МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №42

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | Д.О. Шевяков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| ПРИЛОЖЕНИЕ ДЛЯ УЧЁТА ГЕОГРАФИЧЕСКИХ СУБЪЕКТОВ |
| по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Оглавление**

[ВВЕДЕНИЕ 4](#_Toc122619991)

[АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ 5](#_Toc122619992)

[диаграмма классов 7](#_Toc122619993)

[описание программного кода классов 8](#_Toc122619994)

[тестирование функциональности классов 13](#_Toc122619995)

[руководство пользователя 21](#_Toc122619996)

[ЗАКЛЮЧЕНИЕ 22](#_Toc122619997)

[Библиографический список 23](#_Toc122619998)

[приложение 24](#_Toc122619999)

[Код классов: 24](#_Toc122620000)

[Код формы: 26](#_Toc122620001)

[Код дизайна формы: 28](#_Toc122620002)

# 

# ВВЕДЕНИЕ

Цель курсового проектирования как учебной дисциплины: использование полученных знаний и навыков объектно-ориентированного программирования для разработки пользовательской библиотеки классов, предназначенной для программирования приложений пользователя в конкретной предметной области. Библиотека должна учитывать особенности области приложения и располагать необходимой функциональностью для построения интерфейса пользователя, структурирования, записи, чтения данных в различных форматах, обработки, вычисления и визуализации данных. Курсовая работа позволяет:

1. систематизировать теоретические знания и закрепить полученные практические умения по дисциплине «Основы программирования» в соответствии с требованиями к уровню подготовки по направлению 09.03.02 «Информационные системы и технологии»;
2. сформировать умения работы с учебной литературой и иными информационными источниками;
3. развить профессиональную письменную и устную речь;
4. развить системное мышление, творческую инициативу, самостоятельность, организованность и ответственность за принимаемые решения;
5. сформировать навыки планомерной регулярной работы над решением поставленных задач.

# АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

Вариант задания: в приложении пользователь может создать объект класса Государство, используя классы Область, Район, Город. Реализовать методы чтения, записи необходимой информации о государстве, обеспечить поиск областей, районов, городов с соответствующей информацией о каждом объекте. Возможность получить у субъекта такую информацию как: id; название; содержимые субъекты; год создания государства; возраст государства. При наличии в государстве правителя возможность узнать его: ФИО; возраст. Предусмотреть свойства для получения состояния объекта. Обеспечить следующие возможности: вывод информации о государстве по указанному id; вывод информации о областях, в государстве; вывод информации о городах в области; вывод информации о районах в городе; вывод информации о школе в районе.

От архитектуры классов требуется:

1. Возможность создания модели данных государства.
2. Возможность создания списка областей, городов, районов.
3. В классах должна быть представлена связь между субъектами побольше и поменьше.
4. Возможность изменения коллекции:
   1. Добавление внутренних субъектов
   2. Изменение информации в государстве.
5. Возможность получения полей/свойств, характеризующих государство.
6. Возможность установки структуры правителя государства.

От программного кода библиотеки требуется:

1. Программный код должен содержать минимум 4 класса, включая абстрактный(реализация инкапсуляции).
2. Реализация наследования.
3. Реализация полиморфизма: перегруженные и переопределенные методы и свойства классов.
4. Данные класса должны быть описаны полями и свойствами.
5. Наличие развернутых комментариев к программному коду.
6. Реализация событий классов.
7. Реализация отношений ассоциации, композиции, агрегации.
8. Использование интерфейсов.
9. Программный код должен выполнять поставленную задачу для библиотеки.

Также требуется на основе спроектированной библиотеки построить интерфейс и протестировать его. На основе интерфейса показать взаимодействия пользователя с написанной нами библиотекой.

Полученная библиотека должна использоваться для получения информации о географических субъектах. С помощью библиотеки пользователь должен с легкостью контролировать процесс создания субъектов, иметь доступный функционал изменения субъектов, а также иметь возможность синхронизировать информацию одного типа субъекта с другими.

диаграмма классов

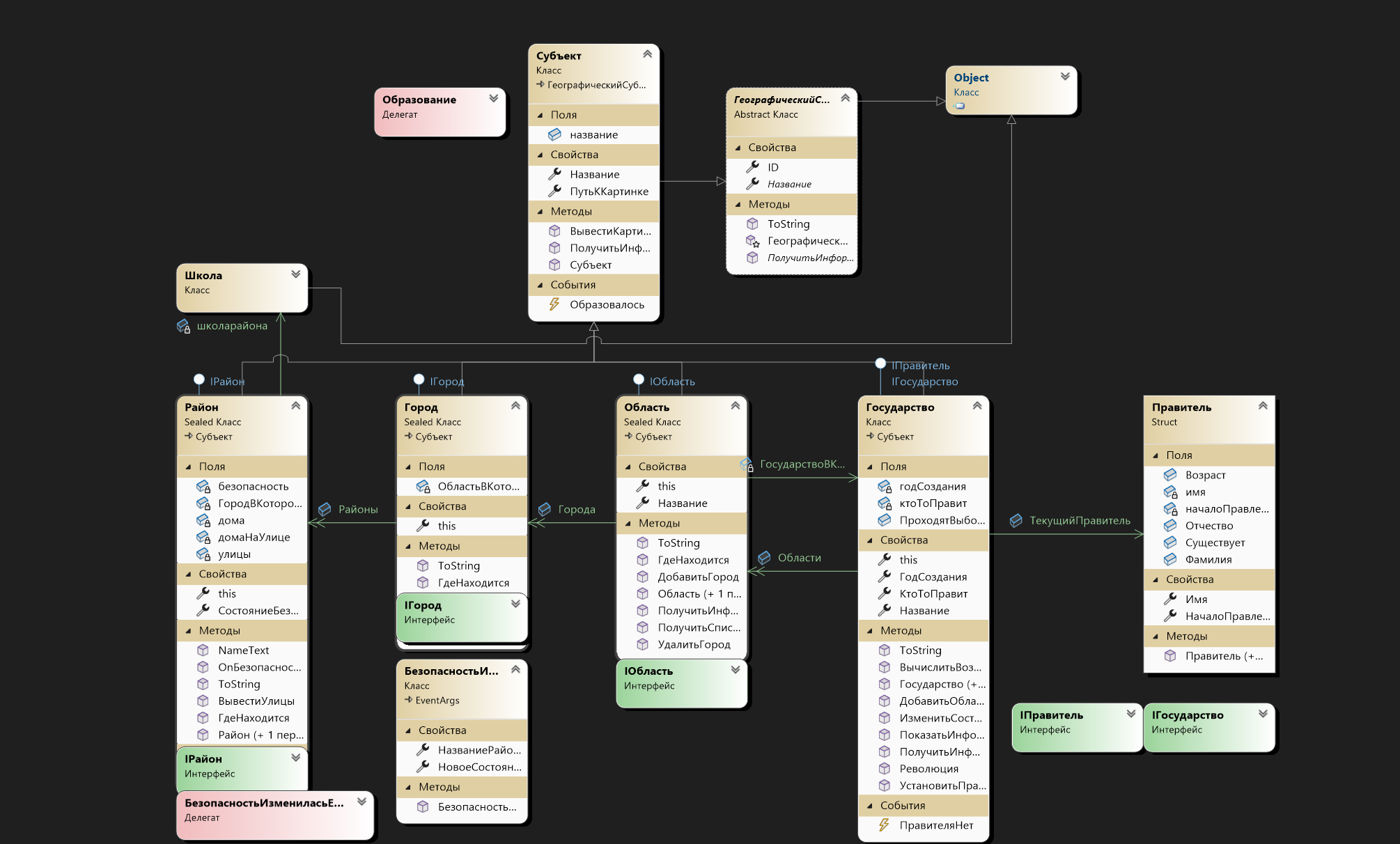


Рисунок 1.1 – диаграмма классов

На рисунке 1 видно, что:

1. Класс Субъект наследует абстрактный класс ГеографическийСубъект и реализует все требуемые поля и свойства.
2. Классы Государство, Область, Город, Район - наследники класса Субъект.
3. Классы Государство, Область, Город, Район реализуют интерфейсы IГосударство, IОбласть, IГород, IРайон, соответственно.
4. Класс Государство реализует интерфейс IПравитель.
5. Класс Школа связан с классом Район посредством композиции. Класс Район содержит в себе объект класса Школа и создаёт его в конструкторе.
6. Класс Район связан с классом Город посредством агрегации. В конструктор объекта класса Район передаётся ссылка на существующий объект класса Город.
7. Класс Город связан с классом Область посредством агрегации.
8. Класс Область связан с классом Государство посредством агрегации.
9. Структура Правитель связана с классом Государство посредством ассоциации.
10. Класс Субъект содержит событие Образовалось с делегатом.
11. Класс Район содержит событие БезопасностьИзменилась с делегатом.
12. Класс Государство содержит событие ПравителяНет без делегата.

описание программного кода классов

Ниже представлен класс Государство, который является страной. Класс Государство содержит свойства: Название(всегда делает первую букву заглавной), ГодСоздания(если пользователь его не ввёл, но передал в конструктор, то устанавливается текущая дата), КтоТоПравит(наличие Правителя); коллекцию объектов класса Область (демонстрируется ассоциация); а также индексатор для доступа к областям государства по индексу, так реализован просмотр коллекции Области.



Рисунок 1.2 – код класса Государство (1)

Конструктор принимает на вход id субъекта, название. Также продемонстрирован полиморфизм - присутствует перегруженный конструктор, который принимает на вход id, название, год создания, а также содержит необязательный аргумент ПроходятВыборы. В конструкторах создаётся коллекция с объектами класса Область, что демонстрирует наличие композиции. Помимо этого в классе Государство есть событие ПравителяНет, которое вызывается при использовании метода Революция – правитель государства сбрасывается. Вызывается уведомление о произошедшем и запускается другой метод, который меняет свойство СостояниеБезопасности у объектов класса Район, которые путём связи Государство-Область-Город-Район, существуют в государстве. Это вызывает событие в классе Район – БезопасностьИзменилась. Таким образом, в библиотеке реализована цепочка событий.

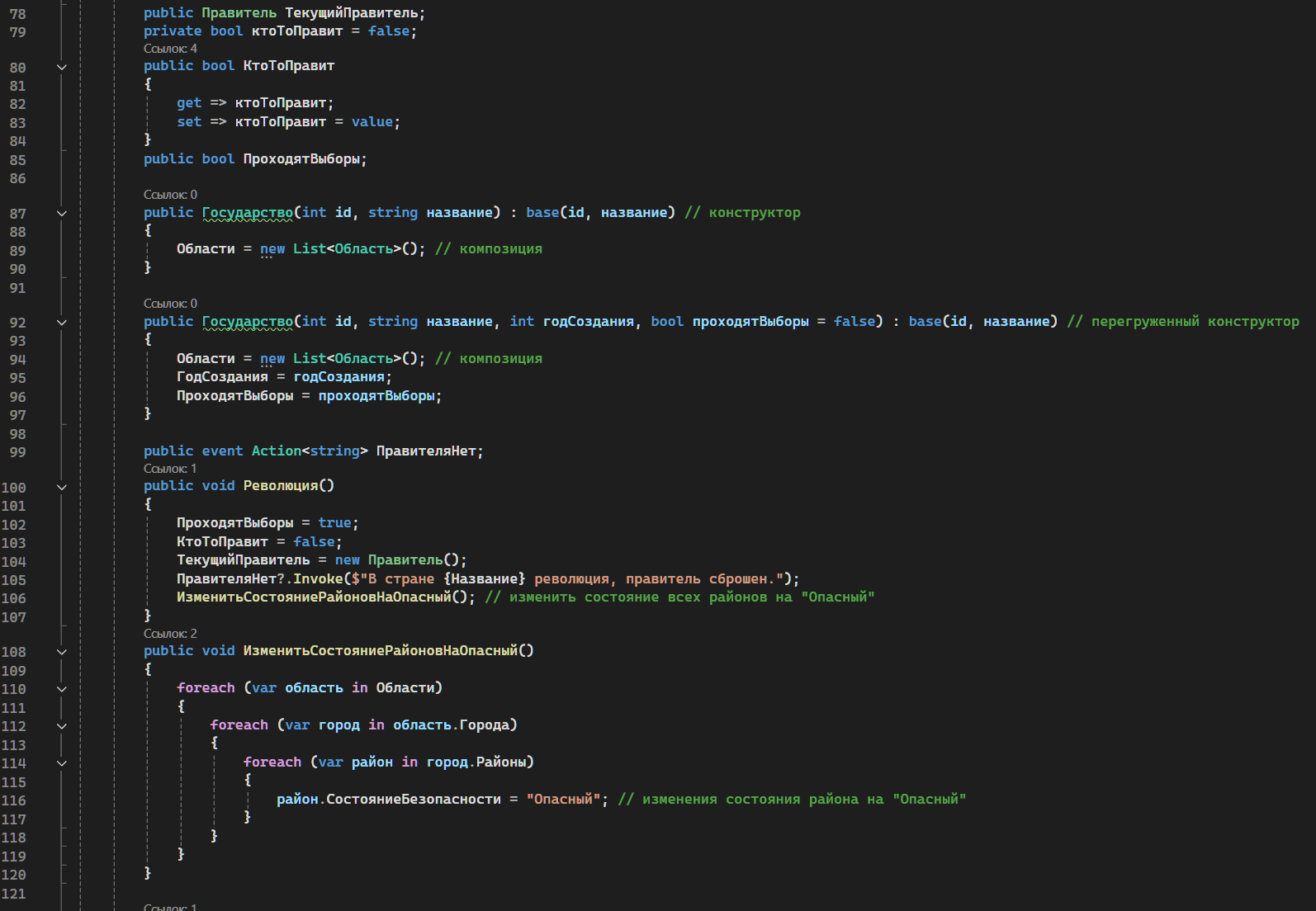


Рисунок 1.3 – код класса Государство (2)

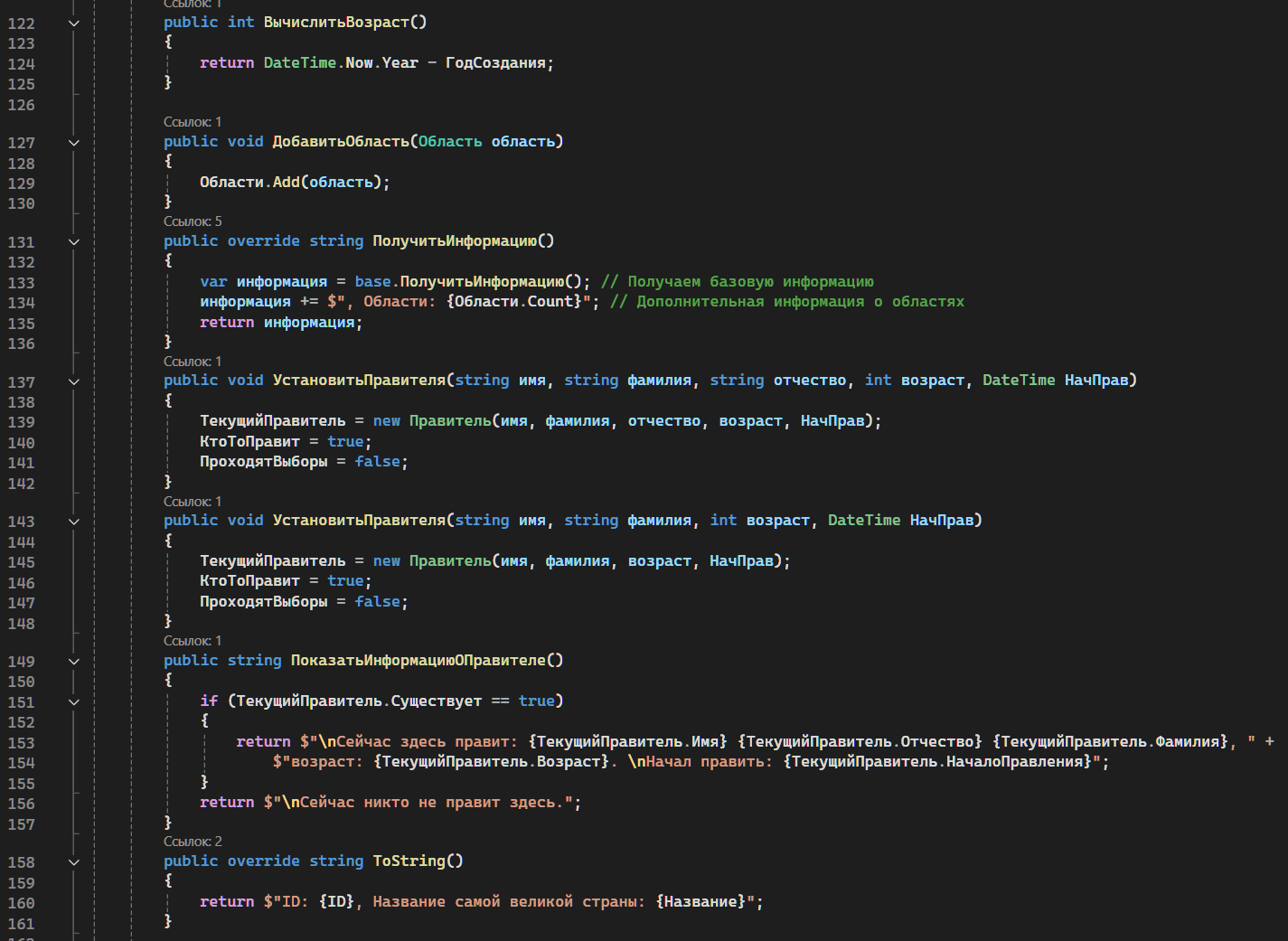
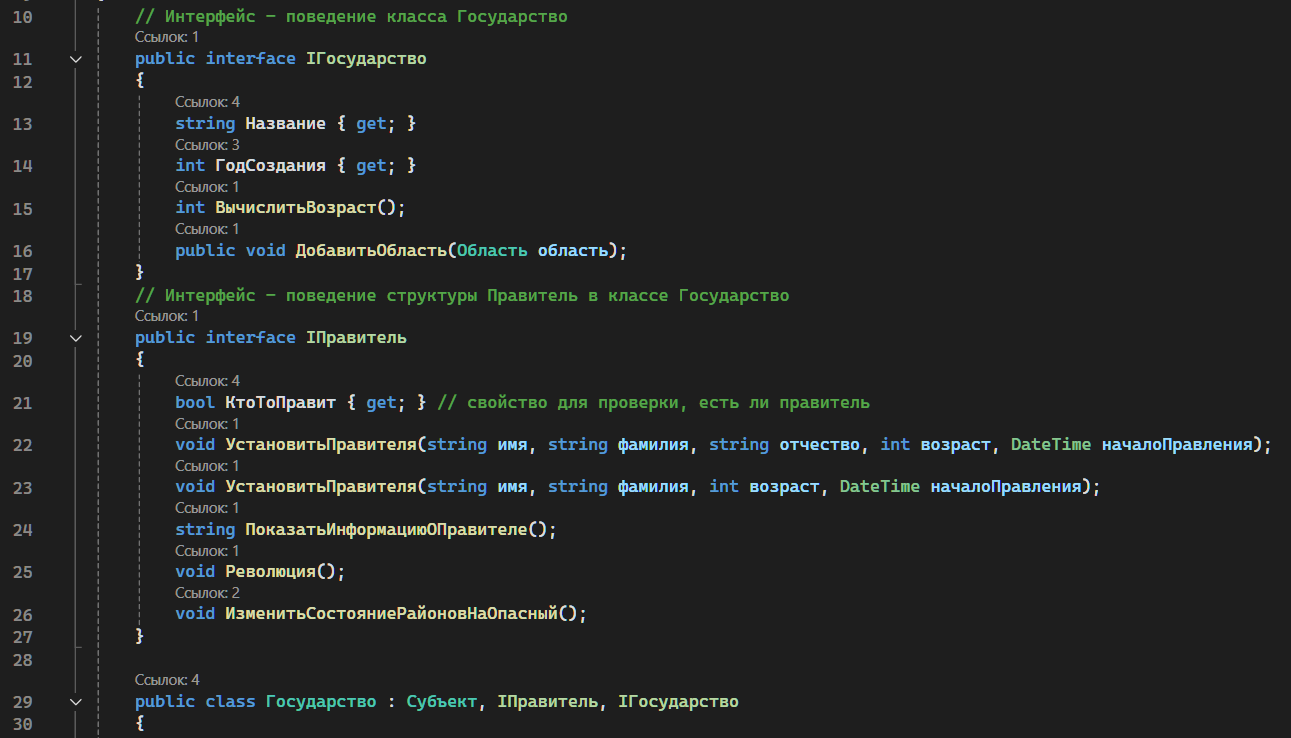


Рисунок 1.4 – код класса Государство (3)

На рис 4 представлены методы класса Государство – ВычислитьВозраст, ДобавитьОбласть, ПолучитьИнформацию(продемонстрирован полиморфизм – метод в классе родителе Субъект virtual, а здесь override - переопределён), УстановитьПравителя и его перегруженная версия(отсутствие/антиотсутствие отчества у правителя), ПоказатьИнформациюОПравителе, а также переопределённый метод ToString.

Рисунок 1.5 – интерфейсы класса Государство

На рис 5 представлен код интерфейсов IГосударство и IПравитель, которые не только демонстрируют возможность реализации множественного наследования, но и описывают поведение класса Государство и структуры Правитель в этом классе. Описанная функциональность: вычисление возраста государства, добавление области в государство; установка правителя, отображение информации о правителе, сброс правителя, изменение состояния районов городов в государстве на «опасный».

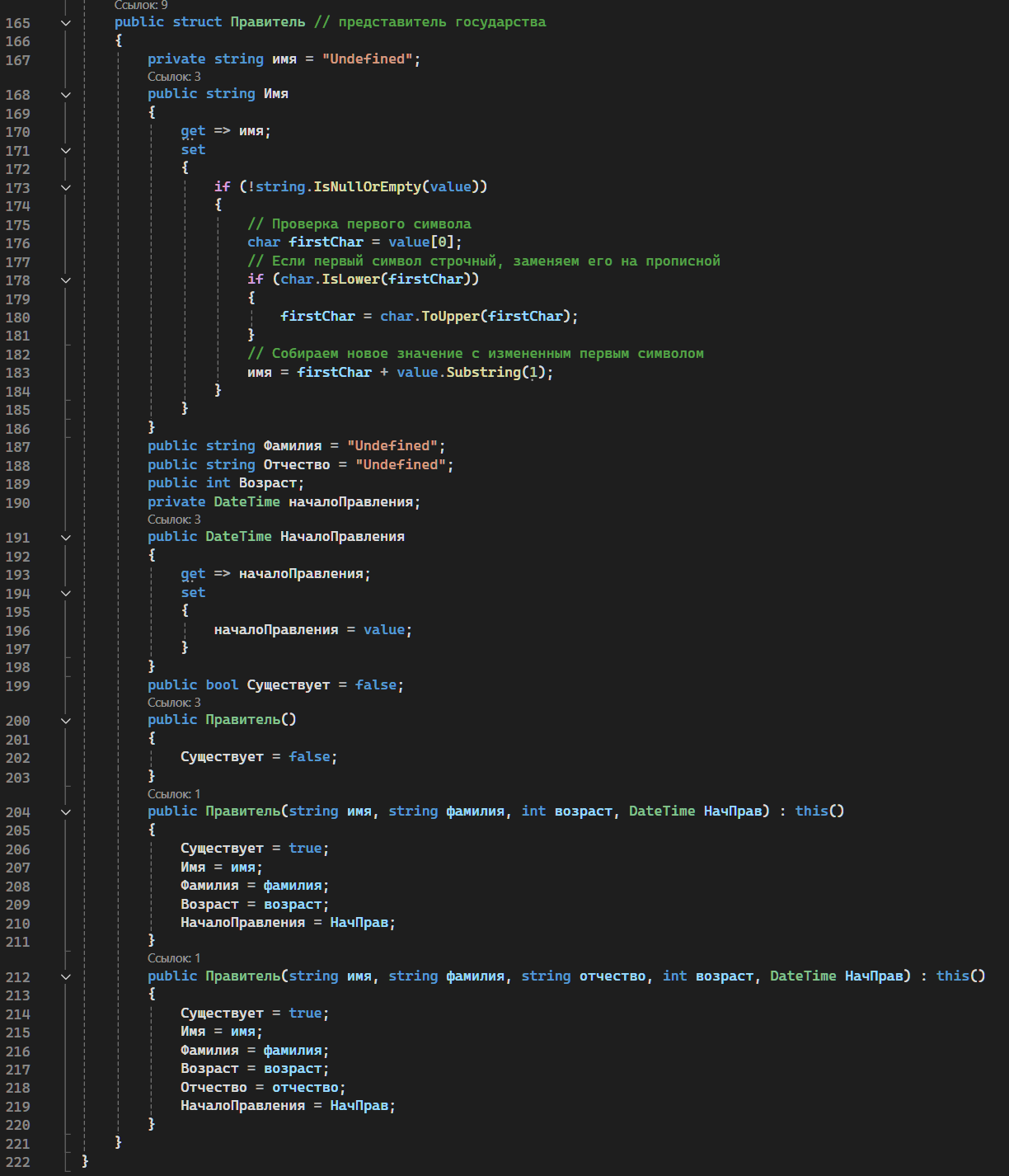


Рисунок 1.6 – код структуры Правитель

Для описания правителя государства была создана структура Правитель, с которой класс Государство связан посредством ассоциации. Установка правителя в государстве реализована методом УстановитьПравителя, описанным выше. Структура содержит поля: фамилия, отчество, возраст, флаг существования; свойства: имя(всегда с заглавной буквы), начало правления(обычно берётся текущая дата); 3 перегруженный конструктора – на случай сброса правителя, на случай установки правителя с отсутствием отчества, на случай установки правителя с антиотсутствием отчества.

Переходим к следующему классу Область. На рис 7 показан интерфейс IОбласть, который описывает поведение класса Область, методы: добавить город, удалить город, получить список городов.

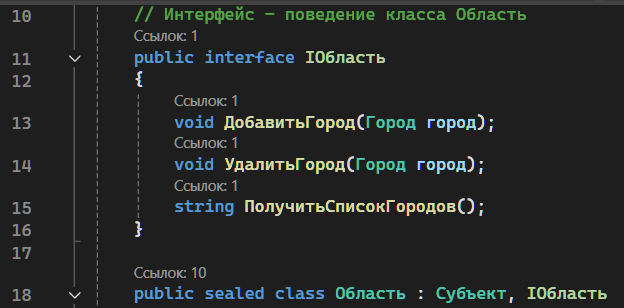


Рисунок 1.7 – код интерфейса IОбласть

Ниже, на рис 8, мы можем наблюдать код класса Область и реализацию заявленной функциональности. Класс Область содержит: коллекцию объектов класса Город, что демонстрирует наличие связи между классами Область и Город – ассоциацию; индексатор для доступа к городам по индексу; свойства: название(всегда добавляет к выводу «обл.»); 2 конструктора – базу, на случай создания области без приписки к государству и основной – с реализацией связи композиция и агрегация. На вход поступает id, название, ссылка на существующий объект класса Государство. Внутри создаётся коллекция городов, которые находятся в созданной области – композиция. Также происходит приписка к государству, в котором находится область – агрегация. Для наглядной демонстрации написан метод ГдеНаходится, который отображает приписку к государству. Представлены также методы, требуемые интерфейсом: добавить город, удалить город, которые изменяют коллекцию городов в области; получить список городов, который выводит коллекцию имеющихся городов. А также переопределённый метод ПолучитьИнформацию, который выводит информацию об области и демонстрирует применение полиморфизма – здесь он override, в классе-родителе Субъект, он virtual. Дополнительно представлен переопределённый метод ToString, чтобы дать несколько вариантов вывода информации.

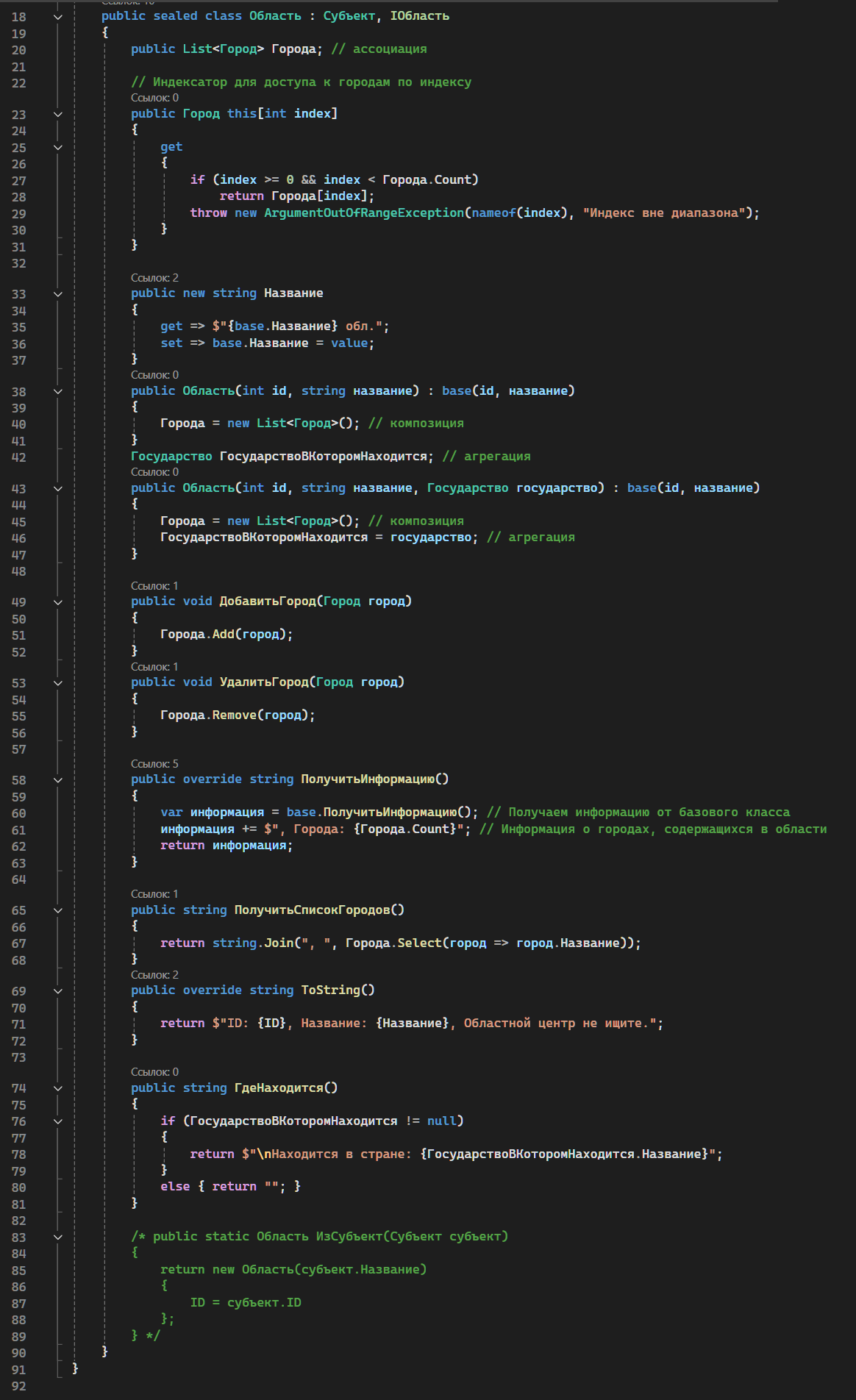


Рисунок 1.8 – код класса Область

Перейдём к классу Город. На представленном рис 9 мы можем наблюдать код класса Город и интерфейс IГород, описывающий его заявленную функциональность: метод ДобавитьРайон, ниже коде класса мы можем заметить, что метод реализован – функции передаётся ссылка на объект класса Район, который затем добавляется в коллекцию Районы, заданному городу.

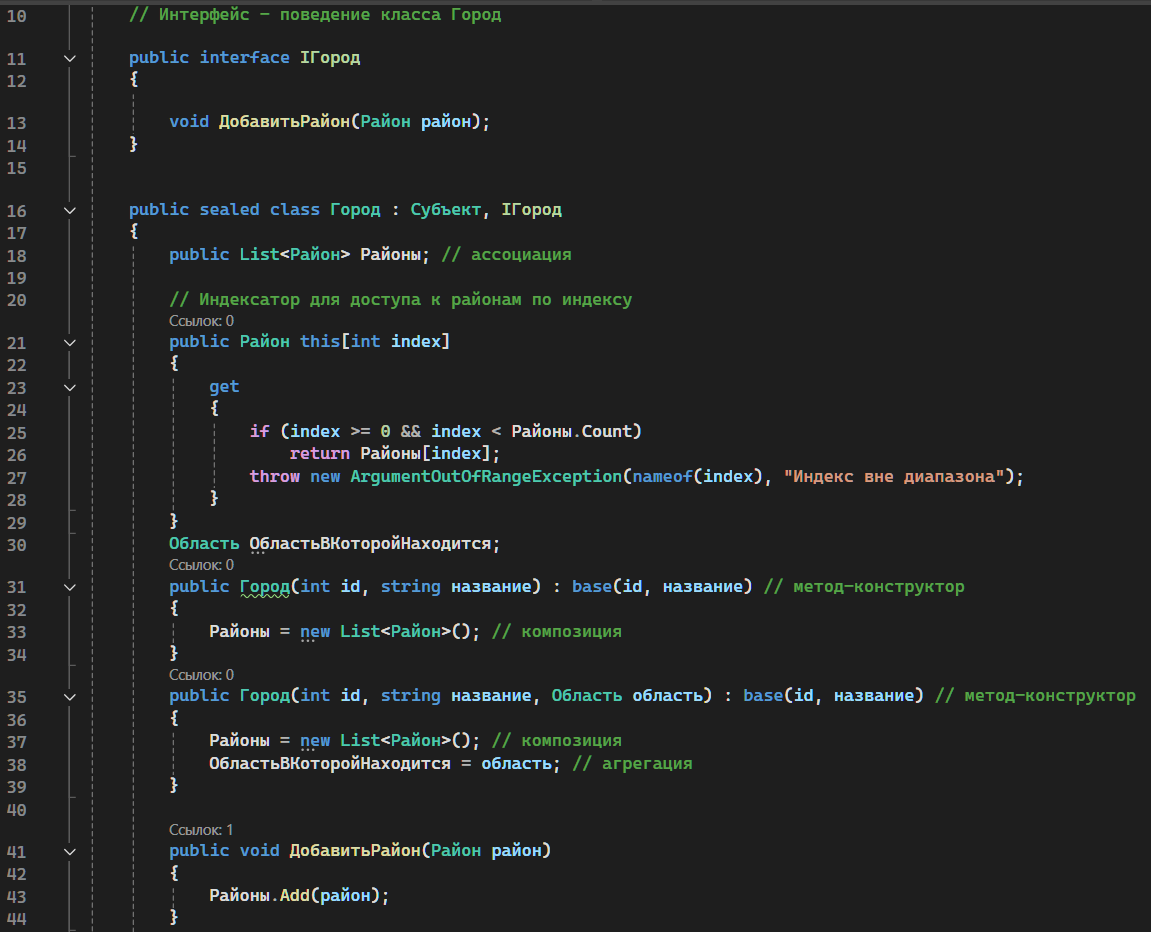


Рисунок 1.9 – код класса Город (1)

Помимо прочего класс Город содержит: коллекцию объектов класса Район – реализуется отношение ассоциации между классами; индексатор для доступа к районам(объектам в коллекции) по индексу; поле ОбластьВКоторойНаходится – объект класса область; базовый конструктор(создаёт отдельный от всего город с id и названием); основной конструктор, которой принимает id, название и ссылку на объект класса Область, а затем создаёт коллекцию объектов класса Район, реализуя композицию, и демонстрирует агрегацию, записывая переданную область в поле ОбластьВКоторойНаходится.

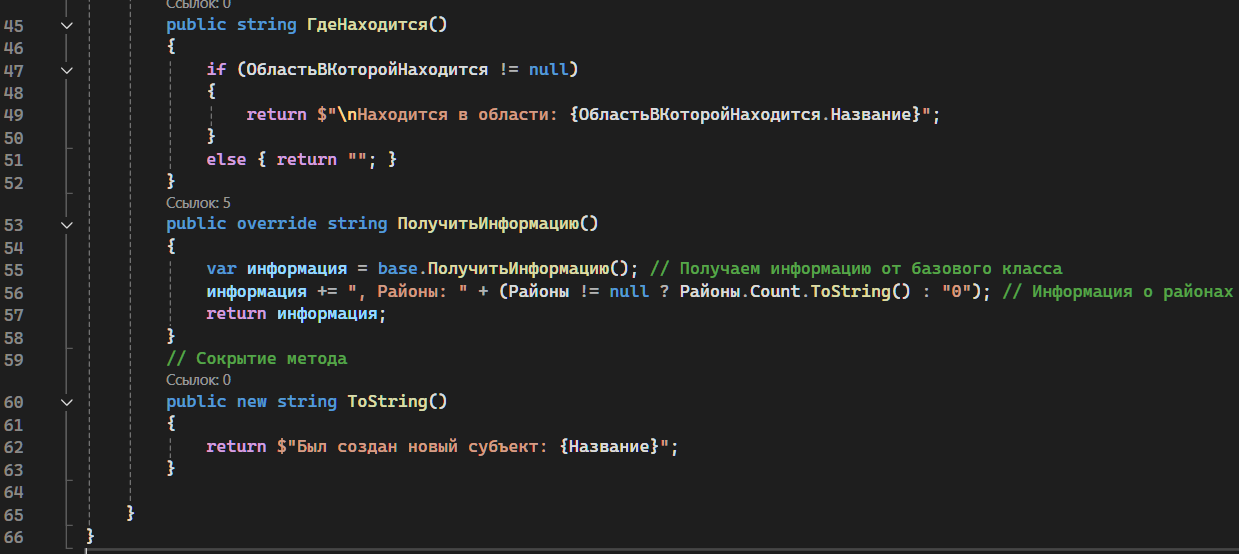


Рисунок 1.10 – код класса Город(2)

Далее показаны методы, возвращающие различную информацию об объекте класса. Метод ГдеНаходится сообщает об области, в которой находится город. Метод ToString демонстрирует сокрытие унаследованного метода, выводит информацию о создании субъекта(наглядно продемонстрировано в приложенной программе с пользовательским интерфейсом). Переопределённый метод ПолучитьИнформацию демонстрирует полиморфизм, выводит помимо базовой информации, информацию о районах, которые содержит в себе город.

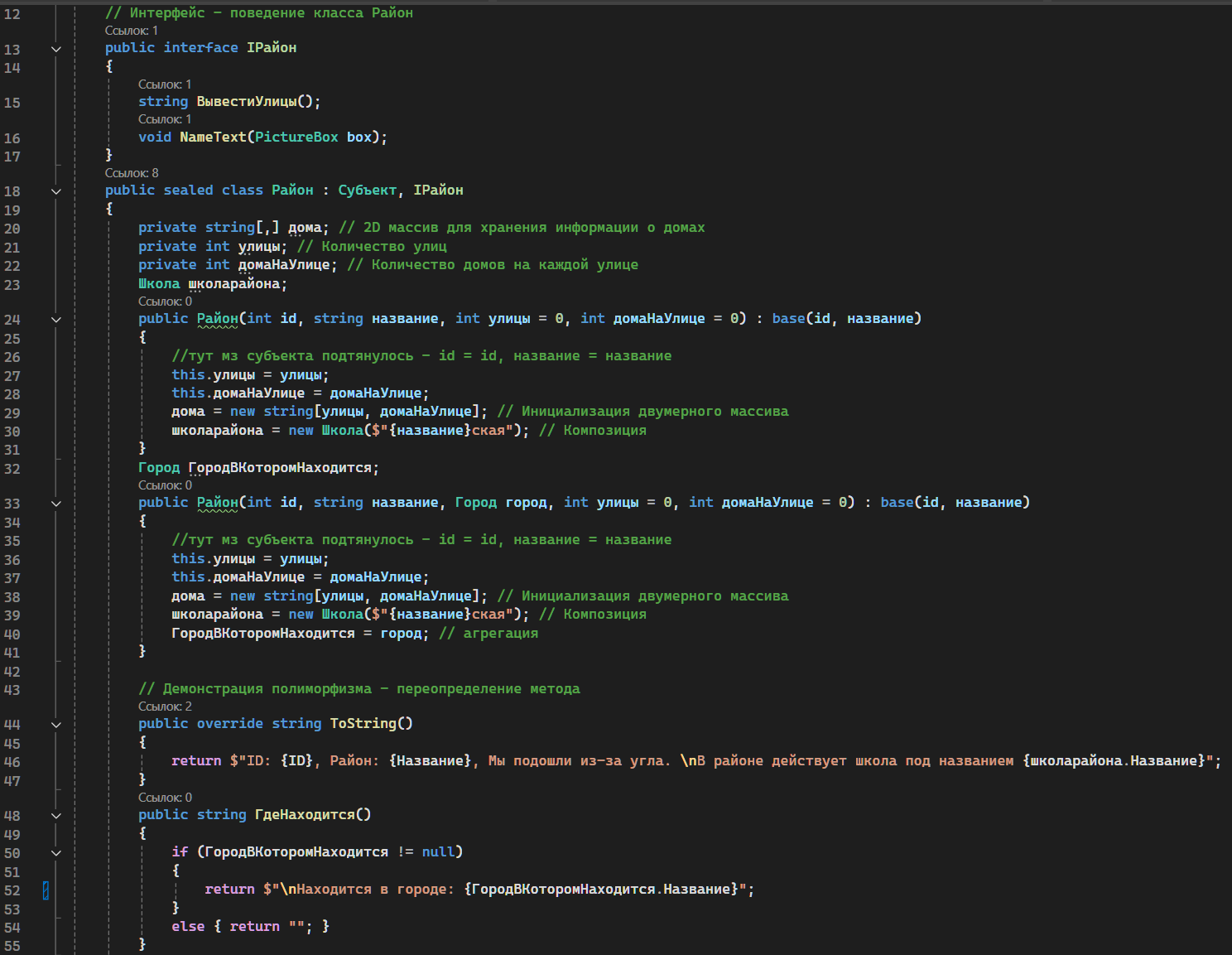


Рисунок 1.11 – код класса Район (1)

На рисунке выше – код класса Район, с интерфейсом IРайон, который заявляет функционал класса: методы ВывестиУлицы и NameText. Далее в коде мы можем наблюдать поля: двумерный массив – дома; улицы; домаНаУлице – так устроена реализация двумерного индексатора, который представлен на рисунке 12, ниже. Он перебирает улицы и дома на них, чтобы в дальнейшем вывести данную информацию.

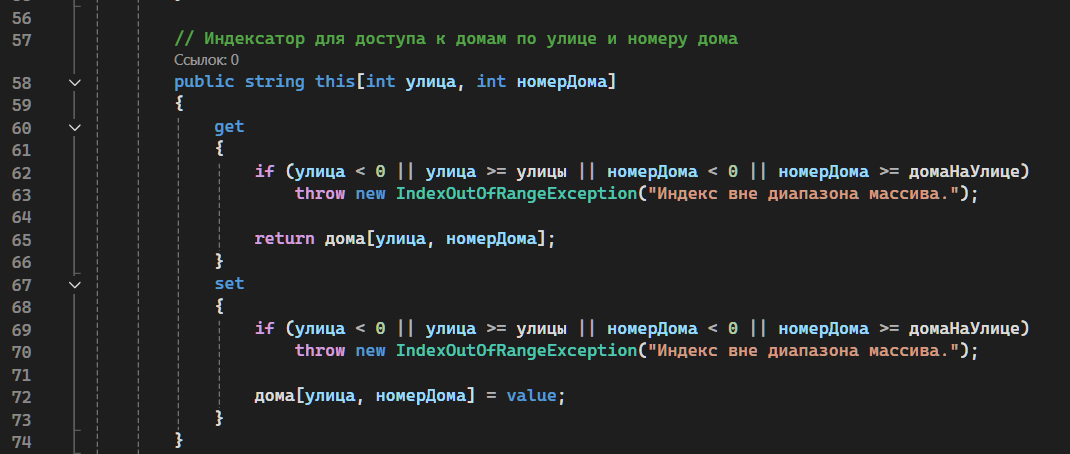


Рисунок 1.12 – код индексатора в классе Район

Также в классе Район находится 2 конструктора: базовый и основной. Отличие состоит в том, что основной конструктор реализует отношение агрегации, в него передаётся ссылка на объект класса Город, в котором находится район, а затем в поле ГородВКоторомНаходится вписывается полученный город. Однако, оба конструктора принимают на вход id, название, необязательные аргументы - количество улиц и домов на улице; создают объект класса Школа, демонстрируя тем самым отношения композиции.

Далее в коде мы можем увидеть переопределённый метод ToString, и метод ГдеНаходится, который возвращает информацию о том, в каком городе находится район.

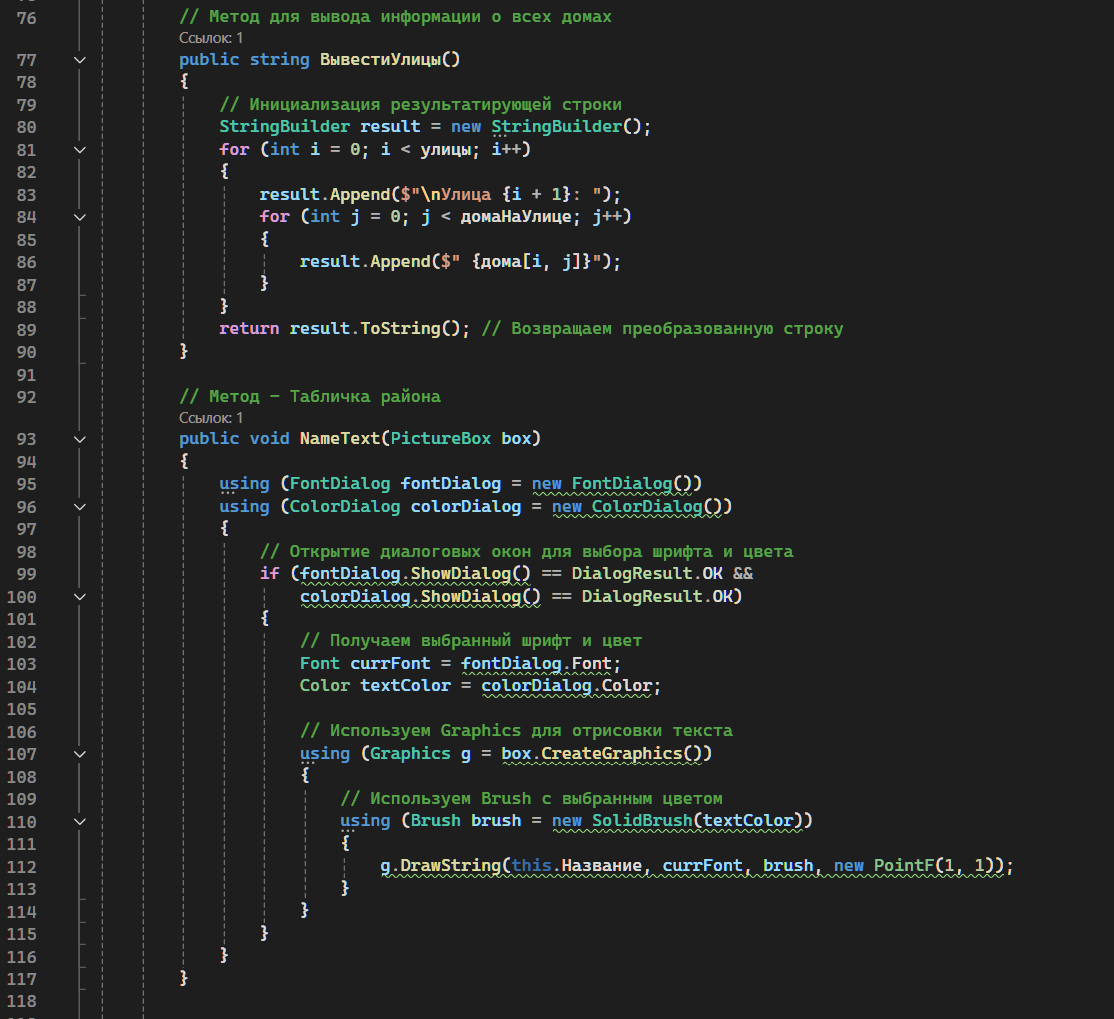


Рисунок 1.13 – код класса Район (2)

На представленном рисунке 13, цветным по чёрному написаны методы ВывестиУлицы и NameText, заявленные в интерфейсе. Первый возвращает символьную строку с информацией о перечислении улиц и домов на ней. Второй реализует отрисовку таблички района – пользователь выбирает шрифт и цвет текста, выводится картинка с кастомизированным названием района.



Рисунок 1.14 – код события БезопасностьИзменилась класса Район

Класс Район также имеет информацию о состоянии безопасности в районе. Для этого созданы поле безопасность и свойство СостояниеБезопасности. А также событие, которые реагируют на изменения безопасности, для упрощения дальнейшего применения библиотеки присутствует делегат, обрабатывающий данное событие. В данном случае, как мы можем помнить, есть возможность вызвать цепочку событий, в случае сброса правителя в государстве (см. начало раздела).

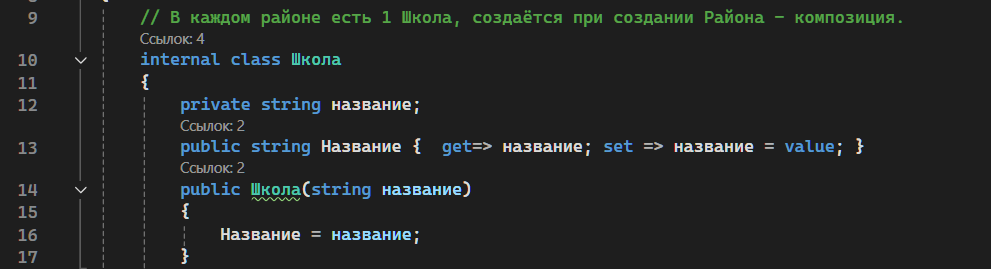


Рисунок 1.15 – код класса Школа

Класс «Школа» хранит название в поле «название», оно поступает на вход в конструктор. Это нужно для уникальности школ и определения их районов. Класс связан с классом «Район» отношением композиции, то есть у каждого района есть своя школа.

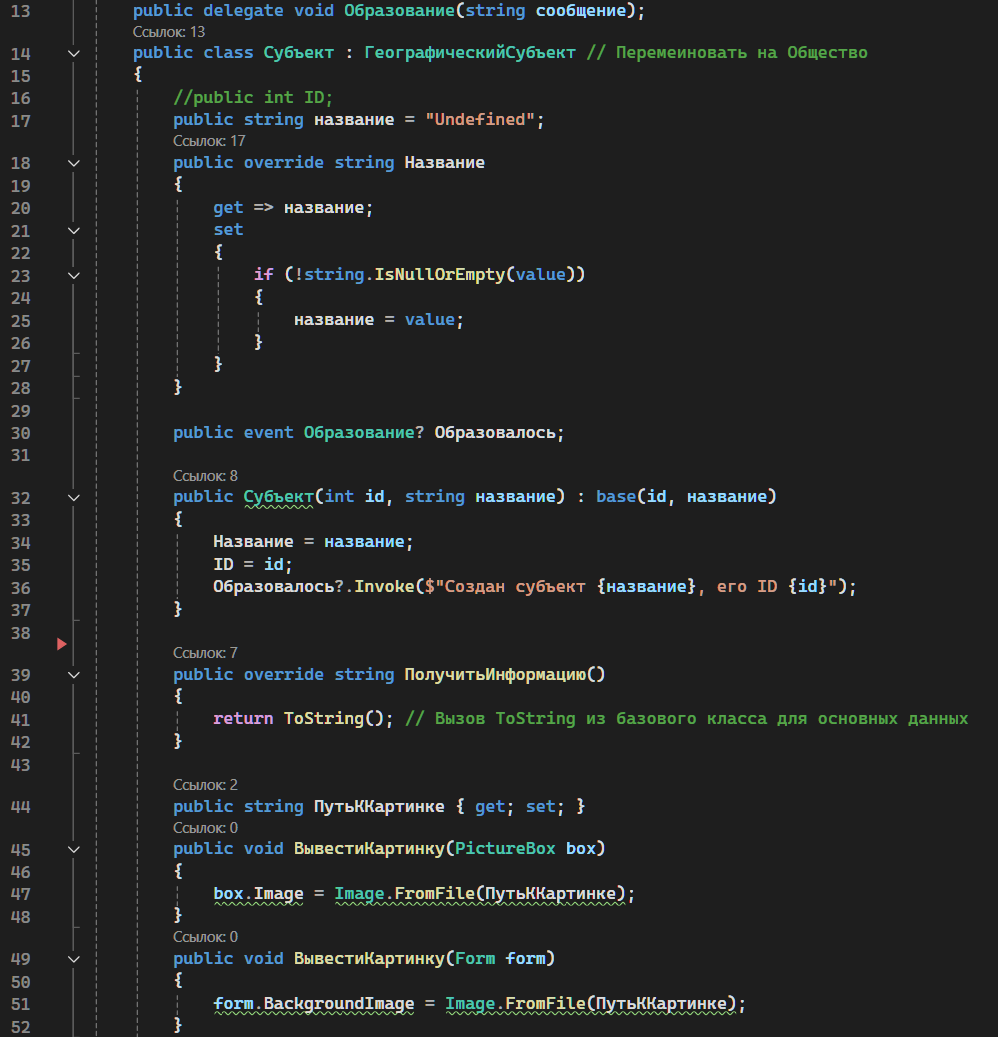


Рисунок 1.16 – код класса Субъект

Класс Субъект: является родительским для классов Государство, Область, Город, Район; наследует абстрактный класс ГеографическийСубъект; имеет поле и свойство название; конструктор, который вызывает событие Образовалось; переопределённый метод ПолучитьИнформацию, возвращающий строку с названием и id субъекта; автоматическое свойство ПутьККартинке и 2 метода ВывестиКартинку, его использующие, первый выводит картинку в PictureBox в Windows Form, второй выводит в задний фон формы.

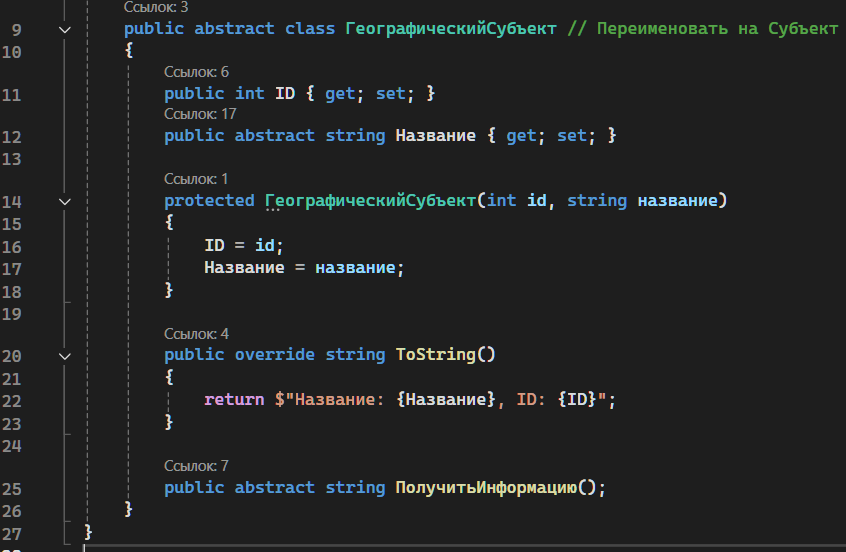


Рисунок 1.17 – код абстрактного класса ГеографическийСубъект

Класс своим существованием реализует одно из принципов объектно-ориентированного программирования – абстракцию. Здесь описывается поля, свойства и методы, которые должны быть реализованы в классе наследнике. В данном случае представлены свойства ID, Название; защищённый конструктор, который не могут использовать классы вне данного класса; методы ToString и ПолучитьИнформацию, призванные возвращать информацию об объектах.

тестирование функциональности классов

Для тестирования функциональности библиотеки, был спроектирован графический интерфейс с возможностью:

1. Создания государств/областей/городов/районов с учётом их связей;
2. Поиска государства/области/города/района по id;
3. Созданию/сбросу правителя в государстве;
4. Вывода информации о субъекте, выбранном в списке или найденном по id.

Для начала протестируем добавление и показ информации о государстве. Для этого создадим субъект и выберем его в списке.

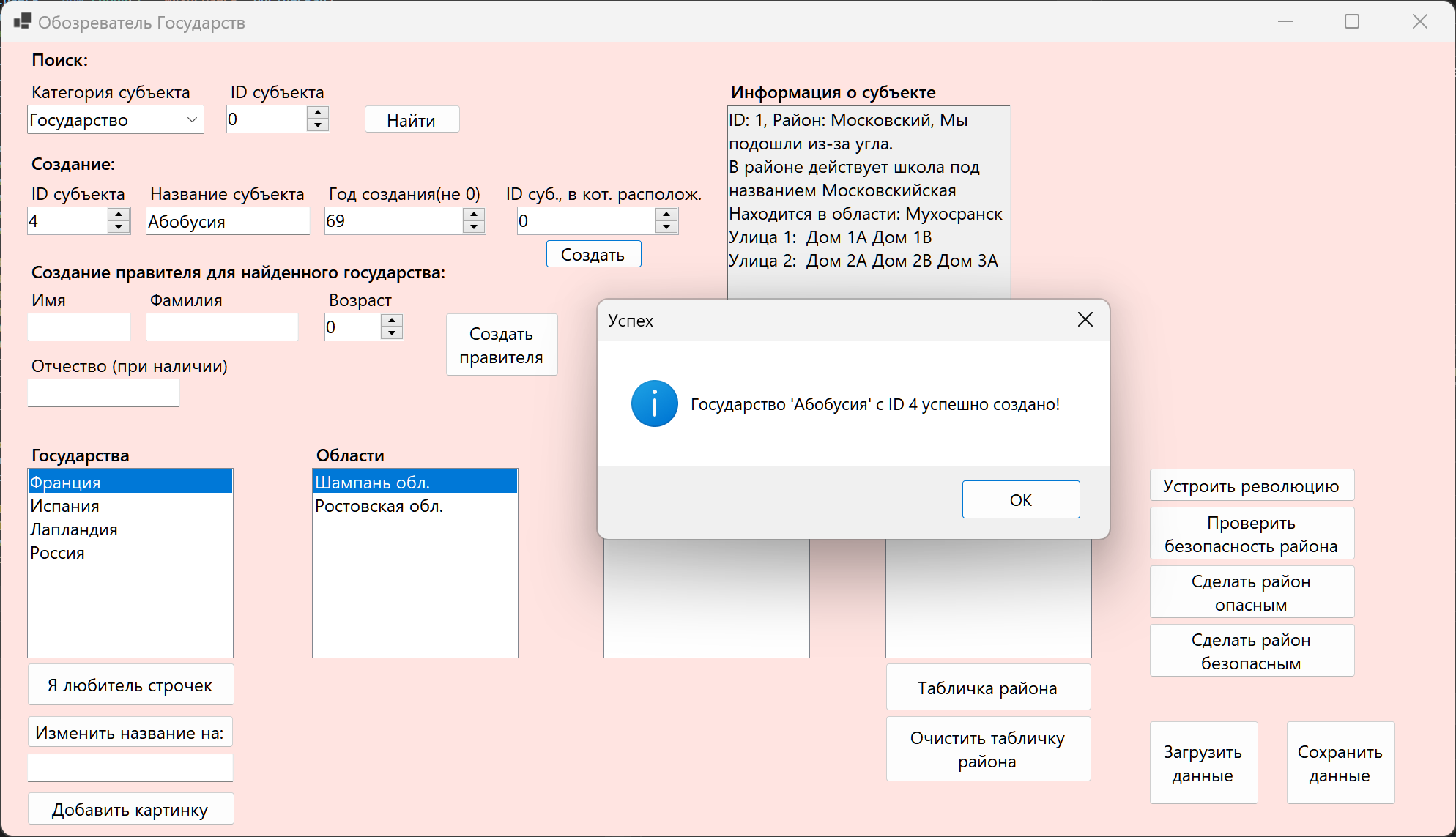


Рисунок 2.1 – тест создания государства

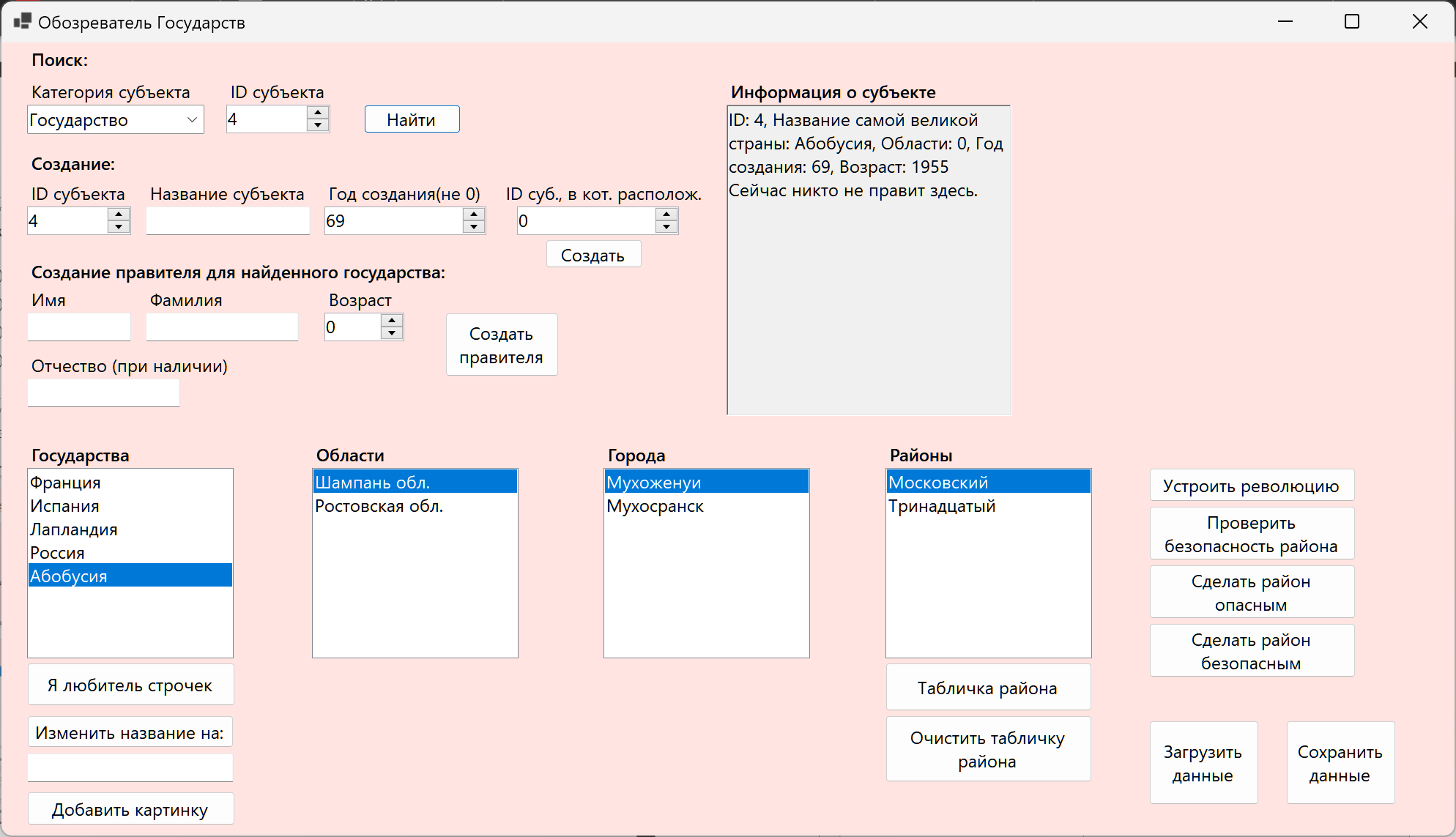


Рисунок 2.2 – тест поиска государства.

Протестируем создание правителя в государстве. Для этого в соответствующих строках введём нужную информацию о правителе и укажем id государства, где будем создавать правителя.

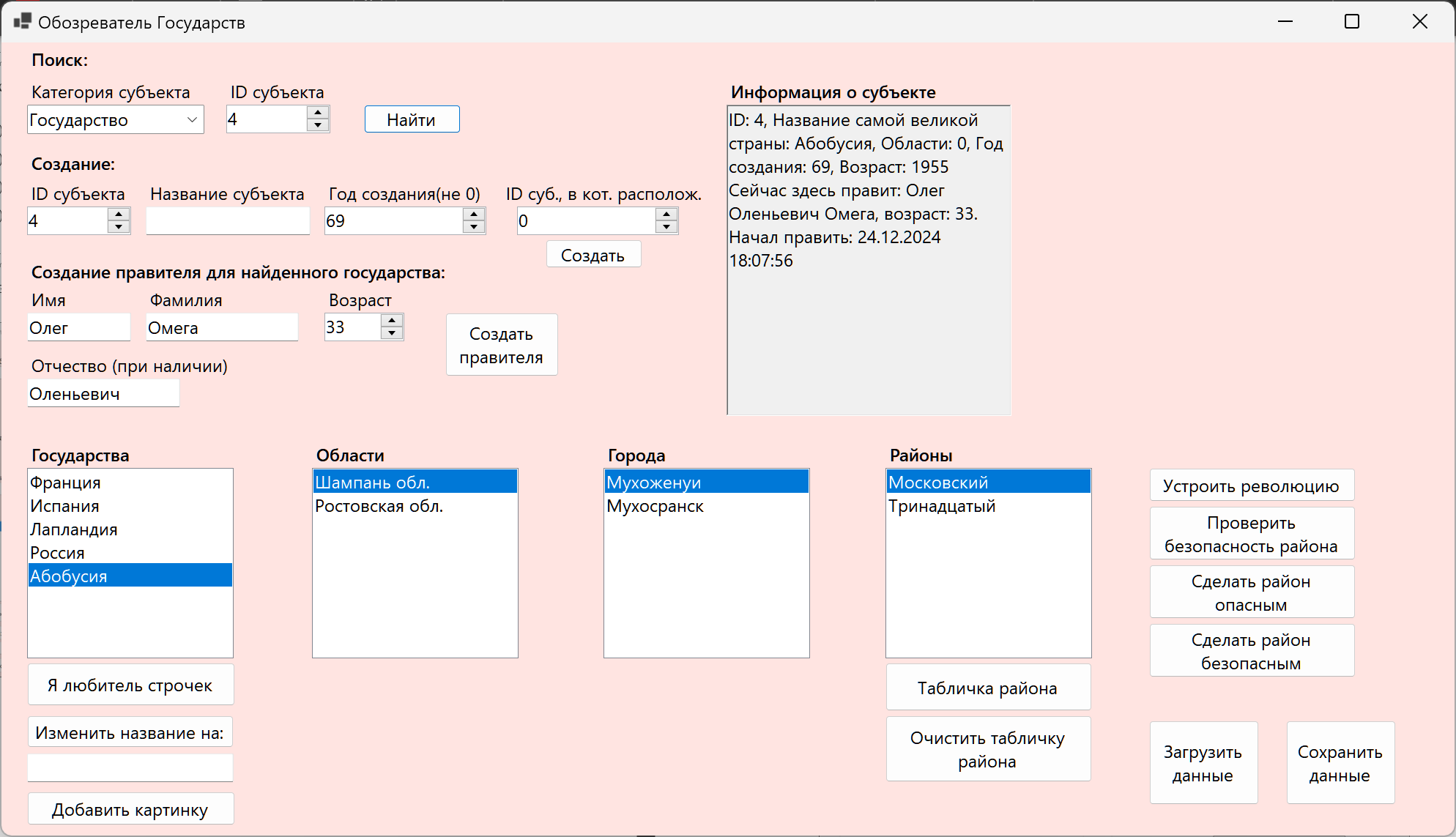


Рисунок 2.3 – тест создания правителя

Протестируем создание района в городе в области в государстве.

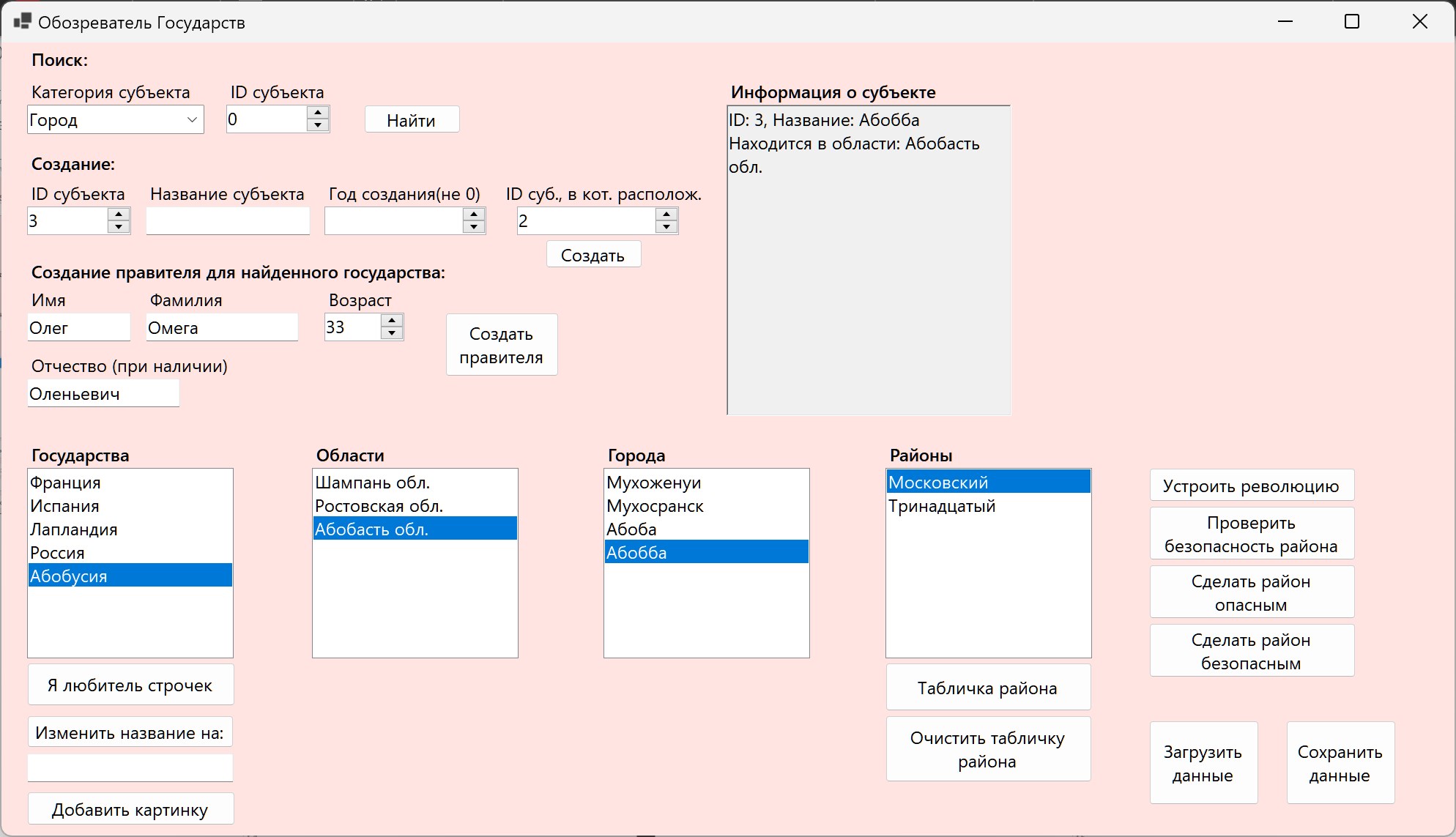


Рисунок 2.4 – тест создания города в области в государстве

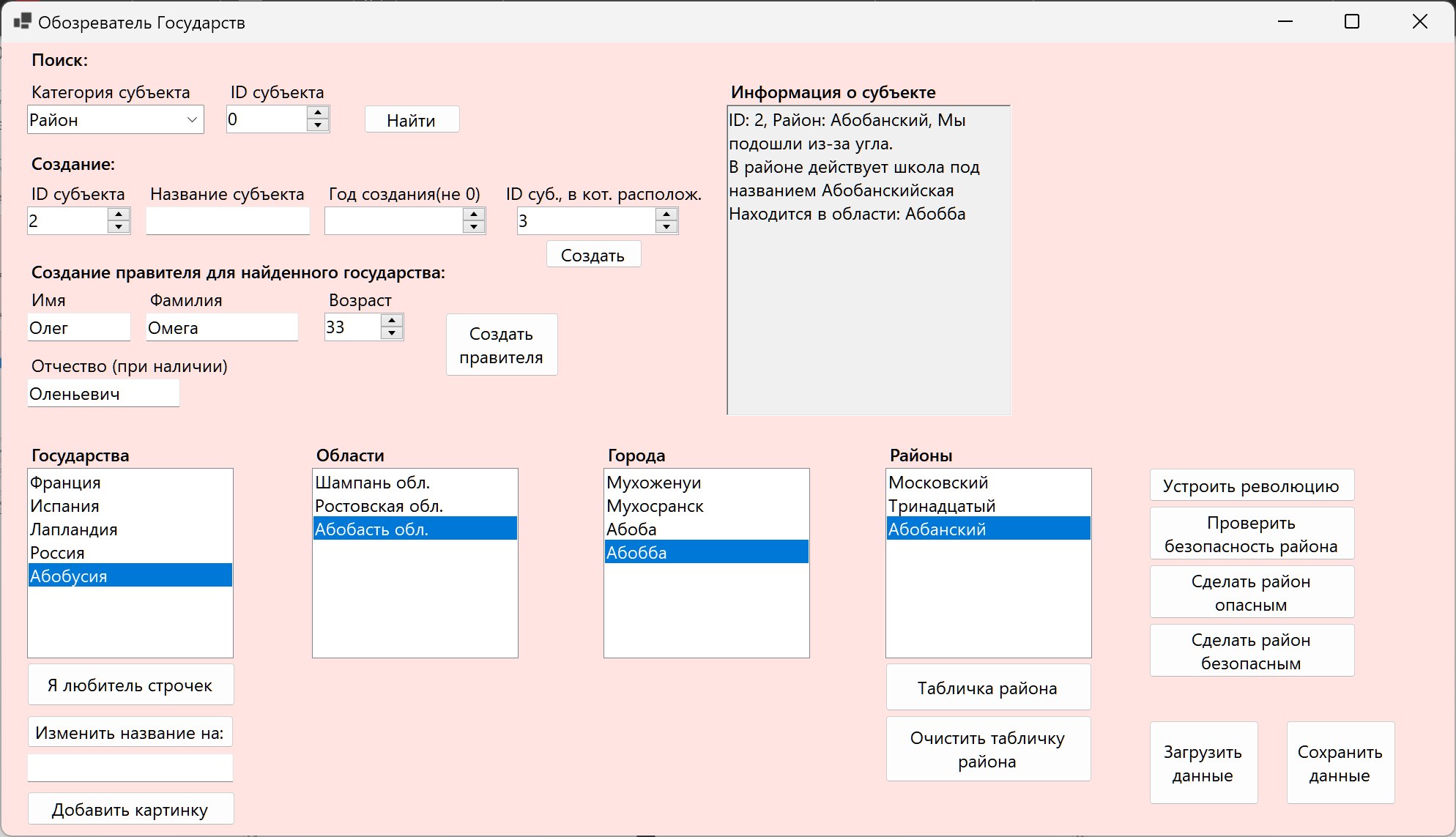


Рисунок 2.5 – тест создания района в городе

Протестируем метод Революция - сброс правителя в государстве.

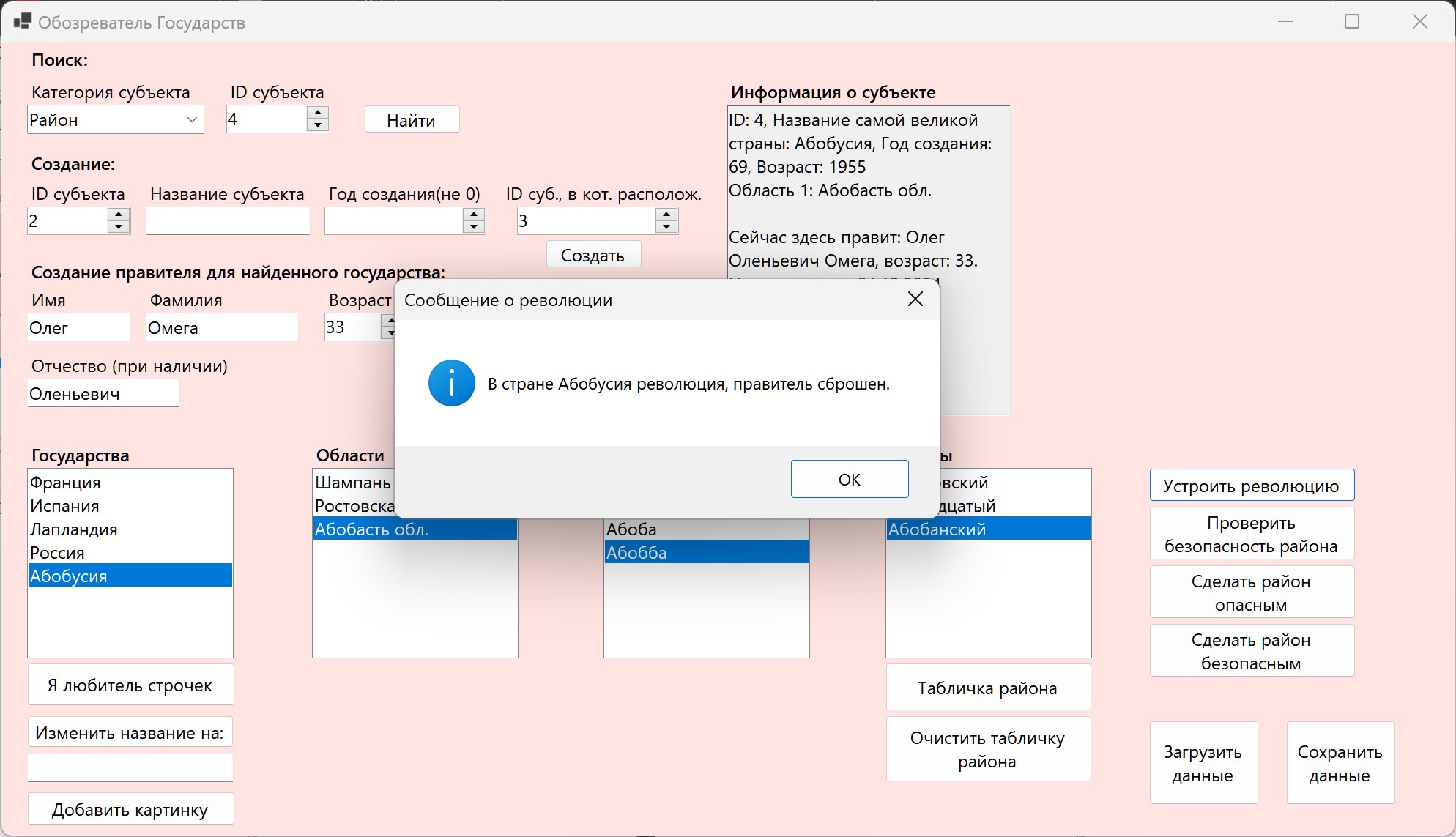


Рисунок 2.6 – устроили революцию

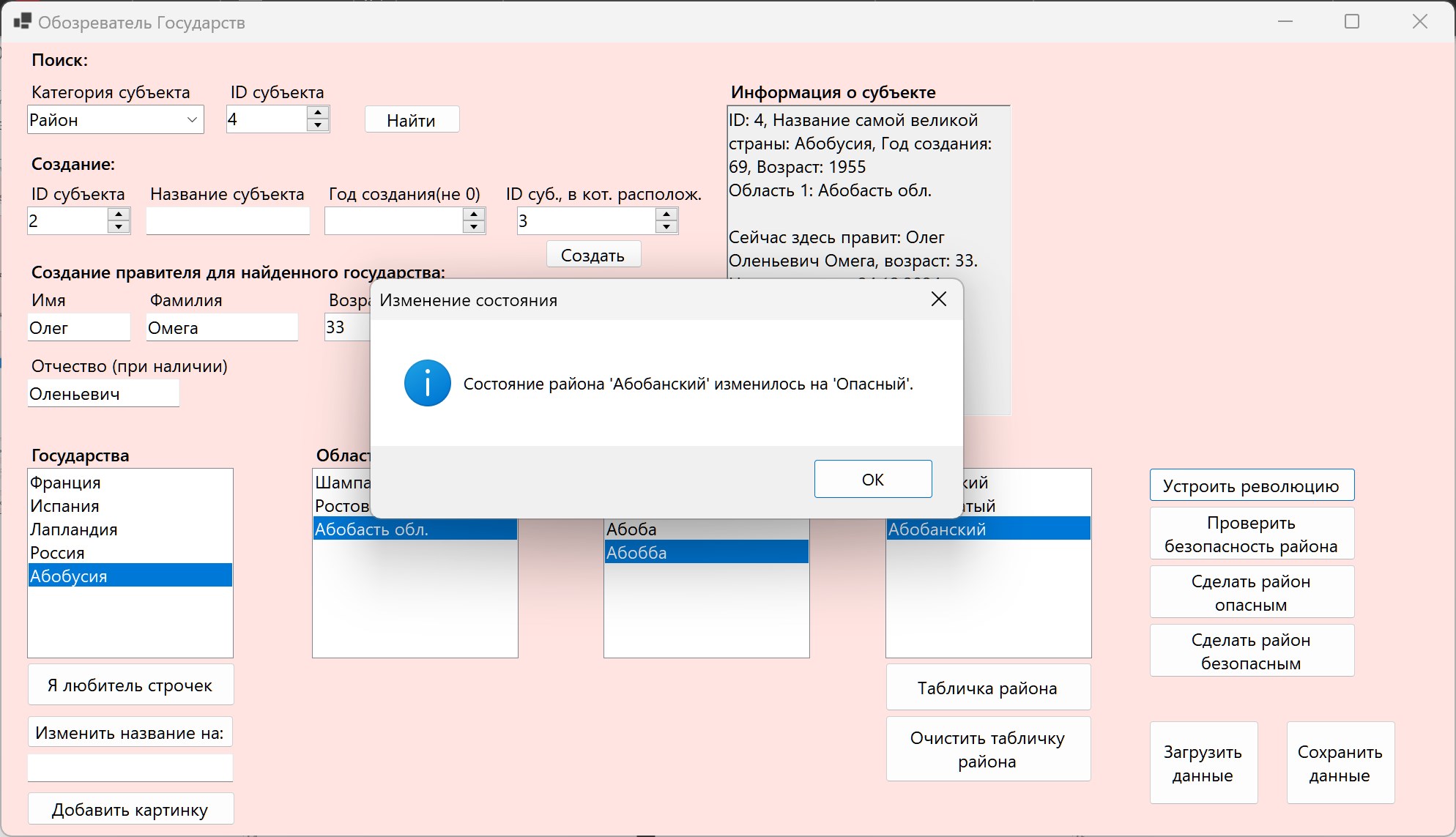


Рисунок 2.7 – район стал опасным

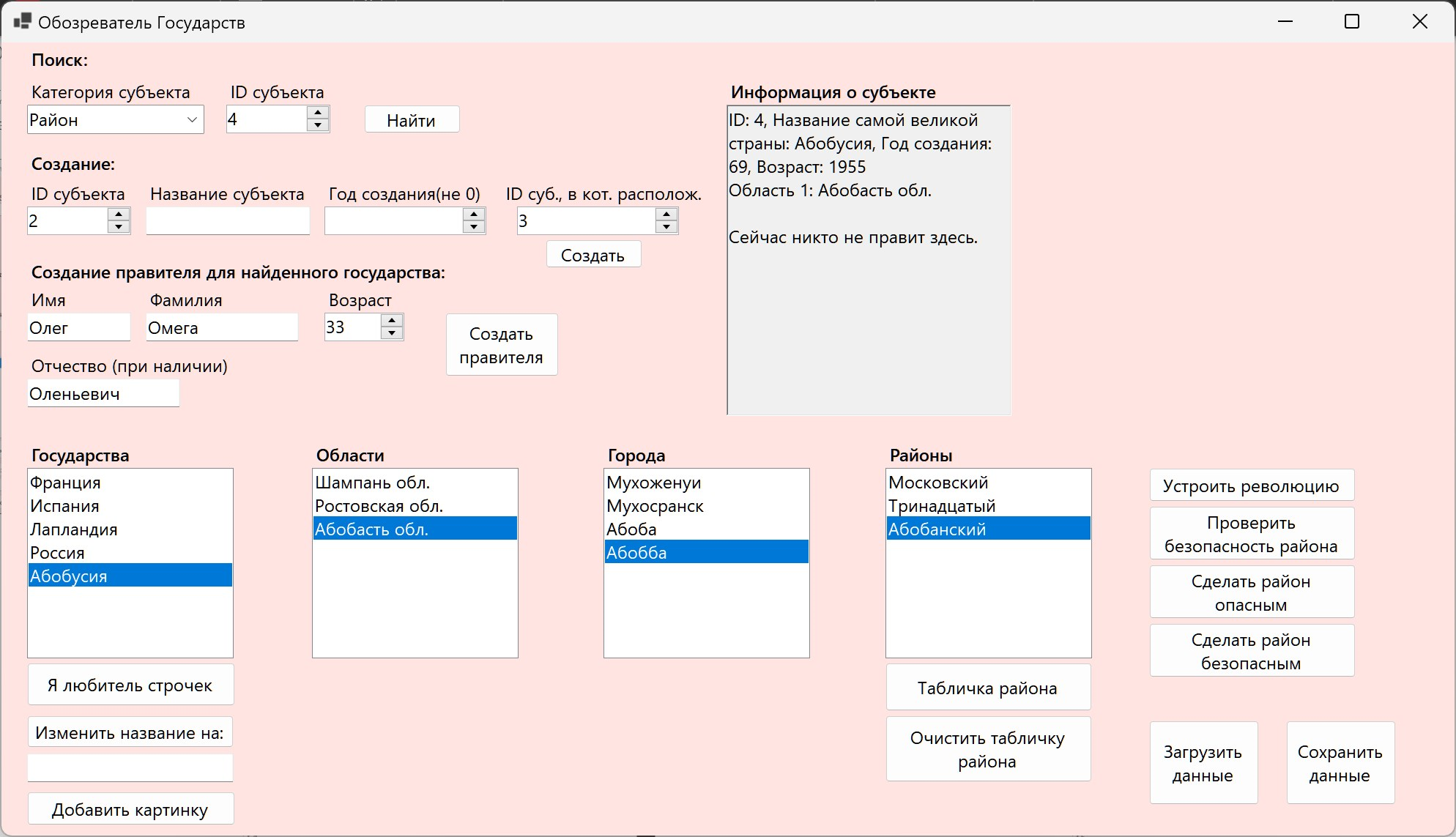
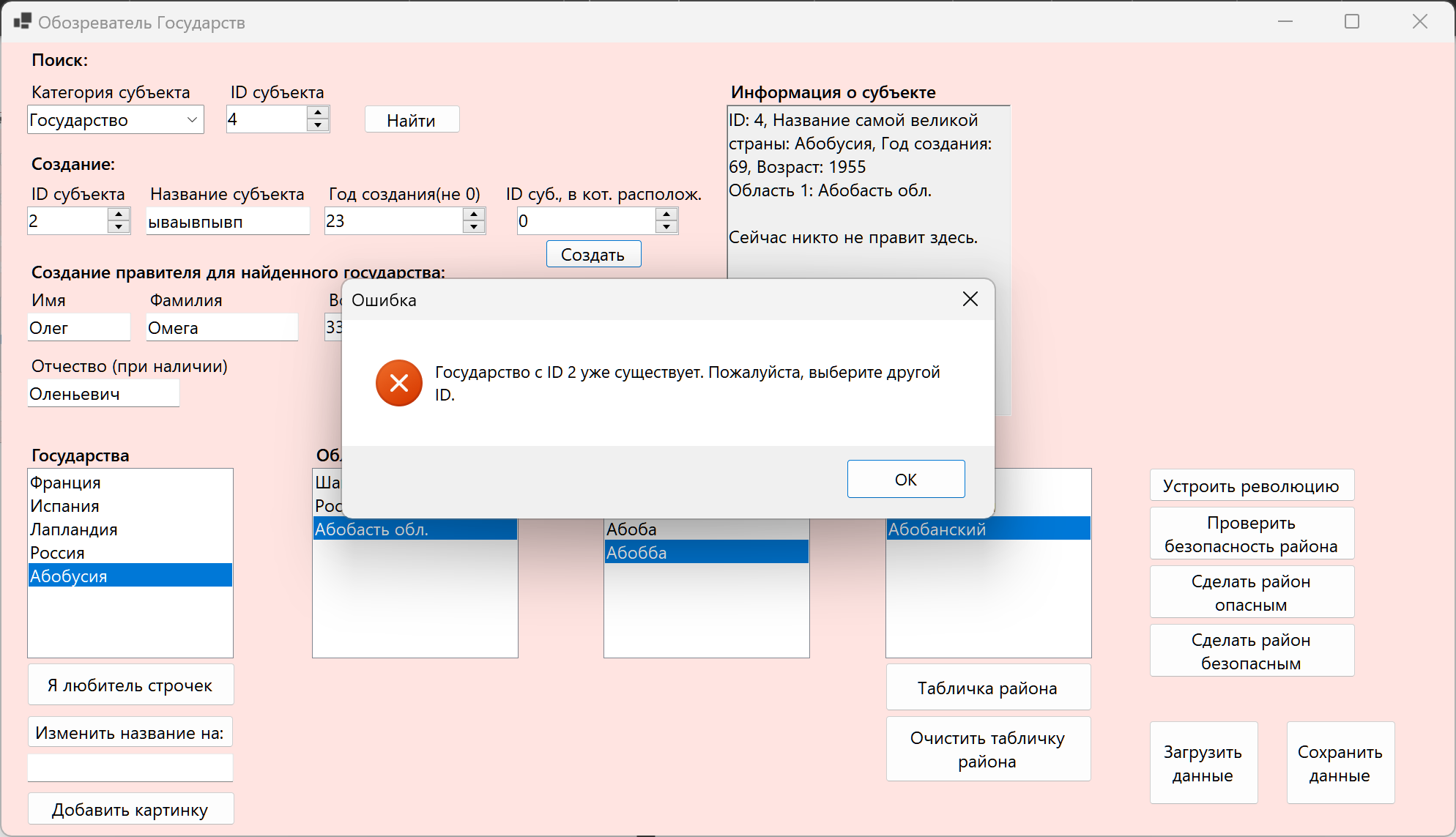
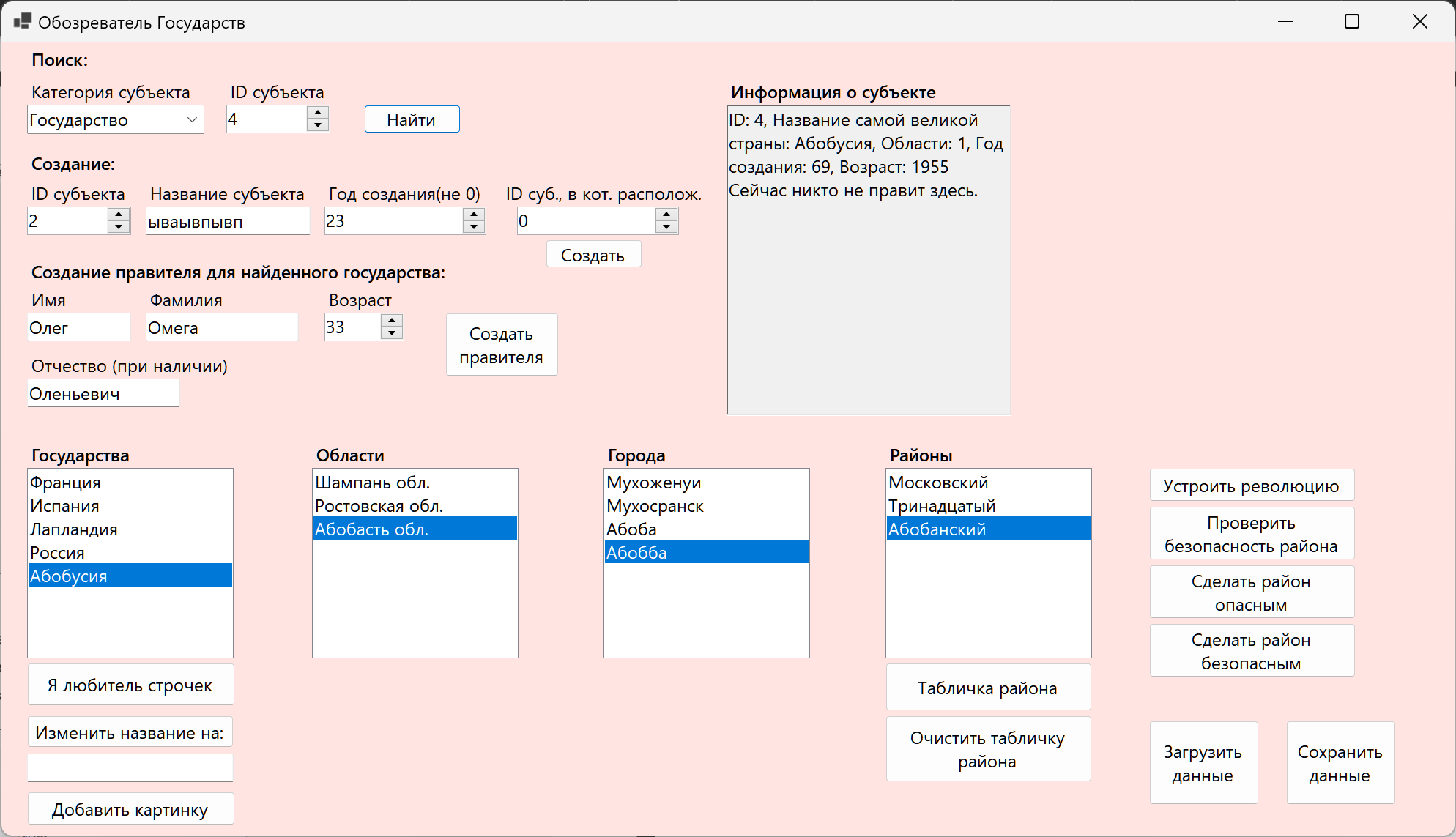


Рисунок 2.8 – в государстве сбросился правитель

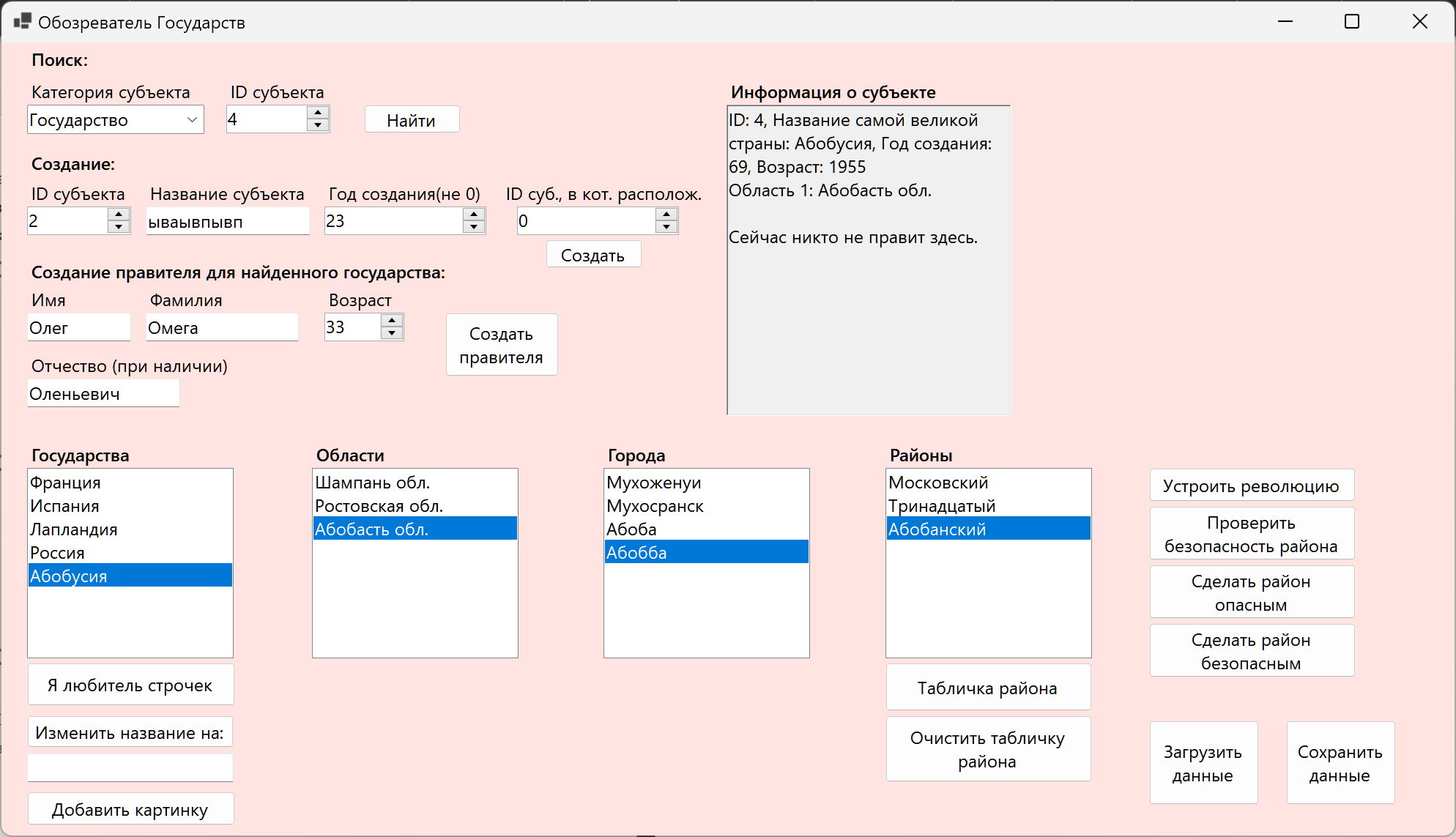
**ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ В ПРИЛОЖЕНИИ:**

Протестируем разработанную библиотеку в разработанном нами интерфейсе. Для начала пользователь создаёт различные субъекты(государства, области, города, районы). Что бы это сделать пользователь выбирает в выпадающем списке тип субъекта, в окошки вводит требуемые данные и нажимает на кнопку “Создать”. Если субъект выбранного типа с введённым id уже существует показывается ошибка.

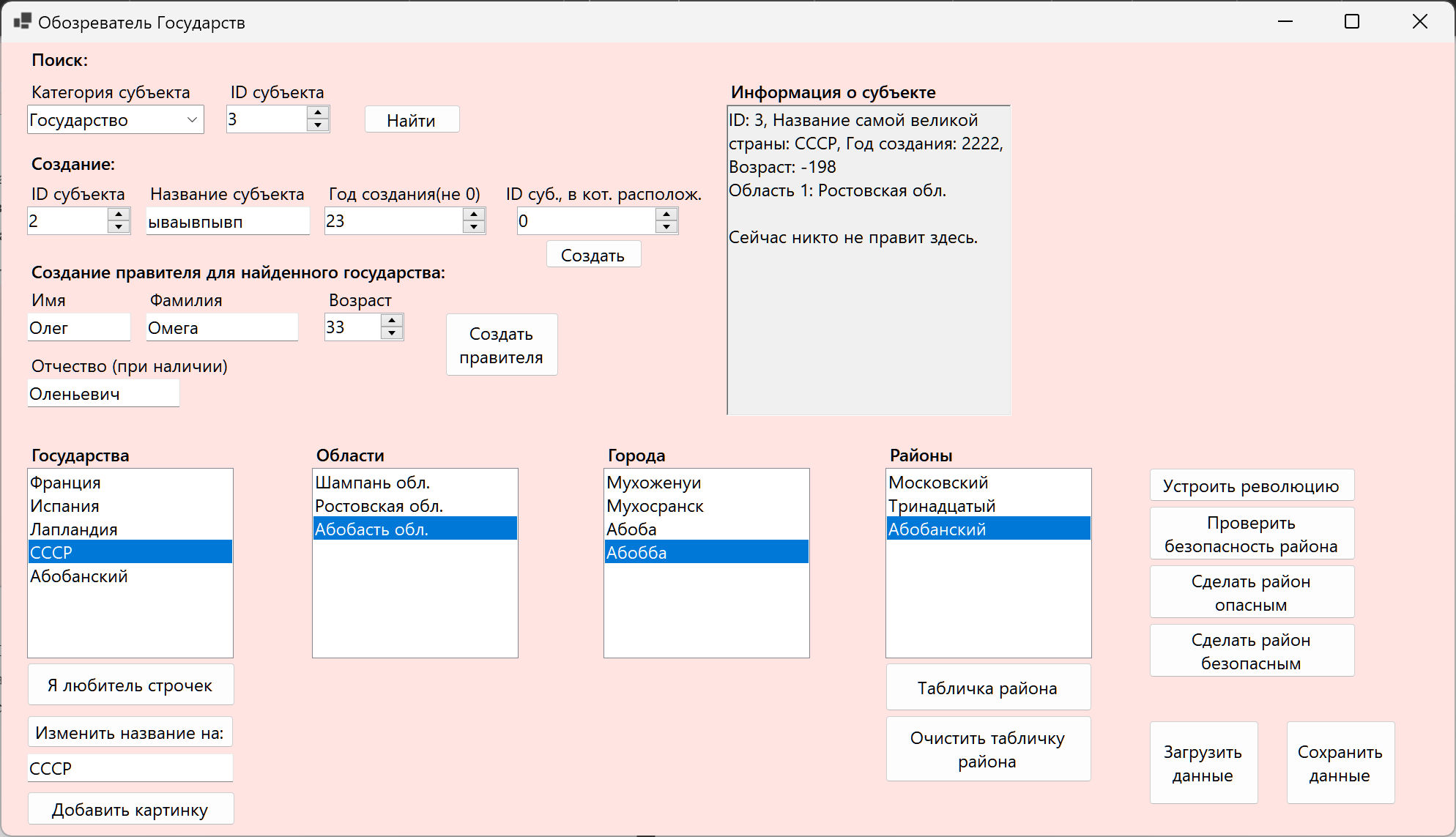
Рисунок 3.1 – показ ошибки при создании субъекта

Рисунок 3.2 – Пользователь нажал кнопку «Найти»

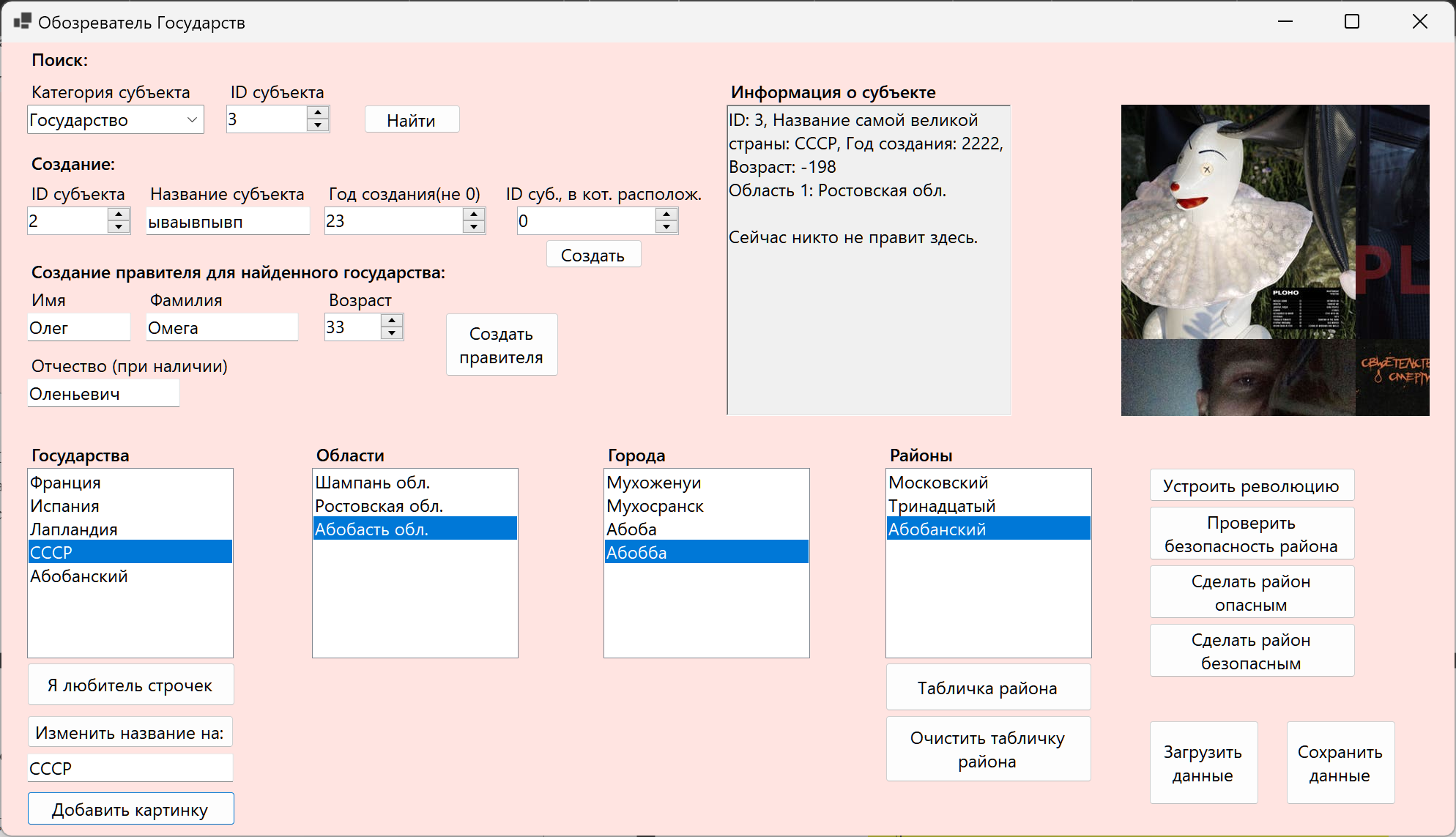
Затем пользователь может проверить информацию о требуемом субъекте просто выбрав в нижнем списке требуемый субъект, списков несколько на каждый тип субъекта. Информация отобразится в окошке “Информация о субъекте”.

Рисунок 3.3 – Пользователь выбрал субъект в списке

Пользователь может захотеть изменить название государства. Для этого пользователь выбирает id в графе «ID субъекта» и в окошко под кнопкой под списком с государствами «Изменить название на» вводит новое название и нажимает кнопку.

Рисунок 3.4 – Пользователь использовал метод «Изменить название»

Также пользователь имеет возможность добавить картинку к государству, например флаг или обложку. Для этого достаточно выбрать государство в списке, нажать на кнопку «Добавить картинку» и выбрать нужный файл.

Рисунок 3.5 – Пользователь добавил государству картинку

Если же пользователю потребуется добавить ещё и табличку района, то ему достаточно выбрать в списке требуемый район и нажать кнопку «Табличка района».

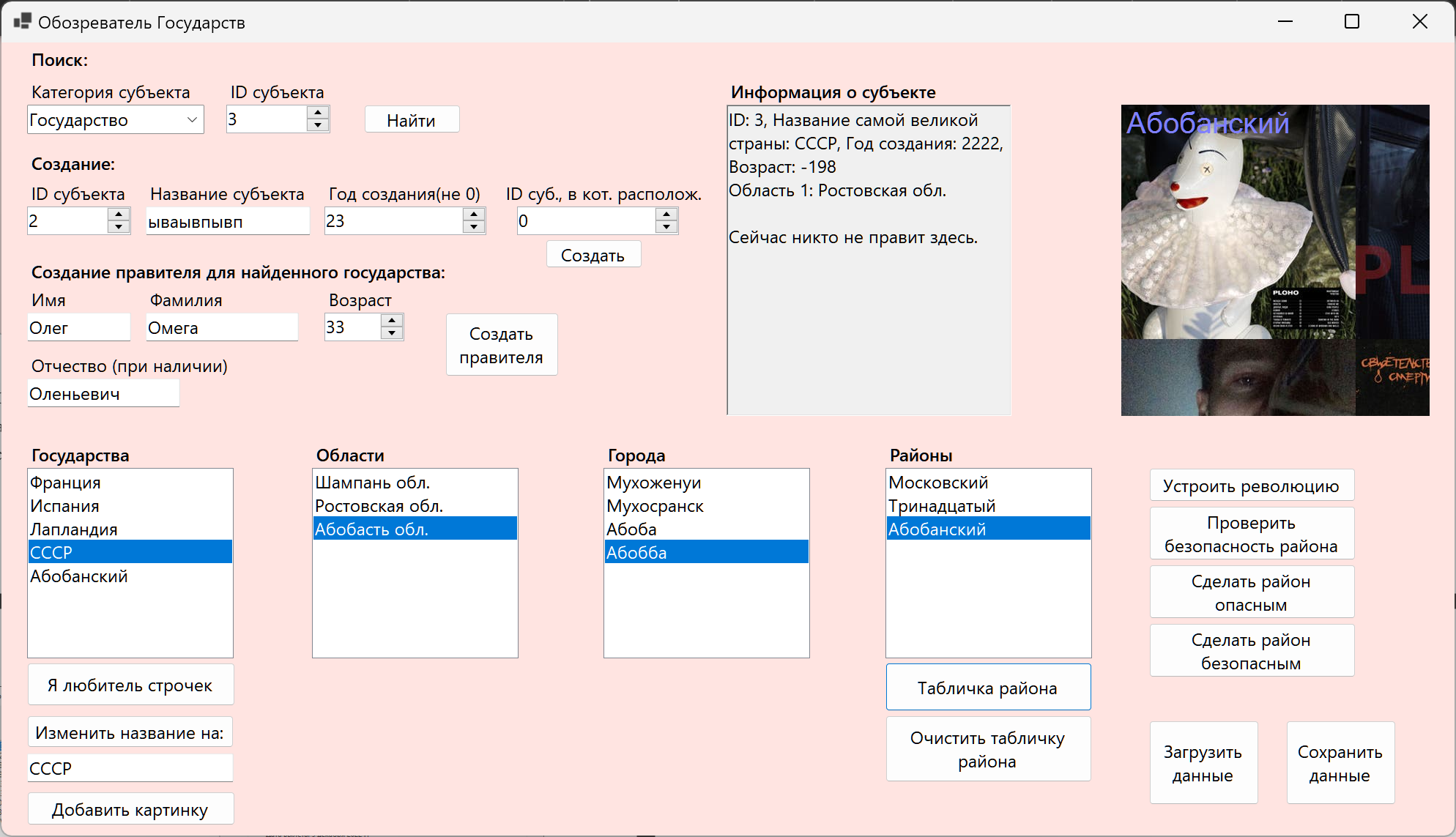


Рисунок 3.6 – Пользователь добавил табличку района

Если пользователи надоели картинки, можно всё удалить, нажав кнопку «Очистить табличку района»

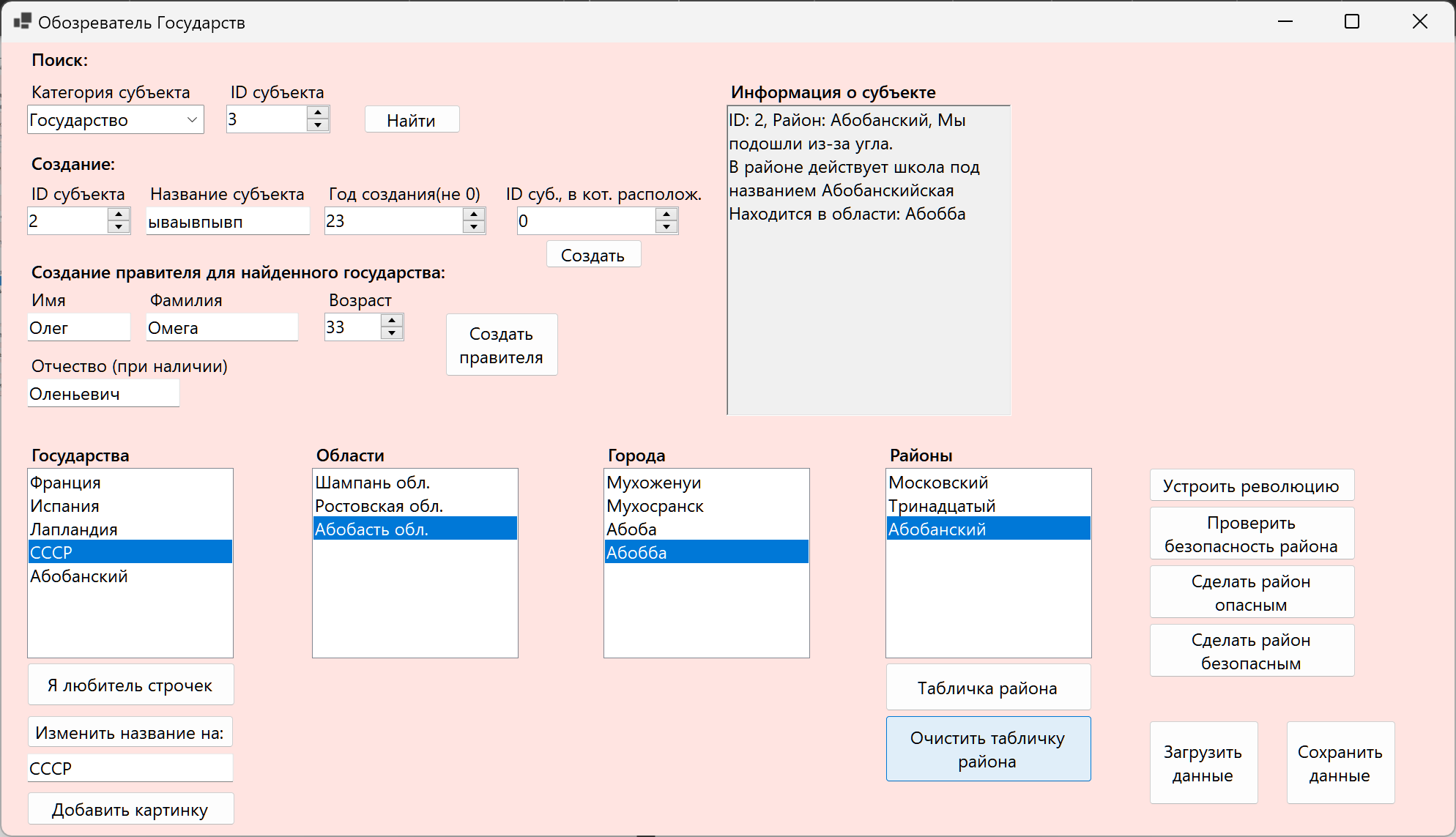


Рисунок 3.7 – Пользователь удалил табличку

руководство пользователя

Для работы с библиотекой необходим компилятор C#, встроенный в среду разработки Microsoft Visual Studio. После установки среды программирования, необходимо установить .Net 6.0 SDK. Библиотека не имеет других зависимостей и готова к использованию.

Библиотека классов ориентирована на отслеживание информации о географических субъектах. Основные элементы разработанной библиотеки классов:

1. Класс Государство. Класс представляет из себя модель данных государства. Для создания объекта класса Государство следует использовать встроенный конструктор. Присутствует метод для установки или сброса Правителя, реализовано как структура.
2. Классы Область, Город, Район. Классы представляют из себя модели данных субъектов область, город, район, соответственно. При создании объекта данных класса можно выбрать субъект (государство, область, город), в котором объект будет находится.
3. Присутствует много методов вывода информации, а также некоторые дополнительные методы, позволяющие добавлять картинки или менять название.

# 

# ЗАКЛЮЧЕНИЕ

Код библиотеки выложен и доступен на удаленном репозитории на GitHub`e: <https://github.com/MyataEtoki/StavteClassy>

В результате выполнения курсового проекта была разработана библиотека классов, для получения информации о географических субъектах. Классы библиотеки содержат все необходимые члены (поля, конструкторы, свойства, методы).

В данной библиотеке реализованы основные принципы объектно-ориентированного программирования: полиморфизм (представлен использованием обобщенных типов), инкапсуляция (представлен методами и свойствами), наследование (представлен реализацией интерфейсов), абстракция (представлен реализацией абстрактного класса).

Функционал библиотеки успешно протестирован. Все поставленные задачи реализованы.

Библиографический список

1. OMG. Unified Modeling Language (OMG UML) Version 2.5.1 (дата посещения 20.12.24): <https://www.omg.org/spec/UML/2.5.1/PDF>
2. Microsoft Docs. Interface (справочник по C#) (дата посещения 12.12.24): <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/interface>
3. События C# (справочник по C#) (дата посещения 12.12.24): <https://metanit.com/sharp/tutorial/3.14.php>
4. «Программирование на С# для начинаюших» сост.: А.Н. Васильев, изд. «Бомбора», 2022г.

приложение

## **Код классов:**

ГеографическийСубъект:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StavteClassy

{

public abstract class ГеографическийСубъект // Переименовать на Субъект

{

public int ID { get; set; }

public abstract string Название { get; set; }

protected ГеографическийСубъект(int id, string название)

{

ID = id;

Название = название;

}

public override string ToString()

{

return $"Название: {Название}, ID: {ID}";

}

public abstract string ПолучитьИнформацию();

}

}

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Xml.Linq;

using System.Windows.Forms;

using System.Security.Cryptography.X509Certificates;

using System.Text;

Субъект:

namespace StavteClassy

{

public delegate void Образование(string сообщение);

public class Субъект : ГеографическийСубъект // Перемеиновать на Общество

{

//public int ID;

public string название = "Undefined";

public override string Название

{

get => название;

set

{

if (!string.IsNullOrEmpty(value))

{

название = value;

}

}

}

public event Образование? Образовалось;

public Субъект(int id, string название) : base(id, название)

{

Название = название;

ID = id;

Образовалось?.Invoke($"Создан субъект {название}, его ID {id}");

}

public override string ПолучитьИнформацию()

{

return ToString(); // Вызов ToString из базового класса для основных данных

}

public string ПутьККартинке { get; set; }

public void ВывестиКартинку(PictureBox box)

{

box.Image = Image.FromFile(ПутьККартинке);

}

public void ВывестиКартинку(Form form)

{

form.BackgroundImage = Image.FromFile(ПутьККартинке);

}

/\* public static readonly Color BackColor;

static Субъект() // статический конструктор

{

DateTime now = DateTime.Now;

if (now.DayOfWeek == DayOfWeek.Thursday || now.DayOfWeek == DayOfWeek.Tuesday)

{

Субъект.BackColor = Color.MistyRose;

}

else

{

Субъект.BackColor = Color.White;

}

}\*/

}

}

Государство:

using StavteClassy;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StavteClassy

{

// Интерфейс - поведение класса Государство

public interface IГосударство

{

string Название { get; }

int ГодСоздания { get; }

int ВычислитьВозраст();

public void ДобавитьОбласть(Область область);

}

// Интерфейс - поведение структуры Правитель в классе Государство

public interface IПравитель

{

bool КтоТоПравит { get; } // свойство для проверки, есть ли правитель

void УстановитьПравителя(string имя, string фамилия, string отчество, int возраст, DateTime началоПравления);

void УстановитьПравителя(string имя, string фамилия, int возраст, DateTime началоПравления);

string ПоказатьИнформациюОПравителе();

void Революция();

void ИзменитьСостояниеРайоновНаОпасный();

}

public class Государство : Субъект, IПравитель, IГосударство

{

public override sealed string Название

{

get => base.Название; // Получаем значение от базового класса

set

{

if (!string.IsNullOrEmpty(value))

{

// Преобразуем первую букву к заглавной

char firstChar = value[0];

if (char.IsLower(firstChar))

{

firstChar = char.ToUpper(firstChar);

}

название = firstChar + value.Substring(1); // Собираем новое значение

}

}

}

public List<Область> Области; // ассоциация

// Индексатор для доступа к областям по индексу

public Область this[int index]

{

get

{

if (index >= 0 && index < Области.Count)

return Области[index];

throw new ArgumentOutOfRangeException(nameof(index), "Индекс вне диапазона");

}

}

private int годСоздания = 0;

public int ГодСоздания

{

get => годСоздания;

set

{

if (value == 0)

{

годСоздания = DateTime.Now.Year;

}

else

{

годСоздания = value;

}

}

}

public Правитель ТекущийПравитель;

private bool ктоТоПравит = false;

public bool КтоТоПравит

{

get => ктоТоПравит;

set => ктоТоПравит = value;

}

public bool ПроходятВыборы;

public Государство(int id, string название) : base(id, название) // конструктор

{

Области = new List<Область>(); // композиция

}

public Государство(int id, string название, int годСоздания, bool проходятВыборы = false) : base(id, название) // перегруженный конструктор

{

Области = new List<Область>(); // композиция

ГодСоздания = годСоздания;

ПроходятВыборы = проходятВыборы;

}

public event Action<string> ПравителяНет;

public void Революция()

{

ПроходятВыборы = true;

КтоТоПравит = false;

ТекущийПравитель = new Правитель();

ПравителяНет?.Invoke($"В стране {Название} революция, правитель сброшен.");

ИзменитьСостояниеРайоновНаОпасный(); // изменить состояние всех районов на "Опасный"

}

public void ИзменитьСостояниеРайоновНаОпасный()

{

foreach (var область in Области)

{

foreach (var город in область.Города)

{

foreach (var район in город.Районы)

{

район.СостояниеБезопасности = "Опасный"; // изменения состояния района на "Опасный"

}

}

}

}

public int ВычислитьВозраст()

{

return DateTime.Now.Year - ГодСоздания;

}

public void ДобавитьОбласть(Область область)

{

Области.Add(область);

}

public override string ПолучитьИнформацию()

{

var информация = base.ПолучитьИнформацию(); // Получаем базовую информацию

информация += $", Области: {Области.Count}"; // Дополнительная информация о областях

return информация;

}

public void УстановитьПравителя(string имя, string фамилия, string отчество, int возраст, DateTime НачПрав)

{

ТекущийПравитель = new Правитель(имя, фамилия, отчество, возраст, НачПрав);

КтоТоПравит = true;

ПроходятВыборы = false;

}

public void УстановитьПравителя(string имя, string фамилия, int возраст, DateTime НачПрав)

{

ТекущийПравитель = new Правитель(имя, фамилия, возраст, НачПрав);

КтоТоПравит = true;

ПроходятВыборы = false;

}

public string ПоказатьИнформациюОПравителе()

{

if (ТекущийПравитель.Существует == true)

{

return $"\nСейчас здесь правит: {ТекущийПравитель.Имя} {ТекущийПравитель.Отчество} {ТекущийПравитель.Фамилия}, " +

$"возраст: {ТекущийПравитель.Возраст}. \nНачал править: {ТекущийПравитель.НачалоПравления}";

}

return $"\nСейчаc никто не правит здесь.";

}

public override string ToString()

{

return $"ID: {ID}, Название самой великой страны: {Название}";

}

}

public struct Правитель // представитель государства

{

private string имя = "Undefined";

public string Имя

{

get => имя;

set

{

if (!string.IsNullOrEmpty(value))

{

// Проверка первого символа

char firstChar = value[0];

// Если первый символ строчный, заменяем его на прописной

if (char.IsLower(firstChar))

{

firstChar = char.ToUpper(firstChar);

}

// Собираем новое значение с измененным первым символом

имя = firstChar + value.Substring(1);

}

}

}

public string Фамилия = "Undefined";

public string Отчество = "Undefined";

public int Возраст;

private DateTime началоПравления;

public DateTime НачалоПравления

{

get => началоПравления;

set

{

началоПравления = value;

}

}

public bool Существует = false;

public Правитель()

{

Существует = false;

}

public Правитель(string имя, string фамилия, int возраст, DateTime НачПрав) : this()

{

Существует = true;

Имя = имя;

Фамилия = фамилия;

Возраст = возраст;

НачалоПравления = НачПрав;

}

public Правитель(string имя, string фамилия, string отчество, int возраст, DateTime НачПрав) : this()

{

Существует = true;

Имя = имя;

Фамилия = фамилия;

Возраст = возраст;

Отчество = отчество;

НачалоПравления = НачПрав;

}

}

}

Область:

using StavteClassy;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StavteClassy

{

// Интерфейс - поведение класса Область

public interface IОбласть

{

void ДобавитьГород(Город город);

void УдалитьГород(Город город);

string ПолучитьСписокГородов();

}

public sealed class Область : Субъект, IОбласть

{

public List<Город> Города; // ассоциация

// Индексатор для доступа к городам по индексу

public Город this[int index]

{

get

{

if (index >= 0 && index < Города.Count)

return Города[index];

throw new ArgumentOutOfRangeException(nameof(index), "Индекс вне диапазона");

}

}

public new string Название

{

get => $"{base.Название} обл.";

set => base.Название = value;

}

public Область(int id, string название) : base(id, название)

{

Города = new List<Город>(); // композиция

}

Государство ГосударствоВКоторомНаходится; // агрегация

public Область(int id, string название, Государство государство) : base(id, название)

{

Города = new List<Город>(); // композиция

ГосударствоВКоторомНаходится = государство; // агрегация

}

public void ДобавитьГород(Город город)

{

Города.Add(город);

}

public void УдалитьГород(Город город)

{

Города.Remove(город);

}

public override string ПолучитьИнформацию()

{

var информация = base.ПолучитьИнформацию(); // Получаем информацию от базового класса

информация += $", Города: {Города.Count}"; // Информация о городах, содержащихся в области

return информация;

}

public string ПолучитьСписокГородов()

{

return string.Join(", ", Города.Select(город => город.Название));

}

public override string ToString()

{

return $"ID: {ID}, Название: {Название}, Областной центр не ищите.";

}

public string ГдеНаходится()

{

if (ГосударствоВКоторомНаходится != null)

{

return $"\nНаходится в стране: {ГосударствоВКоторомНаходится.Название}";

}

else { return ""; }

}

/\* public static Область ИзСубъект(Субъект субъект)

{

return new Область(субъект.Название)

{

ID = субъект.ID

};

} \*/

}

}

Город:

using StavteClassy;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StavteClassy

{

// Интерфейс - поведение класса Город

public interface IГород

{

void ДобавитьРайон(Район район);

}

public sealed class Город : Субъект, IГород

{

public List<Район> Районы; // ассоциация

// Индексатор для доступа к районам по индексу

public Район this[int index]

{

get

{

if (index >= 0 && index < Районы.Count)

return Районы[index];

throw new ArgumentOutOfRangeException(nameof(index), "Индекс вне диапазона");

}

}

Область ОбластьВКоторойНаходится;

public Город(int id, string название) : base(id, название) // метод-конструктор

{

Районы = new List<Район>(); // композиция

}

public Город(int id, string название, Область область) : base(id, название) // метод-конструктор

{

Районы = new List<Район>(); // композиция

ОбластьВКоторойНаходится = область; // агрегация

}

public void ДобавитьРайон(Район район)

{

Районы.Add(район);

}

public string ГдеНаходится()

{

if (ОбластьВКоторойНаходится != null)

{

return $"\nНаходится в области: {ОбластьВКоторойНаходится.Название}";

}

else { return ""; }

}

public override string ПолучитьИнформацию()

{

var информация = base.ПолучитьИнформацию(); // Получаем информацию от базового класса

информация += ", Районы: " + (Районы != null ? Районы.Count.ToString() : "0"); // Информация о районах

return информация;

}

// Сокрытие метода

public new string ToString()

{

return $"Был создан новый субъект: {Название}";

}

}

}

Район:

using StavteClassy;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace StavteClassy

{

// Интерфейс - поведение класса Район

public interface IРайон

{

string ВывестиУлицы();

void NameText(PictureBox box);

}

public sealed class Район : Субъект, IРайон

{

private string[,] дома; // 2D массив для хранения информации о домах

private int улицы; // Количество улиц

private int домаНаУлице; // Количество домов на каждой улице

Школа школарайона;

public Район(int id, string название, int улицы = 0, int домаНаУлице = 0) : base(id, название)

{

//тут мз субъекта подтянулось - id = id, название = название

this.улицы = улицы;

this.домаНаУлице = домаНаУлице;

дома = new string[улицы, домаНаУлице]; // Инициализация двумерного массива

школарайона = new Школа($"{название}ская"); // Композиция

}

Город ГородВКоторомНаходится;

public Район(int id, string название, Город город, int улицы = 0, int домаНаУлице = 0) : base(id, название)

{

//тут мз субъекта подтянулось - id = id, название = название

this.улицы = улицы;

this.домаНаУлице = домаНаУлице;

дома = new string[улицы, домаНаУлице]; // Инициализация двумерного массива

школарайона = new Школа($"{название}ская"); // Композиция

ГородВКоторомНаходится = город; // агрегация

}

// Демонстрация полиморфизма - переопределение метода

public override string ToString()

{

return $"ID: {ID}, Район: {Название}, Мы подошли из-за угла. \nВ районе действует школа под названием {школарайона.Название}";

}

public string ГдеНаходится()

{

if (ГородВКоторомНаходится != null)

{

return $"\nНаходится в городе: {ГородВКоторомНаходится.Название}";

}

else { return ""; }

}

// Индексатор для доступа к домам по улице и номеру дома

public string this[int улица, int номерДома]

{

get

{

if (улица < 0 || улица >= улицы || номерДома < 0 || номерДома >= домаНаУлице)

throw new IndexOutOfRangeException("Индекс вне диапазона массива.");

return дома[улица, номерДома];

}

set

{

if (улица < 0 || улица >= улицы || номерДома < 0 || номерДома >= домаНаУлице)

throw new IndexOutOfRangeException("Индекс вне диапазона массива.");

дома[улица, номерДома] = value;

}

}

// Метод для вывода информации о всех домах

public string ВывестиУлицы()

{

// Инициализация результатирующей строки

StringBuilder result = new StringBuilder();

for (int i = 0; i < улицы; i++)

{

result.Append($"\nУлица {i + 1}: ");

for (int j = 0; j < домаНаУлице; j++)

{

result.Append($" {дома[i, j]}");

}

}

return result.ToString(); // Возвращаем преобразованную строку

}

// Метод - Табличка района

public void NameText(PictureBox box)

{

using (FontDialog fontDialog = new FontDialog())

using (ColorDialog colorDialog = new ColorDialog())

{

// Открытие диалоговых окон для выбора шрифта и цвета

if (fontDialog.ShowDialog() == DialogResult.OK &&

colorDialog.ShowDialog() == DialogResult.OK)

{

// Получаем выбранный шрифт и цвет

Font currFont = fontDialog.Font;

Color textColor = colorDialog.Color;

// Используем Graphics для отрисовки текста

using (Graphics g = box.CreateGraphics())

{

// Используем Brush с выбранным цветом

using (Brush brush = new SolidBrush(textColor))

{

g.DrawString(this.Название, currFont, brush, new PointF(1, 1));

}

}

}

}

}

// Событие БезопасностьИзменилась

private string безопасность = "Undefined";

public string СостояниеБезопасности

{

get { return безопасность; }

set

{

if (безопасность != value)

{

безопасность = value;

OnБезопасностьИзменилась(new БезопасностьИзмениласьEventArgs(Название, безопасность));

}

}

}

public event БезопасностьИзмениласьEventHandler БезопасностьИзменилась;

public void OnБезопасностьИзменилась(БезопасностьИзмениласьEventArgs e)

{

БезопасностьИзменилась?.Invoke(this, e);

}

}

// Событие БезопасностьИзменилась - СостояниеБезопасности

public delegate void БезопасностьИзмениласьEventHandler(object sender, БезопасностьИзмениласьEventArgs e);

public class БезопасностьИзмениласьEventArgs : EventArgs

{

public string НазваниеРайона { get; }

public string НовоеСостояние { get; }

public БезопасностьИзмениласьEventArgs(string названиеРайона, string новоеСостояние)

{

НазваниеРайона = названиеРайона;

НовоеСостояние = новоеСостояние;

}

}

}

Школа:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StavteClassy

{

// В каждом районе есть 1 Школа, создаётся при создании Района - композиция.

internal class Школа

{

private string название;

public string Название { get=> название; set => название = value; }

public Школа(string название)

{

Название = название;

}

}

}

## **Код формы:**

### **Код дизайна формы:**

namespace HopeForPravilnost

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

richTextBox1 = new RichTextBox();

button1 = new Button();

numericUpDown1 = new NumericUpDown();

label1 = new Label();

label2 = new Label();

label3 = new Label();

label4 = new Label();

textBox2 = new TextBox();

label5 = new Label();

numericUpDown2 = new NumericUpDown();

button2 = new Button();

label6 = new Label();

label7 = new Label();

textBox3 = new TextBox();

label8 = new Label();

textBox4 = new TextBox();

label9 = new Label();

numericUpDown3 = new NumericUpDown();

label10 = new Label();

label11 = new Label();

textBox5 = new TextBox();

button3 = new Button();

label12 = new Label();

numericUpDown4 = new NumericUpDown();

listBox1 = new ListBox();

label13 = new Label();

button4 = new Button();

button5 = new Button();

button6 = new Button();

pictureBox1 = new PictureBox();

button7 = new Button();

textBox6 = new TextBox();

label14 = new Label();

button8 = new Button();

listBox2 = new ListBox();

listBox3 = new ListBox();

listBox4 = new ListBox();

label15 = new Label();

label16 = new Label();

label17 = new Label();

comboBox1 = new ComboBox();

label18 = new Label();

button9 = new Button();

button10 = new Button();

button11 = new Button();

button12 = new Button();

button13 = new Button();

button14 = new Button();

label19 = new Label();

numericUpDown5 = new NumericUpDown();

((System.ComponentModel.ISupportInitialize)numericUpDown1).BeginInit();

((System.ComponentModel.ISupportInitialize)numericUpDown2).BeginInit();

((System.ComponentModel.ISupportInitialize)numericUpDown3).BeginInit();

((System.ComponentModel.ISupportInitialize)numericUpDown4).BeginInit();

((System.ComponentModel.ISupportInitialize)pictureBox1).BeginInit();

((System.ComponentModel.ISupportInitialize)numericUpDown5).BeginInit();

SuspendLayout();

//

// richTextBox1

//

richTextBox1.Location = new Point(990, 85);

richTextBox1.Name = "richTextBox1";

richTextBox1.ReadOnly = true;

richTextBox1.Size = new Size(389, 425);

richTextBox1.TabIndex = 1;

richTextBox1.Text = "";

//

// button1

//

button1.Location = new Point(495, 85);

button1.Name = "button1";

button1.Size = new Size(132, 39);

button1.TabIndex = 2;

button1.Text = "Найти";

button1.UseVisualStyleBackColor = true;

button1.Click += button1\_Click;

//

// numericUpDown1

//

numericUpDown1.Location = new Point(307, 85);

numericUpDown1.Name = "numericUpDown1";

numericUpDown1.Size = new Size(142, 39);

numericUpDown1.TabIndex = 3;

//

// label1

//

label1.AutoSize = true;

label1.Location = new Point(35, 50);

label1.Name = "label1";

label1.Size = new Size(231, 32);

label1.TabIndex = 4;

label1.Text = "Категория субъекта";

//

// label2

//

label2.AutoSize = true;

label2.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold, GraphicsUnit.Point, 204);

label2.Location = new Point(990, 50);

label2.Name = "label2";

label2.Size = new Size(294, 32);

label2.TabIndex = 5;

label2.Text = "Информация о субъекте";

//

// label3

//

label3.AutoSize = true;

label3.Location = new Point(307, 50);

label3.Name = "label3";

label3.Size = new Size(142, 32);

label3.TabIndex = 6;

label3.Text = "ID субъекта";

//

// label4

//

label4.AutoSize = true;

label4.Location = new Point(197, 189);

label4.Name = "label4";

label4.Size = new Size(225, 32);

label4.TabIndex = 8;

label4.Text = "Название субъекта";

//

// textBox2

//

textBox2.Location = new Point(197, 224);

textBox2.Name = "textBox2";

textBox2.Size = new Size(225, 39);

textBox2.TabIndex = 7;

//

// label5

//

label5.AutoSize = true;

label5.Location = new Point(35, 189);

label5.Name = "label5";

label5.Size = new Size(142, 32);

label5.TabIndex = 10;

label5.Text = "ID субъекта";

//

// numericUpDown2

//

numericUpDown2.Location = new Point(35, 224);

numericUpDown2.Name = "numericUpDown2";

numericUpDown2.Size = new Size(142, 39);

numericUpDown2.TabIndex = 9;

//

// button2

//

button2.Location = new Point(743, 269);

button2.Name = "button2";

button2.Size = new Size(132, 39);

button2.TabIndex = 11;

button2.Text = "Создать";

button2.UseVisualStyleBackColor = true;

button2.Click += button2\_Click;

//

// label6

//

label6.AutoSize = true;

label6.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold, GraphicsUnit.Point, 204);

label6.Location = new Point(35, 148);

label6.Name = "label6";

label6.Size = new Size(128, 32);

label6.TabIndex = 12;

label6.Text = "Создание:";

//

// label7

//

label7.AutoSize = true;

label7.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold, GraphicsUnit.Point, 204);

label7.Location = new Point(35, 296);

label7.Name = "label7";

label7.Size = new Size(579, 32);

label7.TabIndex = 13;

label7.Text = "Создание правителя для найденного государства:";

//

// textBox3

//

textBox3.Location = new Point(35, 369);

textBox3.Name = "textBox3";

textBox3.Size = new Size(142, 39);

textBox3.TabIndex = 14;

//

// label8

//

label8.AutoSize = true;

label8.Location = new Point(35, 334);

label8.Name = "label8";

label8.Size = new Size(61, 32);

label8.TabIndex = 15;

label8.Text = "Имя";

//

// textBox4

//

textBox4.Location = new Point(197, 369);

textBox4.Name = "textBox4";

textBox4.Size = new Size(209, 39);

textBox4.TabIndex = 16;

//

// label9

//

label9.AutoSize = true;

label9.Location = new Point(197, 334);

label9.Name = "label9";

label9.Size = new Size(113, 32);

label9.TabIndex = 17;

label9.Text = "Фамилия";

//

// numericUpDown3

//

numericUpDown3.Location = new Point(441, 369);

numericUpDown3.Name = "numericUpDown3";

numericUpDown3.Size = new Size(109, 39);

numericUpDown3.TabIndex = 18;

//

// label10

//

label10.AutoSize = true;

label10.Location = new Point(441, 334);

label10.Name = "label10";

label10.Size = new Size(100, 32);

label10.TabIndex = 19;

label10.Text = "Возраст";

//

// label11

//

label11.AutoSize = true;

label11.Location = new Point(35, 424);

label11.Name = "label11";

label11.Size = new Size(282, 32);

label11.TabIndex = 23;

label11.Text = "Отчество (при наличии)";

//

// textBox5

//

textBox5.Location = new Point(35, 459);

textBox5.Name = "textBox5";

textBox5.Size = new Size(209, 39);

textBox5.TabIndex = 22;

//

// button3

//

button3.Location = new Point(606, 369);

button3.Name = "button3";

button3.Size = new Size(155, 87);

button3.TabIndex = 20;

button3.Text = "Создать правителя";

button3.UseVisualStyleBackColor = true;

button3.Click += button3\_Click;

//

// label12

//

label12.AutoSize = true;

label12.Location = new Point(441, 189);

label12.Name = "label12";

label12.Size = new Size(221, 32);

label12.TabIndex = 25;

label12.Text = "Год создания(не 0)";

//

// numericUpDown4

//

numericUpDown4.Location = new Point(441, 224);

numericUpDown4.Name = "numericUpDown4";

numericUpDown4.Size = new Size(221, 39);

numericUpDown4.TabIndex = 24;

//

// listBox1

//

listBox1.FormattingEnabled = true;

listBox1.Location = new Point(35, 581);

listBox1.Name = "listBox1";

listBox1.Size = new Size(282, 260);

listBox1.TabIndex = 26;

listBox1.SelectedIndexChanged += listBox1\_SelectedIndexChanged;

//

// label13

//

label13.AutoSize = true;

label13.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold);

label13.Location = new Point(35, 546);

label13.Name = "label13";

label13.Size = new Size(148, 32);

label13.TabIndex = 27;

label13.Text = "Государства";

//

// button4

//

button4.Location = new Point(1567, 926);

button4.Name = "button4";

button4.Size = new Size(150, 115);

button4.TabIndex = 28;

button4.Text = "Загрузить данные";

button4.UseVisualStyleBackColor = true;

button4.Click += button4\_Click;

//

// button5

//

button5.Location = new Point(1754, 926);

button5.Name = "button5";

button5.Size = new Size(150, 115);

button5.TabIndex = 29;

button5.Text = "Сохранить данные";

button5.UseVisualStyleBackColor = true;

button5.Click += button5\_Click;

//

// button6

//

button6.Location = new Point(35, 847);

button6.Name = "button6";

button6.Size = new Size(284, 59);

button6.TabIndex = 30;

button6.Text = "Я любитель строчек";

button6.UseVisualStyleBackColor = true;

button6.Click += button6\_Click;

//

// pictureBox1

//

pictureBox1.Location = new Point(1529, 85);

pictureBox1.Name = "pictureBox1";

pictureBox1.Size = new Size(421, 425);

pictureBox1.TabIndex = 31;

pictureBox1.TabStop = false;

//

// button7

//

button7.Location = new Point(35, 1023);

button7.Name = "button7";

button7.Size = new Size(284, 46);

button7.TabIndex = 32;

button7.Text = "Добавить картинку";

button7.UseVisualStyleBackColor = true;

button7.Click += button7\_Click;

//

// textBox6

//

textBox6.Location = new Point(35, 971);

textBox6.Name = "textBox6";

textBox6.Size = new Size(282, 39);

textBox6.TabIndex = 33;

//

// label14

//

label14.AutoSize = true;

label14.Location = new Point(809, 28);

label14.Name = "label14";

label14.Size = new Size(0, 32);

label14.TabIndex = 34;

//

// button8

//

button8.Location = new Point(35, 919);

button8.Name = "button8";

button8.Size = new Size(282, 44);

button8.TabIndex = 35;

button8.Text = "Изменить название на:";

button8.UseVisualStyleBackColor = true;

button8.Click += button8\_Click;

//

// listBox2

//

listBox2.FormattingEnabled = true;

listBox2.Location = new Point(424, 581);

listBox2.Name = "listBox2";

listBox2.Size = new Size(282, 260);

listBox2.TabIndex = 36;

listBox2.SelectedIndexChanged += listBox2\_SelectedIndexChanged;

//

// listBox3

//

listBox3.FormattingEnabled = true;

listBox3.Location = new Point(822, 581);

listBox3.Name = "listBox3";

listBox3.Size = new Size(282, 260);

listBox3.TabIndex = 37;

listBox3.SelectedIndexChanged += listBox3\_SelectedIndexChanged;

//

// listBox4

//

listBox4.FormattingEnabled = true;

listBox4.Location = new Point(1207, 581);

listBox4.Name = "listBox4";

listBox4.Size = new Size(282, 260);

listBox4.TabIndex = 38;

listBox4.SelectedIndexChanged += listBox4\_SelectedIndexChanged;

//

// label15

//

label15.AutoSize = true;

label15.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold);

label15.Location = new Point(424, 546);

label15.Name = "label15";

label15.Size = new Size(107, 32);

label15.TabIndex = 39;

label15.Text = "Области";

//

// label16

//

label16.AutoSize = true;

label16.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold);

label16.Location = new Point(822, 546);

label16.Name = "label16";

label16.Size = new Size(93, 32);

label16.TabIndex = 40;

label16.Text = "Города";

//

// label17

//

label17.AutoSize = true;

label17.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold);

label17.Location = new Point(1207, 546);

label17.Name = "label17";

label17.Size = new Size(100, 32);

label17.TabIndex = 41;

label17.Text = "Районы";

//

// comboBox1

//

comboBox1.FormattingEnabled = true;

comboBox1.Location = new Point(35, 85);

comboBox1.Name = "comboBox1";

comboBox1.Size = new Size(242, 40);

comboBox1.TabIndex = 42;

//

// label18

//

label18.AutoSize = true;

label18.Font = new Font("Segoe UI Semibold", 9F, FontStyle.Bold, GraphicsUnit.Point, 204);

label18.Location = new Point(35, 6);

label18.Name = "label18";

label18.Size = new Size(91, 32);

label18.TabIndex = 43;

label18.Text = "Поиск:";

//

// button9

//

button9.Location = new Point(1207, 847);

button9.Name = "button9";

button9.Size = new Size(282, 66);

button9.TabIndex = 44;

button9.Text = "Табличка района";

button9.UseVisualStyleBackColor = true;

button9.Click += button9\_Click;

//

// button10

//

button10.Location = new Point(1207, 919);

button10.Name = "button10";

button10.Size = new Size(282, 91);

button10.TabIndex = 45;

button10.Text = "Очистить табличку района";

button10.UseVisualStyleBackColor = true;

button10.Click += button10\_Click;

//

// button11

//

button11.Location = new Point(1567, 581);

button11.Name = "button11";

button11.Size = new Size(282, 46);

button11.TabIndex = 46;

button11.Text = "Устроить революцию";

button11.UseVisualStyleBackColor = true;

button11.Click += button11\_Click;

//

// button12

//

button12.Location = new Point(1567, 633);

button12.Name = "button12";

button12.Size = new Size(282, 74);

button12.TabIndex = 47;

button12.Text = "Проверить безопасность района";

button12.UseVisualStyleBackColor = true;

button12.Click += button12\_Click;

//

// button13

//

button13.Location = new Point(1567, 713);

button13.Name = "button13";

button13.Size = new Size(282, 74);

button13.TabIndex = 48;

button13.Text = "Сделать район опасным";

button13.UseVisualStyleBackColor = true;

button13.Click += button13\_Click;

//

// button14

//

button14.Location = new Point(1567, 793);

button14.Name = "button14";

button14.Size = new Size(282, 74);

button14.TabIndex = 49;

button14.Text = "Сделать район безопасным";

button14.UseVisualStyleBackColor = true;

button14.Click += button14\_Click;

//

// label19

//

label19.AutoSize = true;

label19.Location = new Point(683, 189);

label19.Name = "label19";

label19.Size = new Size(281, 32);

label19.TabIndex = 51;

label19.Text = "ID суб., в кот. располож.";

//

// numericUpDown5

//

numericUpDown5.Location = new Point(704, 224);

numericUpDown5.Name = "numericUpDown5";

numericUpDown5.Size = new Size(221, 39);

numericUpDown5.TabIndex = 50;

//

// Form1

//

AutoScaleDimensions = new SizeF(13F, 32F);

AutoScaleMode = AutoScaleMode.Font;

BackColor = Color.White;

ClientSize = new Size(1984, 1083);

Controls.Add(label19);

Controls.Add(numericUpDown5);

Controls.Add(button14);

Controls.Add(button13);

Controls.Add(button12);

Controls.Add(button11);

Controls.Add(button10);

Controls.Add(button9);

Controls.Add(label18);

Controls.Add(comboBox1);

Controls.Add(label17);

Controls.Add(label16);

Controls.Add(label15);

Controls.Add(listBox4);

Controls.Add(listBox3);

Controls.Add(listBox2);

Controls.Add(button8);

Controls.Add(label14);

Controls.Add(textBox6);

Controls.Add(button7);

Controls.Add(pictureBox1);

Controls.Add(button6);

Controls.Add(button5);

Controls.Add(button4);

Controls.Add(label13);

Controls.Add(listBox1);

Controls.Add(label12);

Controls.Add(numericUpDown4);

Controls.Add(label11);

Controls.Add(textBox5);

Controls.Add(button3);

Controls.Add(label10);

Controls.Add(numericUpDown3);

Controls.Add(label9);

Controls.Add(textBox4);

Controls.Add(label8);

Controls.Add(textBox3);

Controls.Add(label7);

Controls.Add(label6);

Controls.Add(button2);

Controls.Add(label5);

Controls.Add(numericUpDown2);

Controls.Add(label4);

Controls.Add(textBox2);

Controls.Add(label3);

Controls.Add(label2);

Controls.Add(label1);

Controls.Add(numericUpDown1);

Controls.Add(button1);

Controls.Add(richTextBox1);

Name = "Form1";

Text = "Обозреватель Государств";

((System.ComponentModel.ISupportInitialize)numericUpDown1).EndInit();

((System.ComponentModel.ISupportInitialize)numericUpDown2).EndInit();

((System.ComponentModel.ISupportInitialize)numericUpDown3).EndInit();

((System.ComponentModel.ISupportInitialize)numericUpDown4).EndInit();

((System.ComponentModel.ISupportInitialize)pictureBox1).EndInit();

((System.ComponentModel.ISupportInitialize)numericUpDown5).EndInit();

ResumeLayout(false);

PerformLayout();

}

#endregion

private RichTextBox richTextBox1;

private Button button1;

private NumericUpDown numericUpDown1;

private Label label1;

private Label label2;

private Label label3;

private Label label4;

private TextBox textBox2;

private Label label5;

private NumericUpDown numericUpDown2;

private Button button2;

private Label label6;

private Label label7;

private TextBox textBox3;

private Label label8;

private TextBox textBox4;

private Label label9;

private NumericUpDown numericUpDown3;

private Label label10;

private Label label11;

private TextBox textBox5;

private Button button3;

private Label label12;

private NumericUpDown numericUpDown4;

private ListBox listBox1;

private Label label13;

private Button button4;

private Button button5;

private Button button6;

private PictureBox pictureBox1;

private Button button7;

private TextBox textBox6;

private Label label14;

private Button button8;

private ListBox listBox2;

private ListBox listBox3;

private ListBox listBox4;

private Label label15;

private Label label16;

private Label label17;

private ComboBox comboBox1;

private Label label18;

private Button button9;

private Button button10;

private Button button11;

private Button button12;

private Button button13;

private Button button14;

private Label label19;

private NumericUpDown numericUpDown5;

}

}