Базы данных

# Лабораторная работа №1

Построение объектной модели задачи с использованием языка моделирования UML.

К защите работы должен быть предоставлен проект, созданный в пакете StarUML (<http://staruml.io/download>), включающий три вида диаграмм: прецедентов, классов (интерфейс, данные) и последовательностей для каждой функции.

Общая информация

Построение модели необходимо для того, чтобы лучше понимать разрабатываемую систему.

Моделирование позволяет решить следующие задачи:

* визуализировать систему в ее текущем или желательном для нас состоянии;
* определить структуру или поведение системы;
* получить шаблон, позволяющий затем сконструировать систему;
* документировать принимаемые решения, используя полученные модели.

При разработке программного обеспечения существует несколько подходов к моделированию. Важнейшие из них – алгоритмический и объектно-ориентированный. Наиболее современным подходом к разработке программного обеспечения является объектно-ориентированный. Здесь в качестве основного строительного блока выступает объект или класс. В самом общем смысле объект – это сущность, обычно извлекаемая из словаря предметной области или решения, а класс является описанием множества однотипных объектов. Каждый объект обладает идентичностью (его можно поименовать или как-то по-другому отличить от прочих объектов), состоянием (обычно с объектом бывают связаны некоторые данные) и поведением (с ним можно что-то делать или он сам может что-то делать с другими объектами).

Например, рассмотрим простую двухуровневую архитектуру системы, состоящую из интерфейса пользователя и базы данных. Интерфейс содержит конкретные объекты – кнопки, меню и диалоговые окна. База данных также состоит из конкретных объектов, а именно таблиц, представляющих сущности предметной области.

**Унифицированный язык моделирования** (UML), является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать программные системы.

Словарь языка UML включает три вида строительных блоков:

* сущности;
* отношения;
* диаграммы.

Сущности – это абстракции, являющиеся основными элементами модели. Отношения связывают различные сущности; диаграммы группируют представляющие интерес совокупности сущностей.

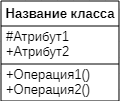
В UML имеется четыре типа сущностей:

* структурные;
* поведенческие;
* группирующие;
* аннотационные.

**Структурные сущности**, как правило, представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует семь разновидностей структурных сущностей:

* класс (Class);
* интерфейс (Interface);
* кооперация (Collaboration);
* прецедент (Use case);
* активный класс (Active class);
* компонент (Component);
* узел (Node).

Подробно будут описаны только те элементы языка, которые будут использоваться при выполнении лабораторной работы.

рис. 1 Классы

**Класс** (Class) – это описание совокупности объектов с общими атрибутами, операциями и отношениями. Графически класс изображается в виде прямоугольника, в котором обычно записаны его имя, атрибуты и операции, как показано на рис. 1. Одной из разновидностей сущности класс является **актер**(Actor). Обычно актер представляет роль, которую в данной системе играет человек, аппаратное устройство или даже другая система. Как показано на рис. 2, актеров изображают в виде человеческих фигурок.



рис.2 Актеры

**Прецедент** (Use Case) - это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного актера (Actor). Прецедент применяется для структурирования поведенческих сущностей модели. Графически прецедент изображается в виде ограниченного непрерывной линией эллипса, обычно содержащего только его имя, как показано на рис. 3.



рис.3 Прецеденты

**Поведенческие сущности** являются динамическими составляющими модели UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует два вида поведенческих сущностей:

* взаимодействие (Interaction);
* автомат (State machine).

**Взаимодействие** (Interaction) – это поведение, суть которого заключается в обмене сообщениями (Messages) между объектами в рамках конкретного контекста для достижения определенной цели. Графически сообщения изображаются в виде стрелки, над которой почти всегда пишется имя соответствующей операции, как показано на рис. 4.

4

рис.4 Сообщения

**Группирующие сущности** являются организующими частями UML. Это блоки, на которые можно разложить модель. Есть только одна группирующая сущность, а именно пакет.



рис. 5 Пакеты

**Пакеты** (Packages) представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить структурные, поведенческие и даже другие группирующие сущности. В отличие от компонентов, существующих во время работы системы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки. Изображается пакет в виде папки с закладкой, содержащей, как правило, только имя (см. рис. 5).

**Аннотационные сущности** – пояснительные части модели UML. Это комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один тип аннотационных элементов – примечания (Note).

**Примечание** – это просто символ для изображения комментариев или ограничений, присоединенных к элементу или группе элементов. Графически примечание изображается в виде прямоугольника с загнутым краем, содержащим текстовый или графический комментарий, как показано на рис. 6.

6

рис. 6 Примечания

В языке UML определены четыре типа отношений:

* зависимость;
* ассоциация;
* обобщение;
* реализация.

Эти отношения являются основными строительными блоками в UML и применяются для создания корректных моделей.

**Зависимость** (Dependency) – это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой. Графически зависимость изображается в виде прямой пунктирной линии, часто со стрелкой (см. рис. 7).

7

рис. 7 Зависимости

**Ассоциация** (Association) – структурное отношение, описывающее совокупность связей; связь – это соединение между объектами. Графически ассоциация изображается в виде прямой линии (иногда завершающейся стрелкой или содержащей метку), рядом с которой могут присутствовать дополнительные обозначения, например кратность и имена ролей. На рис. 8 показан пример отношений этого типа.

8

рис. 8 Ассоциации

Диаграмма в UML – это графическое представление набора элементов, изображаемое чаще всего в виде связного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. В UML выделяют девять видов диаграмм:

* диаграммы классов;
* диаграммы объектов;
* диаграммы прецедентов;
* диаграммы последовательностей;
* диаграммы кооперации;
* диаграммы состояний;
* диаграммы действий;
* диаграммы компонентов;
* диаграммы развертывания.

На **диаграмме классов** показывают классы, интерфейсы, объекты и кооперации, а также их отношения. При моделировании объектно-ориентированных систем этот тип диаграмм используют чаще всего. Диаграммы классов соответствуют статическому виду системы с точки зрения проектирования.

На **диаграмме прецедентов** представлены прецеденты и актеры (частный случай классов), а также отношения между ними. Диаграммы прецедентов относятся к статическому виду системы с точки зрения прецедентов использования. Они особенно важны при организации и моделировании поведения системы.

**Диаграммы последовательностей** являются частным случаем диаграмм взаимодействия. На **диаграммах взаимодействия** представлены связи между объектами; показаны в частности, сообщения, которыми объекты могут обмениваться. Диаграммы взаимодействия относятся к динамическому виду системы. При этом диаграммы последовательности отражают временную упорядоченность сообщений.

Порядок выполнения работы

Порядок выполнения работы будет рассмотрен на примере следующего задания:

*Необходимо обеспечить хранение в базе данных следующей информации:*

* *информация о студентах (включает Ф.И.О., домашний адрес, паспортные данные, номер зачетки, дата рождения, группа);*
* *информация о специальностях (наименование специальности, шифр);*
* *информация о группах (специальность, год поступления, номер группы).*

*Обеспечить выдачу документа “Список группы”, содержащего поля: порядковый номер, Ф.И.О., номер зачетки.*

Построение объектной модели выполняется в пакете Rational Rose (StarUML). Для этого создадим пустой проект (File->New->From Template->Rational. Начинать выполнение работы следует с диаграммы прецедентов. Ее строят в области Main секции Use Case View, как показано на рис.9.

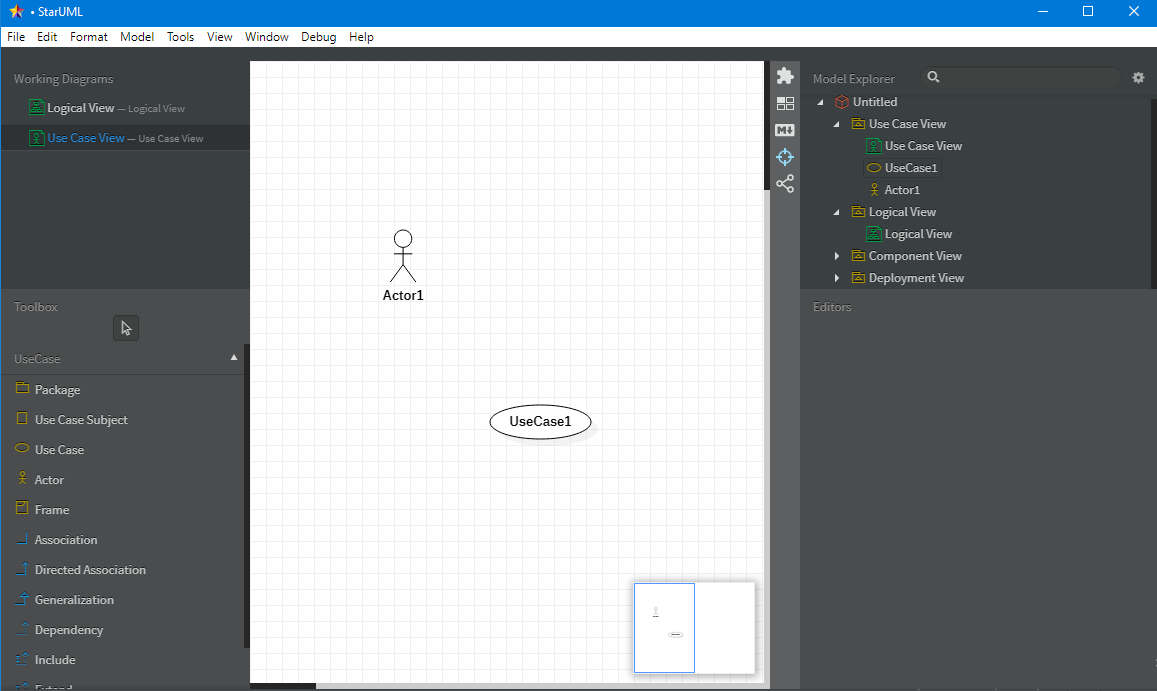


рис. 9 Общий вид интерфейса StarUML

Перед началом построения диаграммы необходимо определить роли пользователей системы (актеров) и их функции (прецеденты). В нашем случае с системой работают два актера – это «Работник учебного отдела» и «Работник деканата». В функции работника учебного отдела входит ведение списка специальностей (под *ведением списка* мы будем понимать добавление записей, их корректировку и удаление). Функции работника деканата включают в себя ведение списка студентов и ведение списка групп.

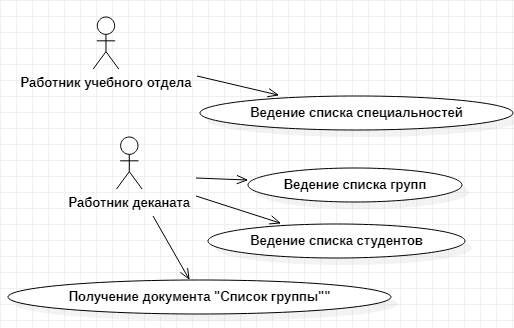


рис. 10 Диаграмма прецедентов (сценариев)

Построенная диаграмма изображена на рис. 10.

Далее в секции Logical View следует создать две диаграммы классов. Для этого можно создать два пакета. Первая диаграмма должна содержать классы интерфейса проектируемого приложения (см. рис. 11). Вторая диаграмма – данные, сущности базы данных (см. рис. 12).

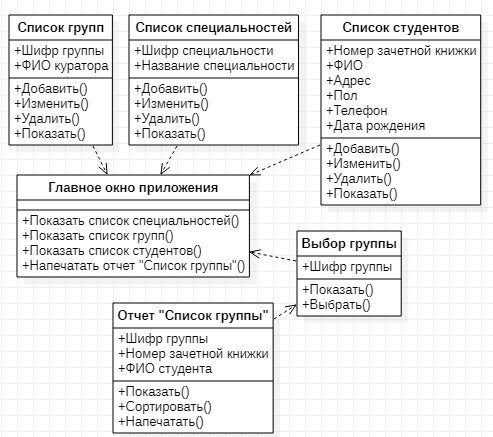


Рис. 11 Диаграмма классов (пакет «Интерфейс приложения»)

В построенной диаграмме классов отображены все формы будущего приложения и их взаимосвязь.

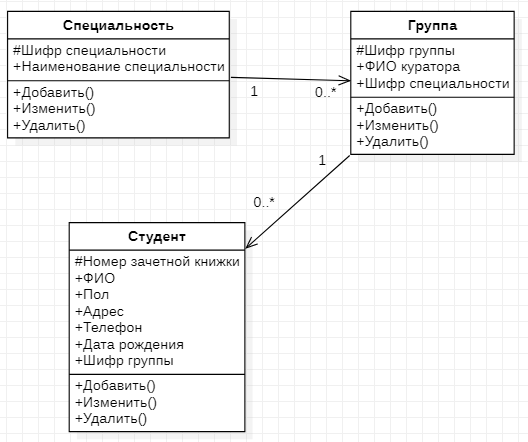


рис. 12 Диаграмма классов (пакет «Данные»)

Следующий этап построения объектной модели – создание диаграмм последовательностей. Диаграммы последовательностей создаются для каждого прецедента на диаграмме прецедентов. Чтобы добавить диаграмму последовательностей к прецеденту необходимо выбрать его в дереве и вызвать на нем контекстное меню (New🡪Sequence Diagram) как показано на рис. 13.

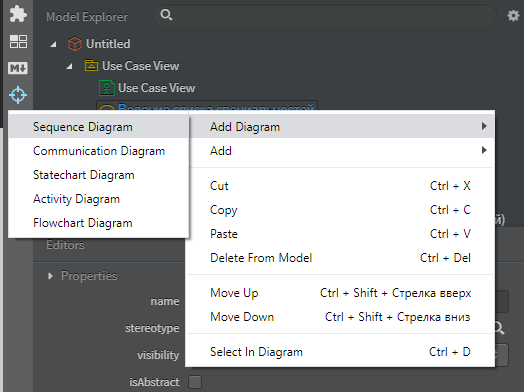


рис. 13 Создание диаграммы последовательностей для прецедента

Данный вид диаграммы строится из уже созданных элементов, включая подписи стрелок, которые выбираются из выпадающих списков (операций классов). Все элементы «перетаскиваются» из правой части главного окна StarUML.

Пример диаграммы последовательностей для прецедента «Ведение списка специальностей» представлен на рис. 14.

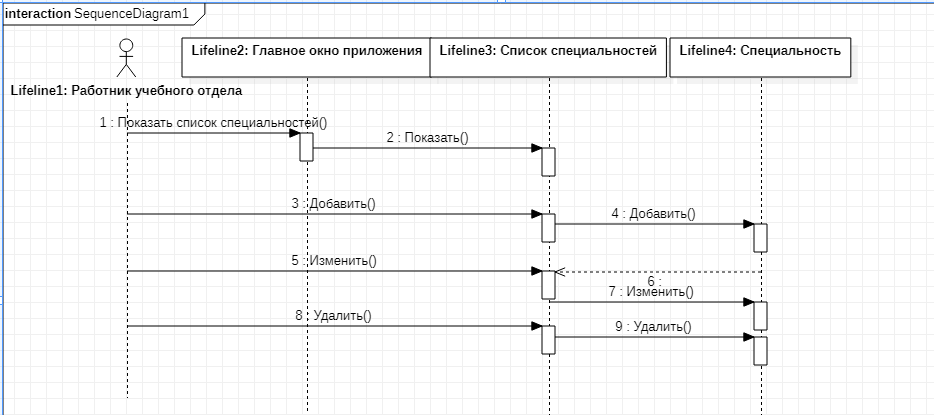


рис. 14 Диаграмма последовательностей для прецедента «Ведение списка специальностей»

Диаграмма читается сверху вниз и слева направо, показывает порядок выполнения функции (прецедента), взаимосвязь пользователя, интерфейса и данных.

На рис. 15 показан пример диаграммы последовательности для выдачи документа «Список группы».

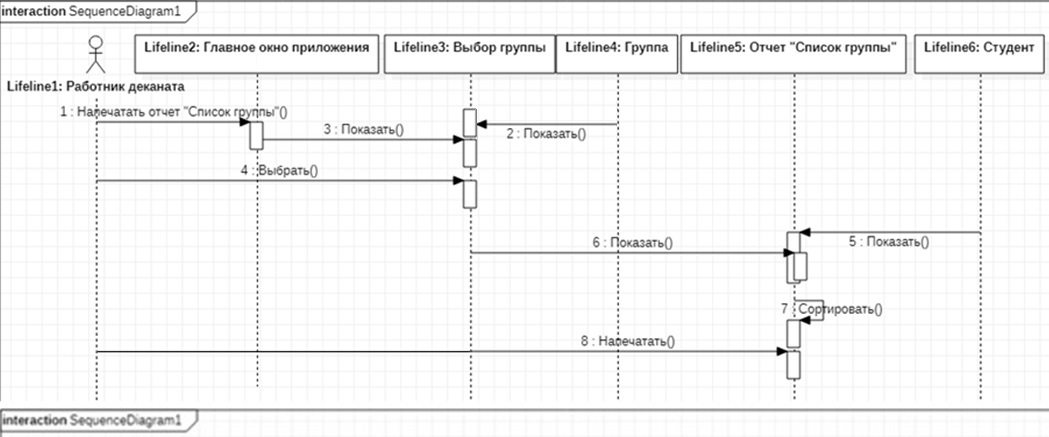


рис. 15 Диаграмма последовательностей для выходного документа