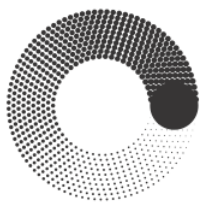


ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет Информационных технологий  
Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»

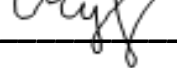
## КУРСОВОЙ ПРОЕКТ

Дисциплина: Объектно-ориентированное программирование

Тема: WPF Кликер

Выполнил(а): студент(ка) группы 221-371

Чуприна С.В.  
(Фамилия И.О.)

Дата, подпись \_\_\_\_\_  
(Дата) (Подпись) 

Проверил: \_\_\_\_\_  
(Фамилия И.О., степень, звание)

Дата, подпись \_\_\_\_\_  
(Дата) (Подпись)

Замечания: \_\_\_\_\_

Москва

2023

## Оглавление

<b>ВВЕДЕНИЕ</b> .....	3
Описание проекта .....	3
Цели проекта.....	3
Задачи проекта.....	3
<b>ГЛАВА 1. Аналитическая</b> .....	5
Анализ требований.....	5
Проектирование .....	6
<b>ГЛАВА 2. Технологическая</b> .....	9
Реализация.....	9
Тестирование.....	13
<b>ЗАКЛЮЧЕНИЕ</b> .....	17
<b>Список литературы</b> .....	19

# ВВЕДЕНИЕ

## Описание проекта

“WPF Кликер” – это приложение, написанное на фреймворке WPF с использованием языка C#. Основной идеей проекта является разработка интерактивного приложения, где пользователь может выполнять клики мыши для сбора достижений в разных игровых режимах. “WPF Кликер” предусматривает регистрацию пользователя для хранения личных достижений.

## Цели проекта

1. Изучение принципов ООП на основе разработки приложения.
2. Использование принципов ООП: инкапсуляция, полиморфизм, наследование и принципов SOLID.
3. Реализация логики и разграничение логики кода для удобочитаемости и поддерживаемости.

Разработка чистого и структурированного кода, обеспечивающего легкость поддержки и модификации приложения.

4. Организация работы с данными через XML.

XML (eXtensible Markup Language — расширяемый язык разметки) — это язык программирования для создания логической структуры данных, их хранения и передачи в виде, удобном и для компьютера, и для человека<sup>1</sup>.

Использование XML для эффективной работы с данными, создание моделей данных, и обеспечение взаимодействия с данными, то есть разработать методы для сериализации и десериализации данных в/из формата XML.

## Задачи проекта

1. Разработка интерактивного кликера.

Создание приложения, позволяющего пользователям получать достижения, накапливать очки через клики мыши.

2. Интеграция авторизации.

Реализация системы авторизации для управления доступом к функционалу приложения.

3. Дополнение функционала системой вознаграждения.

Внедрение механизма вознаграждений для расширения возможной активности пользователей, то есть, разработка системы достижений.

4. Создание проработанного графического и визуального сопровождения.

Разработка эстетически приятного пользовательского интерфейса с использованием элементов WPF, включая графические элементы, анимации и звуковые эффекты.

## **ГЛАВА 1. Аналитическая**

### **Анализ требований**

#### **1. Разработка интерактивного кликера:**

Описание функциональности:

Приложение должно предоставлять пользователю возможность взаимодействовать с интерфейсом посредством кликов мыши для подсчёта очков, получения игровых наград при достижении определённых условий и сравнение своих значений с максимальным результатом.

Основные требования:

Отслеживание количества очков, накапливаемых посредством кликов мыши.  
Обнуление очков при отсутствии нажатий более 1с.

Механизм накопления достижений и их отображения.

#### **2. Интеграция авторизации:**

Описание функциональности:

Возможность зайти через личный аккаунт (логин, пароль) и просмотреть полученные достижения.

Основные требования:

Создание учетных записей с обязательным вводом уникального логина и пароля.

#### **3. Дополнение функционала системой вознаграждения:**

Описание функциональности:

Механизм вознаграждения призван стимулировать активность пользователя и улучшить игровой опыт. Это включает в себя систему достижений.

Основные требования:

Система отслеживания достижений пользователя.

Выдача наград.

Отображение рейтинга.

#### **4. Создание проработанного графического и визуального сопровождения:**

Описание функциональности:

Дизайн и визуальное оформление приложения должны быть не топорными, минималистичными и визуально приятными. Это включает в себя работу с графическими элементами, анимацией, звуковыми эффектами и общей стилистикой, создающей приятное визуальное восприятие.

Основные требования:

Редактирование элементов WPF для создания современного пользовательского интерфейса.

Подбор и использование звуковых эффектов, соответствующих событиям в приложении.

### **Проектирование**

(Проектирование классов и взаимодействия между ними согласно принципам ООП)

При проектировании классов необходимо учитывать и следовать таким принципам ООП, в том числе SOLID, как:

- инкапсуляция – сокрытие внутренней реализации класса и предоставление интерфейса для взаимодействия с классом;
- абстракция – создание абстрактных типов данных на основе выделения общих характеристик и функциональности классов;
- наследование – наследование полей и методов другого класса для расширения функциональности;

- полиморфизм – возможность объектов разных типов использовать одинаковый интерфейс;<sup>ii</sup>
- принцип единой ответственности (Single Responsibility Principle - SRP) – каждый класс имеет только одну причину для изменения, то есть выполняет только одну задачу;<sup>iii</sup>
- принцип открытости/закрытости (Open/Closed Principle - OCP) – каждый класс должен быть открыт для расширения и закрыт для изменения;
- принцип подстановки Лисков (Liskov substitution principle - LSP) – функции, использующие базовый тип, должны принимать подтипы базового типа, не зная об этом, то есть, работая корректно;
- принцип разделения интерфейса (Interface Segregation Principle - ISP) – лучше много специальных интерфейсов, чем один интерфейс общего назначения;
- принцип инверсии зависимостей (Dependency Inversion Principle - DIP) – создание слабосвязанных систем, где высокоуровневые модули не зависят от низкоуровневых модулей, благодаря зависимости обоих типов модулей от абстракции.<sup>iv</sup>

Исходя из этих принципов, были выделены следующие классы:

Класс `PlayerData` – работа с данными пользователей, XML-форматом.

Класс `UserData` – класс полей пользователей, сериализация и десериализация которых выполняется классом `PlayerData`.

Класс `Sounds` – работа со звуком – его воспроизведение - при нажатии на кнопки, переключение между окнами, наведение на некоторые графические элементы.

Класс `Points` – подсчёт очков, обновление и предоставление информации об их количестве, максимальном значении.

Класс `ButtonClick` – выполнение действий в соответствии с нажатой кнопкой.

Класс `TimeCheck` – проверка задержки между нажатиями на кнопку пользователем, для дальнейшего увеличения или обнуления количества очков.

Класс `BooleanToVisibilityConverter` – реализует встроенный интерфейс `IValueConverter`, необходим для отображения наличия или отсутствия достижений у пользователя.

Это логические компоненты приложения, а также имеются классы, отвечающие за визуальные элементы, такие как окна. В этих классах вызываются методы, определенные в логических компонентах, для обеспечения взаимодействия между логикой и пользовательским интерфейсом.

Для соблюдения принципа DIP (Принципа инверсии зависимостей) и предотвращения прямой зависимости между классами, например, такими как `ButtonClick` и `Sounds`, были реализованы следующие интерфейсы:

- `IButtonService`,
- `IPoint`,
- `ISounds`.

Предварительная структура проекта:

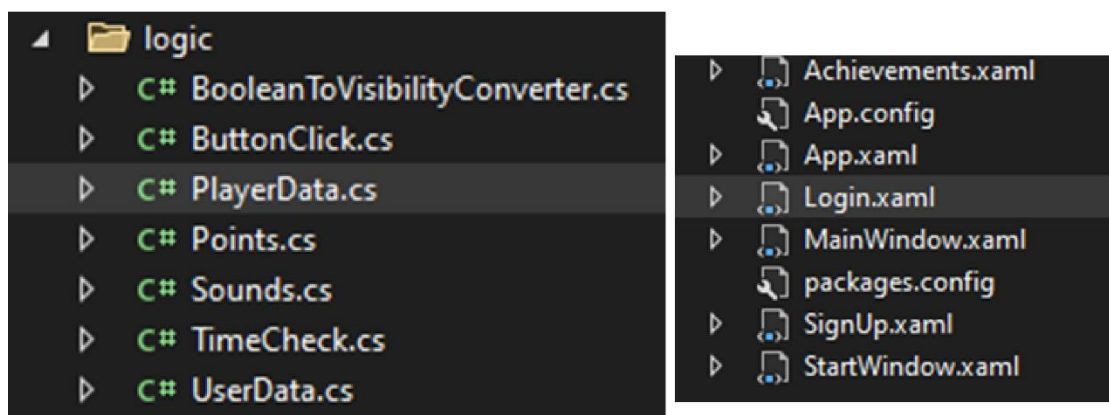


Рисунок 1.1



## ГЛАВА 2. Технологическая

### Реализация

Описание реализации каждого класса и функциональности

Класс UserData - структура включает в себя необходимую информацию для идентификации и хранения достижений пользователя. Ключевыми атрибутами класса являются имена пользователей, соответствующие пароли, а также четыре булевых свойства, отражающих наличие или отсутствие определенных достижений в контексте игры.

Сериализация является важной чертой класса UserData, предоставляя возможность представления объектов этого класса в форме, пригодной для сохранения и передачи данных. Этот процесс обеспечивает сохранение состояния пользователя, что может быть полезным, например, при сохранении прогресса игрока.

PlayerData, в свою очередь, выступает в роли класса, который реализует функциональность, связанную с регистрацией и аутентификацией пользователей, а также сохранением их данных в формате XML.

Ключевые компоненты класса включают константу, предназначенную для хранения пути к файлу формата XML, где будут храниться данные пользователей и список объектов типа UserData, который используется для хранения информации о пользователях.

Таблица 2.1

```
private const string FilePath = "E:\\учёба\\3  
семестр\\00П\\Кликер\\credentials.xml";  
private List<UserData> userCredentials;
```

В классе содержатся два открытых метода, связанные с регистрацией и аутентификацией. В качестве параметров они принимают две строки - username и password, которые пользователь вводит в диалоговом окне. Ниже приведён пример метода, который проверяет наличие username в словаре. При наличии совпадения выполняется проверка введенного пользователем пароля на соответствие тому, что хранится в словаре по указанному имени.

Таблица 2.2

```
public bool AuthenticateUser(string username, string password)
{
    if (userCredentials.TryGetValue(username, out string
storedPassword))
    {
        return storedPassword == password;
    }
    else
    {
        Console.WriteLine("Authentication failed.. Incorrect
password.");
    }
    return false; // пользователь с таким именем не найден
}
```

Помимо представленных примеров, в классе также реализованы методы, отвечающие за сохранение и доступ к данным в формате XML.

Sounds занимается воспроизведением звуков в приложении, таких как звуки кнопок, переключение между окнами и обратная связь при взаимодействии с графическими элементами. Для соблюдения принципов ООП и SOLID класс Sounds реализует интерфейс ISounds.

Интерфейс включает в себя необходимые методы, предназначенные для использования в событиях. Такой подход обеспечивает гибкость взаимодействия с различными реализациями звуковых эффектов, обеспечивая при этом единообразие интерфейса для взаимодействия с классом.

Внутри класса реализован приватный метод PlayButtonSounds, который принимает в себя экземпляр класса SoundPlayer. Этот метод отвечает за воспроизведение звуковых эффектов, специфических для взаимодействия с кнопками, и использует SoundPlayer для обработки звуковых файлов формата WAV.

Ниже представлен пример реализации этого метода с использованием обработчика ошибок – блока try-catch:

Таблица 2.3

```
private void PlayButtonSounds(SoundPlayer soundPlayer)
{
    try
    {
        soundPlayer.Play();
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Ошибка воспроизведения звука:
{ex.Message}");
    }
}
```

Остальная логика подобна этому фрагменту:

Таблица 2.4

```
private readonly SoundPlayer click = new SoundPlayer("zz/клик.wav");
public void AllSounds()
{
    PlayButtonSounds(click);
}
```

Класс Points ответственен за систему подсчета и управления количеством очков игрока. Кроме того, он обладает функциональностью хранения информации о максимальном достигнутом значении очков за текущий сеанс игры. Реализуя интерфейс IPoint, класс обеспечивает единый набор методов, необходимых для эффективного взаимодействия с системой управления очками.

Пример некоторых методов класса – увеличение очков (вызывается при клике), обнуление очков (при паузе более 1с между нажатиями), сравнение с максимальным значением и его обновление. Кроме того, реализованы методы, связанные с отображением информации класса на графических элементах.

Класс ButtonClick является реализацией интерфейса IButtonService, что позволяет ему предоставлять единый набор методов для взаимодействия с элементами управления в различных сценариях.

В частности, метод MainButtonClick обрабатывает событие основного нажатия кнопки, при котором происходит подсчет и обновление очков, а также визуальное отображение результата. Методы MainClickMenu и StartClick отвечают за переход между главным меню и стартовым окном приложения.

Методы SignSaveClick и LoginClick связаны с регистрацией и аутентификацией пользователя, предоставляя визуальные обратные связи об успешности или ошибке этих процессов.

И последнее - метод MainAchievementsClick обеспечивает переход к окну достижений приложения.

Таблица 2.5

```
public void MainAchievementsClick(MainWindow window)
{
    Achievements achievements = new Achievements(window);
    achievements.Show();
    window.Hide();
}
```

TimeCheck представляет собой класс, специализированный для выполнения проверок временных задержек между последовательными действиями, а именно - нажатиями кнопок пользователем. В контексте данного класса акцент сделан на аспекте управления временем, что позволяет регулировать и контролировать интервалы времени между событиями. Более функционала в классе нет.

Ниже приведён пример одного из методов:

Таблица 2.6

```
DateTime lastTime = DateTime.MaxValue; //по дефолту - значение,
которое не может быть достигнуто
TimeSpan distinctionOnClick = TimeSpan.Zero; //"расстояние" во
времени между нажатиями
TimeSpan maxDistinction = TimeSpan.FromSeconds(1);
//проверка на задержку между нажатиями
//если значение True, то значение очков необходимо обнулить
```

```
public bool CheckedTimeClick()
{
    if (lastTime != DateTime.MaxValue)
    {
        distinctionOnClick = DateTime.Now - lastTime;
        if (CheckDistinction())
        {
            lastTime = DateTime.Now;
            return true;
        }
        lastTime = DateTime.Now;
    }
    else { lastTime = DateTime.Now; }
    return false;
}
```

## Дизайн

Для создания приятного дизайна, была выбрана общая цветовая палитра — мягкие пастельные цвета, пиксельная тематика.

Были созданы разные текстовые и графические элементы, добавляющие интересные детали, и добавлены небольшие анимации для кнопок — теперь они меняют цвет при наведении или нажатии, а также установлены видео-обои в соответствующей пиксельной тематике.

Были округлены кнопки для создания более приятного, уникального и гармоничного дизайна.

Пример:



Рисунок 2.1

Дополнительно, для большего погружения пользователя, были добавлены звуковые эффекты: при нажатии на кнопку-кликер и кнопки навигации, а также фоновые аудио-эффекты. Эти звуки гармонично сочетаются с общей тематикой, обогащая визуальный опыт звуковой составляющей.

Ниже представлено, как вписаны в дизайн созданные элементы.

Кнопка “Start” отличается по цвету именно потому, что на неё был наведен курсор в момент скриншота.

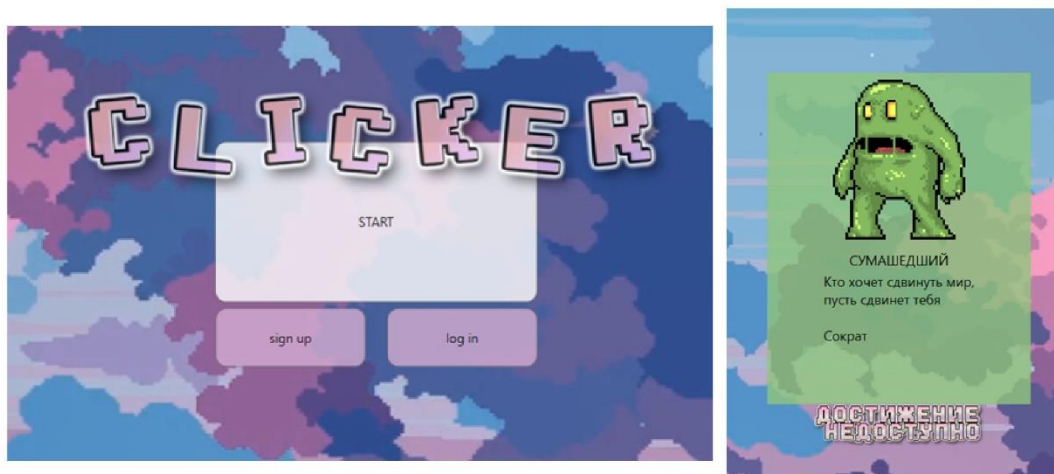


Рисунок 2.2

## Тестирование

(Описание проведенных тестов и результатов)

В связи с простотой логики приложения и его графическим характером основной акцент в процессе тестирования делается на визуальной проверке с использованием метода проб и ошибок.

### 1. Тесты на проверку работы кнопок.

Работу с кнопками реализует класс `ButtonClick`.

В приложении есть кнопки разной функциональности: кнопка-кликер, кнопки перехода между окнами, кнопки регистрации и входа.

Все кнопки выполняют необходимые действия: кнопка-кликер верно считает количество кликов, кнопки перехода между окнами закрывают активное окно и открывают нужное, кнопка регистрации сохраняет пользователя, а кнопка входа – позволяет войти в аккаунт зарегистрированному пользователю.

Тесты были проведены визуально.

### 2. Тест на проверку работы класса `TimeCheck`.

Класс ответственен за то, чтобы при кликании пользователя на кнопку проверять, не больше ли 1 секунды разница времени между нажатиями пользователя на кнопку-кликер. Класс работает корректно, и всегда происходят соответствующие задержки последствия – либо увеличение

количества очков, либо их обнуление (за действия, которые произойдут после задержки в 1с ответственен класс Point, а за вызов этих действий для нужной кнопки – класс ButtonClick).

Пример обнуления:



Рисунок 2.3

### 3. Обновление очков, их отображение – класс Points.

Класс, как было озвучено выше, верно считает количество кликов.

Все итерации нажатий на кнопку выводятся в TextBox, соответственно, визуально видно, что класс верно отображает количество очков каждой итерации.

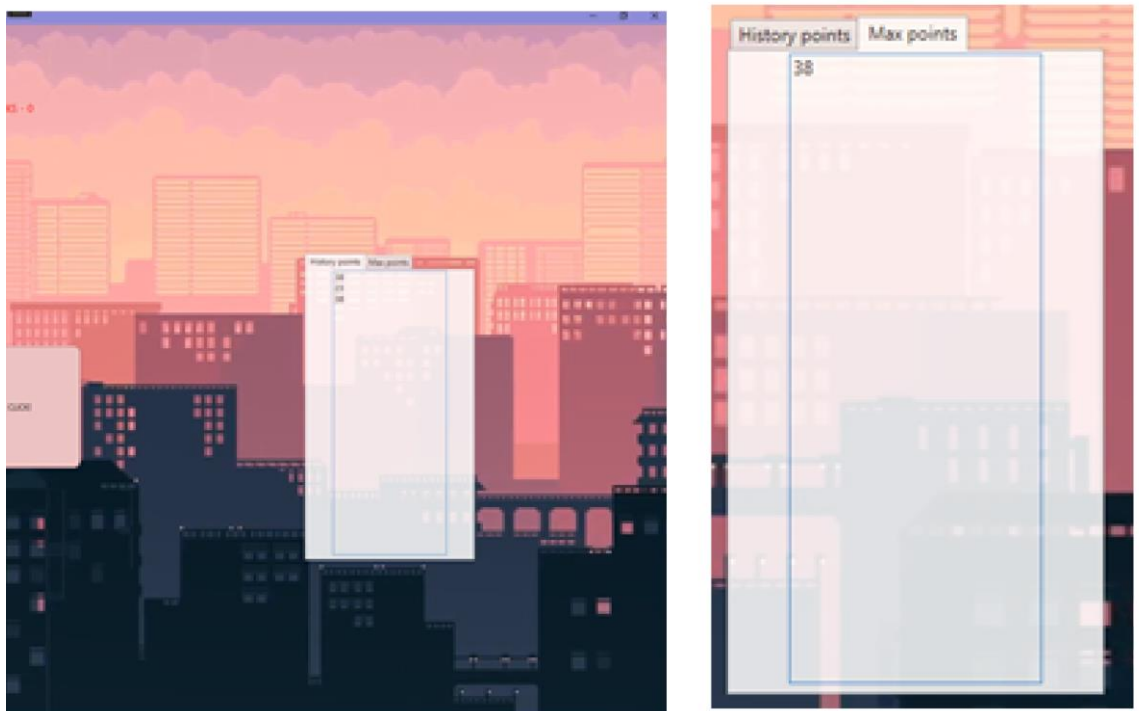


Рисунок 2.4

### 4. Воспроизведение звуков – класс Sounds.

Воспроизведение аудиофайлов должно быть в трёх случаях: нажатие на обычные кнопки (то есть все, кроме кнопки-кликера), нажатие на кнопку-кликер, наведение на панель (StackPanel) с достижением.

Во всех случаях воспроизводились ожидаемые звуки, независимо от количества нажатий и других факторов.

#### 5. Авторизация с сохранением данным в XML формате.

Для проверки работы потребовалось найти файл, в котором хранятся данные пользователей – credentials.xml (учетные данные) и произвести регистрацию пользователя. Ниже представлено содержание файла.

Все данные записываются в нужном формате и с верным значением.

Таблица 2.7

```
<ArrayOfUserData
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <UserData>
    <Username>soia</Username>
    <Password>1111</Password>
    <BoolAchievement1>true</BoolAchievement1>
    <BoolAchievement2>false</BoolAchievement2>
    <BoolAchievement3>false</BoolAchievement3>
    <BoolAchievement4>false</BoolAchievement4>
  </UserData>
</ArrayOfUserData>
```

Все вышеуказанные тесты были проведены преимущественно методом визуальной проверки, что подразумевает непосредственное взаимодействие с графическим интерфейсом приложения. Полученные результаты тестов подтвердили корректное функционирование всех рассмотренных элементов и соответствие ожидаемым результатам.

Таким образом, тестирование позволило подтвердить корректность работы приложения в рамках предназначения и функциональности.



## ЗАКЛЮЧЕНИЕ

(Обобщение результатов, описание полученного опыта).

Опыт разработки:

### 1) Использование принципов ООП

Классы были организованы с соблюдением принципа инкапсуляции. Это позволило эффективно управлять доступом к данным и методам, обеспечивая безопасность и структурированность кода. Код с применением инкапсуляции стал чище, легче для понимания.

Также применение наследования способствовало повторному использованию кода и созданию иерархии классов для более логичной организации функциональности, обеспечивая гибкость и легкость расширения приложения.

Следование принципу полиморфизма позволило создавать перегруженные методы и использовать интерфейсы, что упростило взаимодействие между классами.

Соблюдение этих принципов также связано с принципами SOLID, которые более глубоко и конкретно рассматривают принципы ООП.

Так, например, не возникло трудностей с реализацией принципа открытости/закрытости (ОСР), так как он достаточно схож с принципами ООП и интуитивно понятен, единой ответственности (SRP). Однако такие принципы SOLID, как принцип разделения интерфейса (ISP) и принцип инверсии зависимости (DIP) были новы, а последний – сложен в понимании его реализации. Однако их использование, несмотря на сложность на начальных этапах работы, сделало код более простым и модульным.

Реализация этих принципов требовала более тщательного проектирования и планирования. Однако, благодаря этому, код стал более легким для понимания и поддержки. Все изменения функциональности стало проще внедрять, не затрагивая существующую логику.

### 2) Структурирование кода

Разделение логики приложения на отдельные классы, соответствующие конкретным функциональным блокам, обеспечило легкость поддержки и модификации кода, простоту реализации логики.

А применение SOLID-принципов позволило создать гибкую архитектуру, способствующую расширению функциональности приложения.

### 3) Организация взаимодействия объектов

Использование событий обеспечило эффективное взаимодействие между различными элементами приложения.

Работа с интерфейсами способствовала созданию гибких механизмов взаимодействия и реализации принципов SOLID.

Разработка WPF-кликера в парадигме ООП позволила не только создать функциональное приложение, но и приобрести ценный опыт в области объектно-ориентированного проектирования. Применение принципов ООП сделало код более читаемым, поддерживаемым и гибким для будущих изменений. А новый уровень понимания SOLID-принципов позволил мне увидеть разработку программного обеспечения как более стратегический и интеллектуальный процесс.

Этот опыт стал полезным шагом в познании особенностей разработки в WPF и подтвердил эффективность принципов ООП в создании качественных приложений.

## Список литературы

- 
- i XML // SKILLFACTORY MEDIA [Электронный ресурс]. URL: <https://goo.su/oVRmph8> (дата обращения 2.12.2023)
- ii Концепции объектно-ориентированного программирования JAVA // JAVA RUSH [Электронный ресурс]. URL: <https://goo.su/mvDn> (дата обращения 2.12.2023)
- iii Принципы SOLID на примерах // Хабр [Электронный ресурс]. URL: <https://goo.su/iBu5FmE> (дата обращения 2.12.2023)
- iv SOLID — принципы объектно-ориентированного программирования // Web Creator [Электронный ресурс]. URL: <https://goo.su/dmfbxz> (дата обращения 5.12.2023)