**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ МАРИЙ ЭЛ**

"**ЙОШКАР**-**ОЛИНСКИЙ** **ТЕХНОЛОГИЧЕСКИЙ** **КОЛЛЕДЖ**"

**ОТЧЁТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«Диаграмма компонентов по теме (космос)»**

Студента 3 курса, А-31 группы

Специальность 09.02.07 – «Информационные системы и программирования»

Квалификация «Администратор БД»

Руководитель: Пинешкин Ю.С.

Разработал:Тимофеев М.А.

г. Йошкар-Ола, 2022

# **Диаграмма компонентов**

Построение диаграммы компонентов — следующий этап объектно-ориентированного анализа и проектирования. Для построения диаграммы компонентов необходимо иметь построенные диаграммы прецедентов, диаграммы классов и диаграммы последовательности (и (или) диаграммы кооперации).

Диаграмма компонентов (*Component Diagram)* показывает структурные компоненты ИС и связи между ними. В качестве компонентов рассматриваются только информационные объекты: файлы, модули, библиотеки, пакеты и т.п.

**Компонент**— это физически существующая часть системы, благодаря которой обеспечивается реализация классов и отношений, а также достигается функциональность моделируемой ИС.

Для компонентов UML определяет следующие стереотипы:

* *file {файл)* — самая распространенная разновидность компонента, принимающая вид какого-либо файла;
* *executable {исполнимый) —* разновидность файла, являющегося исполнимым; может исполняться на какой-либо компьютерной платформе;
* *document {документ) —* разновидность файла в формате документа, который не является исполнимым и не содержит исходный код программы;
* *library {библиотека) —* разновидность файла в формате библиотеки (динамической или статической);
* *source {источник) —* разновидность файла, который содержит в себе исходный код программы;
* *table {таблица) —* разновидность компонента в формате таблицы БД.

Компонент изображается в виде большого прямоугольника, слева на котором расположены два маленьких вытянутых вдоль прямоугольника. Графическое изображение компонента происходит от изображения модуля системы.

Интерфейс — следующий элемент диаграммы компонентов. Интерфейс отображается в виде окружности (не обязательно замкнутой), которая связана с компонентом определенным отношением. Рекомендуется начинать название интерфейса с буквы «I» (например, Idialog). Интерфейс может либо реализовываться, либо использоваться компонентом.

Соответственно, диаграмма компонентов содержит следующие типы отношений:

* отношение зависимости;
* отношение реализации.

Наименование интерфейса напрямую зависит от его отношения к компоненту. Если компонент X реализует интерфейс Y, то интерфейс Y называется *экспортируемым* (поддерживаемым), поскольку данный интерфейс будет предоставляться другим компонентам в виде сервиса. Если компонент X использует интерфейс Y, реализуемый другими компонентами, то интерфейс Y называется *импортируемым.* Импортируемые интерфейсы отражаются при помощи отношения зависимости, а экспортируемые — при помощи отношения реализации.

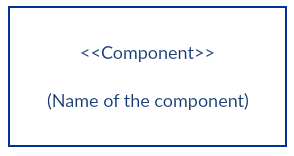
**Символы на диаграмме компонентов**

Ниже мы объяснили общие обозначения компонентной диаграммы, которые используются для построения компонентной диаграммы.

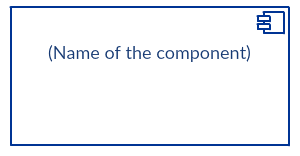
*Компонент*

Существует три способа использования символа компонента.

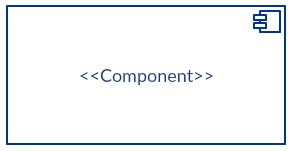
1. Прямоугольник со стереотипом компонента (текст <<компонент>>). Стереотип компонента обычно используется над именем компонента, чтобы не перепутать форму со значком класса.



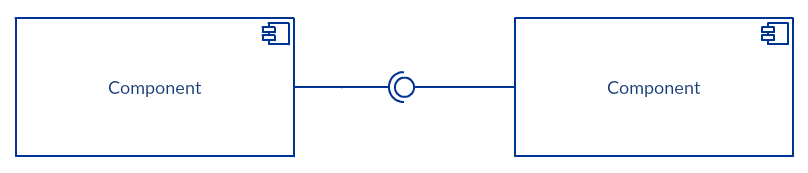
1. Прямоугольник со значком компонента в правом верхнем углу и названием компонента.



1. Прямоугольник со значком компонента и стереотипом компонента.

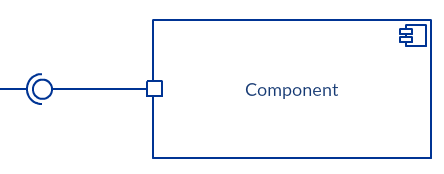


Предоставляемый интерфейс и требуемый интерфейс



Интерфейсы на компонентных схемах показывают, как компоненты соединены друг с другом и взаимодействуют друг с другом. Соединитель сборки позволяет соединить требуемый интерфейс компонента (представленный полукругом и сплошной линией) с предусмотренным интерфейсом (представленный окружностью и сплошной линией) другого компонента. Это показывает, что один компонент предоставляет услугу, которая требуется другому.

*Порт*



Порт (представленный маленьким квадратом в конце требуемого интерфейса или предоставляемого интерфейса) используется, когда компонент делегирует интерфейсы внутреннему классу.

*Зависимости*



Хотя вы можете показать более подробную информацию об отношениях между двумя компонентами с помощью нотации ball-and-socket (предусмотренный интерфейс и требуемый интерфейс), вы можете также использовать стрелку зависимостей, чтобы показать отношения между двумя компонентами.

**Диаграмма компонентов по теме Космос**

