**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа энергетики

Отделение электроэнергетики и электротехники

Направление – 09.04.03 Прикладная информатика

**ПРОКТНАЯ ДОКУМЕНТАЦИЯ**

Отчёт по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил |  |  |  | Д. С. Байкалова |
| Студент группы: О-5КМ11 |  | (подпись, дата) |  |  |
|  |  |  |  |  |
| Проверил преподаватель |  |  |  | А. А. Калентьев |
|  |  | (подпись, дата) |  |  |

Томск 2023

# **ВВЕДЕНИЕ**

Цель лабораторной работы направлена на обучение разработки проектной документации на созданный программный продукт. Для достижения поставленной цели необходимо выполнить следующее:

* Составление UML диаграмму вариантов использования для разработанной программы;
* Составление UML диаграммы классов;
* Описание классов, образующих связь типа «общее-частное» (наследование, реализация).

# **UML диаграмма вариантов использования**

Диаграмма вариантов использования (англ. Use case diagram) в UML – диаграмма, отражающая отношения между акторами и вариантами использования (ВИ) и являющаяся составной частью модели ВИ, позволяющей описать систему на концептуальном уровне.

ВИ соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой.

Основное назначение диаграммы – это описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Диаграмма вариантов использования для разработанного ПО приведена на рисунке 1.

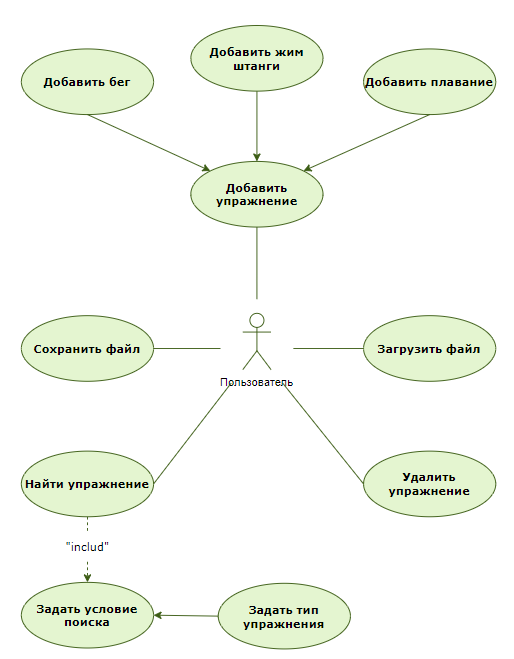


Рисунок 1. Диаграмма вариантов использования

# **UML диаграмма классов**

Диаграмма классов (от англ. «class diagram») предназначена для представления внутренней структуры программы в виде классов и связей между ними. Применяется при проектировании архитектуры, документировании системы, уточнений требований, а также для поддержки системы.

Диаграмма классов представляет описание структуры классов в системе и их взаимосвязи. Она отображает как статические аспекты системы, включая классы, атрибуты и методы, а также динамические аспекты, такие как связи между объектами и выполнение методов во время выполнения программы.

У каждого класса должно быть только одно назначение и уникально осмысленное имя, которое будет связано с этой целью.

Связи между классами на диаграмме могут представлять отношение, такие как ассоциация, наследование и реализацию. Ассоциация показывает отношения между классами, которые могут быть однонаправленными или двунаправленными. Наследование показывает. Как класс наследует свойства другого класса, который является его родительским классом. Реализация показывает, как класс реализует интерфейс или абстрактный класс.

Использование диаграммы классов помогает лучше понимать структуру системы и её компоненты.

Диаграмма классов приведена на рисунке 2.

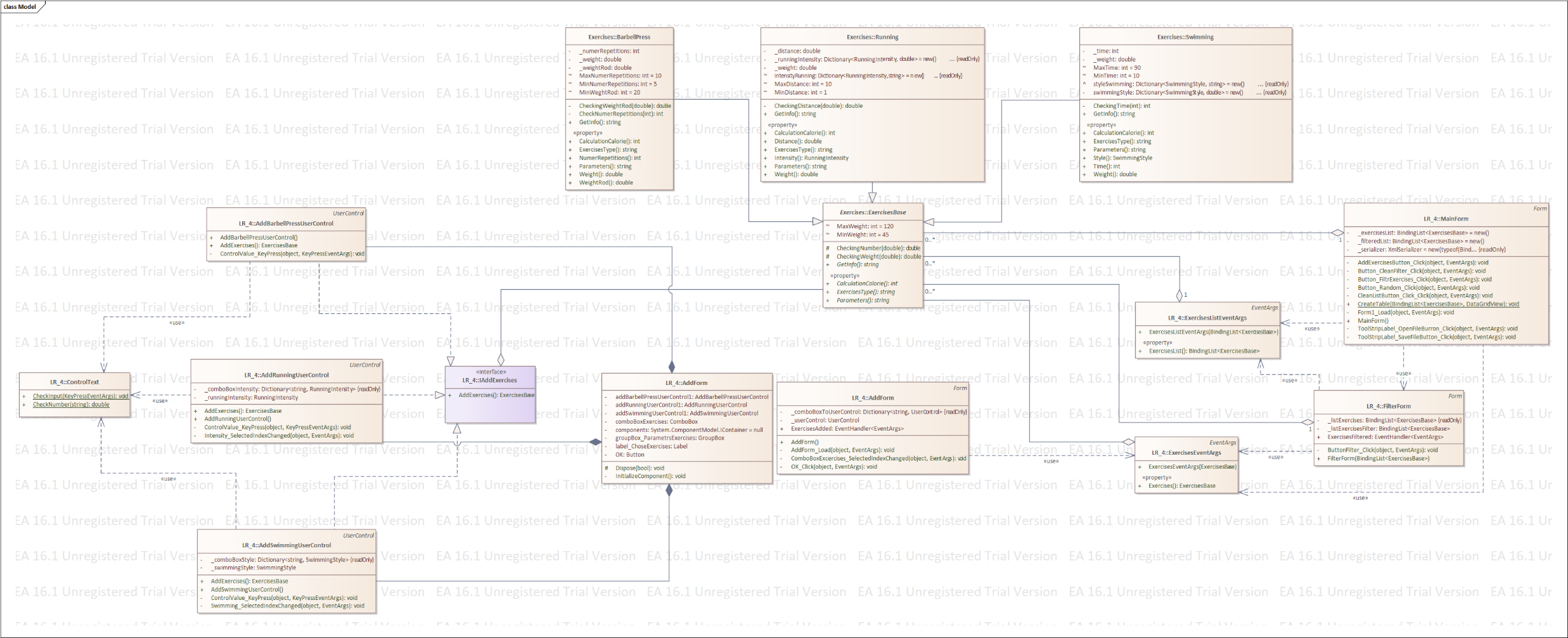


Рисунок 2. Диаграмма классов

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

В таблице 1 приведено описание абстрактного класса *ExercisrsBase* с его имеющимися полями, свойствами и методы класса.

Далее в таблицах приведены описания классов *BarbellPress*, *Running*, *Swimming*, которые наследуются от *ExercisrsBase*.

Таблица 1. Описание класса *ExercisrsBase*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Класс *ExercisrsBase* – сущность для описания абстрактного упражнения в программе | | |
| **Свойства** | | |
| *+ ExercisesType* | *string* | Тип упражнения |
| *+* *Parameters* | *string* | Параметры упражнения |
| *+* *CalculationCalorie* | *int* | Затраченные калории |
| **Методы** | | |
| *# CheckingNumber(double)* | *double* | Проверка введённого параметра на ноль и NuN;  *double* – любой параметр упражнения. |
| *#* *CheckingWeight(double)* | *double* | Проверяет на корректность введённый вес;  *double* – вес человека. |
| *+* *GetInfo* | *string* | Метод вывода информации об упражнении.  Абстрактный метод, переопределяется в производных классах. |

Таблица 2. Описание класса *BarbellPress*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Класс *BarbellPress* – сущность для описания жима штанги в программе | | |
| **Поля** | | |
| *-* *\_weight* | *double* | Вес человека |
| *- \_weightRod* | *double* | Вес штанги |
| *- \_numerRepetitions* | *int* | Количество повторений |
| **Свойства** | | |
| *+ Weight* | *double* | Вес человека |
| *+* *WeightRod* | *double* | Вес штанги |
| *+ NumerRepetitions* | *int* | Количество повторений |
| *+ ExercisesType* | *string* | Тип упражнения |
| *+* *Parameters* | *string* | Параметры упражнения |
| *+* *CalculationCalorie* | *int* | Затраченные калории |
| **Методы** | | |
| *+* *GetInfo* | *string* | Метод вывода информации об упражнении.  Абстрактный метод, переопределяется в производных классах. |
| *- CheckingWeightRod(double)* | *double* | Проверка веса штанги  *double* – вес штанги. |
| - *CheckNumerRepetitions(double)* | *double* | Проверка на количество повторений;  *double* – количество повторений. |

Таблица 3. Описание класса *Swimming*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Класс *Swimming* – сущность для описания плавания в программе | | |
| **Поля** | | |
| *-* *\_weight* | *double* | Вес человека |
| *- \_time* | *int* | Время плавания |
| **Свойства** | | |
| *+ Weight* | *double* | Вес человека |
| *+* *Time* | *double* | Время плавания |
| *+ Style* | *SwimmingStyle* | Стиль плавания |
| *+ ExercisesType* | *string* | Тип упражнения |
| *+* *Parameters* | *string* | Параметры упражнения |
| *+* *CalculationCalorie* | *int* | Затраченные калории |
| **Методы** | | |
| *+* *GetInfo* | *string* | Метод вывода информации об упражнении.  Абстрактный метод, переопределяется в производных классах. |
| *- CheckingTime (double)* | *int* | Метод проверки времени при плавании.  *double* – ведённая время. |

Таблица 4. Описание класса *Running*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Класс *Running* – сущность для описания бега в программе | | |
| **Поля** | | |
| *-* *\_weight* | *double* | Вес человека |
| *- \_distance* | *double* | Расстояние |
| **Свойства** | | |
| *+ Weight* | *double* | Вес человека |
| *+* *Distance* | *double* | Расстояние |
| *+ Intensity* | *RunningIntensity* | Интенсивность бега |
| *+ ExercisesType* | *string* | Тип упражнения |
| *+* *Parameters* | *string* | Параметры упражнения |
| *+* *CalculationCalorie* | *int* | Затраченные калории |
| **Методы** | | |
| *+* *GetInfo* | *string* | Метод вывода информации об упражнении.  Абстрактный метод, переопределяется в производных классах. |
| *- CheckingDistance (double)* | *double* | Метод проверки дистанции бега.  *double* – ведённая дистанция. |