query);

if (searchResultDel.Count == 0)

{

Console.WriteLine("Контакты не найдены.");

}

else

{

foreach (var contact in searchResultDel)

{

notebook.RemoveContact(contact);

Console.Write(contact);

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(" -- УДАЛЕНО");

Console.ResetColor();

}

}

Console.WriteLine();

Console.WriteLine("Все контакты:");

notebook.ShowContacts();

}

}

}

Анализ результатов:

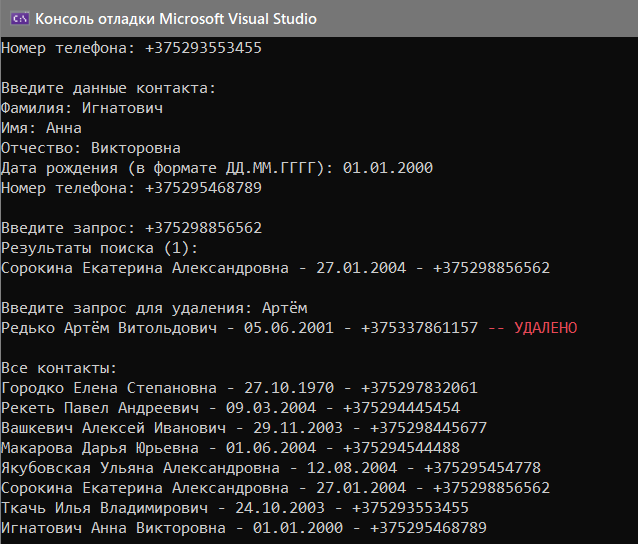


Рисунок 9.2 – Результат работы программы