Министерство образования и науки

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Кафедра программных систем

Отчет

по лабораторной работе № 4

# «Разработка проекта корпоративной инфокоммуникационной системы»

**Система управления проектами**

по дисциплине «Проектирование инфокоммуникационных систем»

**Выполнил: студент группы K4220**

**Проверил: к.т.н., доцент Н.А. Осипов**

Санкт-Петербург

2017

## Цель работы.

Создание структурных диаграмм моделей средствами Visual Studio.

## Задачи, решаемые при выполнении работы.

### Разработать основные UML-диаграммы.

### Определить элементы, отображаемые на UML-схемах.

## Объект исследования.

Средства Visual Studio Ultimate для построения основных диаграмм UML.

## Метод экспериментального исследования.

Имитационное визуально ориентированное моделирование.

## Рабочие формулы и исходные данные.

Visual Studio Ultimate предоставляет шаблоны для структурных UML-диаграмм:

* классов,
* компонентов,

Кроме того, можно создавать схемы слоев, которые помогают определить структуру системы.

UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования.

* Все проекты моделирования содержат общую UML-модель и несколько UML-диаграмм. Каждая диаграмма является представлением части модели. UML-модель содержит все элементы, отображаемые на UML-схемах, и может просматриваться с помощью обозревателя моделей UML.

## Схема.

### Создание диаграмм в проекте моделирования.

UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования.

Все проекты моделирования содержат общую UML-модель и несколько UML-диаграмм. Каждая диаграмма является представлением части модели. UML-модель содержит все элементы, отображаемые на UML-схемах, и может просматриваться с помощью обозревателя моделей UML.

## Окончательные результаты.

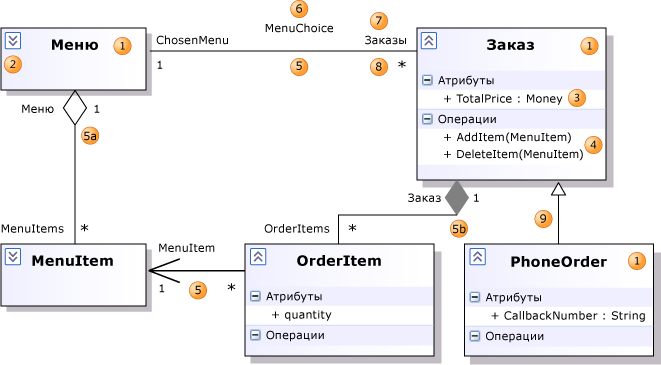
### Диаграмма классов

UML-схема классов описывает структуры объектов и сведений, используемые для внутренней организации приложения и для взаимодействия с пользователями. Кроме того, схема предоставляет сведения об этих структурах безотносительно какой-либо конкретной реализации. Ее классы и отношения могут реализовываться несколькими способами, например, в таблицах базы данных, XML-узлах или сочетаниях программных объектов.

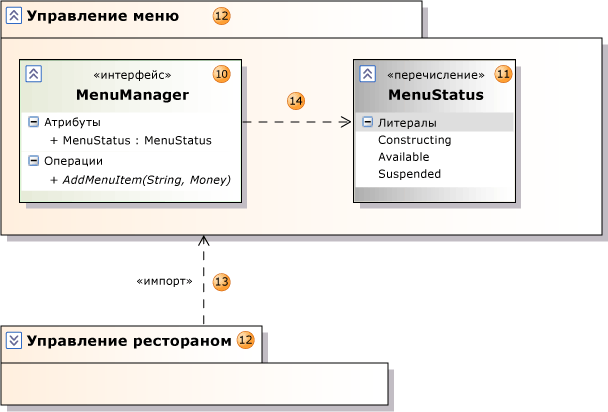
#### Чтение схем классов



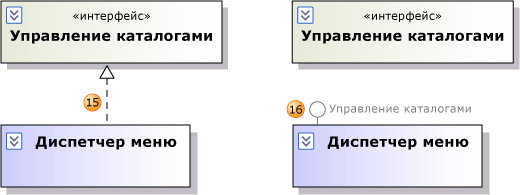
В этом разделе в таблице описаны элементы, которые можно увидеть на UML-схеме классов.



| **Фигура** | **Элемент** | **Описание** |
| --- | --- | --- |
| 1 | **Класс** | Определение объектов, совместно обладающих данными характеристиками структуры и поведения. |
| 1 | **Классификатор** | Общее имя для класса, интерфейса или перечисления. Компоненты, варианты использования и субъекты также являются классификаторами. |
| 2 | **Элемент управления "свернуть/развернуть"** | Используется для просмотра подробностей классификатора |
| 3 | **Атрибут** | Типизированное значение, прикрепленное к каждому экземпляру классификатора. |
| 4 | **Операция** | Метод или функция, которую можно выполнить с помощью экземпляров классификатора. Чтобы добавить операцию, щелкните раздел **Операции** и нажмите **ВВОД**. Введите сигнатуру операции. |
| 5 | **Ассоциация** | Отношение между членами двух классификаторов. |
| 5a | **Агрегат** | Ассоциация, представляющая отношение совместного владения. Свойству **Агрегат** роли-владельца присвоено значение **Сделано общим**. |
| 5б | **Композиция** | Ассоциация, представляющая отношение целого и части. Свойству **Агрегат** роли-владельца присвоено значение **Составной**. |
| 6 | **Имя ассоциации** | Имя ассоциации. Имя может оставаться пустым. |
| 7 | **Имя роли** | Имя роли, т. е. одного из окончаний ассоциации. Может использоваться для ссылки на связанный объект. |
| 8 | **Количество элементов** | Указывает, сколько объектов на этом окончании можно связать с объектами на другом окончании. |
| 9 | **Обобщение** | *Конкретный* классификатор наследует часть своего определения от *общего* классификатора. Общий классификатор находится на окончании соединителя с указателем стрелки. Атрибуты, ассоциации и операции наследуются конкретным классификатором. Используется инструмент **Наследование**, для создания обобщения между двумя классификаторами. |



| Фигура | Элемент | Описание |
| --- | --- | --- |
| 10 | **Интерфейс** | Определение части внешне видимого поведения объекта |
| 11 | **Перечисление** | Классификатор, состоящий из набора строковых литералов. |
| 12 | **Пакет** | Группа классификаторов, ассоциаций, действий, линий жизни, компонентов и пакетов. Логическая схема классов показывает, что членами данного пакета являются классификаторы и пакеты. |
| 13 | **Импорт** | Отношение между пакетами, указывающее, что один пакет включает все определения другого. |
| 14 | **Зависимость** | Определение или реализация зависимого классификатора может измениться, если изменяется классификатор на окончании с наконечником стрелки. |



|  |  |  |
| --- | --- | --- |
| Фигура | **Элемент** | Описание |
| 15 | **Реализация** | Класс реализует операции и атрибуты, определенные интерфейсом. Воспользуйтесь инструментом **Наследование**, чтобы создать реализацию между классом и интерфейсом. |
| 16 | **Реализация** | Альтернативное представление того же отношения. Метка на символе обозначения указывает на интерфейс.  Чтобы создать эту презентацию, выделите существующее отношение реализации. Рядом с ассоциацией появляется тег действия. Щелкните тег действия и выберите **Показывать без описания операций**. |

### Диаграмма компонентов

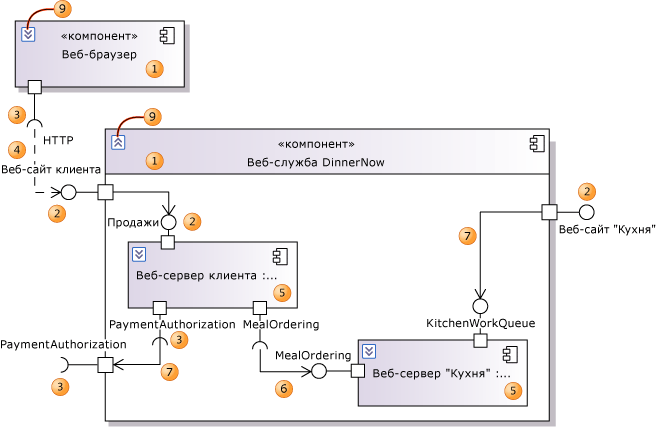
В Visual Studio Ultimate на *схеме компонентов* показаны части конструкции программной системы. Схема компонентов помогает визуализировать высокоуровневую структуру системы и поведение служб, предоставляемых и потребляемых этими элементами через интерфейсы.

Схему компонентов можно использовать, чтобы описать конструкцию системы, реализуемую на любом языке и в любом стиле. Нужно только определить части конструкции, взаимодействующие с другими частями через ограниченный набор входных и выходных каналов. Можно использовать компоненты любого масштаба, взаимосвязанные любым способом.

#### Чтение схем компонентов



Ниже в таблице описаны элементы, которые можно использовать на схеме компонентов, и их основные свойства.



| **Фигура** | **Элемент** | **Описание и основные свойства** |
| --- | --- | --- |
| 1 | **Компонент** | Допускающий повторное использование функциональный элемент системы. Компонент предоставляет и потребляет поведение через интерфейсы и может использовать другие компоненты.  Можно скрывать или отображать внутренние части компонента с помощью элемента управления "развернуть/свернуть" (9).  Компонент — это вид класса.   * **Является неявно создаваемым экземпляром**. Если значение true (по умолчанию), компонент существует только как артефакт конструкции. Во время выполнения существует только ее часть. |
| 2 | **Предоставленный порт интерфейса** | Представляет группу сообщений или вызовов, реализуемых компонентом и доступных для использования другими компонентами или внешними системами. Порт — это свойство компонента, имеющее в качестве типа интерфейс. |
| 3 | **Требуемый порт интерфейса** | Представляет группу сообщений или вызовов, отправляемых компонентом другим компонентам или внешним системам. Компонент предназначен для соединения с компонентами, которые предоставляют хотя бы эти операции. Порт имеет в качестве типа интерфейс. |
| 4 | **Зависимость** | Может использоваться для указания, что требуемый интерфейс одного компонента может соответствовать предоставленному интерфейсу другого компонента.  Зависимости также можно использовать в более общем случае при работе с элементами модели, чтобы показать, что конструкция одного зависит от конструкции другого. |
| 5 | **Часть** | Атрибут компонента, тип которого, как правило, является другим компонентом. Часть используется при внутреннем проектировании ее родительского компонента. Графически части изображаются вложенными в родительский компонент.  Чтобы создать часть существующего типа компонента, перетащите компонент из Проводника по моделям UML в компонент-владелец.  Чтобы создать часть нового типа, выберите инструмент **Компонент** и щелкните компонент-владелец. Например, компонент Car имеет части engine:CarEngine, backLeft:Wheel, frontRight:Wheel и т. д.  Несколько частей могут иметь один и тот же тип, и разные компоненты могут иметь части одного типа.   * **Тип**. Тип части, определяемый в другом месте модели. Как правило, типом является другой компонент. * **Количество элементов**. По умолчанию используется значение 1. Можно задать значение **0..1**, чтобы указать, что часть может иметь значение **null**, или задать значение **\***, чтобы указать, что часть является коллекцией экземпляров данного типа. Также в качестве значения можно задать любое выражение, которое можно оценить в числовом диапазоне. |
| 6 | **Сборка части** | Соединение между требуемыми портами интерфейса одной части и предоставленными портами интерфейса другой. |
| 7 | **Делегирование** | Связывает порт с интерфейсом одной из частей компонента. Указывает, что сообщения, отправленные компоненту, обрабатываются этой частью, или что сообщения, отправленные этой частью, отсылаются из родительского компонента. |
| 8 | **Обобщение** | Указывает, что один компонент наследуется от другого. Части и интерфейсы наследуются. |
| 9 | **Элемент управления "**развернуть/свернуть**"** | Позволяет скрывать или отображать внутренние части компонента. |

## Выводы и анализ результатов работы.

В результате работы было осуществлено ознакомление с нотацией UML для разработки проекта корпоративной инфокоммуникационной системы.

UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования.

Цель и сценарии включенного варианта использования должны иметь смысл независимо друг от друга, чтобы их можно было включать в варианты использования, создаваемые позже.

Разделение вариантов использования на включающие и включенные части позволяет достичь следующих целей:

• Структурировать описания вариантов использования по уровню детализации.

• Избежать дублирования общих сценариев в разных вариантах использования.