Государственное бюджетное профессиональное образовательное учреждение Республики Марий Эл

«ЙОШКАР-ОЛИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ»

|  |
| --- |
| Специальность: 09.02.07 Информационные системы и программирование  Группа: А-31  Дисциплина: Технология разработки ПО |

ОТЧЕТ ПО

«Диаграмма классов»

Руководитель: Пинешкин Ю.С

Выполнил(а): студентки А-31

Николаева С.В, Мочалова К.Е.

Йошкар-Ола

2025

**Предметная область**

**Диаграмма классов**— структурная диаграмма языка моделирования UML, которая описывает типы объектов системы и существующие между ними отношения.

**Некоторые особенности диаграммы классов:**

* Используется для изучения концепций предметной области, понимания требований к программному обеспечению и описания подробных проектов.
* Основной компонент — класс, который представляет собой абстракцию реального объекта или сущности в системе.
* Класс изображается в виде прямоугольника, разделённого на три части:
  + Первая часть — название класса. Как правило, оно состоит из одного или максимум двух слов.
  + Вторая часть — перечень атрибутов класса, которые характеризуют тот или иной объект этого класса в модели предметной области.
  + Третья часть — перечень операций, отражающих поведение класса в модели предметной области.

**Некоторые типы отношений, которые отображаются на диаграмме классов:**

* Зависимость — отношение между двумя классами, где изменения в одном классе могут повлиять на другой класс.
* Ассоциация — отношение между объектами двух классов, которое описывает, что объекты одного класса могут использоваться в объектах другого класса.
* Обобщение — связь между двумя классами, где один класс является более общим (родительским), а другой класс является более конкретным (дочерним).
* Реализация — показывает, что класс реализует интерфейс или абстрактный класс.



Пример диаграммы классов



Пример моей диаграммы классов

**Вывод:** Диаграммы классов применяются для визуализации структуры программного обеспечения. Они отображают основные строительные блоки системы: классы, их атрибуты и операции.

Некоторые области применения диаграмм классов:

* Планирование архитектуры приложений. Диаграмма позволяет определить, какие данные и функции необходимы каждому компоненту, как будет обеспечена их связываемость друг с другом.
* Моделирование данных. Диаграммы классов являются основой для построения концептуальной схемы базы данных.
* Представление навигации экранов. На таких диаграммах показываются пограничные классы и их логическая взаимосвязь.
* Моделирование логики программных компонент.
* Моделирование логики обработки данных.
* Эффективная коммуникация внутри команды. Разработчики могут более точно описывать требования и архитектурные решения, что ускоряет процесс создания и интеграции новых функций.
* Выявление потенциальных узких мест в проекте. Наглядные схемы помогают проанализировать, как изменения в одной части системы могут повлиять на другие компоненты.

|  |  |
| --- | --- |
| **Плюсы** | **Минусы** |
| * Визуализация. Наглядное представление о структуре программного обеспечения. * Структурирование. Диаграмма помогает разработчикам структурировать код, сделать его более читаемым. * Устранение ошибок. Позволяет обнаружить и исправить ошибки на ранних этапах разработки. * Коммуникация. Помогает лучше осмыслить требования конкретного проекта. * Иллюстрирует модели данных. Подходит для очень сложных информационных систем. * Сокращение времени обслуживания. Диаграмма предоставляет обзор структуры приложения перед изучением фактического кода. | * Сложность и избыточность. В языке UML, на котором создаются диаграммы классов, много вариантов диаграмм и нотаций, поэтому для их изучения и применения на практике требуется время. * Недостаточная гибкость. Моделирование ограничено набором строгих правил и нотаций, которые могут не работать при решении нестандартных задач или использовании новых технологий. * Отсутствие внятной интеграции. В UML нет функций автоматизации, упрощающих моделирование. * Недостаточная универсальность. В основном UML используется для разработки программного обеспечения, для моделирования других систем, например, бизнес-процессов или инженерных систем, его эффективность не так очевидна. * Трудность использования со стороны заказчиков или менеджеров. Люди, далёкие от непосредственной разработки ПО, могут столкнуться со сложностями в освоении этого инструмента. * Отсутствие общепринятой интерпретации. Несмотря на стандартизацию, UML может быть интерпретирован разными способами, что иногда приводит к непониманию между разработчиками. |