|  |  |
| --- | --- |
|  | МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ ТАТАРСТАН **Государственное автономное профессиональное образовательное учреждение** **«ЗЕЛЕНОДОЛЬСКИЙ МЕХАНИЧЕСКИЙ КОЛЛЕДЖ»** (ГАПОУ «ЗМК») |

**09.02.03 «Программирование в компьютерных системах»**

# **МДК 03.01. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**Отчет о практических работах**

**Исполнитель**: Темуров Ибрахим Равшанович

**Группа**: 205

**Преподаватель:** Алемасов Евгений Павлович

**Дата сдачи** 12.12.2023  **Оценка** \_\_\_\_\_\_\_\_\_\_\_\_

**Подпись преподавателя**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗЕЛЕНОДОЛЬСК – 2023**

**Работа №6. Создание диаграмм классов Цель** работы – получение навыков построения Class Diagram.

**Диаграмма классов** в языке моделирования UML относится к структурному типу диаграмм, используется для визуализации структуры классов в системе, атрибутов, методов, интерфейсов и отношений между ними. Применяется при проектировании архитектуры, документировании системы, уточнении требований, а также для поддержки системы.

Связи между классами на диаграмме классов могут представлять отношения, такие как ассоциация, наследование и реализация. 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 ее компоненты.

**Класс** — это основной элемент моделирования, который представляет собой абстракцию реального объекта или сущности в системе. Он определяет свойства (атрибуты) и поведение (методы) объектов, которые являются экземплярами данного класса. На диаграмме классов обычно представляется в виде прямоугольника, разделенного на три части:

Название класса (располагается в верхней части прямоугольника). Атрибуты класса (указываются в средней части прямоугольника). Методы класса (располагаются в нижней части прямоугольника).

**Атрибуты класса** — это характеристики объекта, например, имя, возраст, адрес и т.д. Каждый атрибут имеет имя и тип данных. Тип данных может быть примитивным (например, целочисленным, строковым) или ссылочным, то есть типом другого класса.

**Методы класса** — это действия, которые может выполнять объект, например, получение или изменение значения атрибутов, выполнение какой-либо

операции или обработка событий. Каждый метод имеет имя, тип возвращаемого значения (если есть) и список параметров.

В диаграмме классов UML можно выделить следующие виды классов:

**Обычный класс** — это класс, который представляет собой абстракцию реального объекта или сущности в системе. Он имеет свои атрибуты и методы, которые описывают характеристики и поведение объектов данного класса.

**Абстрактный класс** — это класс, который содержит хотя бы один абстрактный метод. Абстрактный метод не имеет реализации в самом классе, а определяется только его сигнатура (имя, параметры, тип возвращаемого значения). Абстрактные классы могут использоваться для определения общих характеристик для нескольких классов, от которых наследуются другие классы. На диаграмме название абстрактного класса выделяется курсивом.

**Интерфейс** — это специальный вид класса, который определяет набор методов без их реализации. Интерфейсы используются для определения общих методов, которые должны быть реализованы в классах, которые их имплементируют. Один класс может имплементировать несколько интерфейсов.

**Класс перечисления** — это класс, который содержит константы, описывающие возможные значения для определенного атрибута. Классы перечисления используются, когда требуется ограничить множество допустимых значений атрибутов.

**Класс шаблона** — это класс, который представляет общий шаблон для создания конкретных классов, которые определяются позже. Классы шаблоны используются для обобщения классов и уменьшения повторяемости кода.

**Видимость (visibility)** в языке моделирования UML определяет уровень доступности элементов модели. Она определяет, какие части программы могут обращаться к конкретному элементу модели.

В языке UML используется четыре уровня видимости:

**+ Public (публичный)** — элементы с такой видимостью доступны из любой точки модели и внешних модулей. Они могут быть использованы как внутри класса, так и снаружи.

**# Protected (защищенный)** — элементы с такой видимостью не могут быть использованы вне класса, кроме классов-наследников.

**– Private (приватный)** — элементы с такой видимостью доступны только внутри класса, в котором они объявлены. Они не доступны из других классов и объектов.

**~ Package (пакетный)** — элементы с такой видимостью доступны всем классам находящимся внутри одного пакета.

Классы могут быть связаны друг с другом на диаграмме классов различными типами отношений. Такие отношения позволяют определить связи и взаимодействия между классами в системе.

**Ассоциация** — это отношение между объектами двух классов, которое описывает, что объекты одного класса могут использоваться в объектах другого класса. Она представляется на диаграмме классов в виде линии, соединяющей два класса, с указанием на концах ассоциации их ролей.

**Наследование** — это отношение на диаграмме классов UML, которое представляет собой связь между двумя классами, где один класс является более общим (родительским), а другой класс является более конкретным (дочерним). Описывает иерархию классов и позволяет наследовать атрибуты и методы от родительского класса к дочерним классам.

**Реализация** показывает, что класс реализует интерфейс или абстрактный класс. Класс, который реализует интерфейс, обязан предоставить реализацию всех методов, объявленных в этом интерфейсе. Класс, который наследует абстрактный класс, также должен реализовать все его абстрактные методы.

**Зависимость** — это отношение между двумя классами, где изменения в одном классе могут повлиять на другой класс. Зависимость также может возникать в результате передачи объектов как параметров методов других классов или при использовании переменных других классов внутри методов. Она является

наименее сильным типом отношений между классами и обычно не включается в диаграммы классов в большом количестве.

**Агрегация** — это отношение между объектами классов, когда один объект является частью другого объекта. В этом случае, объект, содержащий другой объект, называется контейнером, а содержащийся объект — содержимым. Контейнер может содержать несколько объектов-содержимых, а объект- содержимое может быть частью нескольких контейнеров.

**Композиция** — это более строгий вариант агрегации, когда объект класса- контейнера создает объект класса-части и полностью управляет его жизненным циклом. Объекты класса-части могут принадлежать только одному объекту класса-контейнера. При удалении объекта класса-контейнера все объекты класса-части также удаляются.

**Пример диаграммы классов:**

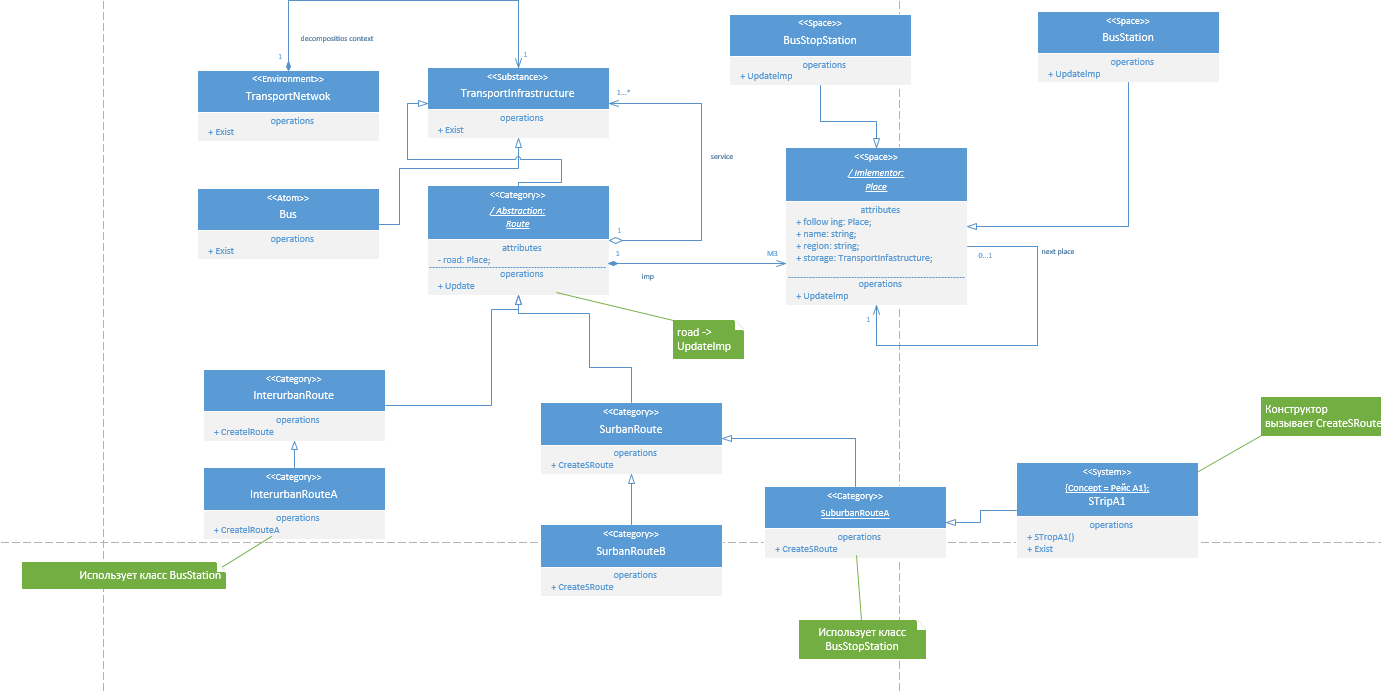


Рисунок 1- Диаграмма Классов

**Задания на самостоятельную работу:**

1. Разработайте диаграмму классов UML для системы управления задачами. Система должна поддерживать следующий функционал: Пользователи:

Каждый пользователь имеет уникальный идентификатор, имя и электронную почту.

Пользователь может создавать задачи, назначать их себе или другим пользователям, устанавливать статус (выполнено, в процессе, отложено), устанавливать сроки выполнения.

Задачи:

Задачи имеют уникальный идентификатор, описание, статус выполнения и срок выполнения.

Задачи могут быть созданы, просмотрены и изменены пользователями. Категории задач:

Задачи могут быть связаны с одной или несколькими категориями (например, работа, личные дела, обучение).

Каждая категория имеет уникальный идентификатор и название. Дополнительные требования (с уточнением по видимости): User:

id, name, email - приватные атрибуты. create\_task - публичный метод. assign\_task - защищенный метод. set\_task\_status - публичный метод.

Task:

id, description, status, deadline - приватные атрибуты. get\_priority - публичный метод.

set\_priority - защищенный метод. link\_to\_category - публичный метод. Category:

id, name - приватные атрибуты. get\_category\_name - публичный метод. Композиция:

User и Task: Каждый пользователь владеет своим списком задач. Task и Category: Задача состоит из категории.

Обобщение:

Task обобщает Category (задача - это более общий понятийный класс, чем категория).

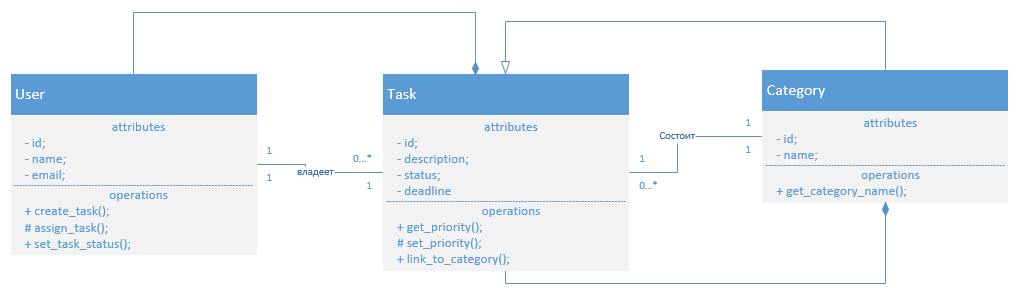


Рисунок 2- Диаграмма Классов

1. Разработайте диаграмму классов UML для системы школьной библиотеки. Система предназначена для учета книг, их выдачи студентам и управления каталогом библиотекарем.

Книги (Book):

isbn: str (private) title: str (private) author: str (private) genre: str (private) Реализуйте методы:

get\_book\_info(): void (public) - выводит информацию о книге. check\_out\_book(student: Student): void (public) - регистрирует выдачу книги студенту.

return\_book(): void (public) - регистрирует возврат книги в библиотеку. Студенты (Student):

student\_id: int (private) name: str (private) grade: int (private) Реализуйте методы:

borrow\_book(book: Book): void (public) - регистрирует взятие книги студентом.

return\_book(book: Book): void (public) - регистрирует возврат книги студентом.

get\_student\_info(): void (public) - выводит информацию о студенте. Библиотекарь (Librarian):

librarian\_id: int (private) name: str (private) Реализуйте методы:

add\_book(book: Book): void (public) - добавляет книгу в каталог библиотеки.

remove\_book(book: Book): void (public) - удаляет книгу из каталога библиотеки.

get\_librarian\_info(): void (public) - выводит информацию о библиотекаре. Отношения:

Агрегация:

Student агрегирует Book (студент может взять несколько книг). Librarian агрегирует Book (библиотекарь управляет несколькими книгами).

Ассоциация:

Student и Book: Связь между студентами и книгами (например, "взял книгу из библиотеки").

Librarian и Book: Связь между библиотекарем и книгами (например, "добавил книгу в каталог").

Зависимость:

Student зависит от Librarian при взятии или возврате книги.

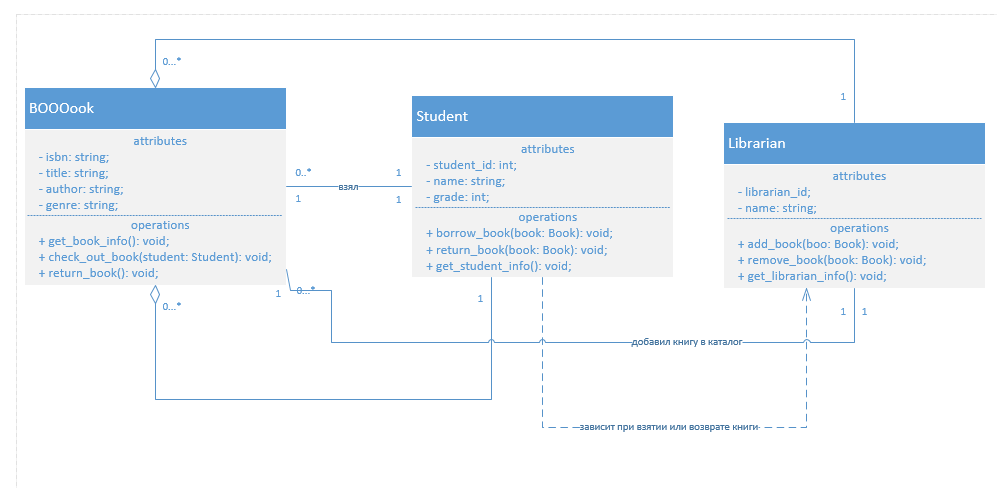


Рисунок 3- Диаграмма Классов

1. Класс MyWindow уточняет абстрактный базовый класс Window. MyWindow состоит (композиция) из кнопки класса Button и надписи класса Label.

Отобразите на диаграмме классов:

а. Класс Label имеет закрытый атрибут text типа String и общедоступную операцию setText c параметром text типа String.

б. Композиция между MyWindow и Button называется HoldsButton. Полюс со стороны кнопки имеет имя okButton, кратность 1. Композиция между MyWindow и Label называется HoldsLabel. Название полюса со стороны Label: textLabel, кратность 1.

в. Класс MyWindow реализует интерфейс IClickListener для реакции на нажатие кнопки. Отобразите на диаграмме, что между классом Button и

MyWindow есть ассоциация с именем NotifyListener с направлением от кнопки к окну. Укажите, что полюс со стороны окна называется listener.

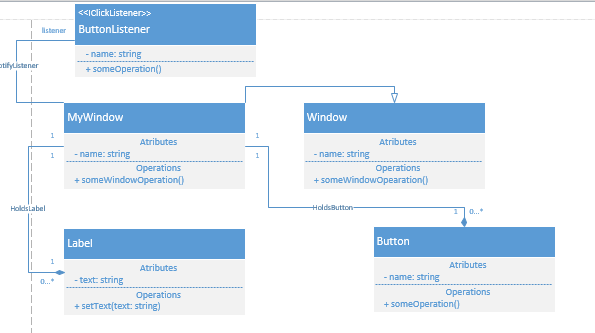


Рисунок 4- Диаграмма Классов

1. \*Класс Товар

с атрибутами:

Название – str – Private

Стоимость – money – Private

Состав – str – Private

Операции:

СозданиеТовара() – Public

ИзменениеТовара() – Public

УдалениеТовара() – Public

\*Класс Каталог

Атрибуты:

Название – str – Private

Операции:  
ВернутьТоварыКаталога() - Public

ДобавитьТоварВКаталог() – Public

\*Класс Заказ

Атрибуты:

ДатаЗаказа –dateTime – Private

СуммаЗаказа – money – Private

СтатусЗаказа

Операции:  
СозданиеЗаказа() – Public

ИзменениеЗаказа() – Public

УдалениеЗаказа() – Public

\*Класс Клиент

Атрибуты:

Название – str – Private

ВыбратьТовары() – Public

СовершитьЗаказ(товар: Товар) - Public

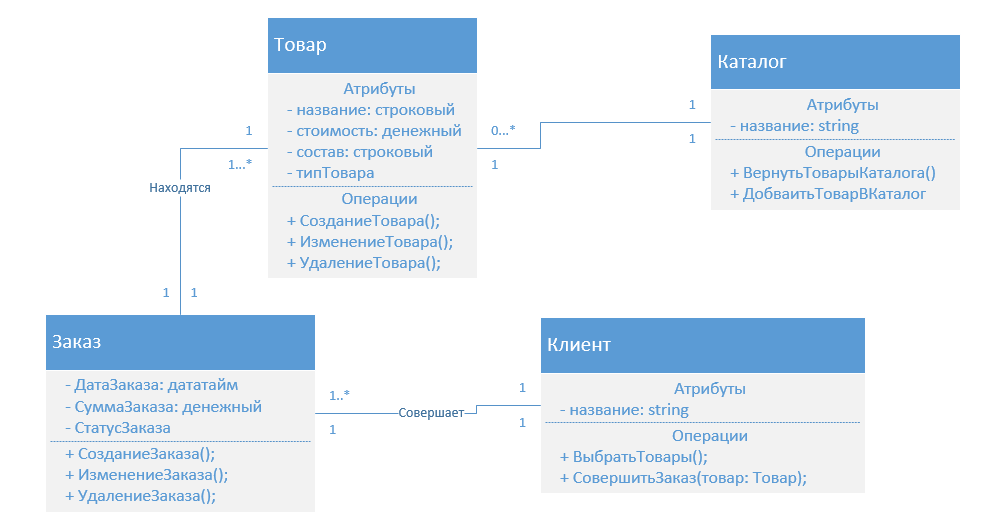


Рисунок 5- Диаграмма Классов