	<p>Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)</p>
---	--

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Студент _____ Шегай Павел Игоревич
Фамилия имя отчество

Группа ИУ6-25Б

Тип практики Проектно-технологическая практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Студент _____
(Подпись, дата) _____ П. И. Шегай
(И.О. Фамилия)

Руководитель практики _____
(Подпись, дата) _____ А. А. Веселовский
(И.О. Фамилия)

Оценка отлично (90 из 100)

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ЗАДАНИЕ на учебную практику

по теме Проектирование и реализация программного обеспечения с использованием
структурного и объектного подходов

Студент группы ИУ6-25Б

Шегай Павел Игоревич
(Фамилия, имя, отчество)

Тип практики Проектно-технологическая практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Техническое задание:

Задание 1. Создание информационной программной системы с графическим интерфейсом на C++

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

База данных (файл) платежного шлюза содержит сведения о платежах: дата платежа, номер счета получателя, сумма платежа, комиссия в процентах. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать сведения о платежах, проведенных в заданный период.
2. Определить общую сумму платежей, совершенных в указанную дату.
3. Определить суммарную комиссию, полученную от переводов на указанный счет.
4. Построить график зависимости суммы комиссии от суммы платежа.

Задание 3 Изучение средств создания графических интерфейсов на C#

Задание выполняется по методическим указаниям к заданию 3.

Задание 4 Создание программной системы с графическим интерфейсом на C#

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму

последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу.

База данных (файл) платежного шлюза содержит сведения о платежах: дата платежа, номер счета получателя, сумма платежа, комиссия в процентах. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.


1. Показать сведения о платежах, проведенных в заданный период.
2. Определить общую сумму платежей, совершенных в указанную дату.
3. Определить суммарную комиссию, полученную от переводов на указанный счет.

Оформление отчета по практике:

Отчет на 25-35 листах формата А4 должен включать титульный лист, задание (печатать с двух сторон), оглавление, введение, 2-4 главы, заключение и список использованных источников. Отдельная глава по каждому заданию должна содержать анализ задания, требуемые чертежи, текст программы, результаты тестирования и выводы.


Дата выдачи задания « 07 » февраля 2024 г.

Руководитель практики

 17.02.24
(Подпись, дата)

О. А. Веселовская
(И.О. Фамилия)

Студент

 17.02.24
(Подпись, дата)

П. И. Шегай
(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

ОГЛАВЛЕНИЕ

Введение	5
1 Создание информационной программной системы С графическим интерфейсом на С++	6
1.1 Объектная декомпозиция приложения	6
1.2 Разработка форм интерфейса	7
1.3 Разработка диаграммы состояний интерфейса	8
1.4 Разработка диаграммы классов интерфейсной и предметной областей приложения	10
1.5 Разработка диаграммы последовательности действий выполнения операции	10
1.6 Разработка кода приложения	11
1.7 Тестирование приложения	13
Вывод.....	15
2 Изучение средств создания графических интерфейсов на С#	16
3 Создание программной системы с графическим интерфейсом на С#	23
Заключение.....	34
Список литературы	35

ВВЕДЕНИЕ

Целью учебной практики является получение навыков создания небольших программных систем с оконными и консольными интерфейсами.

Задачами практики являются:

- более глубокое изучение средств реализации проектов программ на одном из изучаемых универсальных языках программирования высокого уровня;
- овладение методикой и получение практических навыков проектирования небольших программных систем при структурном и объектном подходах;
- воспитание внимания, аккуратности, систематичности, а также формирование интереса к изучаемой профессиональной деятельности.

Выполнение практикума должно способствовать формированию и развитию следующих навыков и умений:

- выделение объектов предметной области, обобщение их в классы, определение связей между классами;
- проектирование эргономичного обеспечения информационных систем;
- разработка и отладка компонентов программных комплексов и систем с помощью современных автоматизированных средств проектирования;
- разработка проектной и эксплуатационной документации на программную и техническую продукцию;
- выполнение контроля разрабатываемых проектов и технической документации на соответствие стандартам и техническим требованиям;
- разработка интерфейсов «человек - ЭВМ».

1 СОЗДАНИЕ ИНФОРМАЦИОННОЙ ПРОГРАММНОЙ СИСТЕМЫ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ НА C++

Задание:

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу.

База данных (файл) платежного шлюза содержит сведения о платежах: дата платежа, номер счета получателя, сумма платежа, комиссия в процентах. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать сведения о платежах, проведенных в заданный период.
2. Определить общую сумму платежей, совершенных в указанную дату.
3. Определить суммарную комиссию, полученную от переводов на указанный счет.
4. Построить график зависимости суммы комиссии от суммы платежа.

1.1 Объектная декомпозиция приложения

При проектировании программного продукта были выделены следующие объекты предметной области:

- Таблица
- Файл
- График
- Окно

Диаграмма объектов представлена на рисунке 1.

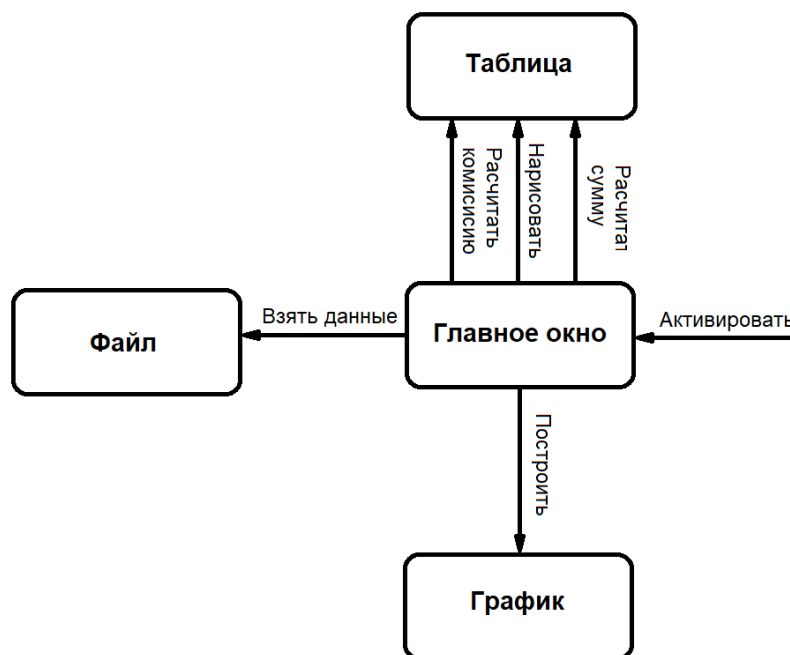
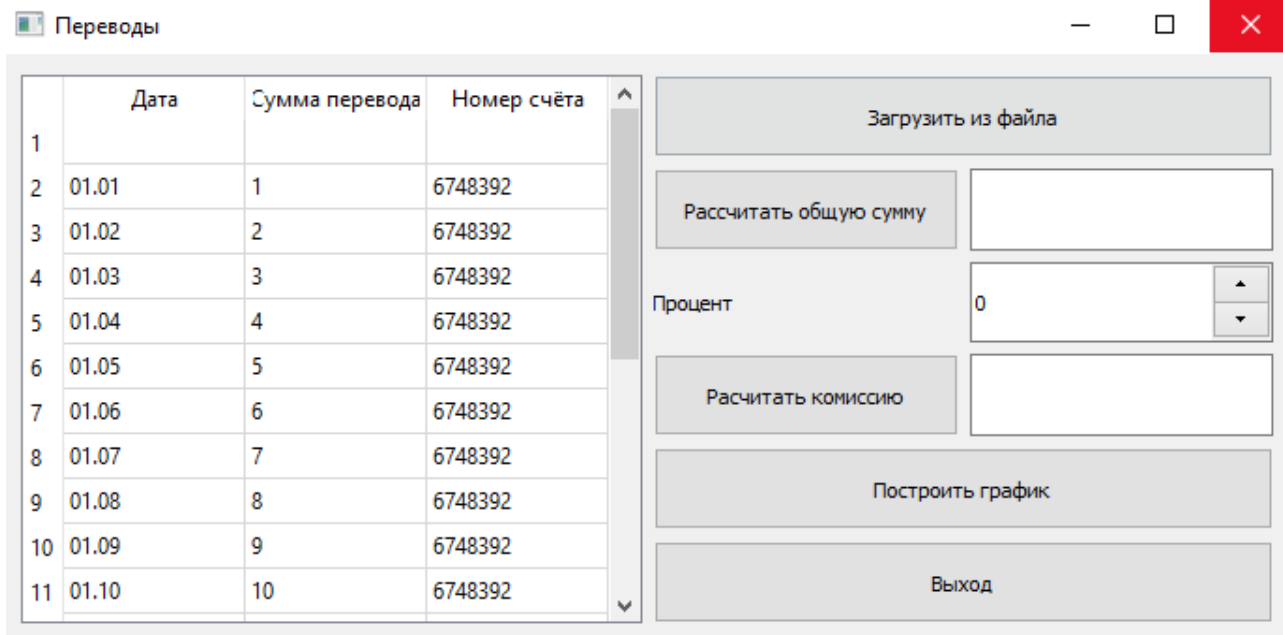


Рисунок 1 – Диаграмма объектов приложения

1.2 Разработка форм интерфейса

Разрабатываемые формы интерфейса должны обеспечивать пользователю возможность выполнения заданных функций.

Форма “Переводы” является главным окном, соответственно она должна включать кнопки, обеспечивающие функционал приложения (рисунок 2).



	Дата	Сумма перевода	Номер счёта
1			
2	01.01	1	6748392
3	01.02	2	6748392
4	01.03	3	6748392
5	01.04	4	6748392
6	01.05	5	6748392
7	01.06	6	6748392
8	01.07	7	6748392
9	01.08	8	6748392
10	01.09	9	6748392
11	01.10	10	6748392

Рисунок 2 – Внешний вид формы «Переводы»

Форма «График» вызывается нажатием кнопки «Построить график». В нем предоставлен график на основе данных, содержащихся в файле.

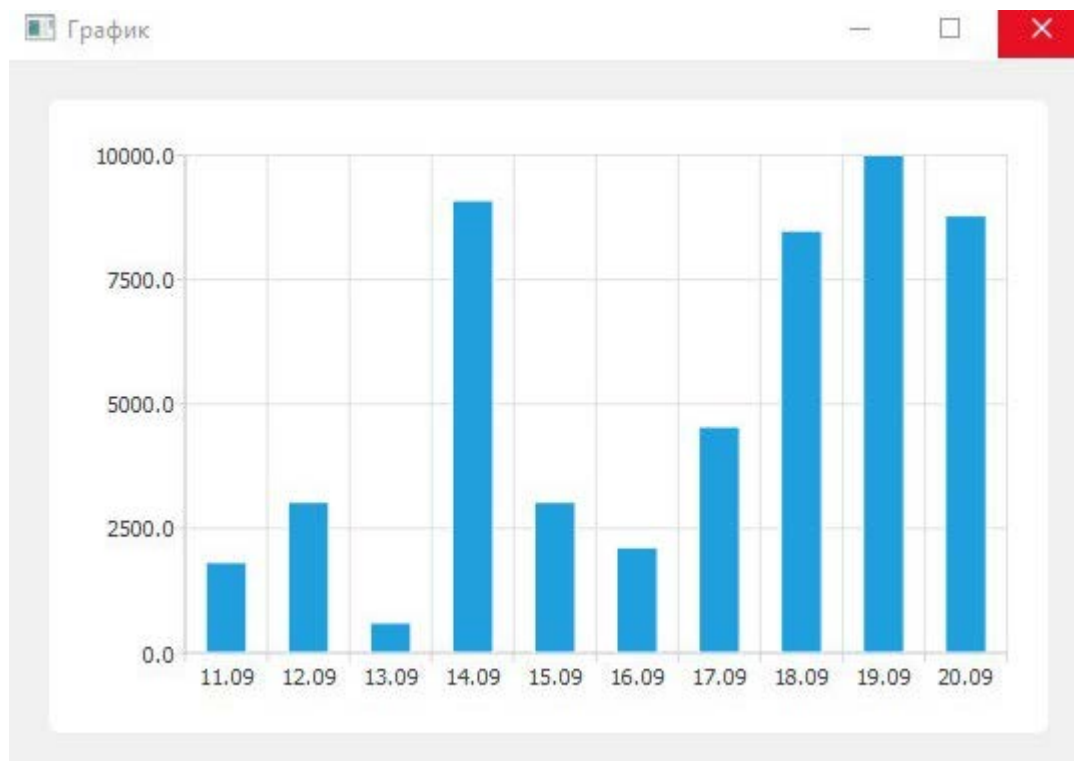


Рисунок 3 – Внешний вид формы «График»

Остальные формы были спроектированы аналогично.

1.3 Разработка диаграммы состояний интерфейса

Диаграмма состояний интерфейса показывает возможные варианты переключения форм интерфейса (рисунок 3).

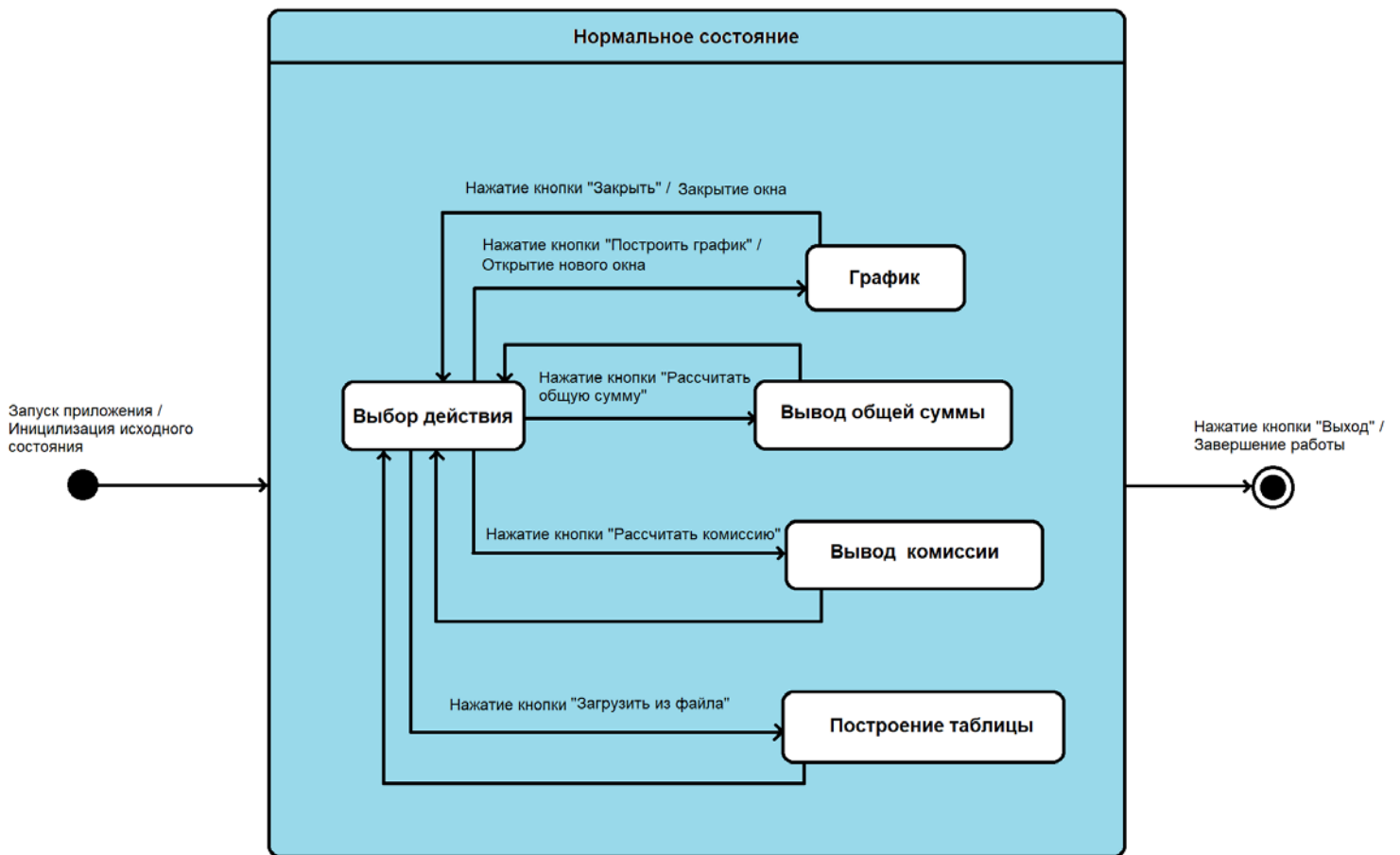


Рисунок 4 – Диаграмма состояний интерфейса

Аналогично разрабатываются диаграммы состояний интерфейса для отдельных форм.

1.4 Разработка диаграммы классов интерфейсной и предметной областей приложения

Диаграмма классов интерфейсной и предметной областей показывает связи между классами (рисунок 4).

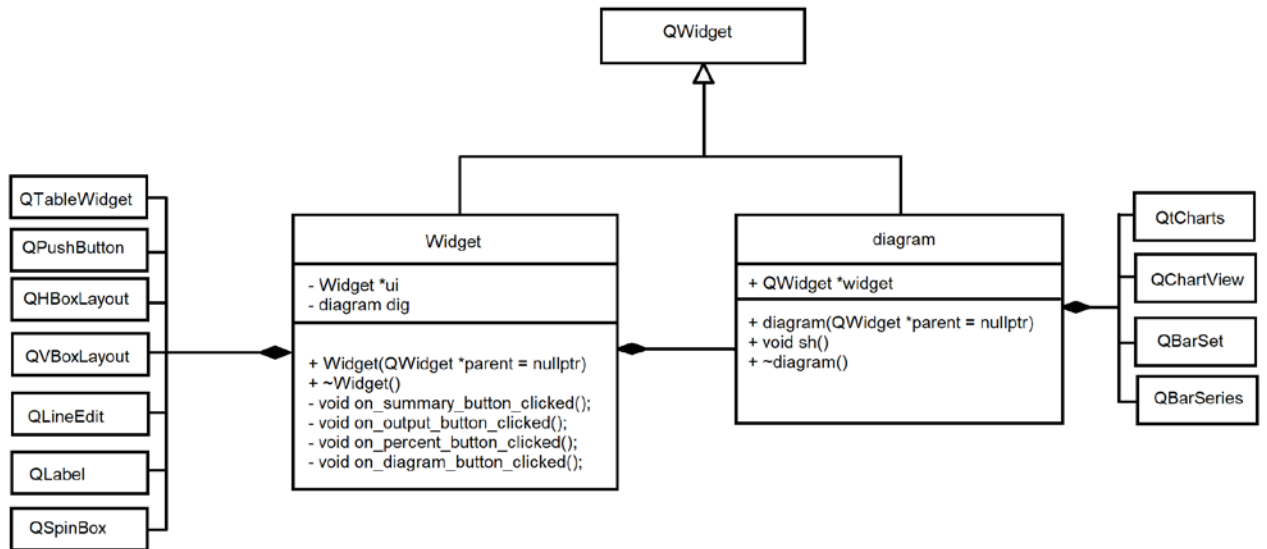


Рисунок 5 – Диаграмма классов приложения

1.5 Разработка диаграммы последовательности действий выполнения операции загрузки данных из файла

Диаграмма последовательностей действий позволяет уточнить порядок выполнения фрагментов операции различными объектами (рисунок 5).

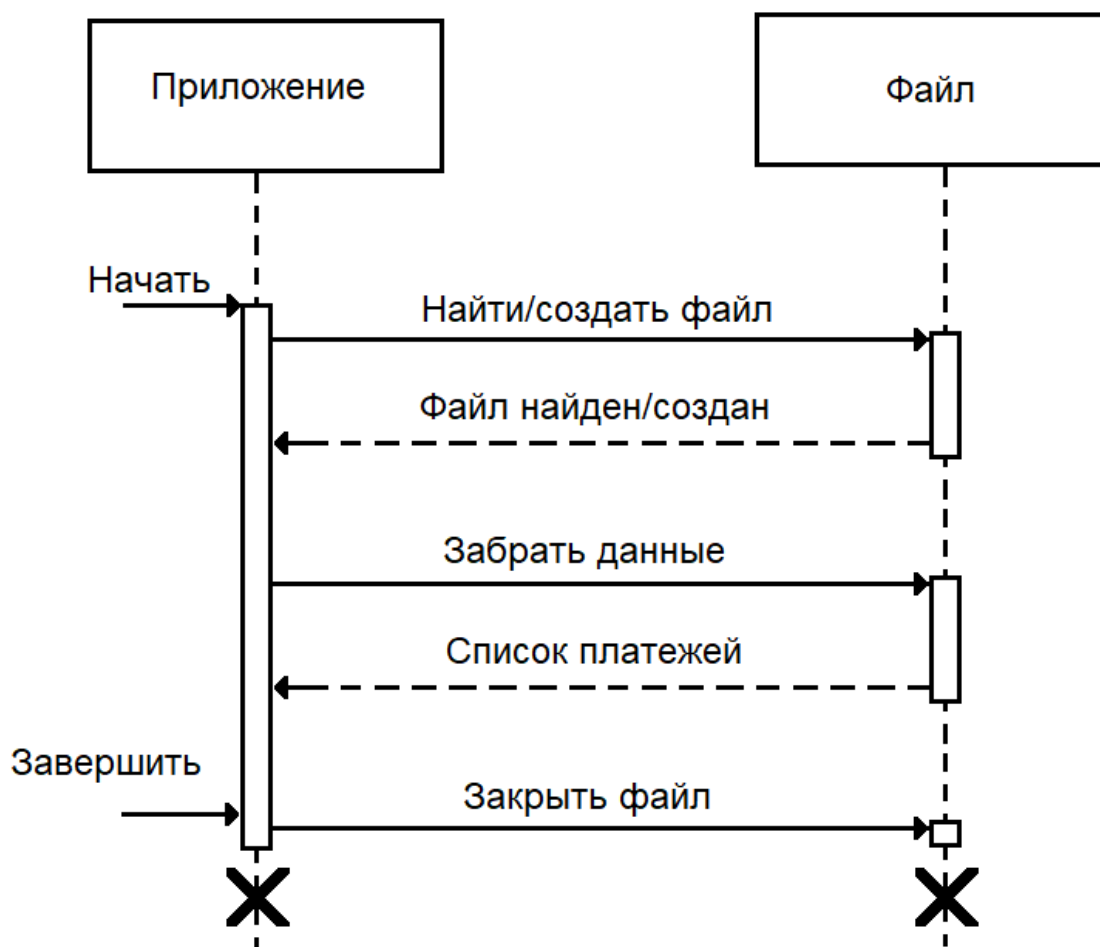


Рисунок 6 – Диаграмма последовательностей действий выполнения операции загрузки данных из файла

1.6 Разработка кода приложения

Уточнение поведения объектов во время выполнения программы позволило разработать код программы.

Ниже в качестве примера приведен код модуля widget:

Заголовочный файл widget.h:

```

#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include "diagram.h"
#include <iostream>
#include <fstream>
#include <string>

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

```

```

class Widget : public QWidget
{
    Q_OBJECT

public:
    Widget(QWidget *parent = nullptr);
    ~Widget();

private slots:

    void on_summary_button_clicked();

    void on_output_button_clicked();

    void on_percent_button_clicked();

    void on_diagram_button_clicked();

private:
    Ui::Widget *ui;
    diagram dig;

};
#endif // WIDGET_H

```

Файл реализации widget.cpp:

```

#include "widget.h"
#include "ui_widget.h"
#include <QTableWidget>

using namespace std;

Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    ui->tableWidget->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
    ui->tableWidget->verticalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);

}

Widget::~~Widget()
{
    delete ui;
}

void Widget::on_summary_button_clicked()
{
    int c = 0;
    for (int i = 0; i < ui->tableWidget->rowCount(); i++){
        if (ui->tableWidget->item(i,1) != 0){
            c += ui->tableWidget->item(i,1)->text().toInt();
        }
        ui->lineEdit->setText(QString::number(c));
    }

}

void Widget::on_output_button_clicked()

```

```

{
    ifstream f("base.txt");
    int i = 0;
    string date, summ, num;
    while (f >> date >> summ >> num){
        i++;
        ui->tableWidget->insertRow(i);
        ui->tableWidget->setItem(i, 0, new
QTableWidgetItem(QString::fromStdString(date)));
        ui->tableWidget->setItem(i, 1, new
QTableWidgetItem(QString::fromStdString(summ)));
        ui->tableWidget->setItem(i, 2, new
QTableWidgetItem(QString::fromStdString(num)));
    }
    f.close();
}

void Widget::on_percent_button_clicked()
{
    float k = ui->spinBox->text().toInt();
    int c = 0;
    for (int i = 0; i < ui->tableWidget->rowCount(); i++){
        if (ui->tableWidget->item(i,1) != 0){
            c += ui->tableWidget->item(i,1)->text().toInt();
        }
    }
    ui->lineEdit_2->setText(QString::number(c*k/100));
}

void Widget::on_diagram_button_clicked()
{
    dig.sh();
}

```

Аналогично был разработан остальной код приложения.

1.7 Тестирование приложения

Разработанное приложение позволяет выполнять функции, предусмотренные заданием (рисунки 7-9) .

Переводы

	Дата	Сумма перевода	Номер счёта
1			
2	11.09	600	639852743109
3	12.09	1000	639852743109
4	13.09	200	639852743109
5	14.09	3000	639852743109
6	15.09	1000	639852743109
7	16.09	700	639852743109
8	17.09	1500	639852743109
9	18.09	2800	639852743109
10	19.09	3300	639852743109
11	20.09	2900	639852743109

Загрузить из файла

Рассчитать общую сумму 17000

Процент 0

Рассчитать комиссию

Построить график

Выход

Рисунок 7 – Расчёт общей суммы переводов

Переводы

	Дата	Сумма перевода	Номер счёта
1			
2	11.09	600	639852743109
3	12.09	1000	639852743109
4	13.09	200	639852743109
5	14.09	3000	639852743109
6	15.09	1000	639852743109
7	16.09	700	639852743109
8	17.09	1500	639852743109
9	18.09	2800	639852743109
10	19.09	3300	639852743109
11	20.09	2900	639852743109

Загрузить из файла

Рассчитать общую сумму 17000

Процент 10

Рассчитать комиссию 1700

Построить график

Выход

Рисунок 8 – Расчёт комиссии

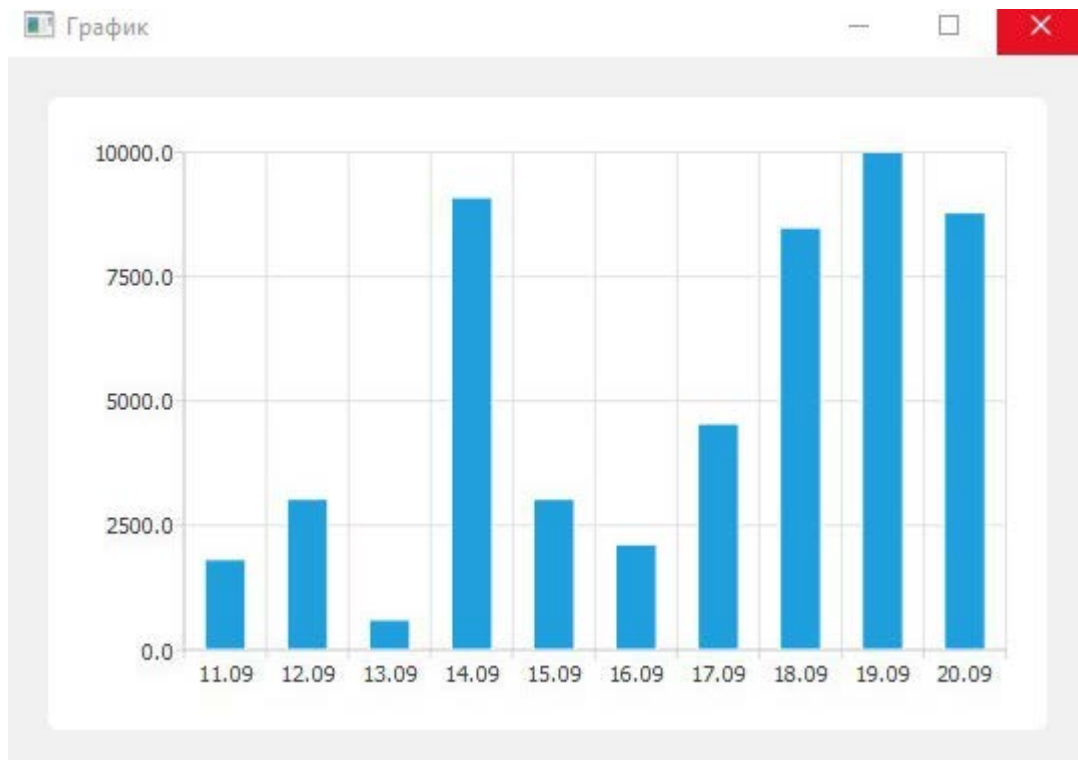


Рисунок 9 – Построение графика

Вывод

При разработке приложения изучены средства разработки приложений с графическим интерфейсом на языке C++, используемые при объектном подходе, а также основные диаграммы, сопровождающие процесс разработки.

2 ИЗУЧЕНИЕ СРЕДСТВ СОЗДАНИЯ ГРАФИЧЕСКИХ ИНТЕРФЕЙСОВ НА C#

2.1 Игра «Крестики-нолики»

Задание:

Разработать приложение «Крестики-нолики». Интерфейс приложения будет состоять:

1. Из игрового поля: 9 клеток, в которых пользователями будут выставляться крестики – символ «X» и нолики – символ «O»;
2. Поля вывода результатов ходов, в которое будет выводиться номер клетки, которую установил свой символ пользователь, и результаты игры;
3. Кнопки «Начать заново», перезапускающей игру.

Для упрощения программа будет разработана для двух игроков-людей. Она будет проверять, победил ли один из игроков после каждого хода, а в конце игры, когда все клетки будут заполнены, будет проверять, выиграл ли

один из игроков или игра закончилась ничьей.

После завершения игры либо победой, либо ничьей, программа не будет реагировать на любые действия пользователей до нажатия ими кнопки

«Начать заново». По нажатию той же кнопки игра может быть завершена досрочно.

Клетки игрового поля будут реализованы с помощью кнопок, по нажатию которых в названиях кнопок будут появляться символы «X» или «O».

2.2 Создание проекта с графическим интерфейсом

Разрабатываемые формы интерфейса должны обеспечивать пользователю возможность выполнения заданных функций.

Форма “Крестики-нолики” является главным окном, соответственно она должна включать кнопки, обеспечивающие функционал приложения

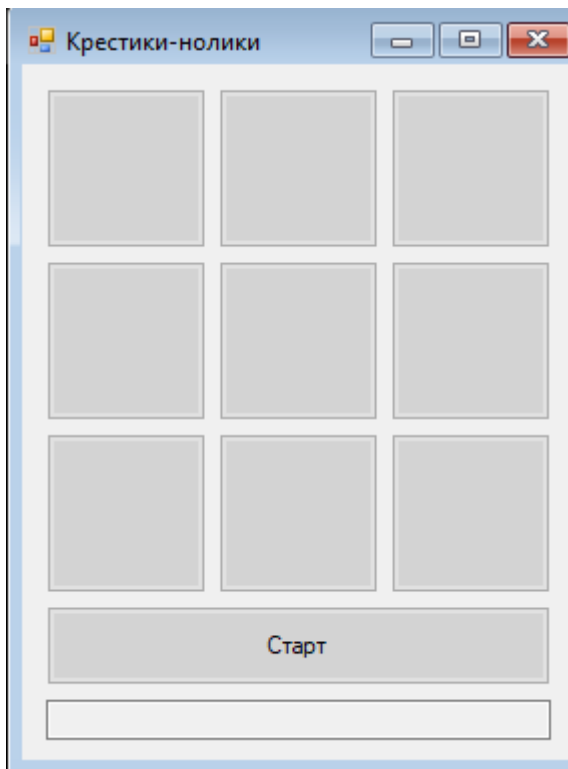


Рисунок 10 – Форма “Крестики-нолики”

2.3 Полный текст исходного кода программы

Для контроля ниже приведен исходный код файла реализации приложения

Form1.cs:

```
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 PTP3
{
    public partial class Form1 : Form
    {
        string curSign;
        int movenum;
        bool gameEnded;
        public Form1()
        {
```

```

InitializeComponent();
curSign = "X";
movenum = 1;
gameEnded = false;
}

private void ClickTicTacToeButton(object sender, EventArgs e)
{
    Button button = (Button)sender;
    if (string.IsNullOrEmpty(button.Text) & !gameEnded)
    {
        button.Text = curSign;
        if (!string.IsNullOrEmpty(this.button1.Text)
            & this.button1.Text == this.button2.Text
            & this.button2.Text == this.button3.Text
            |
            !string.IsNullOrEmpty(this.button6.Text)
            & this.button6.Text == this.button5.Text
            & this.button5.Text == this.button4.Text
            |
            !string.IsNullOrEmpty(this.button7.Text)
            & this.button7.Text == this.button8.Text
            & this.button8.Text == this.button9.Text
            |
            !string.IsNullOrEmpty(this.button1.Text)
            & this.button1.Text == this.button6.Text
            & this.button6.Text == this.button7.Text
            |
            !string.IsNullOrEmpty(this.button2.Text)
            & this.button2.Text == this.button5.Text
            & this.button5.Text == this.button8.Text
            |
            !string.IsNullOrEmpty(this.button3.Text)
            & this.button3.Text == this.button4.Text
            & this.button4.Text == this.button9.Text
            |
            !string.IsNullOrEmpty(this.button1.Text)
            & this.button1.Text == this.button5.Text
            & this.button5.Text == this.button9.Text
            |
            !string.IsNullOrEmpty(this.button3.Text)
            & this.button3.Text == this.button5.Text
            & this.button5.Text == this.button7.Text
        )
        {
            gameEnded = true;
            this.txtbox.Text = "Ход " + movenum.ToString() +
                ": Победил " + curSign + "!";
            this.button1.BackColor = Color.LightGreen;
            this.button2.BackColor = Color.LightGreen;
            this.button3.BackColor = Color.LightGreen;
            this.button6.BackColor = Color.LightGreen;
            this.button5.BackColor = Color.LightGreen;
            this.button4.BackColor = Color.LightGreen;
            this.button7.BackColor = Color.LightGreen;
            this.button8.BackColor = Color.LightGreen;
            this.button9.BackColor = Color.LightGreen;
        }
        else if (!string.IsNullOrEmpty(this.button1.Text) &
            !string.IsNullOrEmpty(this.button2.Text) &
            !string.IsNullOrEmpty(this.button3.Text) &
            !string.IsNullOrEmpty(this.button6.Text) &
            !string.IsNullOrEmpty(this.button5.Text) &
            !string.IsNullOrEmpty(this.button4.Text) &
            !string.IsNullOrEmpty(this.button7.Text) &

```

```

!string.IsNullOrEmpty(this.button8.Text) &
!string.IsNullOrEmpty(this.button9.Text)
)
{
    gameEnded = true;
    this.txtbox.Text = "Ход " + movenum.ToString() +
        ": Ничья!";
    this.button1.BackColor = Color.LightPink;
    this.button2.BackColor = Color.LightPink;
    this.button3.BackColor = Color.LightPink;
    this.button6.BackColor = Color.LightPink;
    this.button5.BackColor = Color.LightPink;
    this.button4.BackColor = Color.LightPink;
    this.button7.BackColor = Color.LightPink;
    this.button8.BackColor = Color.LightPink;
    this.button9.BackColor = Color.LightPink;
}
else
    this.txtbox.Text = "Ход " + movenum.ToString() +
        ": " + curSign + " сходил на " +
        button.Name[6] + " клетку";
if (curSign == "X")
    curSign = "O";
else
    curSign = "X";
movenum += 1;
}
}
private void button10_Click(object sender, EventArgs e)
{
    this.button10.Text = "Заново";
    curSign = "X";
    movenum = 1;
    gameEnded = false;
    this.button1.Text = "";
    this.button1.BackColor = Color.LightGray;
    this.button2.Text = "";
    this.button2.BackColor = Color.LightGray;
    this.button3.Text = "";
    this.button3.BackColor = Color.LightGray;
    this.button6.Text = "";
    this.button6.BackColor = Color.LightGray;
    this.button5.Text = "";
    this.button5.BackColor = Color.LightGray;
    this.button4.Text = "";
    this.button4.BackColor = Color.LightGray;
    this.button7.Text = "";
    this.button7.BackColor = Color.LightGray;
    this.button8.Text = "";
    this.button8.BackColor = Color.LightGray;
    this.button9.Text = "";
    this.button9.BackColor = Color.LightGray;
    this.txtbox.Text = "Игра началась!";
}
}
}

```

2.3 Разработка диаграммы состояний интерфейса

Диаграмма состояний интерфейса показывает возможные варианты переключения форм интерфейса (рисунок 11).

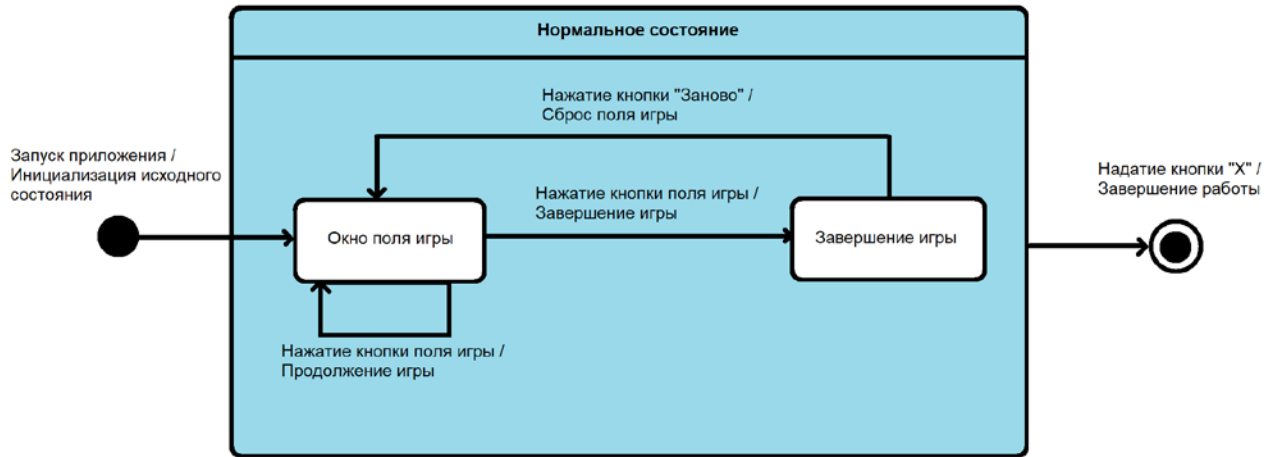


Рисунок 11 – Диаграмма состояний интерфейса

2.4 Разработка диаграммы классов интерфейсной и предметной областей приложения

Диаграмма классов интерфейсной и предметной областей показывает связи между классами (рисунок 12).

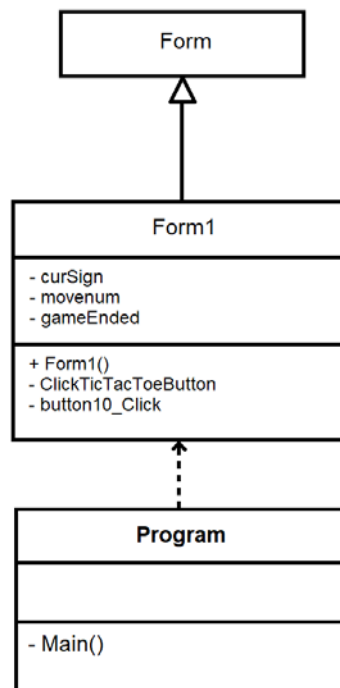


Рисунок 12 – Диаграмма классов приложения

2.5 Тестирование приложения

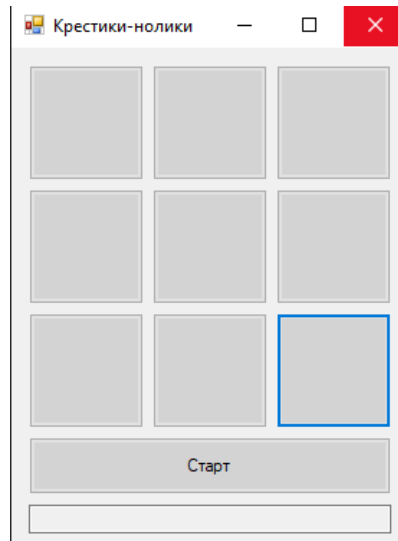


Рисунок 13 – Начальное состояние

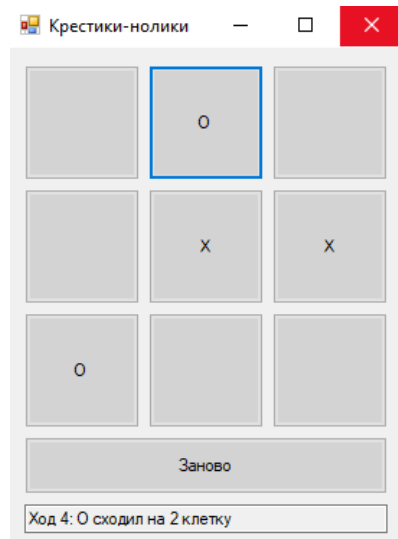


Рисунок 14 – Состояние главного окна после 4 ходов

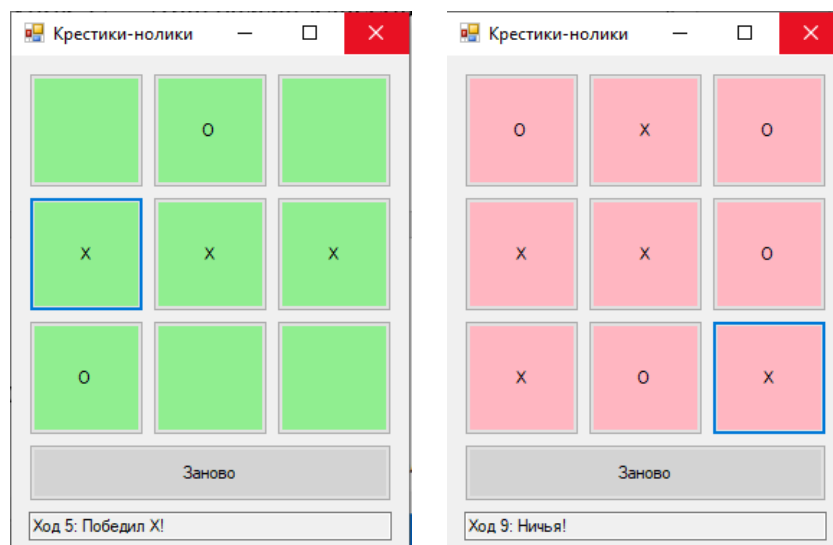


Рисунок 15 – Состояние главного окна после победы и ничьи

Вывод

При разработке приложения изучены средства разработки приложений с графическим интерфейсом на языке C#, используемые при объектном подходе, а также основные диаграммы, сопровождающие процесс разработки.

3 СОЗДАНИЕ ПРОГРАММНОЙ СИСТЕМЫ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ НА C#

Задание:

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу.

База данных (файл) платежного шлюза содержит сведения о платежах: дата платежа, номер счета получателя, сумма платежа, комиссия в процентах. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать сведения о платежах, проведенных в заданный период.
2. Определить общую сумму платежей, совершенных в указанную дату.
3. Определить суммарную комиссию, полученную от переводов на указанный счет.

3.1 Объектная декомпозиция приложения

При проектировании программного продукта были выделены следующие объекты предметной области:

- Таблица
- Файл
- Окно

Диаграмма объектов представлена на рисунке 16.

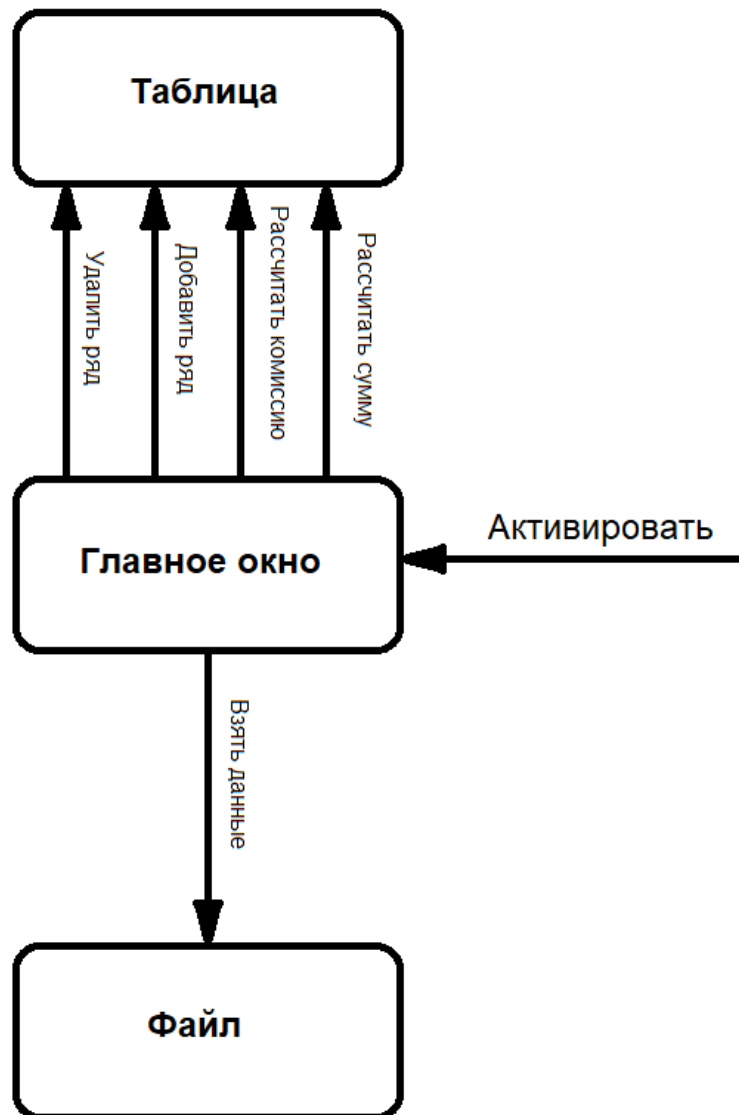


Рисунок 16 – Диаграмма объектов приложения

3.2 Разработка форм интерфейса

Разрабатываемые формы интерфейса должны обеспечивать пользователю возможность выполнения заданных функций.

Форма “Переводы” является главным окном, соответственно она должна включать кнопки, обеспечивающие функционал приложения (рисунок 17).

	Date	Number	Summ	Comission
▶	01.01.22	12345678901	5000	50
	01.01.22	23456789012	6000	60
	02.01.22	12345678901	7000	70
	03.01.22	45678901234	8000	80
	03.01.22	12345678901	9000	90
	04.01.22	67890123456	1000	10
	05.01.22	78901234567	2000	20
	05.01.22	89012345678	3000	30
	08.01.22	90123456789	4000	40
	10.01.22	12345678901	9500	95
	12.01.22	12345678901	8500	85
	12.01.22	23456789012	7500	75
	12.01.22	34567890123	6500	65
	15.01.22	45678901234	5500	55
	15.01.22	56789012345	4500	45
	16.01.22	67890123456	3500	35
	17.01.22	78901234567	2500	25
	18.01.22	89012345678	1500	15
	19.01.22	90123456789	5000	50

Дата: 12.01.22

Расчитать сумму

Сумма: 5425

Номер счёта: 12345678901

Расчитать комиссию

31750

Добавить ряд

Номер ряда:

Удалить ряд

Выход

Рисунок 17 – Внешний вид формы “Переводы”

3.3 Разработка диаграммы состояний интерфейса

Диаграмма состояний интерфейса показывает возможные варианты переключения форм интерфейса (рисунок 18).

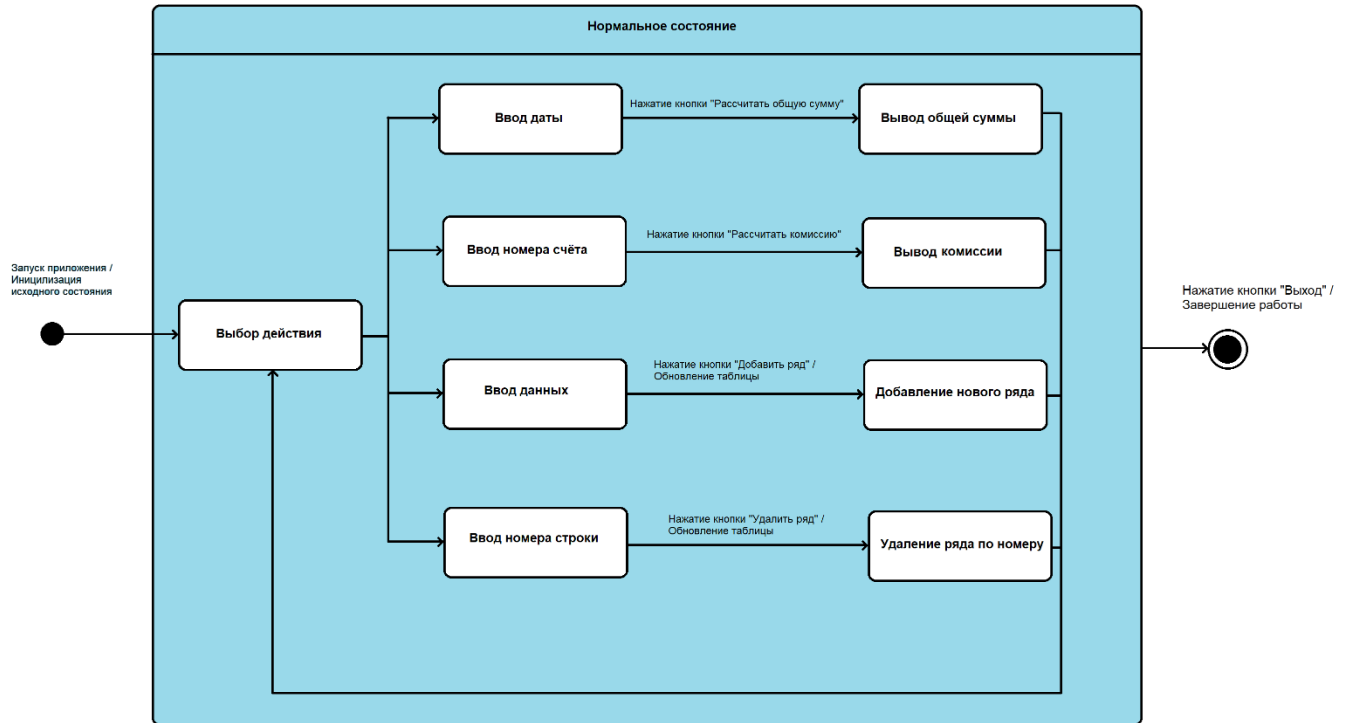


Рисунок 18 – Диаграмма состояний интерфейса

3.4 Разработка диаграммы классов интерфейсной и предметной областей приложения

Диаграмма классов интерфейсной и предметной областей показывает связи между классами (рисунок 19).

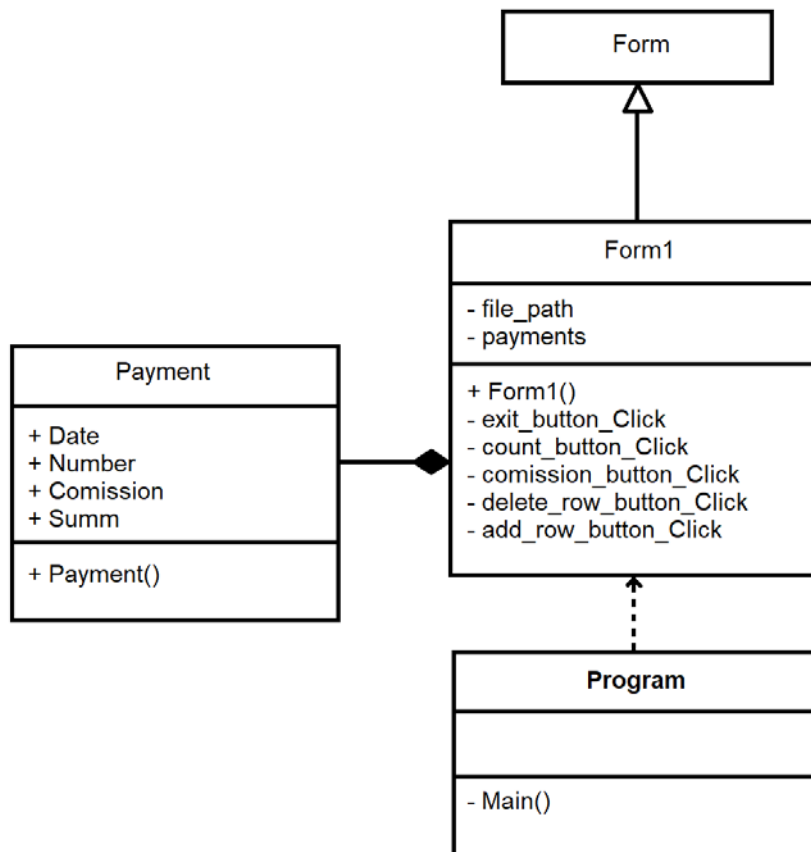


Рисунок 19 – Диаграмма классов приложения

3.5 Разработка диаграммы последовательности действий выполнения операции загрузки данных из файла

Диаграмма последовательностей действий позволяет уточнить порядок выполнения фрагментов операции различными объектами (рисунок 20).

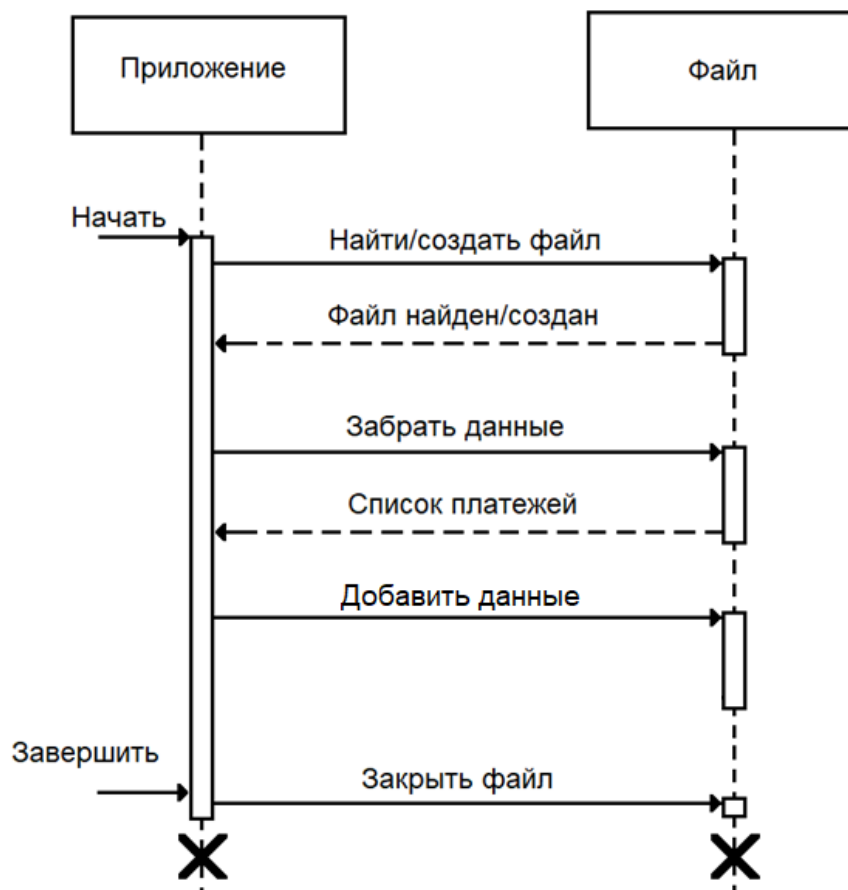


Рисунок 20 – Диаграмма последовательностей действий выполнения операции загрузки данных из файла

1.6 Разработка кода приложения

Уточнение поведения объектов во время выполнения программы позволило разработать код программы.

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Web;
using System.Windows.Forms;

namespace PTP_4
{
    public partial class Form1 : Form
  
```

```

{
    string file_path = "C:/Users/Администратор/source/repos/UTP_4/database.txt";
    List<Payment> payments = new List<Payment>();
    public Form1()
    {
        InitializeComponent();

        if (File.Exists(file_path))
        {
            string[] all_lines = File.ReadAllLines(file_path);
            foreach (string line in all_lines)
            {
                string[] parts = line.Split(' ');
                Payment payment = new Payment(parts[0], parts[1],
double.Parse(parts[2]), double.Parse(parts[3]));

                payments.Add(payment);
            }
        }
        else
        {
            File.Create(file_path);
        }
        dataGridView1.DataSource = payments;
        dataGridView1.Columns[0].Width = 150;
        dataGridView1.Columns[1].Width = 200;
        dataGridView1.Columns[2].Width = 150;
    }

    private void exit_button_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void count_button_Click(object sender, EventArgs e)
    {
        double summ = 0;
        if (date_input.Text == "")
        {
            summ_output.Text = "Введите дату";
        }
        else
        {
            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
                if (Convert.ToString(dataGridView1.Rows[i].Cells[0].Value) ==
date_input.Text)
                {
                    summ += Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value)
* (1 - Convert.ToDouble(dataGridView1.Rows[i].Cells[3].Value) / 100);
                }
            }
            summ_output.Text = summ.ToString();
        }
    }

    private void comission_button_Click(object sender, EventArgs e)
    {
        double summary_comission = 0;
        if (number_input.Text == "")
        {
            comission_output.Text = "Введите Number";
        }
    }
}

```

```

else
{
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        if (number_input.Text ==
Convert.ToString(dataGridView1.Rows[i].Cells[1].Value))
        {
            summary_comission +=
Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value) *
(Convert.ToDouble(dataGridView1.Rows[i].Cells[3].Value) / 100);
        }
    }
    comission_output.Text = summary_comission.ToString();
}

private void delete_row_button_Click(object sender, EventArgs e)
{
    List<Payment> payments2 = new List<Payment>();
    string NewFile = "";
    string[] all_lines = File.ReadAllLines(file_path);
    int deleted = 0;
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        if (i != Convert.ToInt32(delete_row_number.Text) - 1 && i ==
dataGridView1.Rows.Count - 1)
        {
            NewFile += all_lines[i];
            deleted++;
        }
        else if (i != Convert.ToInt32(delete_row_number.Text) - 1)
        {
            NewFile += all_lines[i] + '\n';
            deleted++;
        }
    }
    File.WriteAllText(file_path, NewFile);
    delete_row_number.Clear();
    if (deleted == dataGridView1.Rows.Count)
    {
        delete_row_number.Text = "Не найдено";
    }
    string[] all_lines2 = File.ReadAllLines(file_path);
    foreach (string line in all_lines2)
    {
        string[] parts = line.Split(' ');
        Payment payment = new Payment(parts[0], parts[1],
double.Parse(parts[2]), double.Parse(parts[3]));

        payments2.Add(payment);
    }
    dataGridView1.DataSource = payments2;
}

private void add_row_button_Click(object sender, EventArgs e)
{
    if (new_row_input.Text == "")
    {
        new_row_input.Text = "Введите строку";
    }
    else
    {
        List<Payment> payments2 = new List<Payment>();
        string[] row = new_row_input.Text.Split(' ');
        File.AppendAllText(file_path, '\n' + new_row_input.Text);
    }
}

```

```

        string[] all_lines2 = File.ReadAllLines(file_path);
        foreach (string line in all_lines2)
        {
            string[] parts = line.Split(' ');
            Payment payment = new Payment(parts[0], parts[1],
double.Parse(parts[2]), double.Parse(parts[3]));
            payments2.Add(payment);
        }
        dataGridView1.DataSource = payments2;
        new_row_input.Clear();
    }

}

}

public class Payment
{
    public string Date { get; set; }
    public string Number { get; set; }
    public double Summ { get; set; }
    public double Comission { get; set; }
    public Payment() { }
    public Payment(string date, string accountNumber, double amount, double
comission)
    {
        Date = date;
        Number = accountNumber;
        Summ = amount;
        Comission = comission;
    }
}

}

```

1.7 Тестирование приложения

Разработанное приложение позволяет выполнять функции, предусмотренные заданием (рисунки 21-22) .

The screenshot shows the 'Переводы' (Transfers) application window. It features a table with four columns: Date, Number, Summ, and Comission. The first row is highlighted in blue. To the right of the table is a sidebar with various controls for calculations and data management.

Date	Number	Summ	Comission
01.01.22	12345678901	5000	50
01.01.22	23456789012	6000	60
02.01.22	12345678901	7000	70
03.01.22	45678901234	8000	80
03.01.22	12345678901	9000	90
04.01.22	67890123456	1000	10
05.01.22	78901234567	2000	20
05.01.22	89012345678	3000	30
08.01.22	90123456789	4000	40
10.01.22	12345678901	9500	95
12.01.22	12345678901	8500	85
12.01.22	23456789012	7500	75
12.01.22	34567890123	6500	65
15.01.22	45678901234	5500	55
15.01.22	56789012345	4500	45
16.01.22	67890123456	3500	35
17.01.22	78901234567	2500	25
18.01.22	89012345678	1500	15
19.01.22	90123456789	5000	50

Controls on the right sidebar:

- Дата: 12.01.22
- Расчитать сумму (button)
- Сумма: 5425
- Номер счёта: 12345678901
- Расчитать комиссию (button)
- 31750
- Добавить ряд (button)
- Номер ряда:
- Удалить ряд (button)
- Выход (button)

Рисунок 21 – Главное окно приложения, в котором произведены расчёт суммы и комиссии

This screenshot shows the same application window after a new row has been added to the table. The sidebar controls are also updated to reflect the new data.

Date	Number	Summ	Comission
03.01.22	45678901234	8000	80
03.01.22	12345678901	9000	90
04.01.22	67890123456	1000	10
05.01.22	78901234567	2000	20
05.01.22	89012345678	3000	30
08.01.22	90123456789	4000	40
10.01.22	12345678901	9500	95
12.01.22	12345678901	8500	85
12.01.22	23456789012	7500	75
12.01.22	34567890123	6500	65
15.01.22	45678901234	5500	55
15.01.22	56789012345	4500	45
16.01.22	67890123456	3500	35
17.01.22	78901234567	2500	25
18.01.22	89012345678	1500	15
18.01.22	90123456789	5000	50
20.01.22	01234567890	7000	70
21.02.22	0123456789	6000	10

Updated controls on the right sidebar:

- Дата: 12.01.22
- Расчитать сумму (button)
- Сумма: 5425
- Номер счёта: 12345678901
- Расчитать комиссию (button)
- 31750
- 22.02.22 0123456789 2000 30 (text input)
- Добавить ряд (button)
- Номер ряда:
- Удалить ряд (button)
- Выход (button)

Рисунок 22 – Добавление ряда таблицы

Переводы

Date	Number	Summ	Comission
03.01.22	12345678901	9000	90
04.01.22	67890123456	1000	10
05.01.22	78901234567	2000	20
05.01.22	89012345678	3000	30
08.01.22	90123456789	4000	40
10.01.22	12345678901	9500	95
12.01.22	12345678901	8500	85
12.01.22	23456789012	7500	75
12.01.22	34567890123	6500	65
15.01.22	45678901234	5500	55
15.01.22	56789012345	4500	45
16.01.22	67890123456	3500	35
17.01.22	78901234567	2500	25
18.01.22	89012345678	1500	15
18.01.22	90123456789	5000	50
20.01.22	01234567890	7000	70
21.02.22	0123456789	6000	10
22.02.22	0123456789	2000	30

Дата: 12.01.22

Расчитать сумму

Сумма: 5425

Номер счёта: 12345678901

Расчитать комиссию

31750

Добавить ряд

Номер ряда: 22

Удалить ряд

Выход

Рисунок 23 – Удаление ряда таблицы

Вывод

При разработке приложения изучены средства разработки приложений с графическим интерфейсом на языке C#, используемые при объектном подходе, а также основные диаграммы, сопровождающие процесс разработки.

ЗАКЛЮЧЕНИЕ

Во время выполнения учебной практики более глубоко были изучены средства реализации проектов программ на C++ и C#, получены практические навыки проектирования небольших программных систем при объектном подходе с графическими интерфейсами.

Кроме этого были получены начальные навыки составления программной документации.

СПИСОК ЛИТЕРАТУРЫ

1. Иванова Г.С., Ничушкина Т.Н. Объектно-ориентированное программирование: учебник. – М.: Издательство МГТУ им. Н.Э. Баумана, 2014.
2. Самарев Р.С. Программирование с использованием библиотеки Qt: Методические указания по выполнению лабораторной работы. М.: Издательство МГТУ им. Н.Э. Баумана, 2011. (<http://e-learning.bmstu.ru/moodle/course/view.php?id=129>).
3. Шлее М. Qt4. 5. Профессиональное программирование на C++ / Шлее М. - СПб. : БХВ-Петербург, 2018. - 884 с. + CD. - (В подлиннике). - ISBN 978-5-9775-0398-3.
4. Шилдт Г. Полный справочник по C++, 4 изд. М.: Изд. дом «Вильямс», 2015.