

Лабораторная работа №7

Тема: Отладка отдельных модулей программного проекта.

Организация обработки исключений.

Выполнил: Кузнецов Богдан ПР 23/2

Цель: создать программный проект в среде Visual Studio на языке C#, реализовать организацию обработки исключений для обеспечения надежной работы приложения, а также разработать функционал для оформления предзаказов с использованием базы данных MS SQL Server.

Ход работы

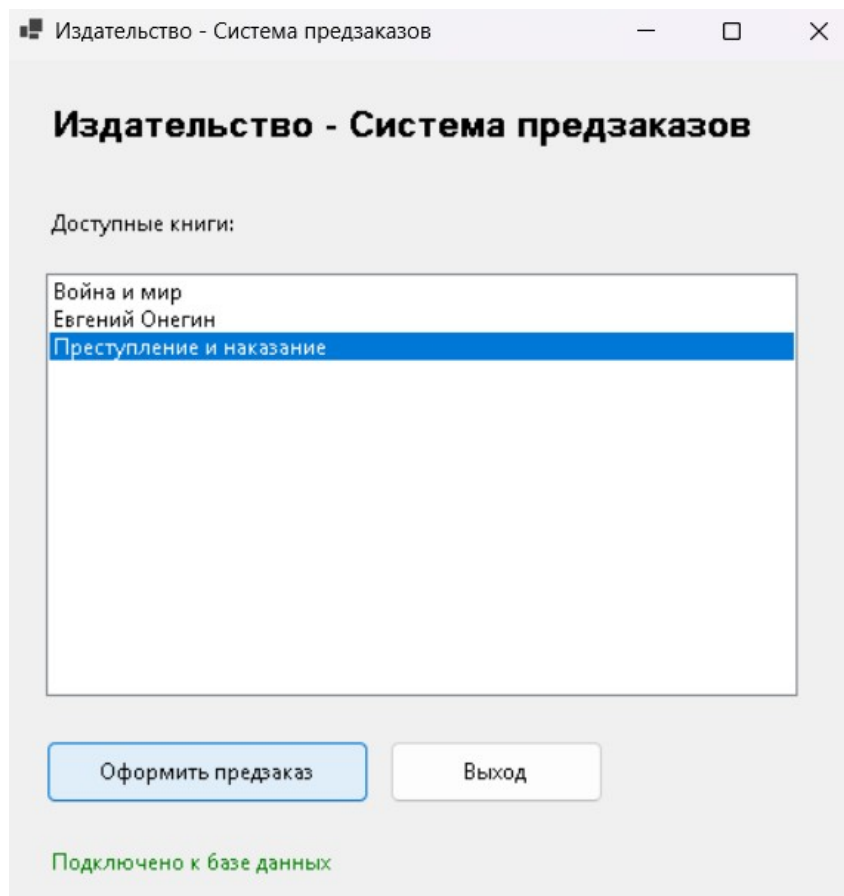


Рисунок 1 — Выбор книги

Оформление предзаказа

Информация о книге

Преступление и наказание
Достоевский Федор
1866
800,00 ₺

Данные клиента

ФИО: Кузнецов Богдан Александрович
Адрес: ул. Пушкина, д. Калатушкина
Телефон: +7 949 567 8948
Офис выдачи: Филиал на Пушкина

Оформить заказ Отмена

Рисунок 2 — Ввод данных клиента

Чек предзаказа

ЧЕК ПРЕДЗАКАЗА

Информация о заказе

Номер заказа: № 3
Дата: 14.12.2025 23:24
Книга: Преступление и наказание
Клиент: Кузнецов Богдан Александрович
Офис выдачи: Филиал на Пушкина
Стоимость: 800,00 ₺

=====Книга:
Преступление и наказаниеКлиент: Кузнецов Богдан
АлександровичОфис получения: Филиал на Пушкина
=====Стоимость:
800,00 ₺=====

Спасибо за заказ!Ожидайте уведомления о поступлении.

Печать Сохранить Закрыть

Рисунок 3 — Формирование чека

Листинг кода:

```
using System;
using Microsoft.Data.SqlClient;
using System.Collections.Generic;

namespace PR7
{
    public class DatabaseHelper : IDisposable
    {
        private string connectionString = @"Data Source=rezonchic_;Initial
Catalog=publishing;Integrated
Security=True;TrustServerCertificate=True;Connect Timeout=30";

        public List<Book> GetBooks()
        {
            var books = new List<Book>();

            try
            {
                using (var connection = new SqlConnection(connectionString))
                {
                    connection.Open();

                    string query = @"
                        SELECT p.id_Publication, p.Name, p.Author,
                               a.Surname + ' ' + a.Name as AuthorName,
                               p.ReleaseYear, p.VolumeOfSheets, p.Circulation, p.Price
                        FROM Publications p
                        LEFT JOIN Authors a ON p.Author = a.id_Author
                        ORDER BY p.Name";

                    using (var command = new SqlCommand(query, connection))
```

```

using (var reader = command.ExecuteReader())
{
    while (reader.Read())
    {
        books.Add(new Book
        {
            Id = (int)reader["id_Publication"],
            Title = reader["Name"].ToString(),
            AuthorId = reader["Author"] != DBNull.Value ?
(int)reader["Author"] : 0,
            AuthorName = reader["AuthorName"].ToString(),
            ReleaseYear = (int)reader["ReleaseYear"],
            Pages = (int)reader["VolumeOfSheets"],
            Circulation = (int)reader["Circulation"],
            Price = (decimal)reader["Price"]
        });
    }
}

catch (Exception ex)
{
    throw new Exception($"Ошибка при загрузке книг: {ex.Message}");
}

return books;
}

public List<Office> GetOffices()
{

```

```

var offices = new List<Office>();

try
{
    using (var connection = new SqlConnection(connectionString))
    {
        connection.Open();

        string query = "SELECT id_Office, Office, Address, Phone FROM
Offices ORDER BY Office";

        using (var command = new SqlCommand(query, connection))
        using (var reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                offices.Add(new Office
                {
                    Id = (int)reader["id_Office"],
                    Name = reader["Office"].ToString(),
                    Address = reader["Address"].ToString(),
                    Phone = reader["Phone"].ToString()
                });
            }
        }
    }
}
catch (Exception ex)
{
    throw new Exception($"Ошибка при загрузке офисов:
{ex.Message}");
}

```

```

    }
    return offices;
}

public int CreateOrder(Order order)
{
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string query = @"
                INSERT INTO Orders (Name, Type, Publication, Office, Customer,
                                   DateOfAdmission, DateOfCompletion, Price)
                VALUES (@Name, 1, @Publication, @Office, @Customer,
                        @DateOfAdmission, @DateOfCompletion, @Price);
                SELECT SCOPE_IDENTITY();";

            using (var command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Name",
order.OrderName);
                command.Parameters.AddWithValue("@Publication",
order.BookId);
                command.Parameters.AddWithValue("@Office", order.OfficeId);
                command.Parameters.AddWithValue("@Customer",
order.CustomerId);
                command.Parameters.AddWithValue("@DateOfAdmission",
order.OrderDate);
            }
        }
    }
}

```

```

        if (order.CompletionDate.HasValue)
            command.Parameters.AddWithValue("@DateOfCompletion",
order.CompletionDate.Value);
        else
            command.Parameters.AddWithValue("@DateOfCompletion",
DBNull.Value);

        command.Parameters.AddWithValue("@Price", order.Price);

        return Convert.ToInt32(command.ExecuteScalar());
    }
}
}
catch (Exception ex)
{
    throw new Exception($"Ошибка при создании заказа: {ex.Message}");
}
}

```

```

public int CreateCustomer(Customer customer)
{
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();

            string query = @"
                INSERT INTO Customers (Name, Type, Address, Phone)
                VALUES (@Name, 1, @Address, @Phone);
            "

```

```
SELECT SCOPE_IDENTITY();
```

```
using (var command = new SqlCommand(query, connection))
{
    command.Parameters.AddWithValue("@Name", customer.Name);
    command.Parameters.AddWithValue("@Address",
        string.IsNullOrEmpty(customer.Address) ?
(object)DBNull.Value : customer.Address);
    command.Parameters.AddWithValue("@Phone",
        string.IsNullOrEmpty(customer.Phone) ? (object)DBNull.Value :
customer.Phone);

    return Convert.ToInt32(command.ExecuteScalar());
}
}
}
catch (Exception ex)
{
    throw new Exception($"Ошибка при создании клиента:
{ex.Message}");
}
}
```

```
public Order GetOrderDetails(int orderId)
{
    try
    {
        using var connection = new SqlConnection(connectionString);
        connection.Open();
        string query = @"
```

```

SELECT o.id_Order, o.Name as OrderName, o.DateOfAdmission,
       o.Price, p.Name as BookTitle, c.Name as CustomerName,
       ofc.Office as OfficeName
FROM Orders o
LEFT JOIN Publications p ON o.Publication = p.id_Publication
LEFT JOIN Customers c ON o.Customer = c.id_Customer
LEFT JOIN Offices ofc ON o.Office = ofc.id_Office
WHERE o.id_Order = @OrderId";

```

```

using (var command = new SqlCommand(query, connection))
{
    command.Parameters.AddWithValue("@OrderId", orderId);

    using (var reader = command.ExecuteReader())
    {
        if (reader.Read())
        {
            return new Order
            {
                Id = (int)reader["id_Order"],
                OrderName = reader["OrderName"].ToString(),
                BookTitle = reader["BookTitle"].ToString(),
                CustomerName = reader["CustomerName"].ToString(),
                OfficeName = reader["OfficeName"].ToString(),
                OrderDate = (DateTime)reader["DateOfAdmission"],
                Price = (decimal)reader["Price"]
            };
        }
    }
}

```

```

        }
    }
    catch (Exception ex)
    {
        throw new Exception($"Ошибка при получении деталей заказа:
{ex.Message}");
    }
    return null;
}

```

```

public bool TestConnection()
{
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();
            return connection.State == System.Data.ConnectionState.Open;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Connection test failed: {ex.Message}");
        return false;
    }
}

```

```

public void Dispose()
{

```

```
        // Очистка ресурсов при необходимости
    }
}
}
```

```
using System;
```

```
using System.Windows.Forms;
```

```
namespace PR7
```

```
{
```

```
    public partial class FormOrder : Form
```

```
    {
```

```
        private DatabaseHelper dbHelper;
```

```
        private Book selectedBook;
```

```
        public FormOrder(DatabaseHelper dbHelper, Book book)
```

```
        {
```

```
            InitializeComponent();
```

```
            this.dbHelper = dbHelper;
```

```
            this.selectedBook = book;
```

```
            LoadOffices();
```

```
            InitializeForm();
```

```
        }
```

```
        private void InitializeForm()
```

```

{
    bookTitleLabel.Text = selectedBook.Title;
    authorLabel.Text = selectedBook.AuthorName;
    yearLabel.Text = selectedBook.ReleaseYear.ToString();
    priceLabel.Text = $" {selectedBook.Price:C}";
}

private void LoadOffices()
{
    try
    {
        var offices = dbHelper.GetOffices();
        officeComboBox.DataSource = offices;
        officeComboBox.DisplayMember = "Name";
        officeComboBox.ValueMember = "Id";
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при загрузке офисов: {ex.Message}",
"Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void createOrderButton_Click(object sender, EventArgs e)
{
    try
    {
        // Валидация данных

```

```
if (string.IsNullOrEmpty(customerNameTextBox.Text))
{
    MessageBox.Show("Введите ФИО клиента", "Внимание",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return;
}
```

```
if (officeComboBox.SelectedItem == null)
{
    MessageBox.Show("Выберите офис", "Внимание",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return;
}
```

```
// Создаем клиента
var customer = new Customer
{
    Name = customerNameTextBox.Text,
    Address = addressTextBox.Text,
    Phone = phoneTextBox.Text
};
```

```
// Сохраняем клиента в БД
int customerId = dbHelper.CreateCustomer(customer);
```

```
// Создаем заказ
var office = (Office)officeComboBox.SelectedItem;
var order = new Order
{
```

```

        OrderName = $"Предзаказ: {selectedBook.Title}",
        BookId = selectedBook.Id,
        OfficeId = office.Id,
        CustomerId = customerId,
        OrderDate = DateTime.Now,
        Price = selectedBook.Price
    };

    // Сохраняем заказ в БД
    int orderId = dbHelper.CreateOrder(order);

    // Показываем чек
    var receiptForm = new FormReceipt(dbHelper, orderId);
    receiptForm.ShowDialog();

    MessageBox.Show("Заказ успешно оформлен!", "Успех",
        MessageBoxButtons.OK, MessageBoxIcon.Information);

    this.Close();
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка при оформлении заказа: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void cancelButton_Click(object sender, EventArgs e)

```

```

    {
        this.Close();
    }

    private void FormOrder_Load(object sender, EventArgs e)
    {

    }

}
}
}

```

```

using System;
using System.Windows.Forms;

namespace PR7
{
    public partial class FormReceipt : Form
    {
        private DatabaseHelper dbHelper;
        private int orderId;

        public FormReceipt(DatabaseHelper dbHelper, int orderId)
        {
            InitializeComponent();

```

```

        this.dbHelper = dbHelper;
        this.orderId = orderId;

        LoadReceiptData();
    }

    private void LoadReceiptData()
    {
        try
        {
            var order = dbHelper.GetOrderDetails(orderId);

            if (order != null)
            {
                orderNumberLabel.Text = $"№ {order.Id}";
                orderDateLabel.Text = order.OrderDate.ToString("dd.MM.yyyy
HH:mm");
                bookTitleLabel.Text = order.BookTitle;
                customerNameLabel.Text = order.CustomerName;
                officeLabel.Text = order.OfficeName;
                priceLabel.Text = $"{order.Price:C}";

                // Генерация текста для печати/сохранения
                receiptTextBox.Text = GenerateReceiptText(order);
            }
        }
        catch (Exception ex)
        {

```

```

        MessageBox.Show($"Ошибка при загрузке данных чека:
{ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

private string GenerateReceiptText(Order order)
{
    return $"=====\\n" +
        $"    ЧЕК ПРЕДЗАКАЗА\\n" +
        $"=====\\n" +
        $"Номер заказа: {order.Id}\\n" +
        $"Дата: {order.OrderDate:dd.MM.yyyy HH:mm}\\n" +
        $"=====\\n" +
        $"Книга: {order.BookTitle}\\n" +
        $"Клиент: {order.CustomerName}\\n" +
        $"Офис получения: {order.OfficeName}\\n" +
        $"=====\\n" +
        $"Стоимость: {order.Price:C}\\n" +
        $"=====\\n" +
        $"Спасибо за заказ!\\n" +
        $"Ожидайте уведомления о поступлении.";
}

```

```

private void printButton_Click(object sender, EventArgs e)
{
    try
    {

```

```

        MessageBox.Show("Функция печати будет реализована позже",
"Информация",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при печати: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void saveButton_Click(object sender, EventArgs e)
{
    try
    {
        SaveFileDialog saveDialog = new SaveFileDialog();
        saveDialog.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|
*. *";
        saveDialog.Title = "Сохранить чек";
        saveDialog.FileName = $"чек_заказа_{orderId}.txt";

        if (saveDialog.ShowDialog() == DialogResult.OK)
        {
            System.IO.File.WriteAllText(saveDialog.FileName,
receiptTextBox.Text);
            MessageBox.Show("Чек успешно сохранен!", "Успех",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    catch (Exception ex)

```

```

        {
            MessageBox.Show($"Ошибка при сохранении: {ex.Message}",
"Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

```

```

private void closeButton_Click(object sender, EventArgs e)

```

```

{
    this.Close();
}

```

```

private void receiptGroupBox_Enter(object sender, EventArgs e)

```

```

{

}

}
}

```

```

using Microsoft.Data.SqlClient;

```

```

using PR7;

```

```

using System.Windows.Forms;

```

```

namespace PR7

```

```

{
    public partial class MainForm : Form

```

```

{
    private DatabaseHelper dbHelper;

    public MainForm()
    {
        InitializeComponent();
        dbHelper = new DatabaseHelper();
        LoadBooks();
        TestConnection();
    }

    private void TestConnection()
    {
        try
        {
            if (dbHelper.TestConnection())
            {
                statusLabel.Text = "Подключено к базе данных";
                statusLabel.ForeColor = System.Drawing.Color.Green;
            }
            else
            {
                statusLabel.Text = "Ошибка подключения к БД";
                statusLabel.ForeColor = System.Drawing.Color.Red;
            }
        }
        catch (Exception ex)
        {
            statusLabel.Text = $"Ошибка: {ex.Message}";
        }
    }
}

```

```

        statusLabel.ForeColor = System.Drawing.Color.Red;
    }
}

private void LoadBooks()
{
    try
    {
        var books = dbHelper.GetBooks();
        booksListBox.DataSource = books;
        booksListBox.DisplayMember = "Title";
        booksListBox.ValueMember = "Id";
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при загрузке книг: {ex.Message}",
            "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void createOrderButton_Click(object sender, EventArgs e)
{
    if (booksListBox.SelectedItem == null)
    {
        MessageBox.Show("Выберите книгу для заказа", "Внимание",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
}

```

```

        var selectedBook = (Book)booksListBox.SelectedItem;
        var orderForm = new FormOrder(dbHelper, selectedBook);
        orderForm.ShowDialog();
    }

    private void exitButton_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void MainForm_Load(object sender, EventArgs e)
    {

    }
}
}

```

```

using System;
namespace PR7
{
    public class Author
    {
        public int Id { get; set; }
        public string Surname { get; set; }
    }
}

```

```
public string Name { get; set; }  
public string FullName => $"{Surname} {Name}";  
}
```

```
public class Book  
{  
    public int Id { get; set; }  
    public string Title { get; set; }  
    public int AuthorId { get; set; }  
    public string AuthorName { get; set; }  
    public int ReleaseYear { get; set; }  
    public int Pages { get; set; }  
    public int Circulation { get; set; }  
    public decimal Price { get; set; }  
}
```

```
public class Customer  
{  
    public int Id { get; set; }  
    public string Name { get; set; }  
    public string Address { get; set; }  
    public string Phone { get; set; }  
    public int Type { get; set; } = 1;  
}
```

```
public class Office  
{  
    public int Id { get; set; }  
    public string Name { get; set; }
```

```

        public string Address { get; set; }
        public string Phone { get; set; }
    }

    public class Order
    {
        public int Id { get; set; }
        public string OrderName { get; set; }
        public int BookId { get; set; }
        public int OfficeId { get; set; }
        public int CustomerId { get; set; }
        public DateTime OrderDate { get; set; }
        public DateTime? CompletionDate { get; set; }
        public decimal Price { get; set; }

        // Для отображения
        public string BookTitle { get; set; }
        public string CustomerName { get; set; }
        public string OfficeName { get; set; }
    }
}

using System;
namespace PR7
{
    public class Author
    {

```

```
public int Id { get; set; }  
public string Surname { get; set; }  
public string Name { get; set; }  
public string FullName => $"{Surname} {Name}";  
}
```

```
public class Book  
{  
    public int Id { get; set; }  
    public string Title { get; set; }  
    public int AuthorId { get; set; }  
    public string AuthorName { get; set; }  
    public int ReleaseYear { get; set; }  
    public int Pages { get; set; }  
    public int Circulation { get; set; }  
    public decimal Price { get; set; }  
}
```

```
public class Customer  
{  
    public int Id { get; set; }  
    public string Name { get; set; }  
    public string Address { get; set; }  
    public string Phone { get; set; }  
    public int Type { get; set; } = 1;  
}
```

```
public class Office  
{
```

```

    public int Id { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string Phone { get; set; }
}

public class Order
{
    public int Id { get; set; }
    public string OrderName { get; set; }
    public int BookId { get; set; }
    public int OfficeId { get; set; }
    public int CustomerId { get; set; }
    public DateTime OrderDate { get; set; }
    public DateTime? CompletionDate { get; set; }
    public decimal Price { get; set; }

    // Для отображения
    public string BookTitle { get; set; }
    public string CustomerName { get; set; }
    public string OfficeName { get; set; }
}
}

```