**Лабораторная работа № 6**

**Создание диаграммы компонентов**

**Теоретическая часть**  **Компонент** - это физическая заменяемая часть системы, реализующая некоторый набор интерфейсов. Они используются для моделирования физических сущностей размещённых в узле: исполняемых модулей, библиотек, таблиц, файлов и документов Обычно компонент представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации. Интерфейсы образуют мост между логическими и физическими моделями. Например, в логической модели можно специфицировать интерфейс некоторого класса, который затем будет реализован каким-либо компонентом в физической модели.

Физическое моделирование системы служит для построения исполняемой системы. В то время как логические сущности присутствуют лишь в концептуальном мире, физические существуют в мире битов, то есть в конечном счёте размещаются в физических узлах и могут либо исполняться непосредственно, либо косвенно участвовать в работе системы. Основные сведенья о компонентах:

компонент имеет *физическую* природу. Он существует в реальном мире битов, а не в мире концепций.

компонент *заменяем*. Вместо одного компонента можно подставить другой, если он совместим с тем же набором интерфейсов. Обычно механизм добавления или замены компонента с целью формирования исполняемой системы прозрачен для пользователя и обеспечивается либо объектными моделями? которые часто совсем не требуют внешнего вмешательства, либо инструментальными средствами, автоматизирующими этот механизм.

компонент - это *часть системы*. Компонент редко выступает в отрыве от остальных: обычно он работает совместно с другими компонентами и, стало быть, встраивается в архитектурный или технологический контекст, для которого предназначен. Компонент является логически и физически способным к сцеплению, то есть представляет собой значимый структурный и/или поведенческий фрагмент некоторой большей системы. Компонент можно повторно использовать в различных системах. Таким образом, компоненты представляют собой фундаментальные строительные блоки, из которых собираются системы. Это определение рекурсивно - система, рассматриваемая на одном уровне абстракции, может быть всего лишь компонентом на более высоком уровне.

Объектные библиотеки, исполняемые программы, компоненты **СОМ+**, **Enterprise JavaBeans** - все это примеры сущностей, которые могут быть непосредственно представлены компонентами в смысле UML.

Вы можете организовывать компоненты, группируя их в пакеты, так же, как это делается для классов. При организации компонентов между ними можно специфицировать отношения зависимости, обобщения, ассоциации (включая агрегирование) и реализации.

В языке UML выделяют три вида компонентов:

*Компоненты развертывания,* которые обеспечивают непосредственное выполнение системой своих функций. Такими компонентами могут быть динамически подключаемые библиотеки с расширением dll, Web-страницы на языке разметки гипертекста с расширением html и файлы справки с расширением hlp.

*Компоненты-рабочие продукты.* Как правило – это файлы с исходными текстами программ, например, с расширениями h или срр для языка C++.

Компоненты исполнения, представляющие исполнимые модули – файлы с расширением ехе. Они обозначаются обычным образом.

Эти элементы иногда называют артефактами, подчеркивая при этом их законченное информационное содержание, зависящее от конкретной технологии реализации соответствующих компонентов. Более того, разработчики могут для этой цели использовать самостоятельные обозначения, поскольку в языке UML нет строгой нотации для графического представления примечаний.

В языке UML для компонентов определены следующие стереотипы:

*Библиотека (library)* – определяет первую разновидность компонента, который представляется в форме динамической или статической библиотеки.

*Таблица (table)* – также определяет первую разновидность компонента, который представляется в форме таблицы базы данных.

*Файл (file)* – определяет вторую разновидность компонента, который представляется в виде файлов с исходными текстами программ.

*Документ (document)* – определяет вторую разновидность компонента, . который представляется в форме документа.

*Исполнимый (executable)* – определяет третий вид компонента, который может исполняться в узле.

# Графическое обозначение

Графически компонент изображается в виде прямоугольника с вкладками в нотации **UML версии 1**, как показано на рис. 1. В **нотации UML версии 2** изображается так, как показано на рис. 2.

|  |  |
| --- | --- |
|  |  |
| **Рис. 1** Компонент в UML версии 1 | **Рис. 2** Компонент в UML версии 2 |

У каждого компонента должно быть имя, отличающее его от других компонентов.

Имя - это текстовая строка; взятое само по себе, оно называется **простым**. К **составному** имени спереди добавляется имя пакета, в котором находится компонент. Обычно при изображении компонента указывают только его имя, как видно из рис. 2.

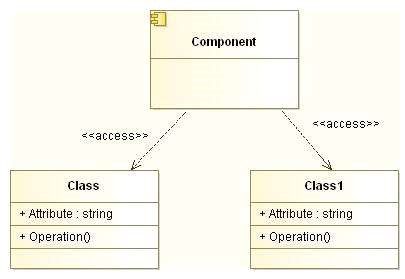
Имя компонента может состоять из любого числа букв, цифр и некоторых знаков препинания (за исключением таких, как двоеточия, которые применяются для отделения имени компонента от имени объемлющего пакета). Имя может занимать несколько строк. Как правило, для именования компонентов используют одно или несколько коротких существительных, взятых из словаря реализации, и в зависимости от выбранной операционной системы добавляют расширения имен файлов (например, java или dll).

# Компоненты и классы

Во многих отношениях компоненты подобны классам. Но есть ряд различий между классами и компонентами:

* классы представляют собой логические абстракции, а компоненты - физические сущности. Таким образом, компоненты могут размещаться в узлах, а классы - нет;
* компоненты представляют собой физическую упаковку логических сущностей и, следовательно, находятся на другом уровне абстракции;
* классы могут обладать атрибутами и операциями. Компоненты обладают только операциями, доступными через их интерфейсы.

Второе различие предполагает существование некоторого отношения между классами и компонентами, а именно: компонент - это физическая реализация множества логических элементов, таких как классы и кооперации. Как показано на рис. 3, отношение между компонентом и классом, который он использует.



**Рис. 3** Компонент и классы

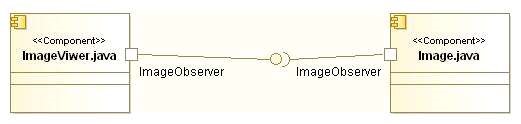
Компоненты похожи на классы еще и потому, что в принципе для них можно задать атрибуты и операции, хотя необходимость в этом возникает редко - разве что при моделировании рефлексивных систем, которые могут манипулировать своими компонентами.

# Компоненты и интерфейсы

Главная задача каждого компонентно-ориентированного средства в любой операционной системе - обеспечить возможность сборки приложений из заменяемых двоичных частей. Иными словами, можно создать систему из компонентов, а затем развивать ее, добавляя новые компоненты или заменяя старые - без перекомпиляции. Именно интерфейсы позволяют достичь этого. Компонент может быть совместимым с одним набором интерфейсов и реализовать другой набор. Это значит, что специфицируя интерфейс, вы можете вставить в уже работающую систему любой компонент, который совместим с этим интерфейсом или предоставляет его. Систему можно расширять, подставляя компоненты, обеспечивающие новые услуги с помощью дополнительных интерфейсов, а также компоненты, способные распознать и использовать эти новые интерфейсы.

**Интерфейс** - это набор операций, которые описывают услуги, предоставляемые классом или компонентом. Существенным является отношение между компонентом и интерфейсом. Все популярные компонентные средства операционных систем используют интерфейсы для "склеивания" различных компонентов. Пользуясь любым из таких средств, можно построить декомпозицию физической реализации путем специфицирования интерфейсов, представляющих основные стыковочные узлы системы. Затем вы можете предоставить компоненты, реализующие интерфейсы, наряду с другими компонентами, которые через эти интерфейсы получают доступ к предоставляемым услугам. Такой механизм позволяет развернуть систему, службы которой в значительной мере независимы от места своего размещения, что будет использовано в лабораторной работе №7 «Диаграмы размещения».

На рис. 4 наиболее распространённый способ изображения интерфейса: круг, который связан линией с компонентом, предоставляющим реализацию этого интерфейса. Запрашиваемый интерфейс изображён в виде полукруга с линией, которая соединяет его с компонентом. Интерфейс рисуется в свёрнутой форме, без таких подробностей, как набор операций, атрибутов доступа и прочего, чем и отличается от интерфейса в диаграмме классов. Компонент, получающий доступ к услугам других компонентов через этот интерфейс, подключается к нему с помощью отношения зависимости.



**Рис. 4** Компоненты и интерфейс

Интерфейс, реализуемый компонентом, называется *предоставляемым интерфейсом (provided interface).* Через данный интерфейс компонент предоставляет ряд услуг другим компонентам. Компонент может предоставлять много интерфейсов.

Интерфейс, которым компонент пользуется, называется *запрашиваемым интерфейсом* (requiredinterface). Это означает, что компонент совместим с таким интерфейсом и зависит от него при выполнении своих функций. Компонент может импортировать различные интерфейсы, причём ему разрешается одновременно экспортировать и импортировать интерфейсы.

Интерфейсы пересекают границы между логическими и физическими сущностями. Тот же самый интерфейс, который используется или реализуется компонентом, будет использоваться или реализовываться классами, которые реализует этот компонент.

# Создание диаграммы компонентов в Modelio

Для примера построения диаграммы компонентов была выбрана бесплатная среда моделирование Modelio от Modeliosoft. Скачать можно с официального сайта:

[http://www.modelio.org](http://www.modelio.org/)

## Внешний вид основного окна приложения с вкладками

**Рис. 6** Внешний вид главного окна приложения Краткое описание:

**Вкладка «Diagram».** Здесь находятся все диаграммы активного проекта.

**Вкладка «Model».** Здесь находятся все элементы на активной диаграмме.

**Вкладка «Element».** Здесь находятся основные настройки активного элемента модели.

**Вкладка «Symbol».** Здесь находятся визуальные и структурные настройки элемента.

Здесь находятся все доступные элементы активной диаграммы.

Здесь находятся элементы активной диаграммы.

Последовательность создания диаграммы компонентов:

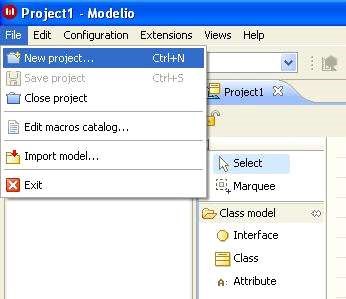
Создайте новый проект:

Выберите в меню *File->New project...* (рис. 7);

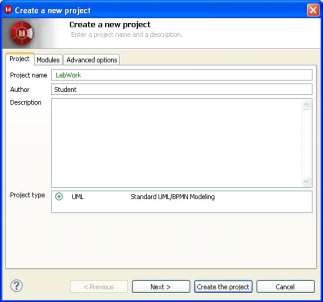
В появившемся окне (рис. 8) введите имя проекта;

Нажмите *Create the project*;

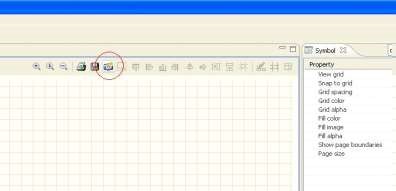
Добавьте необходимые элементы и связи между ними; Скопируйте диаграмму в графическом виде в буфер обмена с помощью кнопки на рис. 9 и отредактируйте потом в любом графическом редакторе.



**Рис 7.** Новый проект



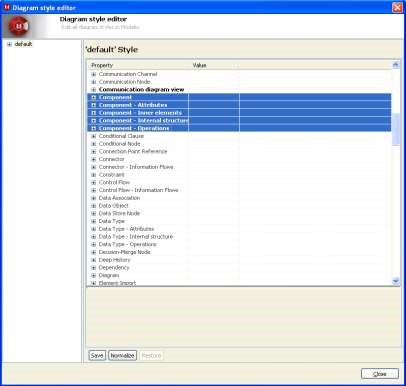
**Рис 8.** Настройки проекта



**Рис 9.** Сохранение диаграммы как графического изображения в буфер обмена

## Настройка отображения компонентов

Все настройки отображения компонентов в диаграмме можно провести в диалоговом окне, которое вызывается из панели меню: *Configuration → Diagram styles...* . Это окно содержит дерево настроек для каждого элемента диаграмм Modelio. Для настройки диаграммы компонентов нам потребуются те узлы дерева, которые содержат в своем имени слово «Component» (рис. 10).



**Рис. 10.** Диалоговое окно настройки элементов

К примеру, для включения отображения внутренних компонентов в компоненте графически (а не в виде списка), нужно в *Component — Inner elements* поменять пункт *Display mode* с *List* на *Diagram*. А что бы включить расширения для имен компонентов: в *Component* поменять пункт *Name display mode* с *Simple* в *Full qualified*.

Эти настройки будут настрйоками по-умоланию для всех элементов. Кроме этого, если для определенного элемента нужно будет провести отдельную настройку отображения, то это следует провести с помощью вкладки №4 из рисунка 6.

## Инструменты

В программе Modelio есть достаточное количество инструментов для построения диаграммы компонентов. Расположение элементов показано в пункте 5 на рисунке 6. Сами компоненты, их порты и интерфейсы находятся во вкладке *Component* инструментария. Для других связей, например «зависимости», можно использовать инструмент *Dependency* из вкладки *Common*.

**Таблица 1.** Основные инструменты диаграммы компонентов

|  |  |
| --- | --- |
| **Графическое отображение** | **Описание** |
|  | **Компонент.**Инструмент расположения компонента на диаграмме. По-умолчанию, имя компонента Component, к которому добавляются цифры в конце, если такое имя уже занято. |
|  | **Порт.**Инструмент добавления порта в компонент. Для этого, выберите этот инструмент и кликните на уже существующий компонент на диаграмме. |
|  | **Предлагаемый интерфейс.**Инструмент позволяет добавить в компонент интерфейс, который он реализовывает. Для этого, компонент должен иметь свободный порт. Что бы добавить интерфейс, кликните этим инструментом на свободный порт. Таким образом, на диагрмме появится интерфейс, который будет соединён с данным компонентом. |
|  | **Запрашиваемый интерфейс.**Инструмент позволяет соединить компонент с другим компонентом через определённый интерфейс. Что бы это сделать, у компонента, запрашивающего определённый интерфейс, должен быть свободный порт. Кликните на этот порт и соедините его с доступным интерфейсом другого компонента. |
|  | **Зависимость.**Инструмент позволяет соединить два компонента, что бы показать их зависимость между собой. Что бы это сделать кликните инструментом на одном из компонентов, а потом на другом, что бы их соединить. |

Это основные инструменты для добавления компонентов. Инструменты для добавления классов можно найти во вкладке *Class model*.

# Методические указания к разработке диаграмм компонентов

Разработка диаграммы компонентов предполагает использование информации как о логическом представлении модели системы, так и об особенностях ее физической реализации. До начала разработки необходимо принять решения о выборе вычислительных платформ и операционных систем, на которых предполагается реализовывать систему, а также о выборе конкретных баз данных и языков программирования.

После этого можно приступать к общей структуризации диаграммы компонентов. В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы не только возможность повторного использования кода за счёт рациональной декомпозиции компонентов, но и создание объектов только при их необходимости. Речь идёт о том, что общая производительность программной системы существенно зависит от рационального использования ею вычислительных ресурсов. Для этой цели необходимо большую часть описаний классов, их операций и методов вынести в динамические библиотеки, оставив в исполняемых компонентах только самые необходимые для инициализации программы фрагменты программного кода.

После общей структуризации физического представления системы необходимо дополнить модель интерфейсами. При разработке интерфейсов следует обращать внимание на согласование (стыковку) различных частей программной системы.

Наконец, завершающий этап построения диаграммы компонентов связан с установлением и нанесением на диаграмму взаимосвязей между компонентами, а также отношений реализации. Эти отношения должны иллюстрировать все важнейшие аспекты физической реализации системы, начиная с особенностей компиляции исходных текстов программ и заканчивая исполнением отдельных частей программы на этапе ее выполнения. Для этой цели можно использовать различные виды графического изображения компонентов.

При разработке диаграммы компонентов следует придерживаться общих принципов создания моделей на языке UML. В частности, в первую очередь необходимо использовать уже имеющиеся в языке UML компоненты и стереотипы. Для большинства типовых проектов этого набора элементов может оказаться достаточно для представления компонентов и зависимостей между ними.

Если же проект содержит некоторые физические элементы, описание которых отсутствует в языке UML, то следует воспользоваться механизмом расширения. В частности, использовать дополнительные стереотипы для отдельных нетиповых компонентов или помеченные значения для уточнения их отдельных характеристик.

# Задания

Ознакомится с теоретическими сведеньями;

Ознакомится со средой моделирования Modelio;

Проанализировать свою предметную область;

Определить основные компоненты и их организацию;

Определить основные интерфейсы;

Построить диаграмму компонентов на основе проведённого анализа; Составить отчёт;

В отчёте должны присутствовать:

Титульная страница с информацией об исполнителе (ФИО, номер группы)

Описание предметной области;

Описание компонентов;

Описание интерфейсов;

Графическое представление диаграммы компонентов;

# Контрольные вопросы

Что такое компонент?

Какие бывают компоненты?

В чем отличие компонента от класса?

Как на диаграмме изображается компонент в UML 2?

Какие стереотипы предопределены в языке UML?

Что такое интерфейс?

Как компоненты связываются через интерфейсы?

Как на диаграмме отображается реализация интерфейса компонентом?

Как на диаграмме отображается подключение компонента к интерфейсу?

Зачем нужны порты?