Министерство образования и науки Российской Федерации

ФГАОУ ВО «КФУ им. В.И. Вернадского»

физико-технический институт (структурное подразделение)

|  |
| --- |
|  |

Кафедра компьютерной инженерии и моделирования

Пугач Давид Сергеевич

отчет по практической работе №5  
по дисциплине **«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»**

Направление подготовки:

09.03.04 "Программная инженерия"

Оценка -



Симферополь, 2023

**Практическая работа №5  
Тема: Наследование классов и использование интерфейсов**

**Цель работы:** Научиться на практике наследовать классы и интерфейсы. Разобраться на практике с вопросами отличия интерфейсов и абстрактных классов, реализовать множественное наследование интерфейсов и явную реализация интерфейсов. Научиться строить архитектуру приложения с помощью интерфейсов, разобраться с преимуществами такого подхода при промышленном программировании.

**Ключевые понятия:**

Interface, Abstract Class, множественное наследование интерфейсов, явная реализация интерфейсов.

**Перед выполнением лабораторной работы изучена следующая литература:**

1. Изучить презентацию лектора курса: «Интерфейсы» (материалы доступны в облаке mail.ru).
2. Сайт Metanit.com
3. Справочник по C#. Корпорация Microsoft.   
   http://msdn.microsoft.com/ru-ru/library/618ayhy6.aspx
4. Биллиг В.А. Основы программирования на C#. Интернет-университет информационных технологий. http://www.intuit.ru/studies/courses/2247/18/info
5. Павловская Т. Программирование на языке высокого уровня C#. http://www.intuit.ru/studies/courses/629/485/info
6. Руководство по программированию на C#. Корпорация Microsoft.   
   http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx
7. Корпорация Microsoft. C#. Спецификация языка.

**Задание 1: Использование интерфейсов**

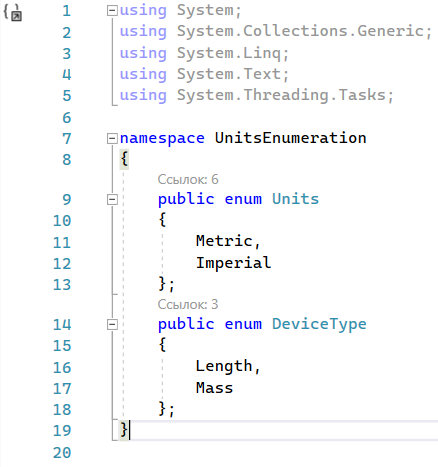


Рисунок 1. Перечисления

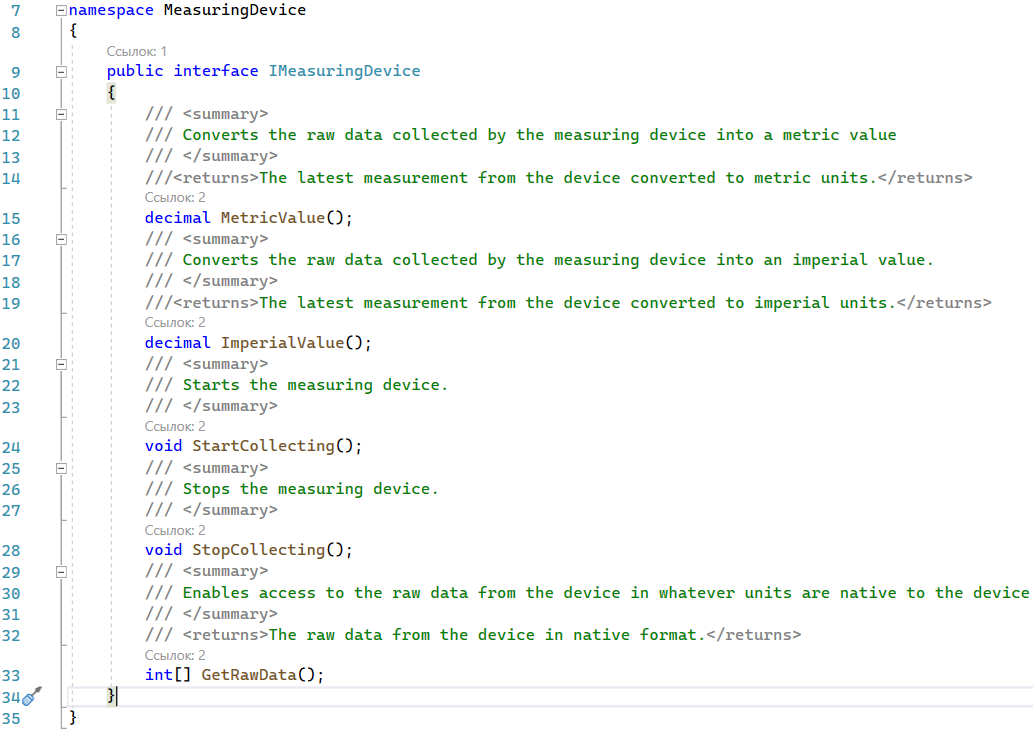


Рисунок 2. Интерфес IMeasuringDevice



Рисунок 3. Класс DeviceController

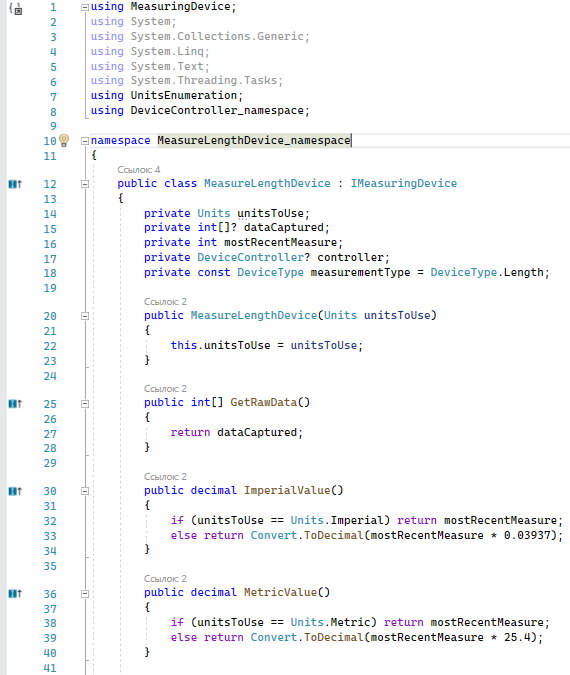
 

Рисунок 4. Класс MeasureLenghtDevice

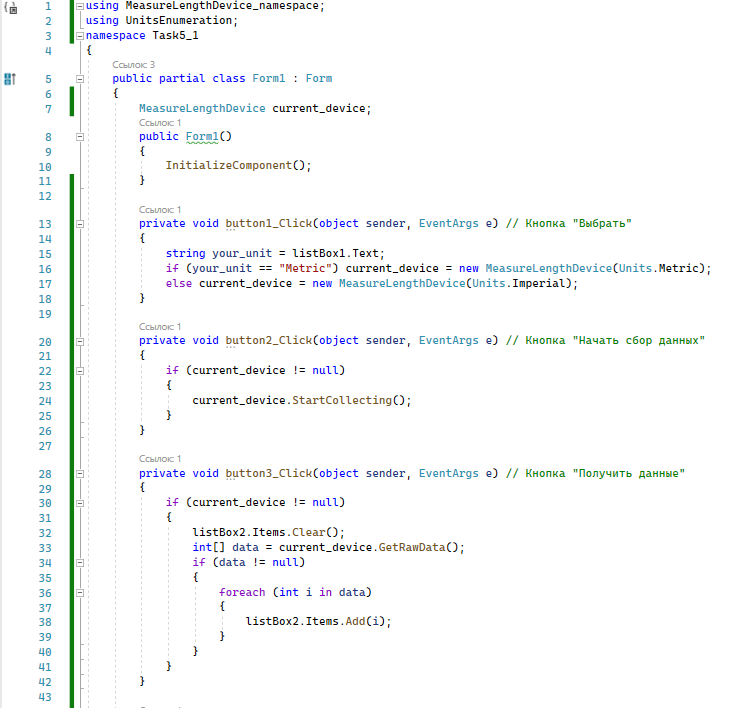
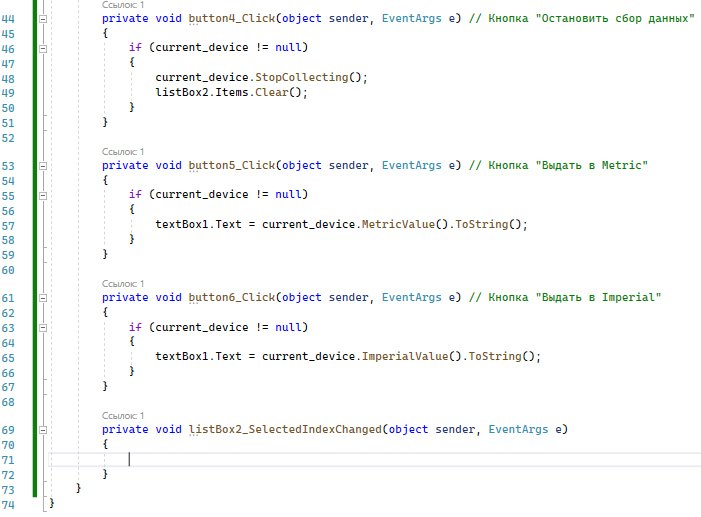
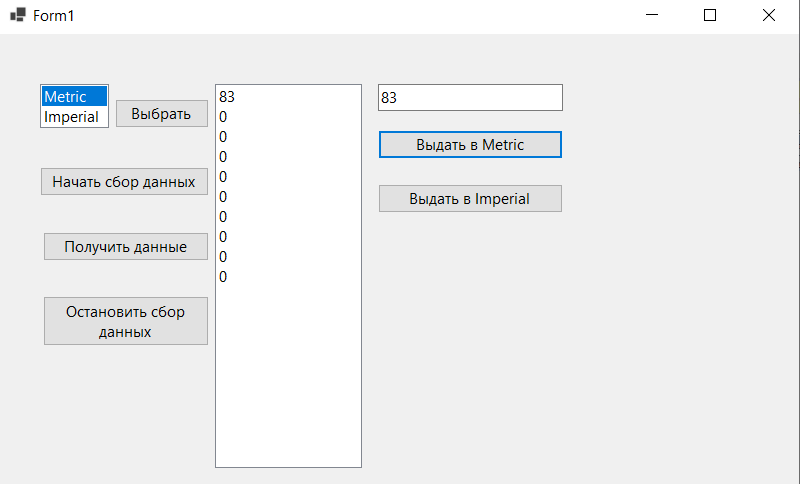
 

Рисунок 5. Form1.cs



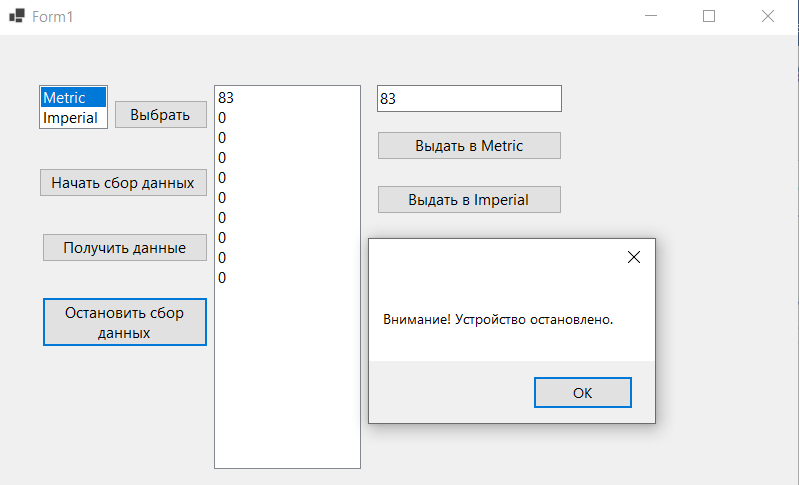


Рисунок 6. Пример работы программы.

**Задание 2: Создание абстрактного класса**

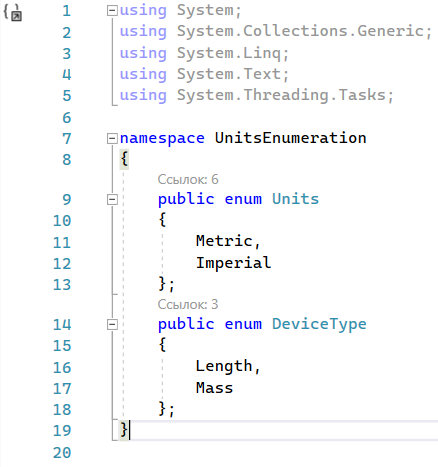


Рисунок 7. Перечисления

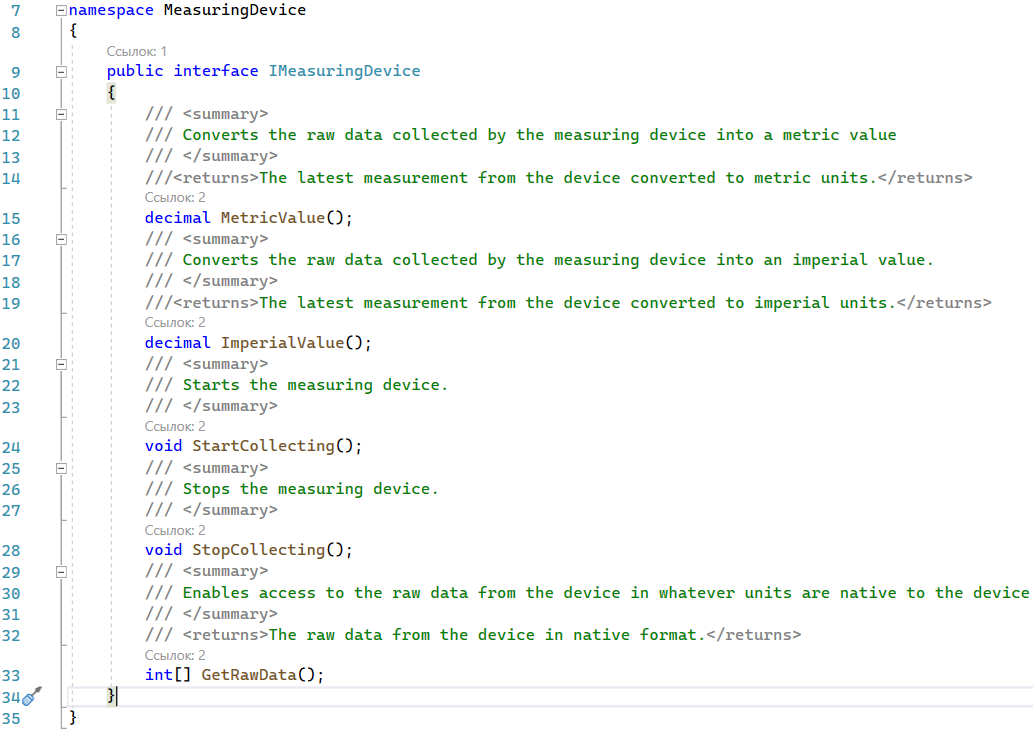


Рисунок 8. Интерфес IMeasuringDevice



Рисунок 9. Класс DeviceController

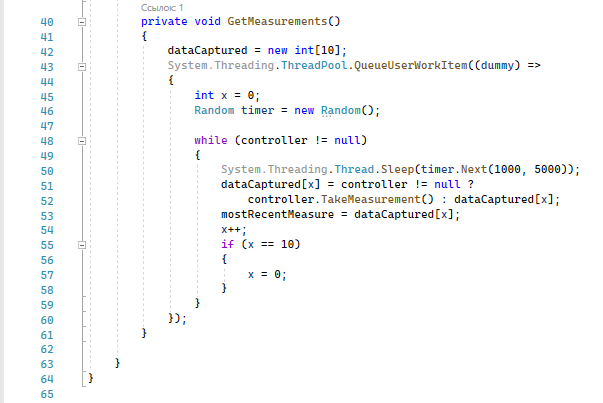


Рисунок 10. Абстрактный класс MeasureDataDevice

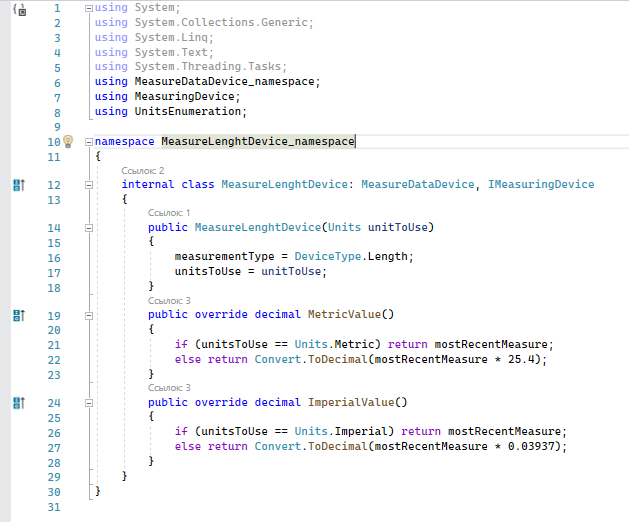


Рисунок 11. Класс MeasureLenght

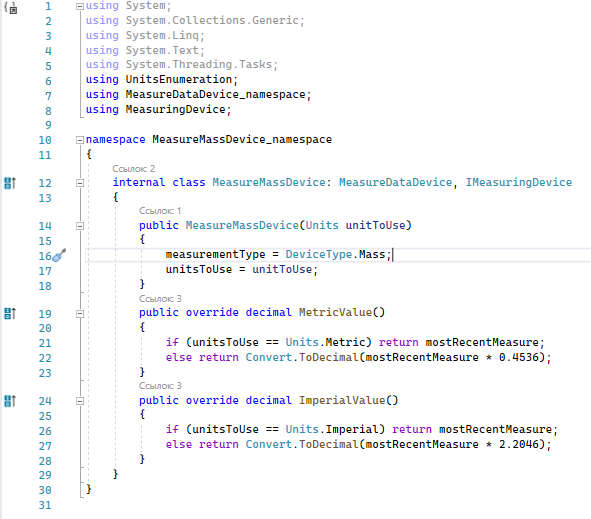


Рисунок 12. Класс MeasureMassDevice

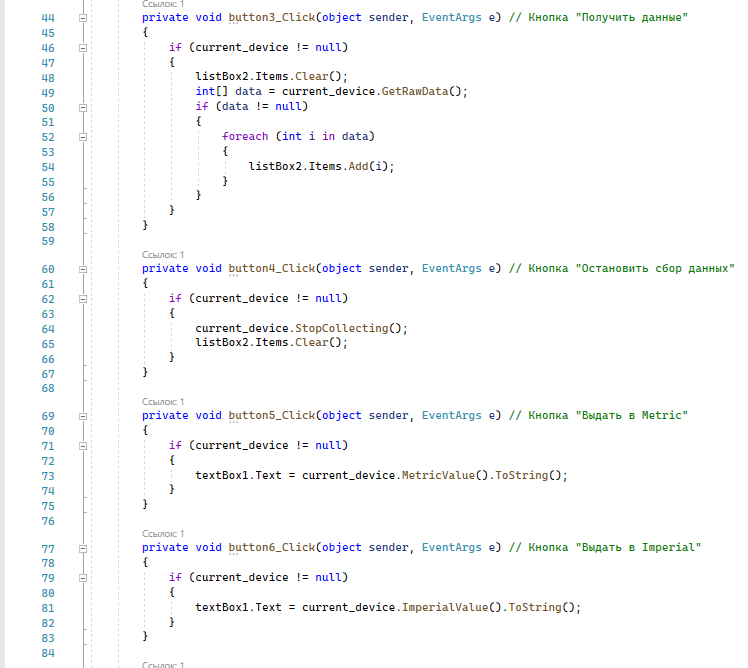
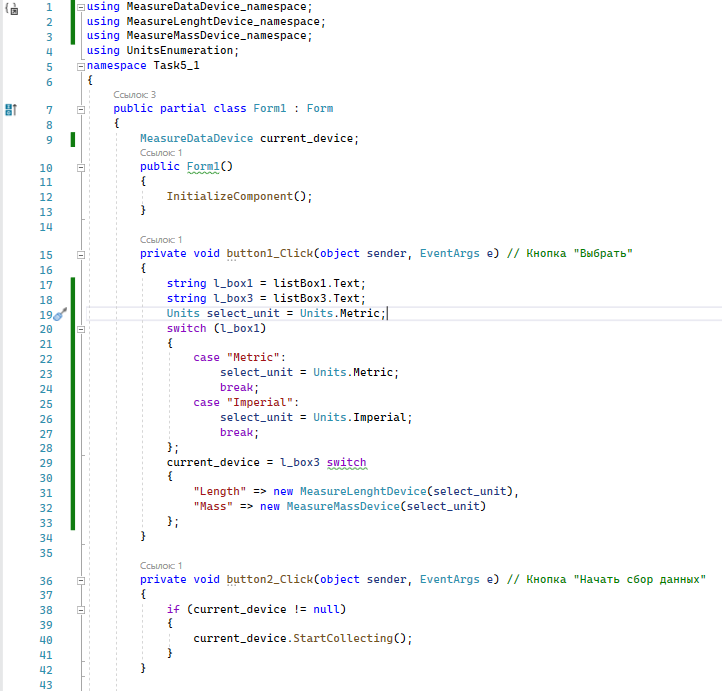
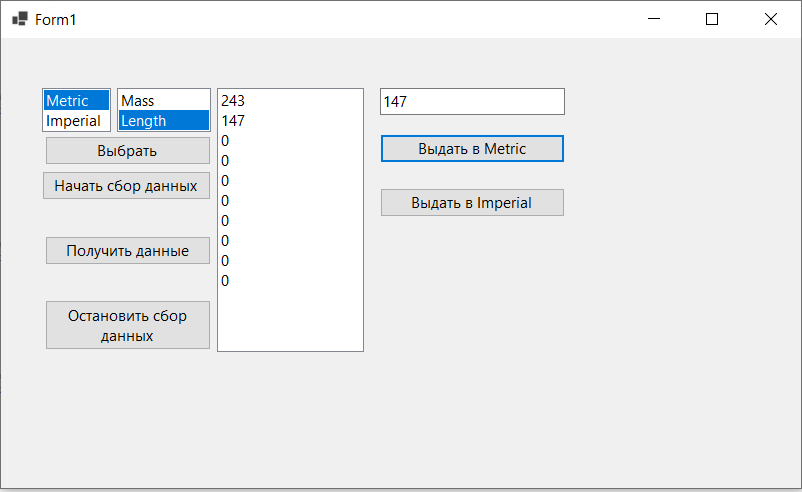
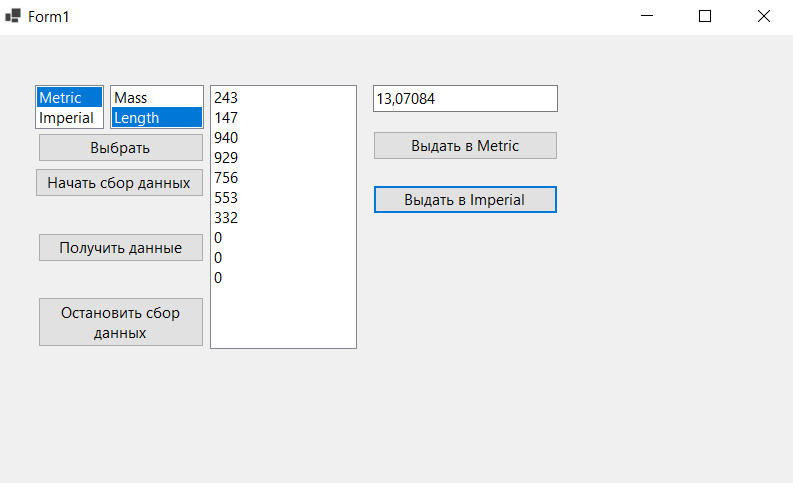
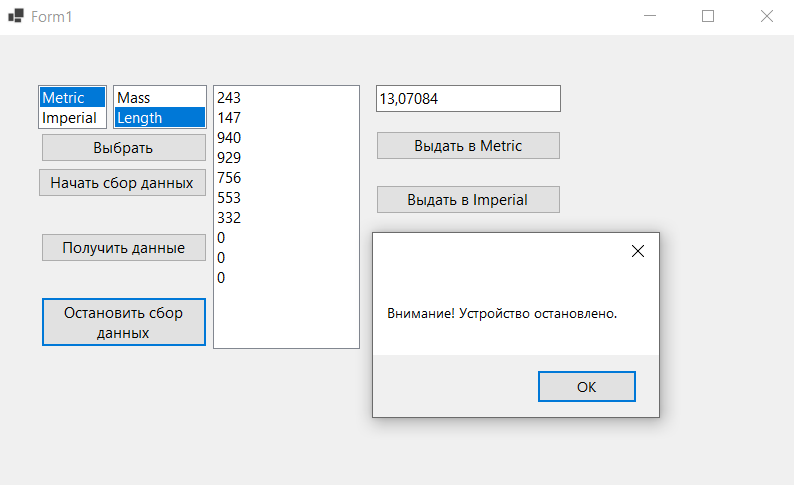


Рисунок 13. Form1.cs







**Задание 3: Создание архитектуры приложения с помощью интерфейсов**

Интерфейсы удобны при:

* Проектировании приложения
* Работе в команде
* Разделении на слои
* Реализации полиморфной связи
* Использовании DI контейнеров
* Unit тестировании

Один из принципов проектирования гласит, что при создании системы классов надо программировать на уровне интерфейсов, а не их конкретных реализаций. Под интерфейсами в данном случае понимаются не только типы C#, определенные с помощью ключевого слова interface, а определение функционала без его конкретной реализации. То есть под данное определение попадают как собственно интерфейсы, так и абстрактные классы, которые могут иметь абстрактные методы без конкретной реализации.

В этом плане у абстрактных классов и интерфейсов много общего. Нередко при проектировании программ в паттернах мы можем заменять абстрактные классы на интерфейсы и наоборот. Однако все же они имеют некоторые отличия.

Когда следует использовать абстрактные классы:

* Если надо определить общий функционал для родственных объектов
* Если мы проектируем довольно большую функциональную единицу, которая содержит много базового функционала
* Если нужно, чтобы все производные классы на всех уровнях наследования имели некоторую общую реализацию. При использовании абстрактных классов, если мы захотим изменить базовый функционал во всех наследниках, то достаточно поменять его в абстрактном базовом классе.
* Если же нам вдруг надо будет поменять название или параметры метода интерфейса, то придется вносить изменения и также во всех классы, которые данный интерфейс реализуют.

Когда следует использовать интерфейсы:

* Если нам надо определить функционал для группы разрозненных объектов, которые могут быть никак не связаны между собой.
* Если мы проектируем небольшой функциональный тип

Ключевыми здесь являются первые пункты, которые можно свести к следующему принципу: если классы относятся к единой системе классификации, то выбирается абстрактный класс. Иначе выбирается интерфейс.

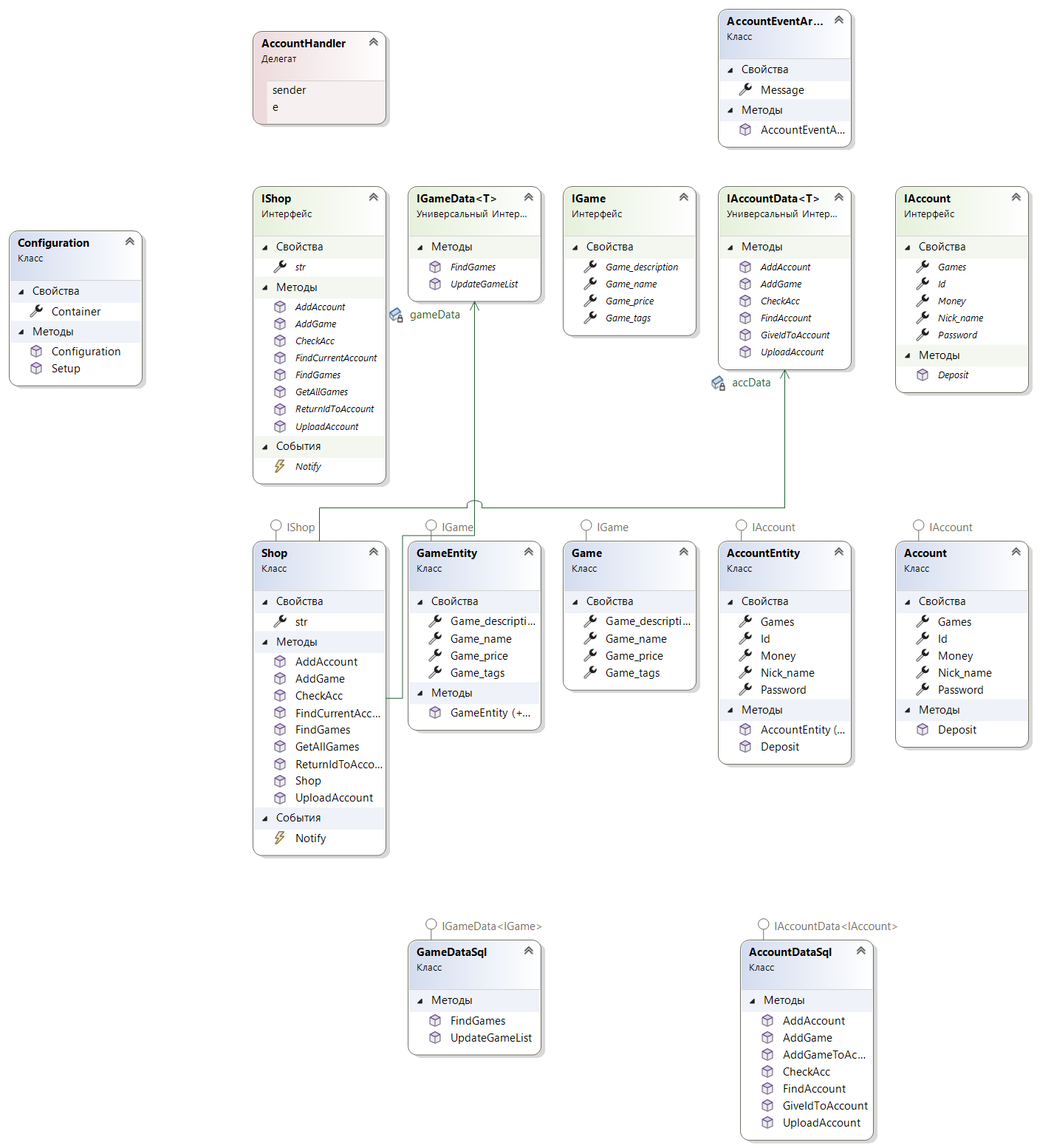
****

Диаграмма классов

**Представлены 3 проекта, реализованных в Visual Studio Community 2022.  
Проекты представлены преподавателю в электронной форме, продемонстрирована их работоспособность, разъяснены детали программного кода.**

**Ссылка на репозиторий с программами**

https://github.com/Bananaws/Object-oriented-programming