**ВВЕДЕНИЕ**

В данной пояснительной записке рассматривается описание программного модуля «Справочник любителя живописи» на основе объектно-ориентированного подхода.

При объектно-ориентированном подходе программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).

Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создавать модель сколь угодно сложной предметной области с заданными свойствами.

Еще одним теоретически интересным и практически важным свойством объектно-ориентированного подхода является поддержка механизма обработки событий, которые изменяют атрибуты объектов и моделируют их взаимодействие в предметной области.

В отличие от предыдущих подходов к программированию, объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. К числу основополагающих понятий ООП обычно относят абстракцию данных, наследование, инкапсуляцию и полиморфизм.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015. Язык программирования C#.

**ЦЕЛЬ И ЗАДАЧИ ВЫПОЛНЕНИЯ РАБОТЫ**

Целью данного курсового проекта является автоматизация деятельности любителя живописи на основе объектно-ориентированного подхода, а также получение навыков в реализации этого подхода, проектировании и реализации схемы данных.

Справочник любителя живописи.

Описание предметной области.

Вы любитель живописи. И вашим хобби является коллекционирование картин различных художников. Для того, чтобы более продуктивно заниматься своим увлечением, вы решили вести справочник, где будете записывать информацию о художниках, интересующих вас картинах и аукционах, где можно эту картину приобрести. Вас интересует следующая информация о художниках: Фамилия, Имя, Отчество, дата рождения и стиль, который он использует. Картины же вы характеризуете по следующим критериям: Название картины, автор картины, дата создания и жанр. В графе с аукционами, вы указываете место проведения, дату проведения, картину, которую хотите купить и цену этой картины.

Задачи выполнения работы:

исследование предметной области с целью выявления основных сущностей и их атрибутов;

проектирование иерархии классов и интерфейсов на основе выделенных сущностей;

использование наследования и агрегации для структуризации классов;

применение принципа инкапсуляции к классам;

использование интерфейсов, описывающих операции в данной предметной области, для обеспечения множественного наследования;

- тестирование созданных классов в клиентском приложении, автоматизирующем деятельность предметной области;

- документация разработанных классов при помощи XML комментариев.

**1 ВНЕШНЕЕ ОПИСАНИЕ**

**1.1 Анализ предметной области**

Анализ предметной области – важнейший этап разработки программного обеспечения.

Анализ работы грузовых перевозок позволяет определить ключевые понятия, выяснить задачи, требующие решения в рамках проекта, проанализировать функциональные требования будущего продукта (набор функций, предоставляемых программой) и нефункциональные (надёжность, эффективность, изучаемость, модифицируемость и т.п.).

Компонентами данной предметной области являются изделия, материалы и продажи. Данное приложение должно позволять пользователю создавать записи, удалять, редактировать, сохранять и открывать файлы создаваемые программой Access.

Нефункциональные требования к программному средству:

- надежность: программа должна быть автономной;

- эффективность: программа должна иметь минимальные требования к аппаратному обеспечению. Для использования данного программного средства не должна требоваться установка на компьютер. Не должно требоваться дополнительных периферийных средств (принтер, сканер, дополнительные дисководы и т.д.)

- изучаемость: программа должна быть интуитивно понятна, иметь удобный пользовательский интерфейс;

- модифицируемость: программа должна быть легко модифицируемой вследствие небольшого исходного размера и объектно-ориентированного подхода.

**1.2. Функциональная спецификация**

Минимальные технические характеристики компьютера, на котором гарантируется стабильная работа программы:

компьютер/процессор: компьютер с процессором класса Pentium II 450 МГц;

память: 64 МБ ОЗУ;

монитор: монитор VGA с разрешением 800x600 точек или более высоким, поддерживающий 256 цветов;

операционная система: операционная система Windows XP с установленным Microsoft .NET Framework 4.5.2;

наличие свободного дискового пространства на жёстком диске.

Внешние функции:

внесение данных и редактирование данных о художниках, картинах и аукционах;

сохранение данных в Access -файл;

считывание данных из Access -файла;

проверка вводимых данных и вывод сообщений об ошибках;

программа должна выполнять все функции за предельно малое количество времени;

Критические случаи:

отсутствие свободного места на жёстком диске. Любая операция с файловой системой должна быть отменена;

некорректный Access -файл. Программа не должна аварийно завершаться;

завершение работы Windows. В этом случае программа должна завершать свою работу, не применяя выбранные пользователем настройки.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА АВТОМАТИЗИРОВАННУЮ СИСТЕМУ**

Техническое задание определяет требования к будущей автоматизированной информационной системе со стороны основных процессов.

Система предназначена для решения перечисленных ниже задач:

изменение списка продаж (добавления, удаление, редактирование);

изменение материалов (добавления, удаление, редактирование);

изменение изделий (добавления, удаление, редактирование);

отображение информации, хранящейся в Access –файле;

сохранение информации в Access –файл.

**3 ПРОГРАММИРОВАНИЕ**

**3.1 Описание классов и интерфейсов**

В ООП главным элементом является класс, включающий множество объектов с одинаковыми свойствами, операциями и отношениями. Класс имеет внутреннее (реализацию) и внешнее представление – интерфейс. В данной программе реализованы классы Auction, Artist, Picture, а также интерфейс WindowsForm.

Рассмотрим остальные классы:

Класс Artist описывает художника. Содержит такие атрибуты, как код художника, имя, фамилия, отчество, дата рождения, используемый стиль.

Класс Picture описывает картину и содержит такие атрибуты и параметры картины, как ее название, автора, дату создания и жанр.

Класс Auction описывает аукцион, где можно приобрести требуемую картину. Содержит в себе такие атрибуты, код аукциона, место проведения, дата проведения, картину, которую вы хотите там приобрести, цена этой картины.

**3.2 Схема данных**

На рисунке 1 представлена схема данных:

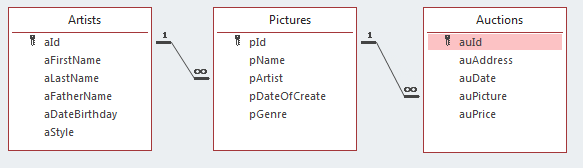


Рисунок 1 – Схема данных

**4 КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ**

**4.1 Структура**

В данном ПП для продуктивной работы реализован удобный пользовательский интерфейс, похожий на большинство Windows – приложений. Данное программное средство можно разбить на модули, взаимодействующие друг с другом. На рисунке 2 представлена структура приложения:

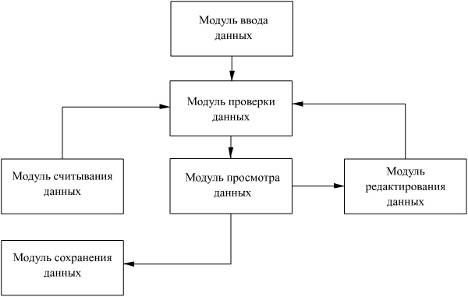


Рисунок 2 – Структура приложения

Модуль просмотра данных предоставляет возможность наглядного отображения данных.

Модуль проверки данных реализует методы:

корректность вводимых пользователем данных;

корректность считываемых из файла данных

Модуль ввода данных, реализует простой графический интерфейс для ввода необходимых значений.

Модуль сохранения данных, формирует все записи в один Access - файл.

Модуль считывания предоставляет возможность открытия Access -файла и заполнения списков в программе.

Модуль редактирования записей предоставляет простой интерфейс для изменения ранее внесенных данных, а также добавления и удаления.

**4.2 Проектирование графического интерфейса**

На рисунке 3 представлена форма приложения и панель, содержащей таблицы данных:

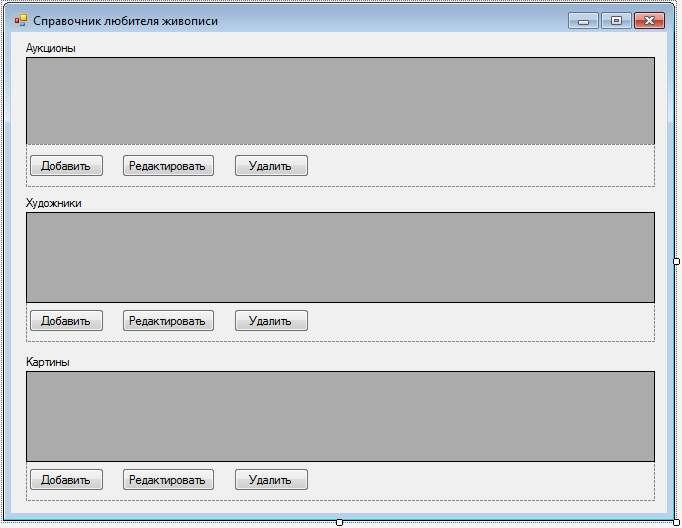


Рисунок 3 – Форма MainForm.

Для редактирования записей о аукционах, художниках и картинах, используются отдельные формы приложения и панели для редактирования таблиц в MainForm. Для редактирования таблицы «Аукционы» используется форма Auctions под названием «Редактирование аукциона», для таблицы «Художники», форма Artists с названием «Редактирование художника», а для последней таблицы используется форма Pictures, название которой «Редактирование картины». Эти формы изображены на рисунках 4-6.

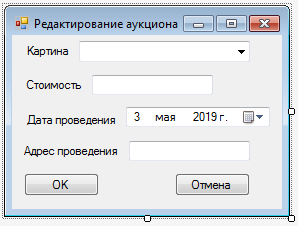


Рисунок 4 – Форма Auctions.

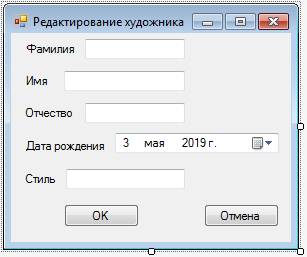


Рисунок 5 – Форма Artists.

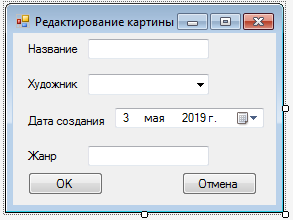


Рисунок 6 – Форма Puctures.

Удаление записи об аукционах, картинах и художниках, осуществляется с помощью специальной кнопки, расположенной на главной форме приложения у каждой таблицы. Нажатием кнопки «Удалить» - удаляем информацию о художнике. Кнопки отображены на рисунках 7-9:

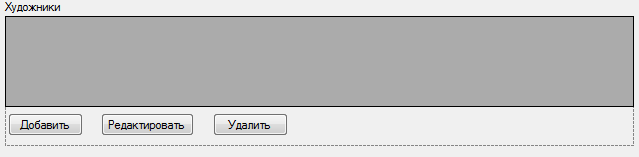


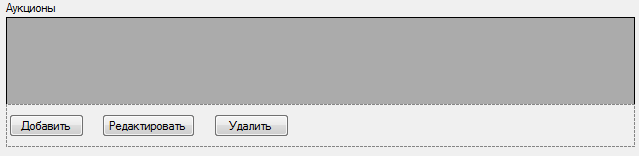
Рисунок 7 – Таблица «Художники»

Рисунок 8 – Таблица «Аукционы»

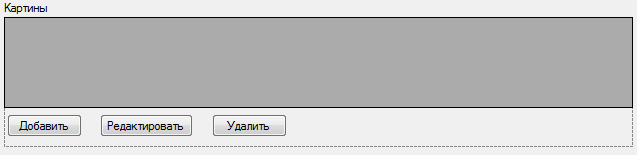


Рисунок 9 – Таблица «Картины»

**5 ТЕСТИРОВАНИЕ**

Для тестирования программы производились различные манипуляции с данными.

Проводились следующие тесты:

Тест 1 – Пустое значение в обязательном поле ввода данных, изображен на рисунке 10:

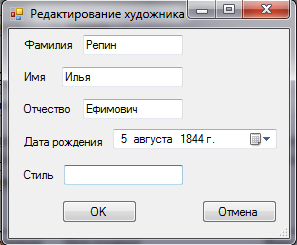


Рисунок 10 – Ввод пустого значения в поле «Стиль»

Сообщение, которое будет показано пользователю после ввода пустого значения, представлено на рисунке 11:

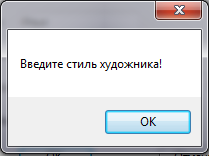


Рисунок 11 – Ошибка при вводе пустого значения.

Тест 2 – Попытка ввода текста в числовое поле, изображен на рисунке 12:

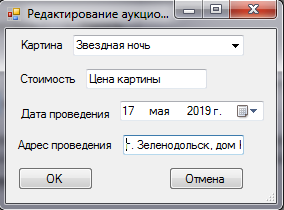


Рисунок 12 – Ввод текста в поле «Стоимость»

Сообщение, которое будет показано пользователю после ввода текста вместо числового значения, представлено на рисунке 13:

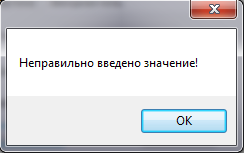


Рисунок 13 – Ввод некорректных данных.

Тест 3 –Добавление значение в поле, которого нет в базе, привязанной к этому полю. Изображено на рисунке 14:

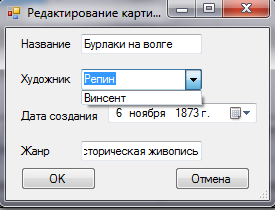


Рисунок 14 – Ввод значения которого нет в базе, привязанный к полю код художника.

Сообщение, которое будет показано пользователю после ввода значения, которого нет в базе, представлено на рисунке 15:

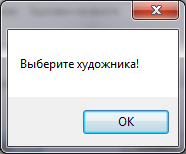


Рисунок 15 – Ошибка неверного выбора.

А так же были проведены другие операции в хаотическом порядке, призванные выявить ошибки в программе.

Тестирование выявило мелкие недочеты, которые впоследствии были устранены.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Э. Троелсен. С# и платформа .NET. Библиотека программиста. - СПб.: Питер, 2015. —796 с.: ил.
2. Е.А. Жоголев. «Технология программирования» - М., Научный мир, 2015, 216 с
3. Т. А. Павловская. «С#. Программирование на языке высокого уровня. Учебник для вузов» - СПб.: Питер, 2015. – 432 с.
4. Microsoft Developer Network(MSDN). [Электронный ресурс]. – Режим доступа: msdn.microsoft.com, свободный.
5. СТО ЗМК 01. – 2017. Работы студентов. Общие требования и правила оформления.

**ПРИЛОЖЕНИЕ А**

**Листинг библиотеки классов**

**Класс Db.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.OleDb;

using System.Configuration;

using System.Windows.Forms;

namespace DirectoryOfPainting

{

/// <summary>

/// класс для создания коннекции к БД

/// </summary>

public static class Db

{

//коннекция к БД

public static OleDbConnection Connection = new OleDbConnection();

/// <summary>

/// процедура соеднинения с БД

/// </summary>

/// <returns></returns>

public static bool Connect()

{

//читаем строку коннекции к БД из файла конфиг

var connectionString = ConfigurationManager.AppSettings["db"];

try

{

//присваиваем строку коннекции

Connection.ConnectionString = connectionString;

//открываем соединение с БД

Connection.Open();

return true;

}

catch (Exception e)

{

//обработка ошибки при коннекции к БД. вызываем сообщение с ошибкой

MessageBox.Show(@"Ошибка соединения с БД. " + e.Message, @"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return false;

}

}

}

}

**Класс Program.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace DirectoryOfPainting

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

//пробуем соединиться к БД, если соединение не прошло то выведется сообщение, дальнейший запуск не актуален

if (!Db.Connect()) return;

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

**Класс Auction.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DirectoryOfPainting

{

/// <summary>

/// описание аукциона

/// </summary>

public class Auction

{

//Идентификатор

public int auId { get; set; }

//Дата проведения аукциона

public DateTime auDate { get; set; }

//Интересующая картина

public int auPicture { get; set; }

//Стоимость картины

public int auPrice { get; set; }

//Адрес проведения

public string auAddress { get; set; }

}

}

**Класс Artist**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DirectoryOfPainting

{

/// <summary>

/// описание художника

/// </summary>

public class Artist

{

//Идентификатор

public int aId { get; set; }

//Имя

public string aFirstName { get; set; }

//Фамилия

public string aLastName { get; set; }

//Отчество

public string aFatherName { get; set; }

//Дата рождения

public DateTime aDateBirthday { get; set; }

//Используемый стиль

public string aStyle { get; set; }

}

}

**Класс Picture.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DirectoryOfPainting

{

/// <summary>

/// описание картины

/// </summary>

public class Picture

{

//Идентификатор

public int pId { get; set; }

//Название картины

public string pName { get; set; }

//Художник

public int pArtist { get; set; }

//Дата создания

public DateTime pDateOfCreate { get; set; }

//Жанр картины

public string pGenre { get; set; }

}

}

**ПРИЛОЖЕНИЕ Б**

**Листинг клиентского приложения**

**Форма приложения MainForm**

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;

using System.Data.OleDb;

namespace DirectoryOfPainting

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

//загружаем данные по аукционам

LoadAuctions();

//загружаем данные по художникам

LoadArtists();

//загружаем данные по картинам

LoadPictures();

}

/// <summary>

/// загрузка художников

/// </summary>

private void LoadArtists()

{

//пишем SQL по отбору данных по художникам, сортируем по фамилии

var sql = @"select \* from Artists order by aLastName ";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом datagridview

dgvArtists.DataSource = ds;

dgvArtists.DataMember = ds.Tables[0].TableName;

//не показываем столбец с ИД

dgvArtists.Columns["aId"].Visible = false;

//устанавливаем заголовок столбца

dgvArtists.Columns["aFirstName"].HeaderText = @"Имя";

//устанавдиваем ширину столбца

dgvArtists.Columns["aFirstName"].Width = 100;

dgvArtists.Columns["aLastName"].HeaderText = @"Фамилия";

dgvArtists.Columns["aLastName"].Width = 100;

dgvArtists.Columns["aFatherName"].HeaderText = @"Отчество";

dgvArtists.Columns["aFatherName"].Width = 100;

dgvArtists.Columns["aDateBirthday"].HeaderText = @"Дата рождения";

dgvArtists.Columns["aDateBirthday"].Width = 200;

dgvArtists.Columns["aStyle"].HeaderText = @"Стиль";

dgvArtists.Columns["aStyle"].Width = 100;

}

/// <summary>

/// загрузка картин

/// </summary>

private void LoadPictures()

{

//пишем SQL по отбору данных по картинам, сортируем по названию

var sql = @"select aId, pId, pName, pArtist, pDateOfCreate, pGenre from Pictures, Artists where pArtist=aId order by pName ";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом datagridview

dgvPictures.DataSource = ds;

dgvPictures.DataMember = ds.Tables[0].TableName;

//не показываем столбец с ИД

dgvPictures.Columns["pId"].Visible = false;

dgvPictures.Columns["aId"].Visible = false;

//устанавливаем заголовок столбца

dgvPictures.Columns["pName"].HeaderText = @"Название";

//устанавдиваем ширину столбца

dgvPictures.Columns["pName"].Width = 100;

dgvPictures.Columns["pArtist"].HeaderText = @"Художник";

dgvPictures.Columns["pArtist"].Width = 70;

dgvPictures.Columns["pDateOfCreate"].HeaderText = @"Дата создания";

dgvPictures.Columns["pDateOfCreate"].Width = 200;

dgvPictures.Columns["pGenre"].HeaderText = @"Жанр";

dgvPictures.Columns["pGenre"].Width = 100;

}

/// <summary>

/// загрузка аукциона

/// </summary>

private void LoadAuctions()

{

//пишем SQL по отбору данных по аукционам, сортируем по дате проведения

var sql = @"select auId, auAddress, auPicture, pId, auPrice, auDate from Auctions, Pictures where auPicture=pId order by auDate desc";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом datagridview

dgvAuctions.DataSource = ds;

dgvAuctions.DataMember = ds.Tables[0].TableName;

//не показываем столбец с ИД

dgvAuctions.Columns["auId"].Visible = false;

dgvAuctions.Columns["pId"].Visible = false;

//устанавливаем заголовок столбца

dgvAuctions.Columns["auAddress"].HeaderText = @"Адрес проведения";

//устанавдиваем ширину столбца

dgvAuctions.Columns["auAddress"].Width = 150;

dgvAuctions.Columns["auPicture"].HeaderText = @"Картина";

dgvAuctions.Columns["auPicture"].Width = 70;

dgvAuctions.Columns["auPrice"].HeaderText = @"Стоимость";

dgvAuctions.Columns["auPrice"].Width = 100;

dgvAuctions.Columns["auDate"].HeaderText = @"Дата проведения";

dgvAuctions.Columns["auDate"].Width = 200;

}

/// <summary>

/// добавление аукциона

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnAuAdd\_Click(object sender, EventArgs e)

{

//создаем форму редактирования

var f = new Auctions();

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по добавлению данных о аукционе

var cmd = new OleDbCommand(@"insert into Auctions (auAddress, auDate, auPicture, auPrice) values (?,?,?,?)")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для добавления данных

cmd.Parameters.AddWithValue(@"auAddress", f.Auction.auAddress);

cmd.Parameters.AddWithValue(@"auDate", f.Auction.auDate);

cmd.Parameters.AddWithValue(@"auPicture", f.Auction.auPicture);

cmd.Parameters.AddWithValue(@"auPrice", f.Auction.auPrice);

//выполняем запрос по добавлению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadAuctions();

}

}

/// <summary>

/// редактирование аукциона

/// </summary>

private void btnAuChange\_Click(object sender, EventArgs e)

{

//если нет выделенной ячейки, то выходим

if (dgvAuctions.CurrentCell == null) return;

//индекс выделенной ячейки

var i = dgvAuctions.CurrentCell.RowIndex;

//создаем форму редактирования

var f = new Auctions();

//заплняем объект класса Auction данными из datagridview

f.Auction.auId = (int)dgvAuctions.Rows[i].Cells["auId"].Value;

f.Auction.auPicture = (int)dgvAuctions.Rows[i].Cells["pId"].Value;

f.Auction.auAddress = Convert.ToString(dgvAuctions.Rows[i].Cells["auAddress"].Value);

f.Auction.auDate = Convert.ToDateTime(dgvAuctions.Rows[i].Cells["auDate"].Value);

f.Auction.auPrice = Convert.ToInt32(dgvAuctions.Rows[i].Cells["auPrice"].Value);

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по изменению данных о аукционе

var cmd = new OleDbCommand(@"update Auctions set auAddress=?, auPicture=?, auDate=?, auPrice=? where auId=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для изменения данных

cmd.Parameters.AddWithValue(@"auAddress", f.Auction.auAddress);

cmd.Parameters.AddWithValue(@"auPicture", f.Auction.auPicture);

cmd.Parameters.AddWithValue(@"auDate", f.Auction.auDate);

cmd.Parameters.AddWithValue(@"auPrice", f.Auction.auPrice);

cmd.Parameters.AddWithValue(@"auId", f.Auction.auId);

//выполняем запрос по изменению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadAuctions();

}

}

/// <summary>

/// удаление аукциона

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnAuDel\_Click(object sender, EventArgs e)

{

//если нет выделенной ячейки, то выходим

if (dgvAuctions.CurrentCell == null) return;

//диалог подтверждения удаления

if (MessageBox.Show(@"Удалить?", @"Удалить", MessageBoxButtons.YesNo) != DialogResult.Yes)

return;

//индекс выделенной ячейки

var i = dgvAuctions.CurrentCell.RowIndex;

//идентификатор записи

var id = dgvAuctions.Rows[i].Cells["auId"].Value;

//sql по удалению аукциона

var cmd = new OleDbCommand(@"delete from Auctions where auId=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для удаления данных

cmd.Parameters.AddWithValue(@"auId", id);

//выполняем запрос по удалению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadAuctions();

}

/// <summary>

/// добавление художника

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnArAdd\_Click(object sender, EventArgs e)

{

//создаем форму редактирования

var f = new Artists();

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по добавлению данных о художнике

var cmd = new OleDbCommand(@"insert into Artists (aFirstName, aLastName, aFatherName, aDateBirthday, aStyle) values (?,?,?,?,?)")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для добавления данных

cmd.Parameters.AddWithValue(@"aFirstName", f.Artist.aFirstName);

cmd.Parameters.AddWithValue(@"aLastName", f.Artist.aLastName);

cmd.Parameters.AddWithValue(@"aFatherName", f.Artist.aFatherName);

cmd.Parameters.AddWithValue(@"aDateBirthday", f.Artist.aDateBirthday);

cmd.Parameters.AddWithValue(@"aStyle", f.Artist.aStyle);

//выполняем запрос по добавлению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadArtists();

}

}

/// <summary>

/// редактирование художника

/// </summary>

private void btnArChange\_Click(object sender, EventArgs e)

{

//если нет выделенной ячейки, то выходим

if (dgvArtists.CurrentCell == null) return;

//индекс выделенной ячейки

var i = dgvArtists.CurrentCell.RowIndex;

//создаем форму редактирования

var f = new Artists();

//заплняем объект класса Artist данными из datagridview

f.Artist.aId = (int)dgvArtists.Rows[i].Cells["aId"].Value;

f.Artist.aFirstName = Convert.ToString(dgvArtists.Rows[i].Cells["aFirstName"].Value);

f.Artist.aLastName = Convert.ToString(dgvArtists.Rows[i].Cells["aLastName"].Value);

f.Artist.aDateBirthday = Convert.ToDateTime(dgvArtists.Rows[i].Cells["aDateBirthday"].Value);

f.Artist.aFatherName = Convert.ToString(dgvArtists.Rows[i].Cells["aFatherName"].Value);

f.Artist.aStyle = Convert.ToString(dgvArtists.Rows[i].Cells["aStyle"].Value);

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по изменению данных о художнике

var cmd = new OleDbCommand(@"update Artists set aFirstName=?, aLastName=?, aFatherName=?, aDateBirthday=?, aStyle=? where aId=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для изменения данных

cmd.Parameters.AddWithValue(@"aFirstName", f.Artist.aFirstName);

cmd.Parameters.AddWithValue(@"aLastName", f.Artist.aLastName);

cmd.Parameters.AddWithValue(@"aFatherName", f.Artist.aFatherName);

cmd.Parameters.AddWithValue(@"aDateBirthday", f.Artist.aDateBirthday);

cmd.Parameters.AddWithValue(@"aStyle", f.Artist.aStyle);

cmd.Parameters.AddWithValue(@"aId", f.Artist.aId);

//выполняем запрос по изменению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadArtists();

}

}

/// <summary>

/// удаление художника

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnArDel\_Click(object sender, EventArgs e)

{

//если нет выделенной ячейки, то выходим

if (dgvArtists.CurrentCell == null) return;

//диалог подтверждения удаления

if (MessageBox.Show(@"Удалить?", @"Удалить", MessageBoxButtons.YesNo) != DialogResult.Yes)

return;

//индекс выделенной ячейки

var i = dgvArtists.CurrentCell.RowIndex;

//идентификатор записи

var id = dgvArtists.Rows[i].Cells["auId"].Value;

//sql по удалению художника

var cmd = new OleDbCommand(@"delete from Artists where aId=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для удаления данных

cmd.Parameters.AddWithValue(@"aId", id);

//выполняем запрос по удалению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadArtists();

}

/// <summary>

/// добавление картины

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnPicAdd\_Click(object sender, EventArgs e)

{

//создаем форму редактирования

var f = new Pictures();

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по добавлению данных о картине

var cmd = new OleDbCommand(@"insert into Pictures (pName, pDateOfCreate, pArtist, pGenre) values (?,?,?,?)")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для добавления данных

cmd.Parameters.AddWithValue(@"pName", f.Picture.pName);

cmd.Parameters.AddWithValue(@"pDateOfCreate", f.Picture.pDateOfCreate);

cmd.Parameters.AddWithValue(@"pArtist", f.Picture.pArtist);

cmd.Parameters.AddWithValue(@"pGenre", f.Picture.pGenre);

//выполняем запрос по добавлению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadPictures();

}

}

/// <summary>

/// редактирование картины

/// </summary>

private void btnPicChange\_Click(object sender, EventArgs e)

{

//если нет выделенной ячейки, то выходим

if (dgvPictures.CurrentCell == null) return;

//индекс выделенной ячейки

var i = dgvPictures.CurrentCell.RowIndex;

//создаем форму редактирования

var f = new Pictures();

//заплняем объект класса Picture данными из datagridview

f.Picture.pId = (int)dgvPictures.Rows[i].Cells["pId"].Value;

f.Picture.pArtist = (int)dgvPictures.Rows[i].Cells["aId"].Value;

f.Picture.pName = Convert.ToString(dgvPictures.Rows[i].Cells["pName"].Value);

f.Picture.pGenre = Convert.ToString(dgvPictures.Rows[i].Cells["pGenre"].Value);

f.Picture.pDateOfCreate = Convert.ToDateTime(dgvPictures.Rows[i].Cells["pDateOfCreate"].Value);

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по изменению данных о картине

var cmd = new OleDbCommand(@"update Pictures set pName=?, pArtist=?, pDateOfCreate=?, pGenre=? where pId=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для изменения данных

cmd.Parameters.AddWithValue(@"pName", f.Picture.pName);

cmd.Parameters.AddWithValue(@"pArtist", f.Picture.pArtist);

cmd.Parameters.AddWithValue(@"pDateOfCreate", f.Picture.pDateOfCreate);

cmd.Parameters.AddWithValue(@"pGenre", f.Picture.pGenre);

cmd.Parameters.AddWithValue(@"pId", f.Picture.pId);

//выполняем запрос по изменению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadPictures();

}

}

/// <summary>

/// удаление картины

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnPicDel\_Click(object sender, EventArgs e)

{

//если нет выделенной ячейки, то выходим

if (dgvPictures.CurrentCell == null) return;

//диалог подтверждения удаления

if (MessageBox.Show(@"Удалить?", @"Удалить", MessageBoxButtons.YesNo) != DialogResult.Yes)

return;

//индекс выделенной ячейки

var i = dgvPictures.CurrentCell.RowIndex;

//идентификатор записи

var id = dgvPictures.Rows[i].Cells["pId"].Value;

//sql по удалению картины

var cmd = new OleDbCommand(@"delete from Pictures where pId=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для удаления данных

cmd.Parameters.AddWithValue(@"pId", id);

//выполняем запрос по удалению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadPictures();

}

private void dgvAuctions\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

btnAuChange\_Click(sender, e);

}

private void dgvArtists\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

btnArChange\_Click(sender, e);

}

private void dgvPicture\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

btnPicChange\_Click(sender, e);

}

}

}

**Форма приложения Auctions.**

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;

using System.Data.OleDb;

namespace DirectoryOfPainting

{

public partial class Auctions : Form

{

/// <summary>

/// форма ввода и редактирования аукциона

/// </summary>

public Auctions()

{

InitializeComponent();

LoadPicture();

Auction.auDate = DateTime.Today;

}

//объект класса Auction

public Auction Auction = new Auction();

/// <summary>

/// метод загрузки комбобокса с картинами

/// </summary>

private void LoadPicture()

{

//пишем SQL по отбору данных по картинам, сортируем по названию

var sql = @"select \* from Pictures order by pName";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом комбобокс

cbPicture.DataSource = ds.Tables[0];

cbPicture.DisplayMember = "pName";

cbPicture.ValueMember = "pId";

}

private void Auctions\_Load(object sender, EventArgs e)

{

//устанавливаем значения компонентов при редактировании

dtpDate.Value = Auction.auDate;

tbPrice.Text = Convert.ToString(Auction.auPrice);

cbPicture.SelectedValue = Auction.auPicture;

tbAddress.Text = Auction.auAddress;

}

/// <summary>

/// обработка кнопки ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnOK\_Click(object sender, EventArgs e)

{

//проверяем выбор картины

if (cbPicture.SelectedIndex < 0)

{

//сообщение

MessageBox.Show(@"Выберите картину!");

//устанавливаем фокус

cbPicture.Focus();

//не закрываем форму

DialogResult = DialogResult.None;

return;

}

//проверяем правильность ввода стоимости

int d = 0;

if (!Int32.TryParse(tbPrice.Text, out d))

{

MessageBox.Show(@"Неправильно введено значение!");

tbPrice.Focus();

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность адреса

if (string.IsNullOrEmpty(tbAddress.Text))

{

MessageBox.Show(@"Введите имя!");

tbAddress.Focus();

DialogResult = DialogResult.None;

return;

}

Auction.auDate = dtpDate.Value;

Auction.auPicture = (int)cbPicture.SelectedValue;

Auction.auPrice = Convert.ToInt32(tbPrice.Text);

Auction.auAddress = tbAddress.Text;

}

}

}

**Форма приложения Artists.**

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 DirectoryOfPainting

{

public partial class Artists : Form

{

/// <summary>

/// форма ввода и редактирования художника

/// </summary>

public Artists()

{

InitializeComponent();

//по-умолчанию дата рождения - сегодняшнее число

Artist.aDateBirthday = DateTime.Today;

}

//объект класса Artist

public Artist Artist = new Artist();

private void Artists\_Load(object sender, EventArgs e)

{

//устанавливаем значения компонентов при редактировании

tbFirstName.Text = Artist.aFirstName;

tbLastName.Text = Artist.aLastName;

tbFatherName.Text = Artist.aFatherName;

tbStyle.Text = Artist.aStyle;

dtpDateBirthday.Value = Artist.aDateBirthday;

}

/// <summary>

/// обработка кнопки ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnOK\_Click(object sender, EventArgs e)

{

//проверяем на заполненность имени

if (string.IsNullOrEmpty(tbFirstName.Text))

{

//сообщение

MessageBox.Show(@"Введите имя!");

//устанавливаем фокус

tbFirstName.Focus();

//не закрываем форму

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность фамилии

if (string.IsNullOrEmpty(tbLastName.Text))

{

MessageBox.Show(@"Введите фамилию!");

tbLastName.Focus();

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность отчества

if (string.IsNullOrEmpty(tbFatherName.Text))

{

MessageBox.Show(@"Введите отчество!");

tbFatherName.Focus();

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность стиля

if (string.IsNullOrEmpty(tbStyle.Text))

{

MessageBox.Show(@"Введите стиль художника!");

tbStyle.Focus();

DialogResult = DialogResult.None;

return;

}

//присваиваем данные из компонентов объекту классса Artist

Artist.aFirstName = tbFirstName.Text;

Artist.aLastName = tbLastName.Text;

Artist.aFatherName = tbFatherName.Text;

Artist.aStyle = tbStyle.Text;

Artist.aDateBirthday = dtpDateBirthday.Value;

}

}

}

**Форма приложения Pictures.**

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;

using System.Data.OleDb;

namespace DirectoryOfPainting

{

public partial class Pictures : Form

{

/// <summary>

/// форма ввода и редактирования картин

/// </summary>

public Pictures()

{

InitializeComponent();

//заполняем комбобокс с художниками

LoadArtists();

//по-умолчанию дата создания - сегодняшнее число

Picture.pDateOfCreate = DateTime.Today;

}

//объект класса Picture

public Picture Picture = new Picture();

/// <summary>

/// метод загрузки комбобокса с художниками

/// </summary>

private void LoadArtists()

{

//пишем SQL по отбору данных по художнику, сортируем по фамилии

var sql = @"select \* from Artists order by aLastName";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом комбобокс

cbArtist.DataSource = ds.Tables[0];

cbArtist.DisplayMember = "aLastName";

cbArtist.ValueMember = "aId";

}

private void Pictures\_Load(object sender, EventArgs e)

{

//устанавливаем значения компонентов при редактировании

tbName.Text = Picture.pName;

tbGenre.Text = Picture.pGenre;

dtpDateOfCreate.Value = Picture.pDateOfCreate;

cbArtist.SelectedValue = Picture.pArtist;

}

private void btnOK\_Click(object sender, EventArgs e)

{

//проверяем выбор художника

if (cbArtist.SelectedIndex < 0)

{

MessageBox.Show(@"Выберите художника!");

cbArtist.Focus();

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность названия картины

if (string.IsNullOrEmpty(tbName.Text))

{

MessageBox.Show(@"Введите название!");

tbName.Focus();

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность жанра картины

if (string.IsNullOrEmpty(tbGenre.Text))

{

MessageBox.Show(@"Введите жанр!");

tbGenre.Focus();

DialogResult = DialogResult.None;

return;

}

//присваиваем данные из компонентов объекту классса Picture

Picture.pArtist = (int)cbArtist.SelectedValue;

Picture.pName = tbName.Text;

Picture.pGenre = tbGenre.Text;

Picture.pDateOfCreate = dtpDateOfCreate.Value;

}

}

}