Федеральное государственное бюджетное образовательное учреждение высшего образования

Пояснительная записка к междисциплинарной курсовой работе

на тему:

Выполнил:
студент группы

_____
(Подпись)

Телефон:

Научный руководитель:

_____
(Подпись)

2020

Оглавление

(оглавление меняй под себя)

Введение

Целью курсовой работы является приобретение навыков в анализе предметной области и в разработке сложного Windows-приложения на объектно-ориентированном языке программирования C# в среде Visual Studio. Разработка приложения позволит улучшить мои знания этого языка, а также методов визуального проектирования и анализа предметной области.

Анализ предметной области позволяет разработать информационную модель задачи. Важным этапом этого анализа является выбор из предметной области наиболее важных объектов и процессов автоматизации, а также разработка новых IT-подходов и функций для решения проблем предметной области.

Второй целью курсовой работы, и не менее важной, является разработка пояснительной записки, отвечающей таким требованиям, как полнота, лаконичность, техническая корректность и т.д. Техническая документация является составной частью программного продукта и, следовательно, определяет его качество. Поэтому приобретение навыков в разработке технической документации, а в данном случае пояснительной записки, также является немаловажной задачей.

1. Постановка задачи

Требования:

1. Разрабатываемое приложение должно представлять собой Windows приложение.
2. Должны быть разработаны пользовательские классы (один или несколько), описывающие объекты предметной области, и несколько форм пользовательского интерфейса.

3. Разработчик самостоятельно определяет интерфейс программы и ее функциональность, однако для получения максимальной оценки приложение в
обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

 Отображать в сетках DataGridView данные предметной области:

 Для информационной модели, основанной на списках класса List<T>, на момент первого запуска программы допускается отсутствие файла. В этом случае списки объектов должны быть созданы в программе на этапе разработки.

 Для информационной модели, основанной на БД (Access, SQL Server), таблицы
должны быть предварительно заполнены записями.

 Реализовать добавление в источник данных нового объекта, удаление объекта из
источника данных, редактирование объекта источника данных.
 Реализовать фильтрацию записей источника данных, удовлетворяющих введенному
пользователем сложному критерию.
 Реализовать сортировку записей источника данных, включая многоуровневую.
 Сохранять источник данных:
 для информационной модели, основанной на списках, в файле, используя XML-сериализацию (или Json).
 для информационной модели, основанной на таблицах БД, в базе данных.
 При запуске программы загрузить сохраненные данные из файла или базы данных.
 Используя меню или панель инструментов, вызвать приложение Блокнот для
просмотра справки о программе: файл Help.txt текущего каталога программы.
 Создать пункт меню «Об авторе» с выводом соответствующей информации
(MessageBox).
 Разработать несколько полезных пользователю функций для отображения статистических данных, например, средних, максимальных или минимальных
значений, данных для построения гистограммы или графика и т.п.

3. Список должен быть реализован в виде коллекции, например, динамического массива типа List<T> или BindingList<T>. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.
4. Программа должна быть читабельной и содержать полезные комментарии.

2. Описание предметной области

3. Актуальность автоматизации

Также программа предназначена для выполнения ряда сервисных функций по работе с Базой Данных, реализованной с помощью MDB.

Программа позволяет вводить, изменять, сохранять и удалять товары и пользователей из файла, а так же полностью откатывать все данные в программе.

4. Описание программы

4.1. Алгоритмические решения

Хранение списка различных вариантов кредитов, клиентов и информации о них организовано в базе данных MDB. Файл отображается на список типа List<Class>. Пользователь может динамически изменять файл со списком выданных кредитов, сортировать его и дополнять, а так же выводить информацию из DataGridView в Excel для дальнейшей аналитики. Программа позволяет сохранять любые действия в базе данных и при новой загрузке данные не теряются и подгружаются.

Чтобы пользователь не заполнял все изменения вручную предусмотрен авто ввод данных по нажатию на ячейку таблицы. В случае внесения неверны данных, программа сообщает об этом пользователю.

Каждый клиент банка имеет свои характеристики (поля), по которым можно сортировать или проводить аналитику, а так же советовать клиенту то, что ему необходимо.

Для сортировки клиентов банка отображается ряд полей, создающих условие для сортировки. Отобраны будут все клиенты, удовлетворяющие условиям сортировки (одноуровневой).

Для работы с таблицами кредитов и клиентов используется интерфейс элемента DataGridView. Базовая функциональность DataGridView поддерживается специальными обработчиками событий таблицы, в которых осуществляется проверка полей таблицы на допустимость значений.

В программе реализована сортировка клиентов по любому атрибуту. Для сортировки необходимо перейти в меню сверху и кликнуть на необходимое условие. Сортировка позволяет расположить строки с искомым признаком в нужном вам порядке. То есть сортировка в какой-то степени может заменить отбор товаров по одному заданному критерию. Многокритериальный отбор сортировка заменить не может.

Программа является задокументированной, пункт меню Помощь выдает полное описание программы.

Для сохранения новых клиентов банка используется MDB. Разработан класс и множество методов, избавляющие разработчика от знания деталей ввода-вывода, а так же предохраняющих от введения некорректных данных.

Использование этого класса повышает надежность программы и производительность труда рядового пользователя.

В программе реализована оптимизация вывода данных на диск. Если за время сеанса не произошло изменения какой либо строки таблицы базы данных (редактирование, удаление, добавление), то при завершении программы, она не нуждается в сохранении в файле.

Для хранения объектов в оперативной памяти выбран список типа List<T>. В пользу такого решения выступает наличие метода сортировки Sort и OrderBy выполняющего упорядочивание объектов списка на месте. У списка BindingList<T> такого метода нет, а его метод расширения OrderBy создает новый список, что, на самом деле, не принципиально.

4.2. Описание интерфейса программы

В приложении для реализации интерфейса используется 10 элементов управления и компонентов:

- DataGridView;
- ComboBox
- Button;
- TextBox;
- Label;
- MessageBox;
- MenuStrip;
- OpenFileDialog;
- BindingSource
- PictureBox(для красоты)

Наиболее сложным элементом управления является таблица (сетка) DataGridView. Загрузка в таблицу записей клиентов осуществляется благодаря привязке таблицы к коллекции List<T> объектов типа Class. Привязка осуществляется с использованием посредника BindingSource.

Элемент управления DataGridView отображает список кредитов и клиентов банка, а так же подробной необходимой информации о них.

Запуская Windows приложение, мы попадаем в форму, на которой нам предложен список различных услуг банка (разных условий кредитов и лимиты по кредитам). Мы можем добавлять новые и менять старые записи в таблице, динамически меняя при этом и базу данных программы и сетку в отображаемом окне программы.

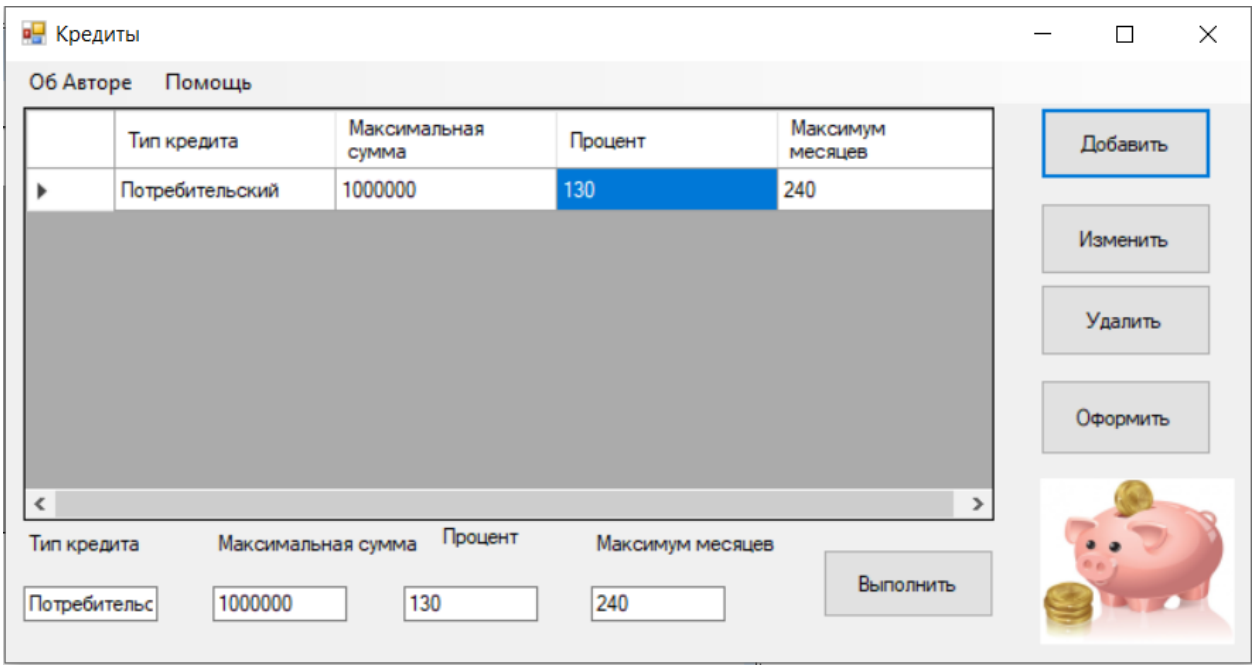

Рис. 4.1. *(придумай название сама)*

Если вы с вашим клиентом сошлись на условиях кредита, переходите по кнопке Оформить для оформления кредита. В данном окне вы можете как оформить кредит на клиента банка, так и посмотреть список всех клиентов нашего банка. Помимо этого вы можете выгрузить данные из таблицы в Excel для дальнейшей работы с ними.

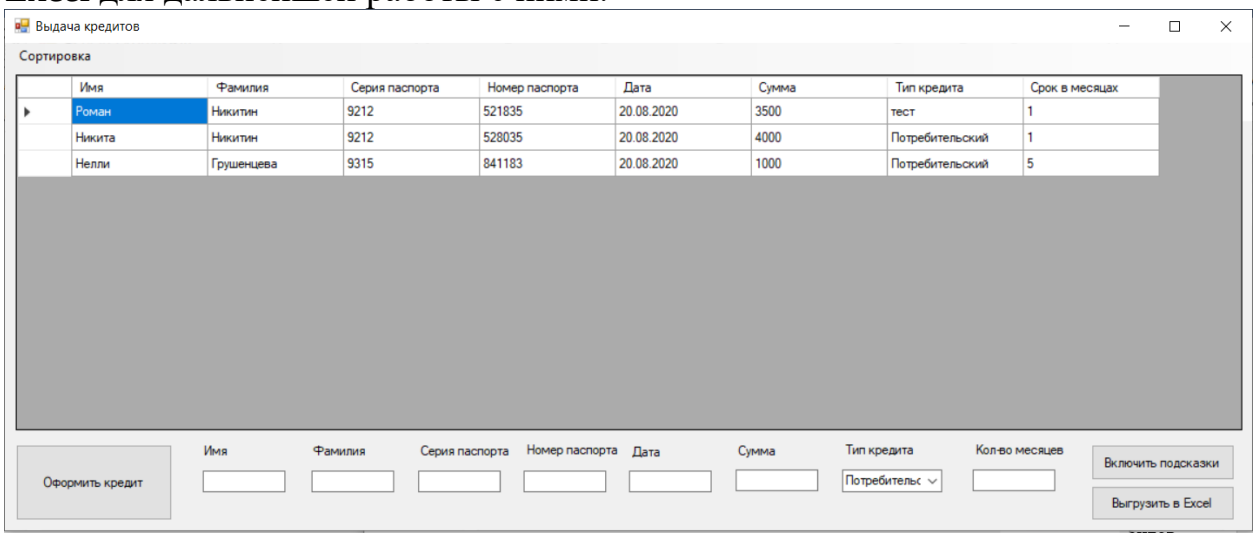

Рис. 4.2.*(Придумаешь сама)*.

Для того, чтобы получить инструкцию по использованию приложения, нужно на панели меню нажать на «Помощь»: будет запущено приложение Блокнот с описанием программы (см. рис. 4.3).

(сюда фото блокнота с помощью)

Рис. 4.3. *Окно Блокнота со справкой.*

(сюда фото блокнота с инфой о тебе)

Рис. 4.4. *Окно Блокнота с информацией о создателе.*

(сюда фотос графика, когда заполнишь бд)

Рис. 4.5 Окно с графиком.

4.3. Состав приложения

В состав приложения входят следующие файлы и папки:

- ....sln – программа;
- WindowsFormsApp4– папка с со всеми файлами программы;
  - ✓ КР-Пояснительная записка– пояснительная записка к программе;
- База Данных–база клиентов и товаров:
  - ✓ Bank.mdb – большая бд, содержащая в себе все основные записи программы.

5. Назначение и состав классов программы

5.2. Класс Printer (товара магазина)

- ✓ Name – свойство, позволяющее возвращать и задавать фирму-производитель принтера(товара);
- ✓ Model – свойство, позволяющее возвращать и задавать модель принтера(товара);
- ✓ Type – свойство, позволяющее возвращать и задавать тип принтера(товара);
- ✓ Price– свойство, позволяющее возвращать и задавать цену принтера(товара);
- ✓ Rate – свойство, позволяющее возвращать и задавать рейтинг принтера(товара);
- ✓ Kol_vo– свойство, позволяющее возвращать и задавать количество принтеров(товара) на складе;
- ✓ Printer(*параметры*) – параметрический конструктор, осуществляющий инициализацию свойств класса.
- ✓ (это ты сама поменяешь, не трудно)

5.3. Служебные классы

- **Program** – класс, содержащий метод Main, который позволяет запустить программу, открывая главную форму Form1.

5.4. Формы

Ниже представлены пользовательские классы, наследующие базовую функциональность от класса Form.

- **Form1** – на данной форме пользователь может посмотреть все существующие предложения кредитов.
- **Form2** – на данной форме мы можем посмотреть историю выданных кредитов, взаимодействовать с ними, а так же добавить новые
- **Form3** – График

Заключение

Разработав Windows приложение, был выполнен ряд поставленных задач. Справочно-информационная система написана на языке программирования C# с использованием классов, методов и свойств. Для реализации некоторых задач потребовалось обратиться к объектам операционной системы: системный реестр, файловая система, процессы.

Программа может модернизироваться и обновляться. К примеру, может быть расширен класс принтеров путем добавления новых объектов.

Программа может быть расширена статистическими и финансовыми функциями.

Список литературы

1. Горелов С.В., Волков А.Г. Разработка Windows-приложений. Часть 1. Учебное пособие. Образовательный портал Финансового университета. 2018.
2. Горелов С.В. Разработка Windows-приложений. Часть 2. Учебное пособие. Образовательный портал Финансового университета. 2018.
3. Г. Шилдт. Полный справочник по C#. - М.: «Вильямс», 2004.
4. Официальный сайт Microsoft: [Интернет-ресурс]. URL: https://msdn.microsoft.com org.
5. Сайт https://sergeygorelov.wixsite.com/projects
6. Отличный канал на youtube «Выполняем домашние задания и курсовые работы». Моего преподавателя Горелова С.В
7. Учебник в 2 томах: Современные технологии программирования. Разработка Windows-приложений на языке C# Автор- Горелов

КОД

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Runtime.CompilerServices;

namespace WindowsFormsApp4_Нелли_Грушенцева
{
/*Класс всех возможных кредитов, предоставляемых нашим банком.*/
   class Kredit
   {
      public string Name { get; set; }
      public int Max_money { get; set; }
      public int Percent { get; set; }
      public int Max_mounth { get; set; }
      //Список, содержащий все объекты данного класса
      public static List<Kredit> Kredits = new List<Kredit>();
      //Подключение к базе данных, из которой программа берет данные для
первоначальной загрузки.
      public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Bank.mdb;";
      public static OleDbConnection myConnection = new
OleDbConnection(connectString);
      public Kredit(string name, int max_money, int percent, int max_mounth)
      {
         Name = name;
         Max_money = max_money;
         Percent = percent;
         Max_mounth = max_mounth;
      }
      public Kredit() { }
      // Функция для загрузки всех данных из базы данных в программу (в
список).
      public static void Load_info()
      {
         // Воизбежание излишних трудностей и ошибок- полностью очищаю
список и заново его перезаписываю, так как затраты по времени
незначительные.
         Kredits.Clear();
         string query = "SELECT * FROM Kredits";
         myConnection.Open();
```

```
            OleDbCommand command = new OleDbCommand(query,
myConnection);
            OleDbDataReader take_info = command.ExecuteReader();
            while (take_info.Read())
            {
                Kredit kred = new Kredit(take_info[1].ToString(),
int.Parse(take_info[2].ToString()), int.Parse(take_info[3].ToString()),
int.Parse(take_info[4].ToString()));
                Kredits.Add(kred);
            }
            take_info.Close();
            myConnection.Close();
        }
        //Функция для добавления информации в базу данных. Все изменения в
базе данных автоматически сохраняются.
        //Вместо xml- сериализации использую access mdb.
        public static string Add_info(string name, int max_money, int percent, int
max_mounth)
        {
            int flag = 0;
            for (int i = 0; i < Kredits.Count; i++)
            {
                if (Kredits[i].Name == name)
                    flag += 1;
            }
            if (flag == 0 & max_money > 0 & percent > 0 & max_mounth > 0 &
max_money <= 99999999 & percent <= 30 & max_mounth <= 360)
            {
                Kredit kred = new Kredit(name, max_money, percent, max_mounth);
                Kredits.Add(kred);
                myConnection.Open();
                OleDbTransaction trans = myConnection.BeginTransaction();
                string sql1 = $"SELECT MAX(ID) FROM Kredits";
                OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection,
trans);
                string x = cmd1.ExecuteScalar().ToString();
                int y = int.Parse(x) + 1;
                string sql2 = $"INSERT INTO Kredits VALUES ({y}, '{name}',
'{max_money}', '{percent}', '{max_mounth}')";
                OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection,
trans);
                cmd2.ExecuteNonQuery();
                trans.Commit();
                myConnection.Close();
```

```
            return "успешно";
        }
        return "неуспешно";
    }
    //Функция удаления информации/строки/записи из базы данных.
Изменения так же автоматически сохраняются.
    public static string Delete_info(string name)
    {
        for (int i = 0; i < Kredits.Count; i++)
        {
            if (Kredits[i].Name == name)
            {
                Kredits.Remove(Kredits[i]);
                myConnection.Open();
                OleDbTransaction trans = myConnection.BeginTransaction();
                string sql2 = $"DELETE FROM Kredits WHERE name = '{name}'";
                OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection,
trans);
                cmd2.ExecuteNonQuery();
                trans.Commit();
                myConnection.Close();
                return "успешно";
            }
        }
        return "неуспешно";
    }
/*Функция для изменения информации об объектах базы данных.*/
    public static string Change_info(string name, int max_money, int percent, int
max_mounth)
    {
        for (int i = 0; i < Kredits.Count; i++)
        {
            if (Kredits[i].Name == name & max_money <= 99999999 & percent <=
30 & max_mounth <= 360 & max_money >= 0 & percent >= 0 & max_mounth >=
0)
            {
                myConnection.Open();
                OleDbTransaction trans = myConnection.BeginTransaction();
                string sql1 = $"UPDATE Kredits SET Max_money = {max_money},
Max_mounth = {max_mounth}, Percents = {percent} where name = '{name}'";
                OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection,
trans);
                cmd1.ExecuteScalar();
                trans.Commit();
```

```
                myConnection.Close();
                Load_info();
                return "успешно";
            }
        }
        return "неуспешно";
    }
}
/*Класс клиентов банка, хранящий в себе информацию о всех деталях оказанных услуг.*/
class Client
{
    public string F_name { get; set; }
    public string S_name { get; set; }
    public int Ser { get; set; }
    public int Num { get; set; }
    public string Data { get; set; }
    public int Money { get; set; }
    public string Type { get; set; }
    public int Months { get; set; }

    public static List<Client> Clients = new List<Client>();
    //Подключение к базе данных, из которой программа берет данные для первоначальной загрузки.
    public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Bank.mdb;";

    public static OleDbConnection myConnection = new OleDbConnection(connectString);

    public Client(string f_name, string s_name, int ser, int num, string data, int money, string type, int months)
    {
        F_name = f_name;
        S_name = s_name;
        Ser = ser;
        Num = num;
        Data = data;
        Money = money;
        Type = type;
        Months = months;
    }
    public Client() { }
    public static void Load_info()
```

```
        {
            // Воизбежание излишних трудностей и ошибок- полностью очищаю
список и заново его перезаписываю, так как затраты по времени
незначительные.
            Clients.Clear();
            string query = "SELECT * FROM Clients";
            myConnection.Open();
            OleDbCommand command = new OleDbCommand(query,
myConnection);
            OleDbDataReader take_info = command.ExecuteReader();
            while (take_info.Read())
            {
                Client cl = new Client(take_info[1].ToString(), take_info[2].ToString(),
int.Parse(take_info[3].ToString()), int.Parse(take_info[4].ToString()),
take_info[5].ToString(), int.Parse(take_info[6].ToString()),
take_info[7].ToString(), int.Parse(take_info[8].ToString()));
                Clients.Add(cl);
            }
            take_info.Close();
            myConnection.Close();
        }
        //Функция для добавления информации в базу данных. Все изменения в
базе данных автоматически сохраняются.
        //Вместо xml- сериализации использую access mdb.
        public static string Add_info(string f_name, string s_name, int ser, int num,
string data, int money, string type, int months)
        {
            int flag = 0;
            int flag_1 = 0;
            for (int i = 0; i < Clients.Count; i++)
            {
                if (Clients[i].S_name == s_name & Clients[i].F_name == f_name)
                    flag += 1;
            }
            for (int i = 0; i < Kredit.Kredits.Count(); i++)
            {
                if (Kredit.Kredits[i].Name == type)
                    flag_1 += 1;
            }

            if (flag == 0 & flag_1 != 0 & f_name != "" & s_name != "" & ser > 0 & ser
<= 9999 & num <= 999999 & money > 0 & money < 99999999 & type != "" &
months > 0 & months <= 360)
            {
```

```
            Client client = new Client(f_name, s_name, ser, num, data, money, type,
months);
            Clients.Add(client);
            myConnection.Open();
            OleDbTransaction trans = myConnection.BeginTransaction();
            string sql1 = $"SELECT MAX(ID) FROM Clients";
            OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection,
trans);
            string x = cmd1.ExecuteScalar().ToString();
            int y = int.Parse(x) + 1;
            string sql2 = $"INSERT INTO Clients VALUES ({y}, '{f_name}',
'{s_name}', '{ser}', '{num}', '{data}', '{money}', '{type}', '{months}')";
            OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection,
trans);
            cmd2.ExecuteNonQuery();
            trans.Commit();
            myConnection.Close();
            return "успешно";
         }
         return "неуспешно";
      }
      public static string Delete_info(string f_name, string s_name)
      {
         for (int i = 0; i < Clients.Count; i++)
         {
            if (Clients[i].F_name == f_name & Clients[i].S_name == s_name)
            {
               Clients.Remove(Clients[i]);
               myConnection.Open();
               OleDbTransaction trans = myConnection.BeginTransaction();
               string sql2 = $"DELETE FROM Clients WHERE F_name =
'{f_name}' and S_name = '{s_name}' ";
               OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection,
trans);
               cmd2.ExecuteNonQuery();
               trans.Commit();
               myConnection.Close();
               return "успешно";
            }
         }
         return "неуспешно";
      }
   }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp4_Нелли_Грушенцева
{
    public partial class Form1 : Form
    {
        public int flag;
        public Form1()
        {
            InitializeComponent();

        }
        private void button1_Click(object sender, EventArgs e)
        {
            flag = 1;
            button5.Show();
            textBox1.Show();
            textBox2.Show();
            textBox3.Show();
            textBox4.Show();
            label1.Show();
            label2.Show();
            label3.Show();
            label4.Show();
        }

        private void Form1_Load_1(object sender, EventArgs e)
        {
            button5.Hide();
            Kredit.Load_info();
            kreditBindingSource.DataSource = Kredit.Kredits;
        }

        private void button5_Click(object sender, EventArgs e)
        {
            try
            {
                if (flag == 1)
                {
                    string answ = Kredit.Add_info(textBox1.Text,
int.Parse(textBox2.Text), int.Parse(textBox3.Text), int.Parse(textBox4.Text));
                    kreditBindingSource.ResetBindings(false);
                    if (answ == "неуспешно")
                    {
                        MessageBox.Show("Неверно введены данные!");
                    }
                }
                else if (flag == 2)
                {
                    string answ = Kredit.Change_info(textBox1.Text,
int.Parse(textBox2.Text), int.Parse(textBox3.Text), int.Parse(textBox4.Text));
                    kreditBindingSource.ResetBindings(false);
                    if (answ == "неуспешно")
                    {
```

```
                    MessageBox.Show("Неверно введены данные!");
                }
            }
            else if (flag == 3)
            {
                string answ = Kredit.Delete_info(textBox1.Text);
                kreditBindingSource.ResetBindings(false);
                if (answ == "неуспешно")
                {
                    MessageBox.Show("Неверно введены данные!");
                }
            }
            button5.Hide();
            textBox1.Hide();
            textBox2.Hide();
            textBox3.Hide();
            textBox4.Hide();
            label1.Hide();
            label2.Hide();
            label3.Hide();
            label4.Hide();
            flag = 0;
        }
        catch
        {
            MessageBox.Show("Неверно введены данные!");
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        flag = 2;
        button5.Show();
        textBox1.Show();
        textBox2.Show();
        textBox3.Show();
        textBox4.Show();
        label1.Show();
        label2.Show();
        label3.Show();
        label4.Show();
    }

    private void dataGridView1_Click(object sender, EventArgs e)
    {
        textBox1.Text = dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString();
        textBox2.Text = dataGridView1[1,
dataGridView1.CurrentRow.Index].Value.ToString();
        textBox3.Text = dataGridView1[2,
dataGridView1.CurrentRow.Index].Value.ToString();
        textBox4.Text = dataGridView1[3,
dataGridView1.CurrentRow.Index].Value.ToString();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        flag = 3;
        button5.Show();
        textBox1.Show();
        textBox2.Show();
        textBox3.Show();
        textBox4.Show();
```

```
            label1.Show();
            label2.Show();
            label3.Show();
            label4.Show();
        }

        private void обАвтореToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start("Об_авторе.txt");
        }

        private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start("Помощь.txt");
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Form2 form2 = new Form2();
            form2.Show();
        }
    }
}
using Microsoft.Office.Interop.Excel;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp4_Нелли_Грушенцева
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            Kredit.Load_info();
            Client.Load_info();
            clientBindingSource.DataSource = Client.Clients;
            kreditBindingSource.DataSource = Kredit.Kredits;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (textBox5.Text.Split('.').Count() == 3)
                {
                    string answ = Client.Add_info(textBox1.Text, textBox2.Text,
int.Parse(textBox3.Text), int.Parse(textBox4.Text), textBox5.Text,
int.Parse(textBox6.Text),
                    comboBox1.Text, int.Parse(textBox7.Text));
```

```
                    if (answ == "неуспешно")
                    {
                        MessageBox.Show("Неправильно введены данные!");
                        label9.Show();
                        label10.Show();
                        label11.Show();
                        label12.Show();
                        label13.Show();
                        label14.Show();
                        label15.Show();
                    }
                    else
                    {
                        clientBindingSource.ResetBindings(false);
                        label9.Hide();
                        label10.Hide();
                        label11.Hide();
                        label12.Hide();
                        label13.Hide();
                        label14.Hide();
                        label15.Hide();
                        textBox1.Clear();
                        textBox2.Clear();
                        textBox3.Clear();
                        textBox4.Clear();
                        textBox5.Clear();
                        textBox6.Clear();
                        textBox7.Clear();
                    }
                }
                else
                {
                    MessageBox.Show("Неправильно введена дата оформления кредита или
другие поля.");
                }
            }
            catch
            {
                MessageBox.Show("Неправильно введены данные!");
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            label9.Show();
            label10.Show();
            label11.Show();
            label12.Show();
            label13.Show();
            label14.Show();
            label15.Show();
        }

        private void суммыToolStripMenuItem_Click(object sender, EventArgs e)
        {
            clientBindingSource.DataSource = Client.Clients =
Client.Clients.OrderByDescending(x => x.Money).ToList<Client>();
            clientBindingSource.ResetBindings(false);
        }

        private void срокаToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            clientBindingSource.DataSource = Client.Clients =
Client.Clients.OrderByDescending(x => x.Months).ToList<Client>();
            clientBindingSource.ResetBindings(false);
        }

        private void суммыToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            clientBindingSource.DataSource = Client.Clients = Client.Clients.OrderBy(x =>
x.Money).ToList<Client>();
            clientBindingSource.ResetBindings(false);
        }

        private void срокаToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            clientBindingSource.DataSource = Client.Clients = Client.Clients.OrderBy(x =>
x.Months).ToList<Client>();
            clientBindingSource.ResetBindings(false);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            // Вывод отчёта в эксель.
            Microsoft.Office.Interop.Excel.Application Excel = new
Microsoft.Office.Interop.Excel.Application();
            Microsoft.Office.Interop.Excel.Workbook ExcelWorkBook;
            Microsoft.Office.Interop.Excel.Worksheet ExcelWorkSheet;
            //Книга.
            ExcelWorkBook = Excel.Workbooks.Add(System.Reflection.Missing.Value);
            //Таблица.
            ExcelWorkSheet =
(Microsoft.Office.Interop.Excel.Worksheet)ExcelWorkBook.Worksheets.get_Item(1);

            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
                for (int j = 0; j < dataGridView1.ColumnCount; j++)
                {
                    Excel.Cells[i + 1, j + 1] = dataGridView1.Rows[i].Cells[j].Value;
                }
            }
            Excel.Visible = true;
            Excel.UserControl = true;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Form3 form3 = new Form3();
            form3.Show();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            string answ = Client.Delete_info(textBox1.Text, textBox2.Text);
            if (answ != "успешно")
            {
                MessageBox.Show("Неверно введены данные! (Такого пользователя нет)");
            }
            else
            {
                clientBindingSource.ResetBindings(false);
            }
        }

        private void dataGridView1_Click(object sender, EventArgs e)
```

```
        {
            for (int i = 1; i < dataGridView1.Rows.Count - 1; i++)
                dataGridView1.Rows[i].Visible = dataGridView1[6, i].Value.ToString() ==
comboBox1.Text;
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp4_Нелли_Грушенцева
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }
        public static List<string> list_of_types = new List<string>();
        public static List<int> list_of_money = new List<int>();

        public static void pieplot()
        {
            for (int i = 0; i < Client.Clients.Count; i++)
            {
                list_of_types.Add(Client.Clients[i].Type);
                list_of_money.Add(Client.Clients[i].Money);
            }
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            pieplot();
            chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;
            chart1.Series[0].Points.DataBindXY(list_of_types, list_of_money);
            list_of_types.Clear();
            list_of_money.Clear();
        }
    }
}
```