|  |
| --- |
| **Министерство образования и науки Российской Федерации**  Федеральное государственное автономное образовательное  учреждение высшего образования  **«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**  **ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»** |

Подразделение: Инженерная школа энергетики

Направление подготовки: 09.04.03 – Прикладная информатика

Отделение: Электроэнергетики и электротехники

**Проектная документация**

**Отчёт по лабораторной работе №5**

по дисциплине: «Основы объектно-ориентированного программирования»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент гр. О-5КМ11 | |  | \_\_\_\_\_\_\_\_\_\_ | |  | Сухов Е.Р. |
|  | |  |  | |  |  |
|  | |  |  |  | | |
|  | |  |  | |  |  |
| Отчёт принял | доцент, к.т.н. |  | \_\_\_\_\_\_\_\_\_\_ | |  | Калентьев А. А. |
|  |  |  |  | |  |  |
|  |  |  |  | \_\_\_ \_\_\_\_\_\_ | | |

Томск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc136563218)

[1 Основная часть 4](#_Toc136563219)

[1.1 UML диаграмма вариантов использования 4](#_Toc136563220)

[1.2 UML диаграмма классов 5](#_Toc136563221)

[1.3 Описание классов, образующих связь типа «общее-частное» 7](#_Toc136563222)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9](#_Toc136563223)

**ВВЕДЕНИЕ**

Корректная и полная документация сопровождает разработку программного обеспечения (далее – ПО) от появления идеи до выпуска конечного продукта. Написание документации является обязательным критерием разработки и последующей поддержки проекта [1].

Целью данной лабораторной работы является разработка проектной документации на созданный программный продукт.

Для достижения поставленной цели должны быть выполнены следующие задачи:

* Составление технического задания (далее – ТЗ) на разработанную программу (Приложение А);
* Составление UML диаграммы вариантов использования для разработанной программы;
* Составление UML диаграммы классов;
* Описание классов, образующих связь типа «общее-частное»;
* Привести дерево ветвлений Git;
* Провести тестирование программы.

**1 Основная часть**

**1.1 UML диаграмма вариантов использования**

Вариант использования (use case) — это описание множества последовательных действий (включая вариации), которые выполняются некоторым субъектом с целью получения результата, значимого для некоторого действующего лица [1]. ВИ предполагает взаимодействие действующих лиц и системы или другого объекта. Действующее лицо представляет собой логически связанное множество ролей, которые играют пользователи системы во время взаимодействия с ней.

Диаграмма вариантов использования для разработанного ПО приведена на рисунке 1.

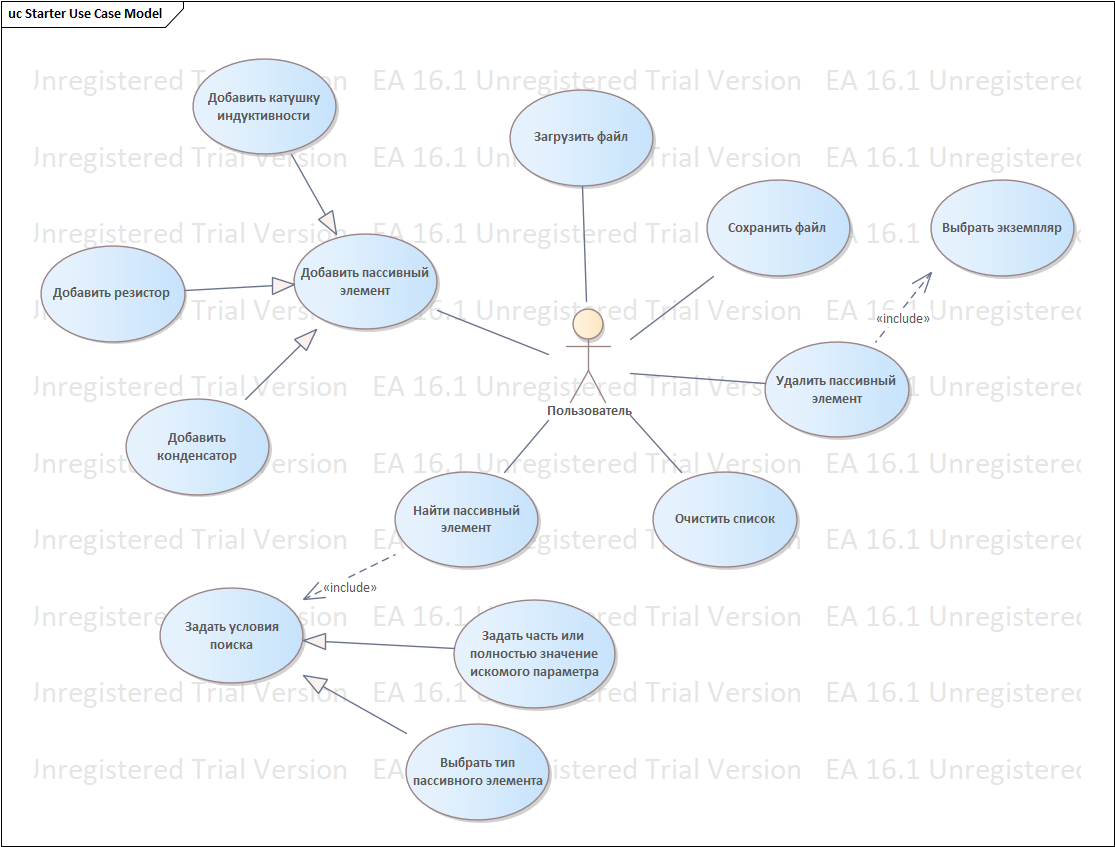


Рисунок 1 – Диаграмма вариантов использования

**1.2 UML диаграмма классов**

Диаграмма классов — это центральная методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов приведена на рисунке 2.

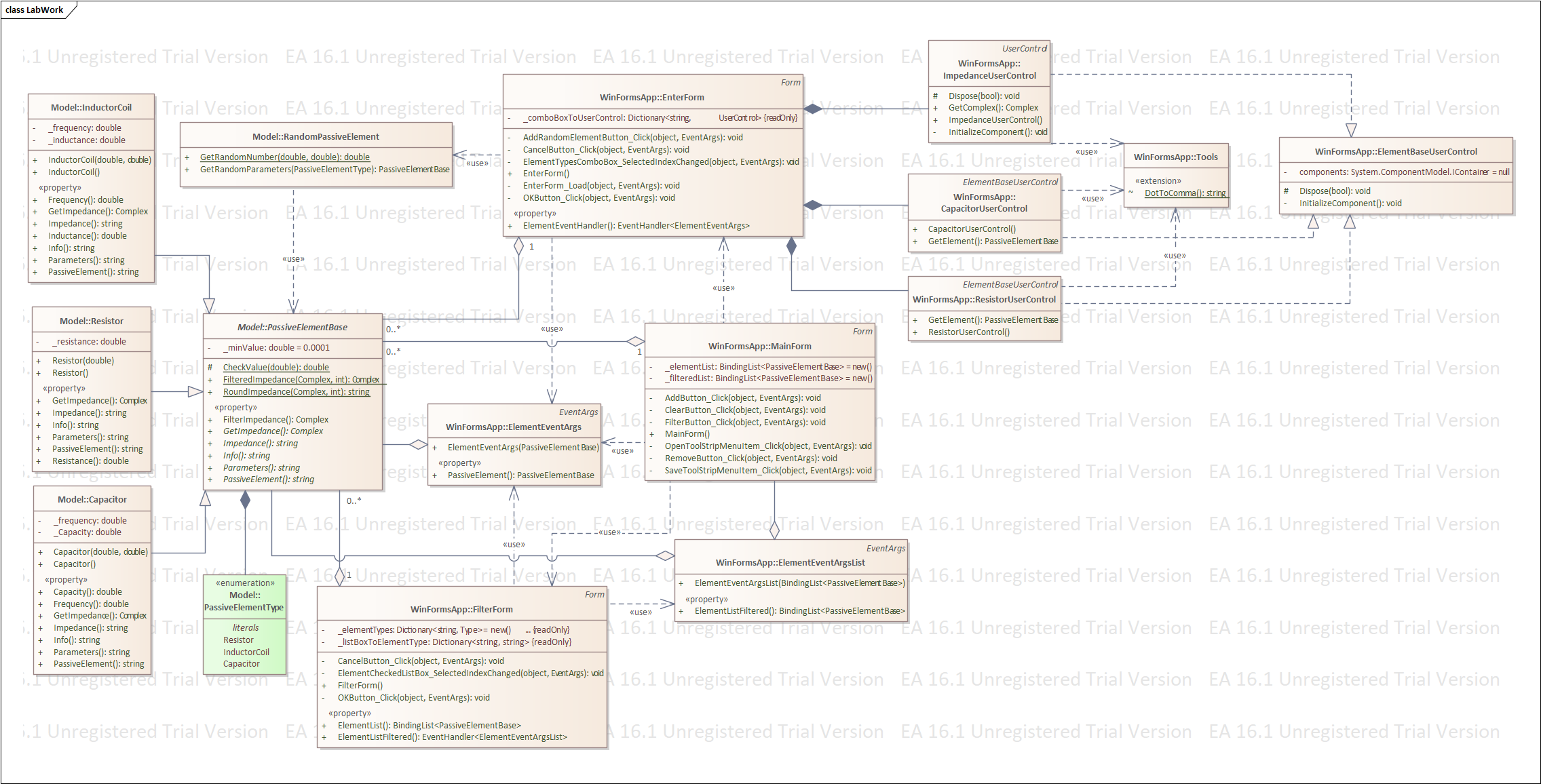


Рисунок 2 – UML диаграмма классов

**1.3 Описание классов, образующих связь типа «общее-частное»**

В таблице 1 приведено описание абстрактного класса *PassiveElementBase* с его полями, свойствами и методами.

Таблица 1 – Описание класса *PassiveElementBase*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *PassiveElementBase* – абстрактный базовый класс для пассивных элементов | | |
| Поля | | |
| -\_minValue | double | Минимальное значение |
| Свойства | | |
| + FilterImpedance | Complex | Комплексное сопротивление элемента для фильтра |
| + GetImpedance | Complex | Рассчитанное комплексное сопротивление элемента.  Абстрактное свойство, переопределяется в производных классах. |
| + Impedance | string | Комплексное сопротивление элемента.  Абстрактное свойство, переопределяется в производных классах. |
| + Info | string | Информация об элементе.  Абстрактное свойство, переопределяется в производных классах. |
| + Parameters | string | Параметры элемента.  Абстрактное свойство, переопределяется в производных классах. |
| + PassiveElement | string | Название элемента.  Абстрактное свойство, переопределяется в производных классах. |
| Методы | | |
| # CheckValue(double) | double | Метод проверки параметров  double – Вводимый параметр |
| + FilteredImpedance(Complex, int) | Complex | Метод округления для фильтра  Complex – комплексное сопротивление элемента,  int – степень округления |
| - RoundImpedance (Complex, int) | string | Метод округления для DataGrid  Complex – комплексное сопротивление элемента,  int – степень округления |

В таблицах 2–4 приведены описания классов 2 *Resistor*, 3 *Capacitor*, 4 *InductorCoil*, которые наследуются от *PassiveElementBase*.

Таблица 2 – Описание класса *Resistor*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *Resistor* – Класс резистора | | |
| Поля | | |
| –\_resistance | double | Сопротивление резистора |
| Свойства | | |
| + GetImpedance | Complex | Рассчитанное комплексное сопротивление элемента. |
| + Impedance | string | Комплексное сопротивление элемента. |
| + Info | string | Информация об элементе. |
| + Parameters | string | Параметры элемента. |
| + PassiveElement | string | Название элемента. |
| + Resistance | double | Свойство для проверки вводимого сопротивления. |
| Методы | | |
| + Resistor(double) |  | Конструктор для создания нового резистора на основе его сопротивления  double - Сопротивление резистора |
| + Resistor() |  | Пустой конструктор для создания нового резистора |

Таблица 3 – Описание класса *Capacitor*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *Capacitor* – Класс конденсатора | | |
| Поля | | |
| –\_capacity | double | Ёмкость конденсатора |
| –\_frequency | double | Частота конденсатора |
| Свойства | | |
| + GetImpedance | Complex | Рассчитанное комплексное сопротивление элемента. |
| + Impedance | string | Комплексное сопротивление элемента. |
| + Info | string | Информация об элементе. |
| + Parameters | string | Параметры элемента. |
| + PassiveElement | string | Название элемента. |
| + Сapacity | double | Свойство для проверки вводимой ёмкости. |
| + Frequency | double | Свойство для проверки вводимой частоты. |
| Методы | | |
| + Capacitor(double, double) |  | Конструктор для создания нового конденсатора на основе его ёмкости и частоты.  double – ёмкость конденсатора,  double – частота конденсатора |
| + Capacitor() |  | Пустой конструктор для создания нового конденсатора |

Таблица 4 – Описание класса *InductorCoil*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *InductorCoil* – Класс конденсатора | | |
| Поля | | |
| –\_ inductance | double | Индуктивность конденсатора |
| –\_frequency | double | Частота конденсатора |
| Свойства | | |
| + GetImpedance | Complex | Рассчитанное комплексное сопротивление элемента. |
| + Impedance | string | Комплексное сопротивление элемента. |
| + Info | string | Информация об элементе. |
| + Parameters | string | Параметры элемента. |
| + PassiveElement | string | Название элемента. |
| + Inductance | double | Свойство для проверки вводимой индуктивности. |
| + Frequency | double | Свойство для проверки вводимой частоты. |
| Методы | | |
| + InductorCoil(double, double) |  | Конструктор для создания новой катушки индуктивности на основе ее индуктивности и частоты.  double – индуктивность катушки индуктивности,  double – частота катушки индуктивности |
| + InductorCoil() |  | Пустой конструктор для создания новой катушки индуктивности |

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Калентьев, А. А. Новые технологии в программировании : учебное пособие / А. А. Калентьев, Д. В. Гарайс, А. Е. Гориянов. – Томск : Эль Контент, 2014. – 176 с. – ISBN 978-5-4332-0185-9.