

А. М. Воронова, Л. В. Щеголева

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ: ОБЪЕКТНЫЙ ПОДХОД



ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ: ОБЪЕКТНЫЙ ПОДХОД

Подписано к изготовлению 26.12.2016.
1 CD-R. 2,5 Мб. Тираж 100 экз. Изд. № 413

Федеральное государственное бюджетное образовательное
учреждение высшего образования
ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
185910, Петрозаводск, пр. Ленина, 33

<https://petsu.ru>
Тел. (8142) 711 001

Изготовлено в Издательстве ПетрГУ
185910, Петрозаводск, пр. Ленина, 33

<https://press.petsu.ru/UNIPRESS/UNIPRESS.html>
Тел. / факс (8142) 781 540
nvpahomova@yandex.ru

ISBN 978-5-8021-1823-8



9 785802 118238

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

А. М. Воронова, А. В. Щеголева

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ: ОБЪЕКТНЫЙ ПОДХОД

Учебное электронное пособие для студентов

Петрозаводск
Издательство ПетрГУ
2017

УДК 004
ББК 32.81
В754

Выпускается по решению редакционно-издательского совета
Петрозаводского государственного университета

Издается в рамках реализации комплекса мероприятий
Программы стратегического развития ПетрГУ на 2017—2021 гг.

Рецензенты:

А. В. Жуков, кандидат технических наук;

А. В. Сусун, кандидат технических наук

Воронова, Анна Михайловна.

В754 Проектирование информационной системы: объектный подход [Электронный ресурс] : учебное электронное пособие для студентов / А. М. Воронова ; А. В. Щеголева ; М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. образования Петрозавод. гос. ун-т. – Электрон. дан. – Петрозаводск : Издательство ПетрГУ, 2017. – 1 электрон. опт. диск (CD-R) ; 12 см. – Систем. требования : PC, MAC с процессором Intel 1,3 ГГц и выше ; Windows, MAC OSX ; 256 Мб ; видеосистема : разрешение экрана 800×600 и выше ; графический ускоритель (опционально) ; мышь или другое аналогичное устройство. – Загл. с этикетки диска.

ISBN 978-5-8021-1864-1

В учебном пособии рассматривается конкретный пример предметной области, для которой строится проект информационной системы на основе объектного подхода к проектированию. Проект включает построение диаграмм сценариев, классов, состояния, активности, последовательности, кооперации, компонентов и топологии.

Издание предназначено для обеспечения самостоятельной работы студентов над индивидуальными проектами, которые они выполняют в рамках дисциплины «Методы и средства проектирования информационных систем».

УДК 004
ББК 32.81

ISBN 978-5-8021-1864-1

© Воронова А. М., Щеголева А. В., 2017
© Петрозаводский государственный университет, 2017

ВВЕДЕНИЕ

Настоящее учебное пособие является продолжением изданного ранее учебного пособия «Проектирование информационной системы: структурный подход» [1] и продолжает описание подходов к проектированию информационной системы. Теперь рассматривается объектно-ориентированный подход к описанию информационной системы. Он включает в себя проектирование объектной базы данных, описание функций и пользовательского интерфейса с помощью классов, сценариев, сообщений и других элементов. В качестве языка для описания системы используется UML – Унифицированный язык моделирования (Unified Modeling Language). С его помощью можно описать различные аспекты системы: ее логическую структуру, поведение системы, физическую структуру и др. Описание системы на языке UML представляет собой набор диаграмм.

Особенностью пособия является его практическая направленность. Краткое изложение теоретического материала сопровождается подробным описанием диаграмм, построенных для предметной области «Деятельность дежурного администратора гостиницы».

ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ «ГОСТИНИЦА»

Приведем краткое описание самой предметной области и бизнес-процессов, выполняемых дежурным администратором.

Гостиница располагает номерами нескольких категорий: люкс, полулюкс, одноместные, двухместные, блок из двух двухместных номеров, трехместные, которые обозначаются соответственно Л, П, 1С, 2С, 2А, 3С. Цена номера определяется только его категорией. Клиент может заранее заказать один или несколько номеров в гостинице. Этот процесс называется бронированием. При бронировании дежурный администратор заполняет карту брони. Клиент может без предварительного бронирования прийти в гостиницу, чтобы снять номер. При поселении клиента на него заполняется карта визита. Если клиент представляет собой группу (семью), то карта визита заполняются на каждого человека, карта брони – одна на всех (в карте брони указываются все бронируемые номера).

Проживающим в гостинице могут быть предоставлены дополнительные услуги (питание, бассейн, тренажерный зал, аренда конференц-зала, аренда места на автостоянке и др.).

Все оказанные услуги вносятся в карту визита клиента. В любой момент времени клиент может оплатить проживание и услуги. Для этого дежурный администратор выписывает наличный счет, который прикрепляется к карте визита. Для оплаты проживания всегда оформляется отдельный счет, в котором нет других услуг, этот счет включает стоимость номера и стоимость бронирования, если этот номер был забронирован клиентом. При выезде из гостиницы для всех неоплаченных ранее услуг формируется счет (или несколько по желанию клиента), а также формируется счет за проживание, если оно не было оплачено ранее, после чего карта визита закрывается.

Если клиент, забронировавший номер в гостинице, не приезжает в указанный срок, то его имя заносится в черный список и в следующий раз ему может быть отказано в бронировании.

Если клиент заранее отказывается от бронирования, его карта брони аннулируется и передается в архив.

Все карты брони, карты визита, счета сохраняются в течение пяти лет.

Ежедневно в конце смены каждый дежурный администратор готовит отчет. Отчет содержит информацию о выписанных дежурным администратором в течение смены картах визита и счетах.

После заполнения карты брони или карты визита, а также после отъезда клиента из гостиницы дежурный администратор вносит соответствующую информацию в Лист заселения, который отражает текущее состояние гостиницы.

Нумерация карт брони, карт визита и счетов начинается с единицы в начале каждого календарного года.

При выполнении бизнес-процессов дежурный администратор заполняет следующие документы: Карта визита (рис. 1), Карта брони (рис. 2), Счет (рис. 3), Лист заселения (рис. 4) и Отчет (рис. 5).

Карта визита № _____ Фамилия _____ Имя _____ Отчество _____ Паспортные данные: серия и номер _____ паспорта _____ кем и когда _____ выдан паспорт _____ Дата приезда _____ Дата отъезда _____ Комната _____	Карта брони № _____ Стоимость _____ <input type="checkbox"/> _____ <div style="text-align: right;">оплачено</div>				
Оказанные услуги					
Наименование	Дата	Цена	Количество	Сумма	Оплачено
Выписанные счета					
Номер счета	Дата	Сумма			

Рис. 1. Карта визита

Карта брони №	_____	Дата	_____
ФИО клиента	_____		
Дата приезда	_____		
Дата отъезда	_____		
Бронируемые комнаты	_____		
Контактная информация клиента	_____		
Деж. адм-р	_____		
<i>подпись</i>			

Рис. 2. Карта брони

Счет №	_____	Дата	_____	
ФИО	_____			
Дата приезда	_____			
Дата отъезда	_____			
Комната	_____			
Наименование	Дата	Цена	Количество	Сумма
Итого				
Сумма прописью	_____			
Деж. адм-р	_____			
<i>подпись</i>				

Рис. 3. Счет

Люкс	101 Иванов 2.10–4.10	102	103 Лукин (бронь) 12.12– 18.12			
Одно- местные	201	202	203 Розова (бронь) 10.10– 14.10	204 Сидоров 30.09–3.10	205	206
...

Рис. 4. Лист заселения по состоянию на 2 октября

Отчет дежурного администратора			ФИО _____
			Дата _____
Комната	ФИО	Дата отъезда	
Номер счета	Комната	ФИО	Сумма
<div style="border-top: 1px solid black; width: 150px; margin-left: 0;"></div> <div style="margin-left: 100px;">подпись</div>			

Рис. 5. Отчет


ДИАГРАММА СЦЕНАРИЕВ

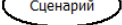
При проектировании информационной системы необходимо определить круг ее пользователей и те сервисы или функции, которая система будет выполнять. Для этого строится диаграмма сценариев.

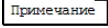
Диаграмма сценариев (use case diagram) отображает отношения между актерами и прецедентами. Актером называют роль пользователя. В качестве пользователя может выступать человек, система, подсистема. Прецедент представляет собой целостный фрагмент поведения системы без учета его внутренней структуры. Диаграммы сценариев описывают функциональное назначение системы, т. е. то, что система будет делать в процессе своего функционирования.

Диаграмма сценариев позволяет заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую систему.

Диаграмма состоит из нескольких основных элементов:

Актер  представляет собой любую внешнюю по отношению к моделируемой ИС сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей. Примерами могут служить: клиент фирмы, сотрудник фирмы, обслуживающий персонал, администратор отеля и т. д.

Сценарий  – сервис, который информационная система предоставляет пользователю (актеру). Примерами могут служить: внести в базу нового клиента, проверить состояние оплаты заказа клиентом, оформить заказ клиента и т. д.

Примечание  необходимо для добавления дополнительной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта. Это могут быть комментарии, уточнения, пояснения и т. д.

В каждой диаграмме между сценариями (или сценариями и актером) обязательно должны быть установлены связи (отношения). Существует четыре основных вида отношений между сценариями:

1. Отношение обобщения (generalization) показывает, что один сценарий может быть обобщен другим сценарием. Обозначается сплошной линией со стрелкой, в виде незакрашенного треугольника.
2. Отношение включения (include) показывает, что один сценарий включает в себя другой сценарий. Обозначается пунктирной линией со стрелкой, направленной от базового сценария к включаемому. Линия помечается ключевым словом «include» («включает»).
3. Отношение расширения (extend) используется, когда один сценарий расширяет возможности, предоставляемые другим сценарием. Обозначается пунктирной линией со стрелкой, направленной от того сценария, который является расширением для исходного сценария. Линия помечается ключевым словом «extend» («расширяет»).
4. Отношение ассоциации (association) показывает, какую конкретную роль играет актер при взаимодействии со сценарием. Обозначается сплошной линией между актером и сценарием. Эта линия может иметь дополнительные условные обозначения, такие, например, как имя и кратность.

Приведем пример фрагмента диаграммы сценариев для разрабатываемой ИС «Гостиница» (рис. 6).

Разрабатываемая информационная система предназначена для автоматизации деятельности дежурного администратора, поэтому **актером** выступает «Дежурный администратор». В случае большего числа ролей пользователей, взаимодействующих с ИС, для каждой такой роли создается отдельный актер.

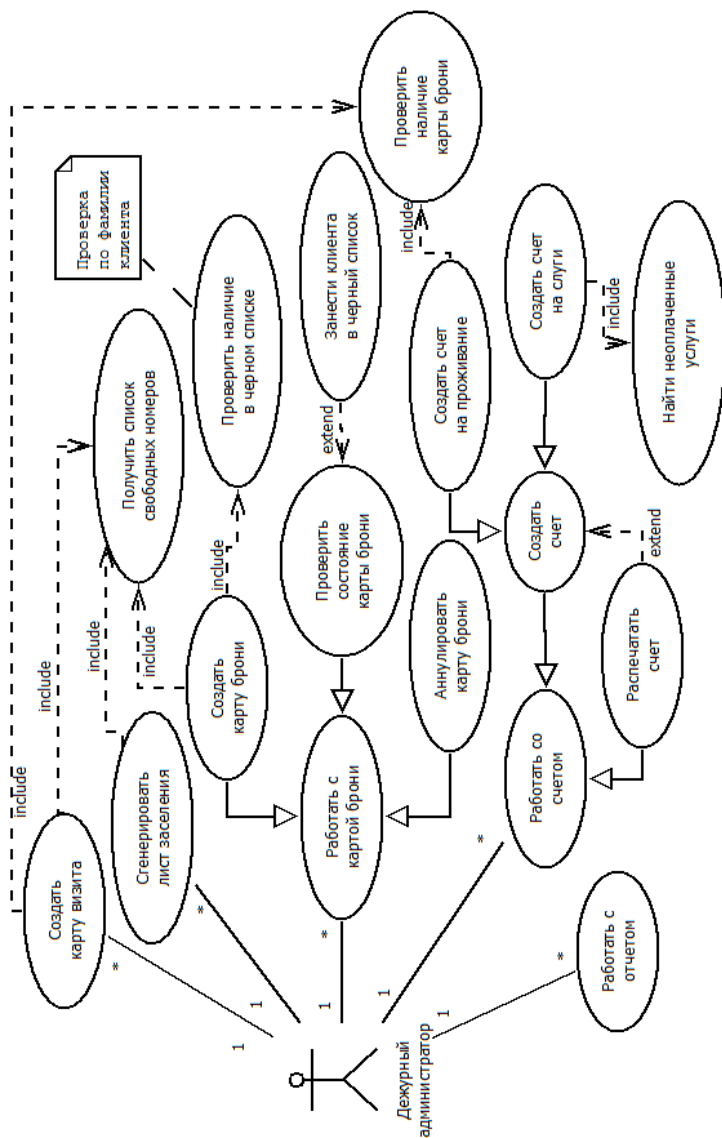


Рис. 6. Диаграмма сценариев

С актером связаны **сценарии** – те сервисы, которые ИС ему предоставля-ет. Сценарии представляют основные функциональные обязанности дежурного администратора: создание карты визита при заселении клиента, работа с картой брони, работа со счетами и отчетами, генерация листа заселения, содержащего текущую информацию о состоянии номерного фонда гостиницы.

Так, на диаграмме появляются сценарии: «Создать карту визита», «Сгенерировать лист заселения», «Работать с картой брони», «Работать со счетом», «Работать с отчетом». Эти сценарии соединены с актером «Дежурный администратор» отношением **ассоциации** с указанием кратности связи один-многим, которая означает, что один дежурный администратор может при работе несколько раз вызывать соответствующие сервисы ИС.

Дежурный администратор при создании карты визита клиента проверяет наличие карты брони, содержащей данные о заселении. Также при создании карты визита проверяется наличие свободных мест в гостинице для заселения клиента. На диаграмме сценариев эти условия отображаются следующим образом. Сценарий «Создать карту визита» соединен со сценарием «Проверить наличие карты брони» отношением **включения**, потому что при выполнении сценария «Создать карту визита» обязательно происходит выполнение сценария «Проверить наличие карты брони». Аналогично сценарий «Создать карту визита» соединен со сценарием «Получить список свободных номеров» отношением **включения**, потому что при выполнении сценария «Создать карту визита» обязательно происходит выполнение сценария «Получить список свободных номеров».

Дежурный администратор имеет доступ к функции автоматической генерации текущего листа заселения. При выполнении этой функции обязательно происходит проверка наличия свободных мест в гостинице. Это отражено на диаграмме следующим образом: сценарий «Сгенерировать лист заселения» соединен со сценарием «Получить список свободных номеров» отношением **включения**, потому что при выполнении сценария «Сгенерировать лист заселения» обязательно происходит вызов функции «Получить список свободных номеров».

При работе с картой брони дежурному администратору доступны следующие функции: создать карту брони, проверить состояние карты брони, аннулировать карту брони. Соответствующие сценарии соединены со сценарием «Работать с картой брони» отношением обобщения. При выполнении сценария «Создать карту брони» обязательно происходит выполнения сценария «Получить список свободных номеров», поэтому на диаграмме эти сценарии соединены отношением **включения**. Также при выполнении сценария «Создать карту брони» обязательно происходит выполнения сценария «Проверить наличие в черном списке», поэтому на диаграмме эти сценарии соединены отношением **включения**.

При проверке состояния карты брони возможен вызов сценария занесения клиента в черный список, если карта брони была просрочена, т. е. клиент не приехал в указанную в брони дату. Если же карта брони не просрочена, то вызов этого сценария не происходит. Поэтому на диаграмме сценарий «Проверить состояние карты брони» связан со сценарием «Занести клиента в черный список» отношением **расширения**, т. е. расширяющий сценарий «Занести клиента в черный список» не обязательно будет выполняться при каждом вызове сценария «Проверить состояние карты брони».

Сценарий «Работать со счетом» **обобщает** следующие сценарии: «Создать счет», «Распечатать счет», т. е. содержит описание основных функций, которые ИС предоставляет при работе со счетом. Сценарий «Создать счет» доступен в двух вариантах: «Создать счет на услуги» и «Создать счет на проживание», поэтому связан отношением **обобщения** с этими сценариями.

При создании счета на проживание обязательно происходит проверка наличия карты брони, чтобы включить в счет услуги бронирования. Поэтому на диаграмме сценарий «Создать счет на проживание» связан отношением **включения** со сценарием «Проверить наличие карты брони».

При создании счета на услуги обязательно происходит проверка наличия неоплаченных услуг, чтобы включить эти услуги в счет. Поэтому на диаграмме

сценарий «Создать счет на услуги» связан отношением **включения** со сценарием «Найти неоплаченные услуги».

При выполнении сценария «Создать счет» есть возможность его распечатать, поэтому сценарий «Создать счет» соединен со сценарием «Распечатать счет» отношением **расширения**.

Комментарий может быть связан пунктирной линией с любым элементом любой диаграммы UML и содержит необходимые пояснения. Например, на диаграмме сценариев комментарий уточняет, что проверка производится только по фамилии клиента.

Следует отметить, что для предметной области «Гостиница» представленная на рис. 6 диаграмма не является полной, т. е. в нее включены не все функции системы.

В качестве самостоятельного задания дополните эту диаграмму сценариями и связями.

ДИАГРАММА КЛАССОВ

Диаграмма классов предназначена для представления статической структуры модели информационной системы в терминологии классов объектно-ориентированного программирования.

Классом называется именованное описание совокупности объектов с общими атрибутами, операциями, связями и семантикой, поведением и отношениями с объектами из других классов. Класс на диаграмме показывается в виде прямоугольника, разделенного на три области. В верхней содержится название класса, в средней – описание атрибутов (свойств), в нижней – названия операций – услуг, предоставляемых объектами этого класса.

У каждого класса должно быть имя (текстовая строка), уникально отличающее его от всех других классов. Рекомендуется в качестве имен классов использовать существительные, записанные по практическим соображениям без пробелов. Имена классов образуют словарь предметной области.

Атрибутом класса называется именованное свойство класса, описывающее множество значений, которые могут принимать экземпляры этого свойства. Класс может иметь любое число атрибутов (в частности, не иметь ни одного атрибута). Свойство, выражаемое атрибутом, является свойством моделируемой сущности, общим для всех объектов данного класса, записывается во второй сверху секции прямоугольника класса. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения.

При реализации объекта в программном коде для атрибутов будет выделена необходимая для хранения память, и каждый атрибут будет иметь конкретное значение в любой момент времени работы программы. Объектов одного класса в программе может быть сколь угодно много, все они имеют одинаковый набор атрибутов, описанный в классе, но значения атрибутов у каждого объекта свои и могут изменяться в ходе выполнения программы.

Для каждого атрибута класса можно задать видимость (visibility). Эта характеристика показывает, доступен ли атрибут для других классов. В UML определены следующие уровни видимости атрибутов:

- открытый (public) – атрибут виден для любого другого класса (объекта);
- защищенный (protected) – атрибут виден для потомков данного класса;
- закрытый (private) – атрибут не виден внешними классами (объектами) и может использоваться только объектом, его содержащим.

Последнее значение позволяет реализовать свойство инкапсуляции данных. Например, объявив все атрибуты класса закрытыми, можно полностью скрыть от внешнего мира его данные, гарантируя отсутствие несанкционированного доступа к ним. Это позволяет сократить число ошибок в программе. При этом любые изменения в составе атрибутов класса никак не скажутся на остальной части системы.

Класс содержит объявления **операций**, представляющих собой определения запросов, которые могут выполнять объекты данного класса. Каждая операция имеет сигнатуру, содержащую имя операции, тип возвращаемого значения и список параметров, который может быть пустым. Реализация операции в виде процедуры – это метод, принадлежащий классу. Для операций, как и для атрибутов класса, определено понятие «видимость». Закрытые операции являются внутренними для объектов класса и недоступны из других объектов. Остальные образуют интерфейсную часть класса и являются средством интеграции класса в систему.

В диаграмме классов могут участвовать связи: зависимость (dependency), обобщение (generalization), ассоциация (association), агрегация и композиция.

Отношение зависимости применяется, когда изменение в спецификации одного класса может повлиять на поведение другого класса, использующего первый класс. Чаще всего зависимости применяют в диаграммах классов, чтобы отразить в сигнатуре операции одного класса тот факт, что параметром этой операции могут быть объекты другого класса. Графически представляется

пунктирной линией со стрелкой на конце, идущей от зависимого элемента к тому, от которого он зависит.

Отношение обобщения описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка. На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на класс-предок, а другой ее конец указывает на класс-потомок.

Отношением ассоциацией называется структурная связь, показывающая, что объекты одного класса некоторым образом связаны с объектами другого или того же самого класса. Допускается, чтобы оба конца ассоциации относились к одному классу. В ассоциации могут связываться два класса, и тогда она называется бинарной.

Данное отношение обозначается сплошной линией с дополнительными специальными символами, которые характеризуют отдельные свойства конкретной ассоциации. В качестве дополнительных специальных символов могут использоваться имя ассоциации, которая выражает суть связи, а также имена и кратность классов-ролей ассоциации. Кратность показывает, сколько объектов каждого класса может участвовать в ассоциации. Кратность указывается у каждого конца ассоциации (полюса) и задается конкретным числом или диапазоном чисел.

Допускается создание ассоциаций, связывающих сразу n классов. N -ассоциация графически обозначается ромбом, от которого ведут линии к символам классов данной ассоциации. В этом случае ромб соединяется с символами соответствующих классов сплошными линиями.

Ассоциация сама может обладать свойствами класса, т. е. иметь атрибуты и операции. В этом случае она называется класс-ассоциацией и может рассматриваться как класс, у которого, помимо явно указанных атрибутов и операций, есть ссылки на все связываемые ею классы.

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности. Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой «целое». Остальные классы являются его «частями».

Отношение композиции является частным случаем отношения агрегации. Это отношение служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части. Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой класс-композицию или «целое». Остальные классы являются его «частями».

Диаграмма классов включает классы, описывающие структуры хранящихся в ИС данных, классы экранных форм, классы обработки данных, классы шаблоны и др.

ПОСТРОЕНИЕ КЛАССОВ-ДАНЫХ

Построение классов, описывающих структуры данных предметной области, осуществляется на основе инфологической модели предметной области.

Для каждого класса инфологической модели необходимо построить класс диаграммы классов, свойства класса инфологической модели становятся свойствами класса диаграммы классов. Каждую связь инфологической модели необходимо отразить в диаграмме классов.

Связь вида «один-ко-многим», «многие-ко-многим» со свойствами и без свойств отражается по-разному в диаграмме классов.

Если класс 1 связан отношением «один-ко-многим» с классом 2, тогда в диаграмме классов можно добавить в класс 1 множественный атрибут кратности «*», содержащий ссылку на класс 2. Реляционная модель запрещает использование множественных атрибутов. В отличие от нее, в объектной модели классы могут содержать множественные атрибуты. В классе 2 можно поместить атрибут, указывающий на класс 1. Классы 1 и 2 нужно соединить одной из подходящих по смыслу связей: ассоциацией с указанием кратности, композицией или агрегацией.

Если класс 1 связан отношением «многие-ко-многим» с классом 2 без свойств связи, тогда в диаграмме классов не нужно создавать отдельный класс: можно добавить в класс 1 множественный атрибут кратности «*», содержащий указание на класс 2. В классе 2 можно поместить множественный атрибут кратности «*», указывающий на класс 1. Установить связь между классами.

Если класс 1 связан отношением «многие-ко-многим» с классом 2 с дополнительными свойствами связи, тогда в диаграмме классов нужно создать ассоциативный класс, куда поместить свойства связи и указание на один или оба связываемые классы. В класс 1 можно добавить атрибут, содержащий указание на класс 2 через ассоциативный класс. В классе 2 можно поместить атрибут, указывающий на класс 1 через ассоциативный класс. Установить связь между классами.

Так построена диаграмма классов, содержащая основные классы предметной области и отражающая связи между классами. При необходимости построенную диаграмму классов можно дополнить введением классов-родителей, содержащих схожие атрибуты и методы разных классов, и связать эти классы-родители с исходными классами-потомками отношением наследования, оставив у классов-потомков отличающиеся свойства и методы.

Построение классов-данных для «Гостиницы»

Инфологическая модель предметной области «Гостиница» содержит классы, приведенные на рис. 7:

- клиент,
- комната,
- категория,
- услуга,
- карта визита,
- карта брони,
- администратор,
- счет,
- отчет.

Эти классы составляют основу для построения объектной модели предметной области в виде диаграммы классов: соответствующие классы-данных с атрибутами переносятся в диаграмму классов, приведенную на рис. 8.

Так появляются классы Клиент, Комната, Категория, Услуга, Карта визита, Карта брони, Администратор, Отчет.

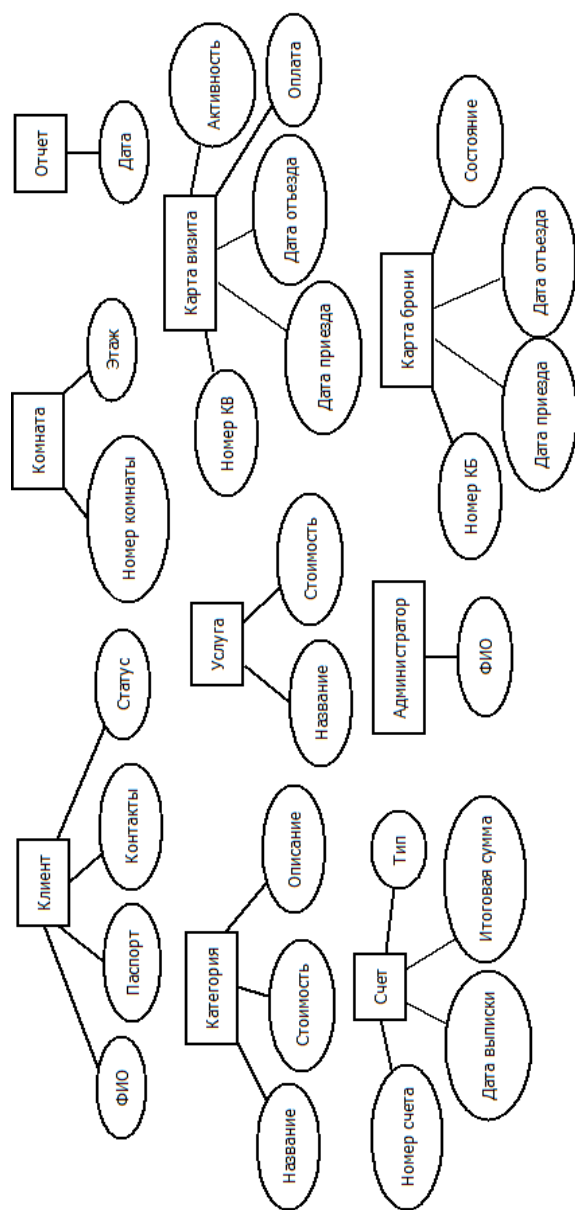


Рис. 7. Классы инфологической модели предметной области

Класс инфологической модели Счет целесообразно разделить на два класса: Счет за проживание, Счет за услугу, так как эти счета содержат разную информацию и создаются администратором независимо друг от друга. Одинаковые атрибуты счета за проживание и счета за услугу переносятся в родительский класс Счет, отличающиеся свойства становятся атрибутами разных классов-потомков. Между родительским классом и классами потомками установлено отношение **обобщения** (наследования): Счет за услугу является потомком класса Счет; Счет за проживание является потомком класса Счет.

Свойства классов инфологической модели становятся атрибутами классов объектной модели. На данном этапе проектирования появляются базовые атрибуты классов, которые в дальнейшем будут дополнены атрибутами, отражающими характер связей между классами.

Класс Категория содержит атрибуты Название, Стоимость, Описание.

Класс Комната содержит атрибуты Номер, Этаж.

Класс Карта брони содержит атрибуты Номер карты брони, Дата приезда, Дата отъезда, Состояние.

Класс Карта визита содержит атрибуты Номер карты визита, Дата приезда, Дата отъезда, Статус оплаты, Статус активности.

Класс Клиент содержит атрибуты ФИО, Паспорт, Контакты, Статус.

Класс Администратор содержит атрибуты ФИО.

Класс Услуга содержит атрибуты Название, Стоимость.

Класс Счет содержит атрибуты Номер счета, Дата выписки.

Класс Отчет содержит атрибут Дата.

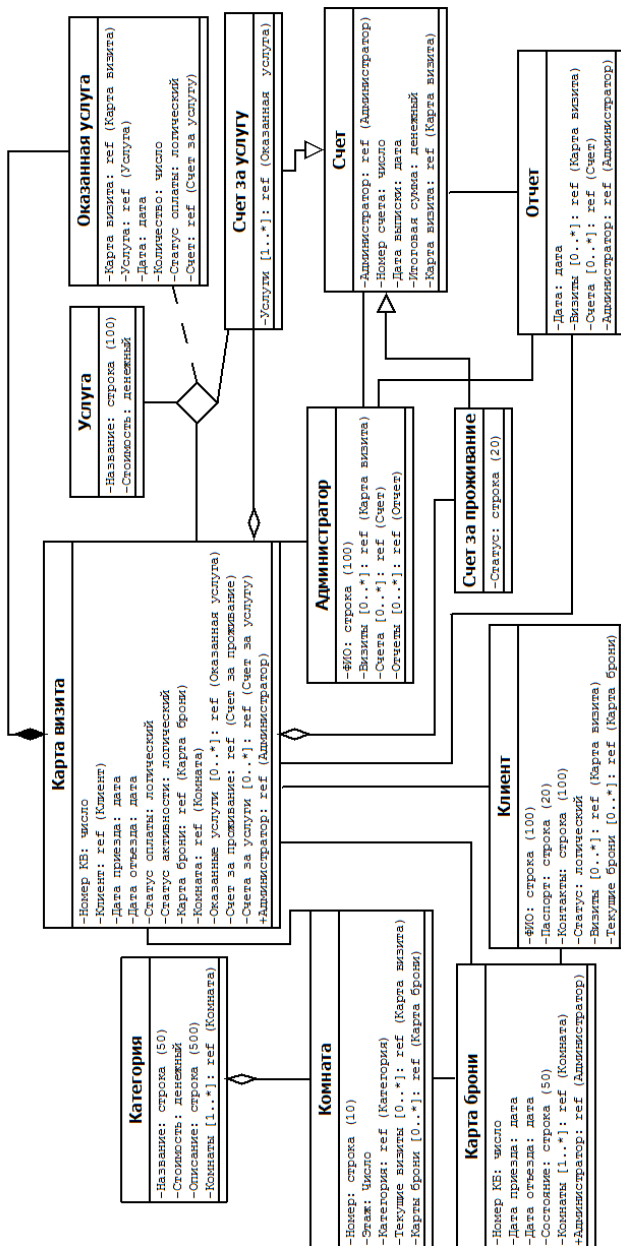


Рис. 8. Диаграмма классов-данных предметной области

Все связи инфологической модели, приведенные на рис. 9 и 10, необходимо отразить в диаграмме классов, а именно дополнить классы атрибутами, отражающими характер связи между классами и нарисовать соответствующие связи между классами объектной модели данных.

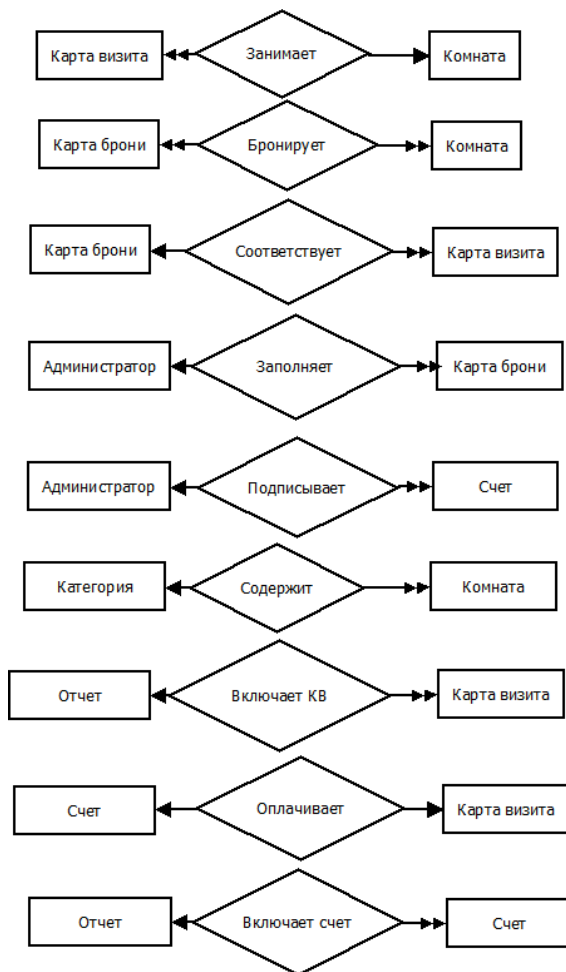


Рис. 9. Связи между классами инфологической модели

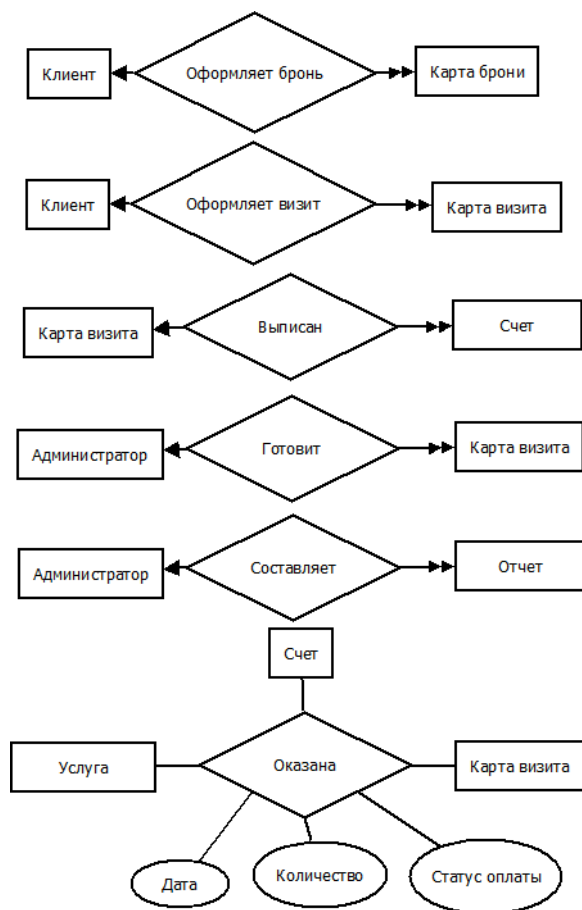


Рис. 10. Связи между классами инфологической модели

Связь «Карта визита занимает Комнату» имеет тип «многие-к-одному» и отражается в объектной модели так: в класс Карта визита добавляется атрибут Комната, ссылающийся на класс Комната (Комната: ref (Комната)); в класс Комната добавляется множественный атрибут Текущие визиты, ссылающиеся на карту визита (Текущие визиты [0..*]: ref (Карта визита)). Класс Карта визита содержит ссылку на занимаемую комнату. Класс Комната содержит массив

ссылок на карты визита за текущий период. Текущих визитов у комнаты может не быть. Между классами Комната и Карта визита устанавливается связь **ассоциация** один ко многим.

Связь «Карты брони бронируют Комнаты» имеет тип «многие-ко-многим» без свойств связи. В объектной модели в класс Карта брони добавляется множественный атрибут Комнаты, ссылающиеся на забронированные комнаты: Комнаты [1..*]: ref (Комната). Хотя бы одна комната должна быть забронирована при создании карты брони. В класс Комната добавляется множественный атрибут Карты брони, ссылающиеся на связанные с комнатой бронирования: Карты брони[0..*]: ref (Карта брони). Бронирований у комнаты может не быть. Между классами Комната и Карта брони устанавливается связь **ассоциация** многие ко многим. В отличие от реляционной модели, требующей создание отдельной таблицы для связи многие-ко-многим, в объектной модели не обязательно создание отдельного ассоциативного класса. Необходимые ссылки добавляются непосредственно в классы, участвующие в связи многие-ко-многим.

Связь «Карте брони соответствуют Карты визита» имеет тип «один-ко-многим». В объектной модели в класс Карта визита добавляется атрибут Карта брони, ссылающиеся на соответствующую карту брони: Карта брони: ref (Карта брони). Обратная связь, именно хранение в карте брони связанных с ней карт визита, является нецелесообразным, поэтому этот атрибут в карту брони не добавляется. Объектная модель допускает такие упрощения, если данное отношение не является важным для рассматриваемого класса. Однако каждая связь инфологической модели должна каким-то образом быть отражена в объектной модели, но не обязательно полностью отражена для каждого класса, участвующего в отношении. Между классами Карта брони и Карта визита устанавливается связь **ассоциация** один ко многим.

Связь «Администратор заполняет Карты брони» имеет тип «один-ко-многим». В объектной модели в класс Карта брони добавляется атрибут Администратор, ссылающийся на соответствующего Администратора: Администрат

тор: ref (Администратор). Обратная связь, именно хранение в классе Администратор заполненных им карт брони не является необходимым, поэтому этот атрибут в класс Администратор не добавляется. Между классами Администратор и Карта брони устанавливается связь **ассоциация** один ко многим.

Связь «Администратор подписывает Счета» имеет тип «один-ко-многим». В объектной модели в класс Счет добавляется атрибут Администратор, ссылающийся на соответствующего Администратора: Администратор: ref (Администратор). В класс Администратор добавляется множественный атрибут Счета, ссылающиеся на подписанные администратором Счета: Счета[0..*]: ref (Счет). У администратора может не быть счетов, которые он выписал. Между классами Администратор и Счет устанавливается связь **ассоциация** один ко многим.

Связь «Категория содержит Комнаты» имеет тип «один-ко-многим». В объектной модели в класс Комната добавляется атрибут Категория, которая показывает категорию комнаты: Категория: ref (Категория). В класс Категория добавляется множественный атрибут Комнаты, которые относятся к соответствующей категории: Комнаты[1..*]: ref (Комната). В каждой категории должна быть хотя бы одна комната. Между классами Категория и Комната устанавливается связь **агрегация**, означающая, что Категория является целым, а комната частью целого, однако Комната может существовать без категории.

Связь «Отчет включает Карты визита» имеет тип «один-ко-многим». В объектной модели в класс Карта визита нет смысловой необходимости добавлять ссылку на отчет, в который эта карта визита включена. В класс Отчет добавляется множественный атрибут Карты визита, которые включены в отчет: Карты визита[0..*]: ref (Карта визита). В отчет может быть не включено ни одной карты визита. Между классами Отчет и Карта визита устанавливается связь **агрегация**, означающая, что Отчет является целым и включает в себя Карты визита, однако Карта визита может существовать без Отчета.

Связь «Отчет включает Счета» имеет тип «один-ко-многим». В объектной модели в класс Счет нет необходимости добавлять ссылку на отчет, в который

этот счет включен. В класс Отчет добавляется множественный атрибут Счета: Счета[0..*]: ref(Счет). В отчет может быть не включено ни одного счета. Между классами Отчет и Счет устанавливается связь **агрегация**, означающая, что Отчет является целым и включает в себя Счета как составные части, однако Счет может существовать без Отчета.

Связь «Клиент оформляет бронь Карты брони» имеет тип «один-ко-многим». В объектной модели в классе Карта брони можно хранить ссылку на клиента, или не хранить ссылку на клиента. В данном случае решили, что в классе Карта брони нет необходимости хранить ссылку на клиента, поэтому эта ссылка не добавляется. В класс Клиент добавляется множественный атрибут Текущие брони, ссылающиеся на Карты брони: Текущие брони [0..*]: ref(Карта брони). У клиента может не быть карт брони, которые он оформил. Между классами Клиент и Карта брони устанавливается связь **ассоциация** один ко многим.

Связь «Клиент оформляет визит Карты визита» имеет тип «один-ко-многим». В объектной модели в классе Карта визита добавляется атрибут Клиент, ссылающийся на соответствующего Клиента: Клиент: ref(Клиент). В класс Клиент добавляется множественный атрибут Карты визита, ссылающиеся на все визиты клиента: Карты визита[0..*]: ref(Карта визита). У клиента может не быть визитов. Между классами Клиент и Карта визита устанавливается связь **ассоциация** один ко многим.

Связь «Счет оплачивает Карту визита» имеет тип один к одному, в данном случае имеется в виду счет за проживание. Связь по «Карте визита выписаны Счета» имеет тип «один-ко-многим», в данном случае имеются в виду счета за услуги. К объектной модели, кроме родительского класса Счет, были добавлены классы-потомки Счет за проживание и Счет за услугу, так как введение этих классов уточняет предметную область. Теперь в класс Карта визита добавляется два атрибута: атрибут Счет за проживание, ссылающийся на класс Счет за проживание: Счет за проживание: ref(Счет за проживание) и множественный атрибут Счета за услуги, ссылающиеся на Счета за услуги: Счета

за услуги[0..*]: ref(Счет за услугу). Счет за проживание обязательно оформляется по карте визита. Счетов по услугам по карте визита может не быть. В класс Счет добавляется атрибут Карта визита, ссылающийся на соответствующую Карту визита: Карта визита: ref(Карта визита). Между классами Карта визита и Счет на проживание устанавливается связь **агрегация**, означающая, что Карта визита является целым и включает в себя Счет за проживание. Между классами Карта визита и Счет на услуги устанавливается связь **агрегация**, означающая, что Карта визита является целым и включает в себя Счета за услуги.

Связь «Администратор готовит Карты визита» имеет тип «один-ко-многим». В объектной модели в класс Карта визита добавляется атрибут Администратор, ссылающийся на соответствующего Администратора: Администратор: ref(Администратор). В класс Администратор добавляется множественный атрибут Карты визита, ссылающиеся на подготовленные администратором Карты визита: Карты визита[0..*]: ref(Карта визита). У администратора может не быть карт визита, которые он готовит. Между классами Администратор и Карта визита устанавливается связь **ассоциация** один ко многим.

Связь «Администратор составляет Отчеты» имеет тип «один-ко-многим». В объектной модели в класс Отчет добавляется атрибут Администратор, ссылающийся на соответствующего Администратора: Администратор: ref(Администратор). В класс Администратор добавляется множественный атрибут Отчеты, ссылающиеся на подготовленные администратором Отчеты: Отчеты[0..*]: ref(Отчет). У администратора может не быть отчетов, которые он составляет. Между классами Администратор и Отчет устанавливается связь **ассоциация** один ко многим.

Связь «Оказанная услуга» соединяет три класса Карта визита, Услуга и Счета, является небинарной связью и имеет свойства: дата, количество, статус оплаты. Связь означает, что определенного типа услуга оказана конкретному клиенту, т. е. вписана в его карту визита и оплачена посредством конкретного счета. Для отражения в объектной модели небинарных связей, а также бинарных связей типа «многие ко-многим» со свойствами необходимо добавление

ассоциативного класса. В объектную модель добавляется ассоциативный класс «Оказанная услуга», который содержит в качестве атрибутов свойства связи дата, количество, статус оплаты, также ссылку на класс Услуга: Услуга:ref(Услуга), ссылку на класс Счет на услугу: Счет [0..1]: ref(Счет на услугу). Счет на услугу выписывается по запросу клиента. В класс Карта визита добавляется множественный атрибут оказанных клиенту услуг: Оказанные услуги [0..*]: ref(Оказанная услуга). Не все клиенты пользуются дополнительными услугами, поэтому их может не быть в карте визита. В класс Счет за услугу добавляется множественный атрибут, ссылающийся на оказанные услуги: Оказанные услуги [1..*]: ref(Оказанная услуга). В класс Услуга нет необходимости добавлять ссылки на класс оказанных услуг.

После того как основные классы и атрибуты классов спроектированы, следующим шагом является проектирование методов классов.

Классы обычно имеют:

- метод, который позволяет создать экземпляр данного класса;
- метод для удаления экземпляра класса;
- методы для работы с закрытыми атрибутами класса, включающие методы получения значений атрибутов класса, методы установки значений атрибутов класса;
- методы получения набора экземпляров данного класса;
- методы для работы со множественными атрибутами класса: метод добавления нового объекта в массив к классу, метод для удаления его из массива;
- другие методы, которые необходимы для поиска, сортировки, обработки экземпляров класса с учетом особенностей проектируемой предметной области.

Один из самых важных классов по предметной области «Гостиница» – это класс «**Комната**» (рис. 11). В нем задается информация по номеру комнаты; этажу, на котором она находится; категории, которой она принадлежит. Кроме того, за комнатой могут быть закреплены Визиты (связанные с данными из

Класса «Карта Визита») и Бронирования (связанные с данными из класса «Карта Брони»). Дополним класс «Комната» атрибутами, содержащими количество броней и количеством визитов для комнаты, которые могут понадобиться при работе с классом.

Комната
-Номер: строка (10) -Этаж: число -Категория: ref (Категория) -Текущие визиты [0..*]: ref (Карта визита) -Карты брони [0..*]: ref (Карта брони) -Количество броней: число -Количество визитов: число
+Создать(in Номер:строка, in Этаж:число,in Категория:ref (Категория)): Комната <class> +Удалить() +ПолучитьИнфКомната(out Номер:строка, out Этаж:число, out Категория:ref (Категория)) +ПолучитьСтатус(in ДатаНачала:дата, in ДатаОкончан:дата): число +ПолучитьВизиты(): List (ref (КартаВизита)) +ПолучитьБрони(): List (ref (КартаБрони)) +УстановитьНомер(in Номер:строка) +УстановитьЭтаж(in Этаж:номер) +УстановитьКатегорию(in Категория:ref (Категория)) +ДобавитьВизит(in Визит:ref (КартаВизита)) +ДобавитьБронь(in Бронь:ref (КартаБрони)) +УдалитьВизит(in Визит:ref (КартаВизита)) +КопироватьВизит(in Визит:ref (КартаВизита)) +РассчитатьБрони(in Комната:ref (Комната)): число +РассчитатьВизиты(in Комната:ref (Комната)): число +ПолучитьКомнаты(): list (ref (Комната))

Рис. 11. Класс «Комната»

Класс «Комната» содержит методы создания и удаления комнат, методы для получения и установления информации о комнатах, метод просмотра текущего статуса, методы работы с множественными атрибутами, включающие методы получения, добавления, удаления текущих визитов и карт брони к конкретной комнате, методы расчета количества броней и визитов по комнате, а также метод получения полного списка всех комнат с описаниями.

В классе «Категория» (рис. 12) хранятся категории номеров гостиницы. В нем задаются параметры распределения комнат по категориям: стоимость, описание того, что есть в комнате. Для каждой категории определено название.

Категория
-Название: строка (50) -Стоимость: денежный -Описание: строка (500) -Комнаты [1..*]: ref (Комната)
<pre> +Создать (in Название:строка, in Стоимость:денежный, in Описание:строка): Категория <class> +Удалить () +ПолучитьИнфКатегории (out Название:строка, out Стоимость:денежный, out Описание:строка) +ПолучитьКомнаты(): list (ref (Комната)) +УстановитьНазвание (in Название:строка) +УстановитьСтоимость (in Стоимость:денежный) +УстановитьОписание (in Описание:строка) +ДобавитьКомнату (in Комната:ref (Комната)) +УдалитьКомнату (in Комната:ref (Комната)) +ПолучитьСвободнКомнаты (in ДатаНачала:дата, in ДатаОкончан:дата): list (ref (Комната)) +ПолучитьКатегории(): list (ref (Категория)) <class> </pre>

Рис. 12. Класс «Категория»

Методы класса «Категория» позволяют получить данные по категориям, установить их, добавить комнаты к категории и удалить комнаты из категории, а также получить свободные комнаты на конкретный заданный период и полную информацию по всем категориям, существующим в Гостинице.

Класс «**Карта брони**» (рис. 13) содержит в себе все данные о бронировании комнат клиентом. Каждая Карта брони имеет номер, даты прибытия и отбытия клиента, сведения о занимаемых комнатах, состояние карты брони (активна, просрочена, аннулирована), и ссылка на администратора, создавшего карту брони.

Карта брони
-Номер КБ: число -Дата приезда: дата -Дата отъезда: дата -Состояние: строка (50) -Комнаты [1..*]: ref (Комната) +Администратор: ref (Администратор)
+Создать(in Дата приезда:дата, in Дата отъезда:дата, in Состояние:строка=активна, in Администратор:ref (Администратор)): Карта Визита <class> +Удалить() +ПолучитьИнфБронь(out Номер:строка, out Дата приезда:дата, out Дата отъезда:дата, out Администратор:ref (Администратор)) +ПолучитьКомнаты(): list (ref (Комната)) +УстановитьДаты(in Дата приезда:дата, in Дата отъезда:дата) +ДобавитьКомнату(in Комната:ref (Комната)) +УдалитьКомнату(in Комната:ref (Комната)) +ОтметитьПросроченной() +Аннулировать() +Закрыть() +ПолучитьПросроченныеБрони(in Дата:дата): list (ref (КартаБрони)) +ПолучитьБрони(): list (ref (КартаБрони))

Рис. 13. Класс «Карта брони»

Класс «Карта брони» содержит методы создания и удаления экземпляров класса, установки значений и получения значений всех атрибутов, включая методы добавления и удаления комнат, методы изменения статуса карты брони (отметить просроченной, аннулировать, закрыть), метод для получения полной информации по всем картам брони, информации по бронированию номеров за некоторый период времени, информации о просроченных картах брони на некоторую дату.

Класс «**Карта визита**» (рис. 14) – основной класс, содержащий в себе все данные по визиту клиента в гостиницу. Каждая карта визита имеет номер, даты прибытия и отбытия клиента, сведения о занимаемой комнате.

Карта визита
-НомерКВ: число -Клиент: ref (Клиент) -Дата приезда: дата -Дата отъезда: дата -Статус оплаты: логический -Статус активности: логический -Карта брони: ref (Карта брони) -Комната: ref (Комната) -Оказанные услуги [0..*]: ref (Оказанная услуга) -Счет за проживание: ref (Счет за проживание) -Счета за услуги [0..*]: ref (Счет за услугу) +Администратор: ref (Администратор)
+Создать(in Клиент:ref (Клиент), in Дата приезда:дата, in Дата отъезда:дата, in Статус оплаты:логический, in Бронь:ref (Карта брони), Комната:ref (Комната), Администратор:ref (Администратор)): Карта визита <class> +СоздатьПоКартеБрони(in Карта брони:ref (Карта брони)): Карта визита <class> +Удалить() +УстановитьБронь(Бронь:ref (Карта брони)) +ПолучитьБронь(): ref (Карта брони) +УдалитьБронь() +ДобавитьСчетПроживание(in Счет:ref (Счет за проживание)) +УдалитьСчетПроживание() +ДобавитьСчетУслуга(in Счет:ref (Счет за услуги)) +УдалитьСчетУслуга(Счет:ref (Счет за услугу)) +ДобавитьУслугу(in Услуга:ref (Оказанная услуга)) +УдалитьУслугу(in Услуга:ref (Оказанная услуга)) +ПолучитьИнфВизит(out НомерКВ:строка, out Клиент:ref (Клиент), out Дата приезда:дата, out Дата отъезда:дата, out Оплата:логический, out Активность:логический, Комната:ref (Комната), Администратор:ref (Администратор)) +ПолучитьУслуги(): list (ref (Оказанные услуги)) +ПолучитьСчетПроживание(): ref (Счет за проживание) +ПолучитьСчетУслуги(): list (ref (Счет за услуги)) +ПолучитьВизиты(): list (ref (Карта Визита)) +РассчитатьИтог(): денежный

Рис. 14. Класс «Карта визита»

В карте визита указаны ссылки на клиента и администратора, полная информация о которых содержится в отдельных классах, а также хранится ссылка на карту брони, если она была оформлена до приезда. В карте визита указывается информация о дополнительных услугах, оказываемых клиенту, выписанных счетах за проживание и за услуги.

В классе «Карта визита» есть атрибут, отвечающий за оплату по визиту – статус оплаты (оплачен, не оплачен), и атрибут активности карты визита – статус активности (активен, не активен).

Класс «Карта визита» содержит методы создания карт визита, создания карты визита по карте брони, удаления карты визита, установки значений и получения значений всех атрибутов, включая методы добавления и удаления бронирования, оказываемых услуг, счетов на проживание и пользование услугами, метод для получения полной информации по всем картам визита, метод расчета итоговой суммы по карте визита.

Класс **«Клиент»** (рис. 15) содержит атрибуты для хранения информации о клиенте: ФИО, паспортные данные, контакты, статус клиента (состоит или не состоит в черном списке).

Клиент

```
-#ИО: строка (100)
-ПаспортНомер: строка (20)
-ПаспортДатаВыдачи: дата
-ПаспортКемВыдан: строка (20)
-Контакты: строка (100)
-Статус: логический
-Визиты [0..*]: ref (Карта визита)
-Текущие брони [0..*]: ref (Карта брони)

+Создать (in #ИО:строка, in ПаспортНомер:строка,
                in ПаспортДатаВыдачи:дата,
                in ПаспортКемВыдан:строка,
                in Контакты:строка,
                in Статус:логический+Истина): Клиент <class>

+Удалить ()

+ПолучитьБрони (in #ИО:строка): list (ref (Карта брони))
+ПолучитьИнфКлиент (out #ИО:строка,
                    out Паспорт:строка,
                    out Контакты:строка)

+ПолучитьВизиты (): list (ref (Карта Визита))
+Установить#ИО (in #ИО:строка)
+УстановитьПаспортНомер (in Номер:строка)
+УстановитьПаспортДата (in Дата:дата)
+УстановитьПаспортКемВыдан (in Выдан:строка)
+УстановитьКонтакты (in Контакты:строка)
+УстановитьСтатус (in Статус:логический)
+ДобавитьВизит (Визит:ref (Карта Визита))
+УдалитьВизит (Визит:ref (Карта Визита))
+ДобавитьБронь (Бронь:ref (Карта Брони))
+УдалитьБронь (Бронь:ref (Карта Брони))
+ПолучитьКлиентов (): list (ref (Клиент))
```

Рис. 15. Класс «Клиент»

Кроме этого в этом классе хранится информация о бронированиях и визитах клиента.

Класс «Клиент» содержит методы для создания клиентов, удаления клиентов, получения информации о клиентах, получения всех визитов и броней, связанных с клиентом, установления значений всех атрибутов, получения списка клиентов.

В Гостинице, помимо проживания, предоставляется ряд дополнительных услуг. Все они занесены в класс «Услуга» (рис. 16). Класс содержит атрибуты: название, стоимость.

Услуга	
-Название: строка (100)	
-Стоимость: денежный	
+Создать (in Название:строка, in Стоимость:денежный) : Услуга <class> +Удалить () +ПолучитьНазвание () : строка +ПолучитьСтоимость () : денежный +УстановитьНазвание (in Название:строка) +УстановитьСтоимость (in Стоимость:денежный) +ПолучитьУслуги () : list (ref (Услуга))	

Рис. 16. Класс «Услуга»

Класс «Услуга» содержит методы создания, удаления, получения атрибутов, установки атрибутов, получения списка услуг.

Класс «Оказанная услуга» (рис. 17) содержит атрибуты: дата, количество, статус оплаты, ссылку на карту визита, услугу и ссылку на счет за услугу.

Класс «Оказанная услуга» содержит методы создания, удаления экземпляров, получения информации об оказанной услуге, установки атрибутов, получения списка оказанных услуг, методы для работы со счетами (добавление и удаление счета), метод получения всех оказанных услуг, метод поиска неоплаченных услуг по карте визита.

Класс «Счет» содержит методы создания, удаления экземпляров, получения информации по счетам за указанный период времени, установки атрибутов, ссылающихся на администратора и карту визита, получения неоплаченных счетов.

Класс «Счет за проживание» (рис. 19) является потомком класса «Счет», содержит все атрибуты класса «Счет», а также атрибут статус, указывающий включать ли в счет за проживание плату за бронирование.

Счет за проживание
-СтатусБронирования: строка (20)
+Создать(in Счет:ref (Счет), in Визит:ref (Карта визита)): ref (Счет за проживание)
+Удалить()
+УстановитьСтатус(Статус:строка)
+ПолучитьСтатус(): строка
+ПолучитьСчета(): list (ref (Счет за проживание))

Рис. 19. Класс «Счет за проживание»

Класс «Счет за проживание» содержит методы создания, удаления экземпляров, установки и получения статуса бронирования, методы получения счетов за проживание.

Класс «Счет за услугу» (рис. 20) является потомком класса «Счет», содержит все атрибуты класса «Счет», а также атрибуты, ссылающиеся на оказанные услуги.

Счет за услугу
-Услуги [1..*]: ref (Оказанная услуга)
+Создать(in Счет:ref (Счет), in Услуга:ref (Оказанная услуга)): Счет за услугу <class>
+Удалить()
+ДобавитьУслугу(in Услуга:ref (Оказанная Услуга))
+УдалитьУслугу(in Услуга:ref (Оказанная услуга))
+ПолучитьСчета(): list (ref (Счет за услугу))
+НайтиСчет(Счет:ref (Счет)): ref (Счет за услугу)

Рис. 20. Класс «Счет за услугу»

Класс «Счет за услугу» содержит методы создания, удаления экземпляров, добавления и удаления оказанных услуги, методы поиска счетов за услуги.

Класс «Администратор» (рис. 21) содержит личную информацию об администраторе: ФИО, логин и пароль для входа в систему, а также информацию о визитах, счетах и отчетах, которые созданы администратором.

Администратор
-ФИО: строка (100) -Логин: строка (20) -Пароль: строка (20) -Визиты [0..*]: ref (Карта визита) -Счета [0..*]: ref (Счет) -Отчеты [0..*]: ref (Отчет)
+Создать(in ФИО:строка): Администратор <class> +Удалить() +ПолучитьВизиты(): list (ref (Карта Визита)) +ПолучитьСчета(): list (ref (Счет)) +ПолучитьОтчеты(): list (ref (Отчет)) +ПолучитьФИО(): строка +ДобавитьВизит(in ref (Карта визита)) +УдалитьВизит(in Визит:ref (Карта визита)) +ДобавитьСчет(in Счет:ref (Счет)) +УдалитьСчет(in Счет:ref (Счет)) +ДобавитьОтчет(Отчет:ref (Отчет)) +СоздатьОтчет(in Дата:дата) +УстановитьФИО(in ФИО:строка) +УстановитьЛогин(in Логин:строка) +УстановитьПароль(in Пароль:строка) +Проверить(in Логин,in Пароль): логический +ПолучитьАдминистраторов(): list (ref Администратор)

Рис. 21. Класс «Администратор»

Класс «Администратор» содержит методы создания, удаления экземпляров класса, методы установки и получения атрибутов личной информации об администраторе, логинов и паролей, методы для работы с визитами, отчетами и счетами (их добавление, удаление), метод для получения списка администраторов, метод проверки правильности логина и пароля.

Класс «Отчет» (рис. 22) объединяет информацию о картах визита и счетах, выписанных администраторами. Отчет создается за определенную дату администратором, поэтому в классе есть атрибут дата и указание на администратора.

Карты визита				
Номер КВ	Комната	Фамилия Имя Отчество	Приезд	Отъезд
1	403	Арбузов Артем Игоревич	04.07.2012	08.07.2012
3	306	Кузова Виктория Адамовна	05.07.2012	10.07.2012
2	404	Белобоков Эдуард Викторович	05.07.2012	11.07.2012

Состояние: Активна

Поиск Карт визита

Фамилия

Номер КВ

Комната

Дата приезда

Номер КВ

Найти

Очистить

Все КВ

Рис. 23. Экранная форма «Карты визита»

Форма содержит таблицу с информацией о картах визита. Таблица включает столбцы: номер карты визита, номер комнаты, ФИО клиента, даты приезда и даты отъезда. Нижняя часть экрана содержит область поиска, которая включает параметры поиска: состояние карты визита, фамилия клиента, номер карты визита, номер комнаты, дата приезда, номер карты брони. На форме есть кнопки для выполнения поиска, очистки параметров поиска, получения всех карт визита.

На рис. 24 приведен класс «Форма Карты визита», соответствующая экранной форме «Карты визита».

Класс «Форма Карты визита» описывает структуру экранной формы. Атрибуты класса описывают все элементы интерфейса, представленные на форме. Класс содержит методы, которые представляют собой возможности формы, т. е. отражают все, что форма может делать и на что реагирует.

Форма Карты визита
+СписокКартВизита[0..*]: ref (Карта визита) +Карты визита: таблица +Номер KB метка: заголовок столбца +Номер KB поле: поле таблицы +Комната метка: заголовок столбца +Комната поле: поле таблицы +ФИО метка: заголовок столбца +ФИО поле: поле таблицы +Приезд метка: заголовок столбца +Приезд поле: поле таблицы +Отъезд метка: заголовок столбца +Отъезд поле: поле таблицы +Состояние метка: метка +Состояние список: выпадающий список +Поиск метка: метка +Фамилия метка: метка +Фамилия поле: поле ввода +Номер KB метка: метка +Номер KB поле: поле ввода +Комната метка: метка +Комната поле: поле ввода +Дата приезда метка: метка +Дата приезда поле: поле ввода +Номер KB метка: метка +Номер KB поле: поле ввода +Найти: кнопка +Очистить: кнопка +ВсеКартыВизита: кнопка
+Открыть () +Создать () +Удалить () +ПолучитьКартыВизитаПоСостоянию () +Поиск карт визита () +Очистить форму поиска() +Показать все карты визита() +Открыть карту визита () +Закрыть ()

Рис. 24. Класс «Форма Карты визита»

В классе есть метод открытия формы (заполняет все поля формы), метод получения списка карт визита с выбранным состоянием (срабатывает на выбор элемента выпадающего списка), методы, реагирующие на нажатие кнопок, метод открытия формы для просмотра карты визита (срабатывает на нажатие номера выбранной карты визита), метод закрытия формы.

На рис. 25 приведена экранная форма, содержащая подробную информацию об одной карте визита, которая открывается из формы «Карты визита» при выборе соответствующей карты визита.

Номер Карты визита 1		Номер Карты брони	
Фамилия	Арбузов	Паспорт	Серия и номер
Имя	Артём	Кем выдан	УВД г. Москва
Отчество	Игоревич	Когда выдан	12.02.2002
Контактная информация: +79116706747			
Дата приезда	04.07.2012	Номер комнаты	301-а
Дата отъезда	06.07.2012	Категория	2А
Количество суток	5	Стоимость	1 500,00р.
		Сумма	7 500,00р.
		Номер счета	1 <input checked="" type="checkbox"/> Оплачено

Услуги						Счета		
Дата	Название	Цена	Количество	Сумма	Номер счета	Номер счета	Дата	Сумма
06.07.2012	Ужин	400,00р.	1	400,00р.	2	1	06.07.2012	7 500,00р.
07.07.2012	Ужин	400,00р.	2	800,00р.	2	2	08.07.2012	1 200,00р.
07.07.2012	Обед	300,00р.	1	300,00р.				

Рис. 25. Экранная форма «Карта визита»

Форма содержит информацию собственно о визите: номер карты визита, даты заезда и отъезда, номер комнаты. По комнате определяется ее категория и стоимость за сутки. Если визиту предшествовало бронирование, то отображается номер карты брони. Кроме того, форма включает информацию о клиенте: фамилия, имя, отчество, паспортные данные, контактная информация. Также на форме отображаются данные о дополнительных услугах и о счетах, выписанных клиенту.

Услуги представлены в виде таблицы, содержащей поля: дата, название, цена, количество, сумма, номер счета. Счета представлены в виде таблицы, содержащей номер счета, дату и сумму. Количество суток проживания рассчитывается автоматически. Итоговая сумма за проживание рассчитывается автоматически. Номер карты визита подкрашивается цветом в зависимости от статуса карты.

На рис. 26 приведен класс «Форма Карты визита», соответствующая экранной форме «Карта визита».

Форма Карта визита
+Карта визита: ref (Карта визита) +Номер карты визита поле: поле ввода +Номер карты визита метка: метка +Номер карты брони метка: метка +Номер карты брони: поле ввода +фамилия метка: метка +фамилия: поле ввода +Имя метка: метка +Имя: поле ввода +Отчество метка: метка +Отчество: поле ввода +Контактная информация метка: метка +Контактная информация: поле ввода +Паспорт серия метка: метка +Паспорт серия и номер: поле ввода +Паспорт кем выдан метка: метка +Паспорт кем выдан: поле ввода +Паспорт когда выдан метка: метка +Паспорт когда выдан: поле ввода +Дата приезда метка: метка +Дата приезда: календарь +Дата отъезда метка: метка +Дата отъезда: календарь +Количество суток метка: метка +Количество суток: поле ввода +Номер комнаты метка +Номер комнаты: выпадающий список +Категория метка: метка +Категория: выпадающий список +Стоимость метка: метка +Стоимость: поле ввода +Сумма метка: метка +Сумма: поле ввода +Оплачено: флажок +Номер счета метка: метка +Номер счета: Поле ввода +Услуги: таблица +Дата поле: поле таблицы +Название поле: поле таблицы +Цена поле: поле таблицы +Количество поле: поле таблицы +Счет поле: поле таблицы +Счета: таблица +Сумма поле: поле таблицы +Номер счета поле: поле таблицы +Дата поле: поле таблицы +Сумма поле: поле таблицы
+Открыть (in тип=просмотр) +Создать () +Удалить () +Получить комнаты () +Получить категории() +Добавить услугу() +Выделить услугу() +Удалить услугу () +Добавить счет () +Выделить счет () +Удалить счет () +Рассчитать сутки() +Рассчитать сумму() +Закрыть ()

Рис. 26. Класс «Форма Карта визита»

Класс «Форма Карта визита» описывает структуру экранной формы. Атрибуты класса описывают все элементы интерфейса, используемые на форме: метки, таблицы, поля ввода, выпадающие списки, флажки. Класс содержит важный атрибут, ссылающийся на класс с данными о карте визита: Карта визита: `ref` (Карта визита). Эта ссылка обеспечивает связь между классом, описывающим форму, и классом, содержащим данные для этой формы.

Класс содержит методы, которые представляют собой возможности формы, т. е. отражают все, что форма может делать и на что реагирует. В классе есть метод открытия формы в режиме просмотра или редактирования (заполняет все поля формы); метод подцветки номера карты визита в зависимости от состояния карты; метод получения комнат и категорий комнат (срабатывают при открытии выпадающих списков); метод закрытия формы.

Кроме того, в классе есть методы для работы с таблицами услуг и счетов, связанных с картой визита: добавление счета происходит при нажатии зарезервированной клавиши, есть метод для выделения строки таблицы счетов. Метод удаления выделенной строки таблицы счетов запускается при нажатии зарезервированной клавиши, аналогично работают методы по добавлению, выделению, удалению услуг. Кроме того, класс содержит методы автоматического расчета суток проживания и итоговой суммы за проживание.

Подобным образом необходимо описать все классы, соответствующие экранным формам. Далее эти классы добавляются на общую диаграмму классов. Классы, описывающие формы, связывают с классами данных отношением зависимости, так как изменение структуры классов с данными влечет за собой изменение структуры классов экранных форм для отображения этих данных. Обычно классы экранных форм зависят от тех классов данных, информация из которых на них отображается.

На рис. 27 приведена часть диаграммы классов, показывающая установление зависимостей между классами экранных форм и классами с данными. Класс «Форма карта визита» зависит от класса «Карта визита», а также от классов

«Комната», «Категория», «Клиент», «Оказанная услуга», «Счет за проживание», «Счет за услугу».

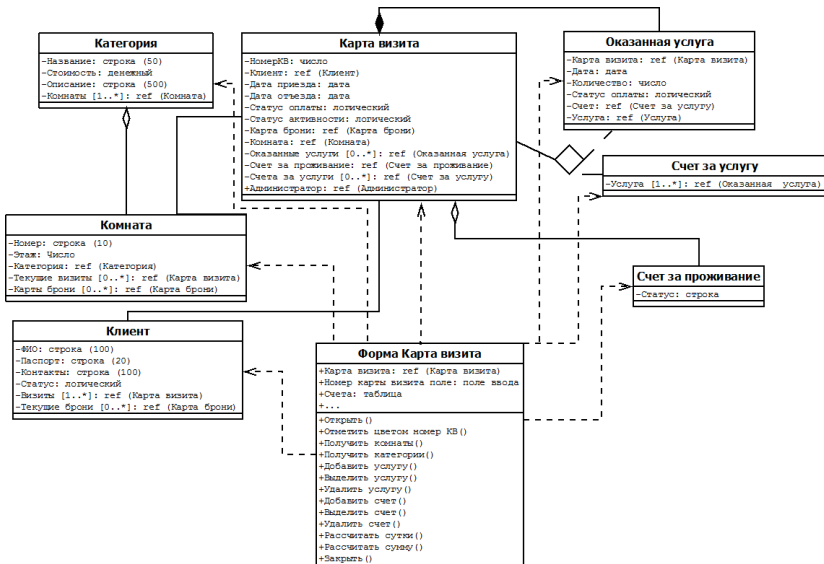


Рис. 27. Фрагмент диаграммы классов, описывающий зависимость класса «Форма Карта визита» от классов с данными

Кроме того, на диаграмме классов возможно отобразить переходы между экранными формами. Для этого классы экранных форм соединяют между собой направленной ассоциацией, где направление ассоциации соответствует направлению перехода между экранными формами.

На рис. 28 переход от формы с информацией обо всех картах визита к форме с единственной картой визита показан направленной ассоциацией от класса «Форма Карты Визита» к классу «Форма карта визита».

Общая диаграмма классов содержит в себе все классы с данными из предметной области, все классы, соответствующие экранным формам и другие необходимые классы.

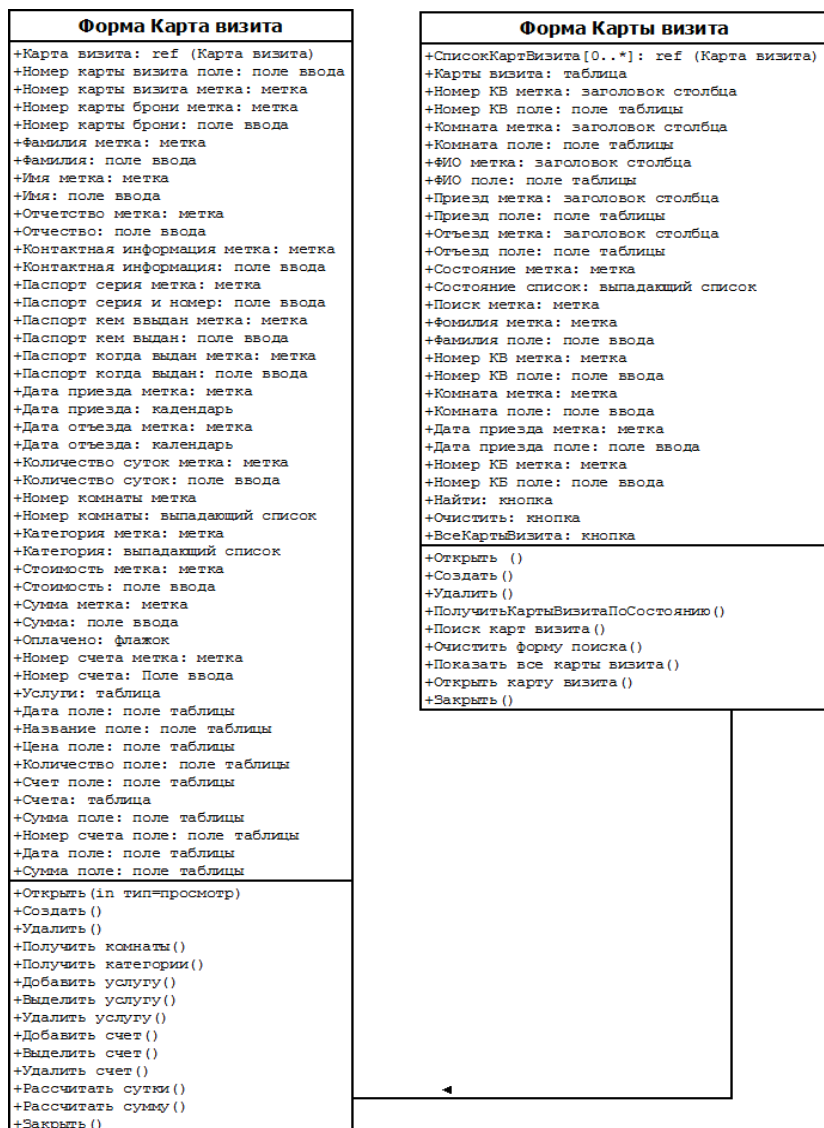


Рис. 28. Фрагмент диаграммы классов, описывающий направленную ассоциацию между классами

ДИАГРАММА СОСТОЯНИЙ

Диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее — одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта. Диаграмма описывает возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Для диаграммы состояний определены следующие ограничения.

- Переход объекта из состояния в состояние происходит мгновенно.
- Автомат не запоминает историю перемещения из состояния в состояние.
- В каждый момент времени автомат может находиться в одном и только в одном из своих состояний.
- При этом автомат может находиться в отдельном состоянии как угодно долго, если не происходит никаких событий.
- Время на диаграмме состояний присутствует в неявном виде, хотя для отдельных событий может быть указан интервал времени и в явном виде.
- Количество состояний автомата должно быть обязательно конечным, и все они должны быть специфицированы явным образом.
- Граф автомата не должен содержать изолированных состояний и переходов. Это условие означает, что для каждого из состояний, кроме начального, должно быть определено предшествующее состояние. Каждый переход должен обязательно соединять два состояния автомата. Допускается переход из состояния в себя.
- Автомат не должен содержать конфликтующих переходов, т. е. таких переходов из одного и того же состояния, когда объект одновременно может перейти в два и более последующих состояний (кроме случая параллельных подавтоматов).

Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта. Не каждый

атрибут класса может характеризовать его состояние. Как правило, имеют значение только такие свойства элементов системы, которые отражают динамический или функциональный аспект ее поведения.

Состояние на диаграмме изображается прямоугольником со скругленными вершинами. Этот прямоугольник может быть разделен на две секции горизонтальной линией. Если указана лишь одна секция, то в ней записывается только имя состояния. В противном случае в верхней секции записывается имя состояния, а в нижней секции – список некоторых внутренних действий или переходов в данном состоянии.

Секция **внутренних действий** содержит перечень внутренних действий или деятельностей, которые выполняются в процессе нахождения моделируемого элемента в данном состоянии. Различают действия следующих типов:

- **entry** – указывает на действие, которое выполняется в момент входа в данное состояние (входное действие);
- **exit** – указывает на действие, которое выполняется в момент выхода из данного состояния (выходное действие);
- **do** – указывает на действие, которое выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится действие;
- **include** – используется для обращения к подавтомату, при этом следующее действие содержит имя этого подавтомата.

Начальное состояние и **конечное состояние** представляют собой состояния, которые не содержат никаких внутренних действий. В начальном состоянии находится объект по умолчанию в начальный момент времени. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить стрелка, соответствующая переходу. В конечном состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени. Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность, в которую может только входить стрелка, соответствующая переходу.

Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности, получения объектом сообщения или приемом сигнала. На переходе указывается имя события. Кроме того, на переходе могут указываться действия, производимые объектом в ответ на внешние события при переходе из одного состояния в другое. На диаграмме состояний переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние. Каждый переход может быть помечен **событием** со сторожевым условием. Событием могут быть сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий.

Сторожевое условие записывается после события-триггера и представляет собой некоторое булевское выражение. Объект перейдет из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение «истина».

Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели. Составное состояние может содержать два или более параллельных подавтомата или несколько последовательных подсостояний.

Последовательные подсостояния используются для моделирования такого поведения объекта, во время которого в каждый момент времени объект может находиться в одном и только одном подсостоянии. Поведение объекта в этом случае представляет собой последовательную смену подсостояний, начиная от начального и заканчивая конечным подсостояниями. Хотя объект продолжает находиться в составном состоянии, введение в рассмотрение последовательных подсостояний позволяет учесть более тонкие логические аспекты его внутреннего поведения.

Параллельные подсостояния позволяют специфицировать два и более подавтомата, которые могут выполняться параллельно внутри составного события. Каждый из подавтоматов занимает некоторую область внутри составного состояния, которая отделяется от остальных горизонтальной пунктирной линией. Если на диаграмме состояний имеется составное состояние с вложенными

параллельными подсостояниями, то объект может одновременно находиться в каждом из этих подсостояний.

Диаграмма состояний не является обязательным представлением в модели и строится для тех классов, которые имеют нетривиальное поведение в течение своего жизненного цикла. Наличие у объекта класса больше двух состояний служит признаком необходимости построения диаграммы состояний.

На рис. 29 приведена диаграмма состояний для класса «Карта брони». Карта брони может находиться в четырех состояниях: активна, реализована, аннулирована, просрочена.

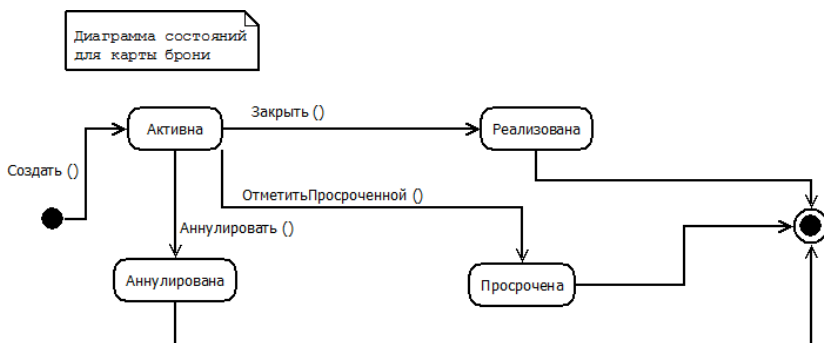


Рис. 29. Диаграмма состояний класса «Карта брони»

При создании карта брони становится активной (состояние «Активна»). При вызове метода «Аннулировать ()» карта брони переходит в состояние «Аннулирована». При вызове метода «Отметить Просроченной ()» карта брони переходит в состояние «Просрочена». При вызове метода «Закреть ()» карта брони переходит в состояние «Реализована». Переход в финальное состояние является безусловным.

ДИАГРАММА ДЕЯТЕЛЬНОСТИ

Следующим шагом проектирования является проектирование всех методов классов. Для этой цели используются **диаграммы деятельности**, которые внешне похожи на блок-схемы. Диаграммы деятельности представляют собой последовательность выполнения определенных действий или элементарных операций, которые в совокупности приводят к получению желаемого результата.

Диаграммы деятельности используют для моделирования методов классов, также они могут быть использованы для моделирования бизнес-процессов организации, проектирования сценариев использования системы.

Деятельность (activity) диаграммы деятельности представляет собой некоторую совокупность отдельных действий. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой. Результат может привести к изменению состояния системы или возвращению некоторого значения.

Графически действие изображается фигурой, напоминающей прямоугольник, боковые стороны которого заменены выпуклыми дугами. Внутри этой фигуры записывается выражение действия, которое должно быть уникальным в пределах одной диаграммы деятельности. Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования.

Каждая диаграмма деятельности должна иметь единственное **начальное** и единственное **конечное состояния**. Они имеют такие же обозначения, как и на диаграмме состояний. Диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз. В этом случае начальное состояние будет изображаться в верхней части диаграммы, а конечное – в нижней части.

Переход имеет такое же значение, как и на диаграмме состояний. При построении диаграммы деятельности используются только переходы, которые срабатывают сразу после завершения деятельности или выполнения соответствующего действия. На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действия выходит единственный переход, то он может быть никак не помечен. Если же таких переходов несколько, то сработать может только один из них. Именно в этом случае для каждого из таких переходов должно быть явно записано **сторожевое условие** в прямых скобках. При этом для всех выходящих из некоторого состояния переходов должно выполняться требование истинности только одного из них. Подобный случай встречается тогда, когда последовательно выполняемая деятельность должна разделиться на альтернативные ветви в зависимости от значения некоторого промежуточного результата. Такая ситуация получила название ветвления.

Графически **ветвление** на диаграмме деятельности обозначается небольшим ромбом с одной входящей стрелкой и несколькими выходящими стрелками. Выходящих стрелок может быть две или более, но для каждой из них явно указывается соответствующее сторожевое условие в форме булевского выражения. Для объединения альтернативных ветвей используется элемент **соединение**, который представляет собой ромб с несколькими входящими стрелками и одной выходящей стрелкой.

В отличие от блок-схем, в диаграмме деятельности есть специальные символы **разделения** и **слияния** для представления параллельных вычислений. Разделение и слияние изображается отрезком горизонтальной линии, толщина которой несколько больше основных сплошных линий диаграммы деятельности. При этом разделение имеет один входящий переход и несколько выходящих. Слияние, наоборот, имеет несколько входящих переходов и один выходящий.

При моделировании бизнес-процессов в виде диаграммы деятельности удобно использовать разделение действий алгоритма на **дорожки**, каждая из которых выполняется отдельным подразделением. При этом все действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии и образуют дорожку, а группа действий между этими линиями выполняется отдельным подразделением организации. Названия подразделений явно указываются в верхней части дорожки. Пересекать линию дорожки могут только переходы, которые

в этом случае обозначают выход или вход потока управления в соответствующее подразделение.

На рис. 30 приведена диаграмма деятельности, которая описывает метод «Рассчитать итог» класса «Карта визита».

Метод «Рассчитать итог» класса «Карта визита» предназначен для расчета итоговой суммы всех неоплаченных счетов по визиту клиента. Итоговая сумма включает сумму за проживание и все оказанные дополнительные услуги, если соответствующие счета за проживание и за услуги еще не были оплачены заранее.

Для расчета итоговой суммы необходимо вначале проверить все счета, выписанные клиенту. Если счет за проживание не был оплачен, то к итоговой сумме добавляется сумма за проживание. Если клиент пользовался услугами и счета за услуги не были оплачены, то проверяется каждый счет по услуге и к итоговой сумме добавляется стоимость услуг каждого вида, равная стоимости одной услуги, умноженной на количество услуг данного вида.

Диаграмма содержит все основные элементы: начальное и конечное состояние, действия, переходы, ветвления, соединения, разделение и слияние.

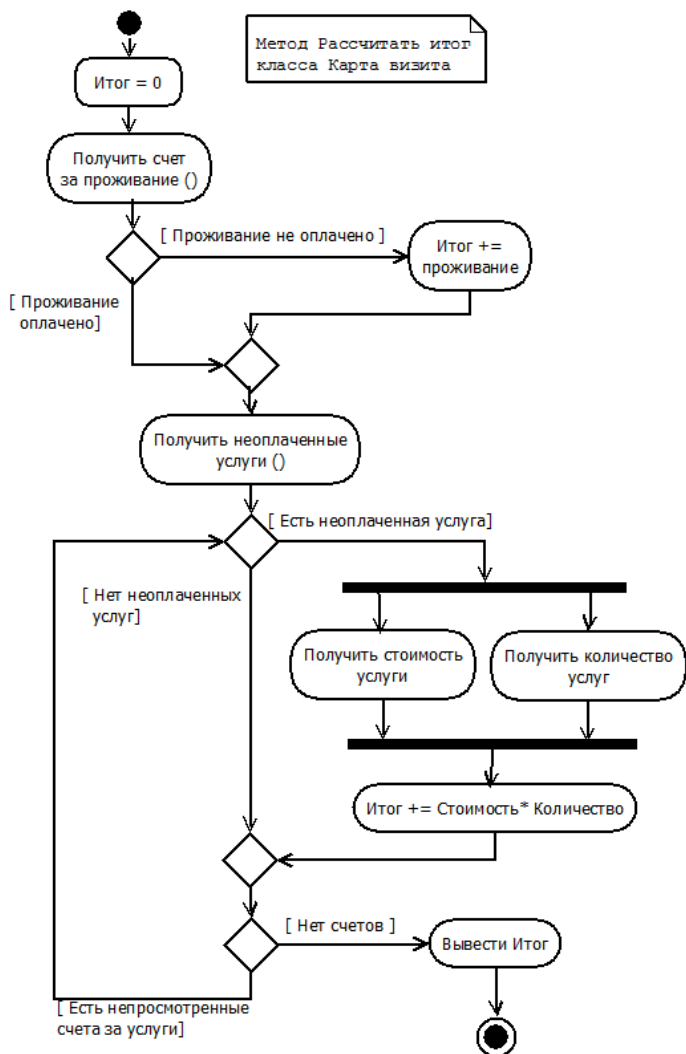


Рис. 30. Диаграмма деятельности для метода «Рассчитать итог» класса Карта визита

На рис. 31 приведена диаграмма деятельности, которая описывает бизнес-процесс поселения клиента в гостиницу.

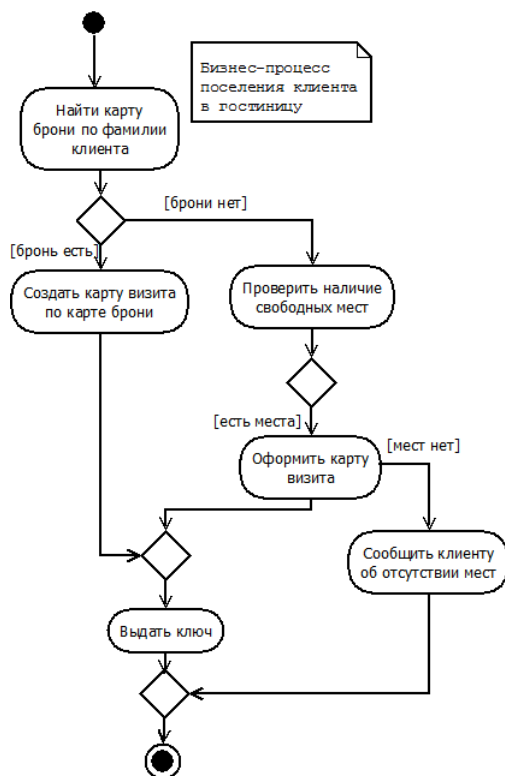


Рис. 31. Диаграмма деятельности «Поселение клиента в гостиницу»

Администратор при поселении клиента в гостиницу сначала проверяет, была ли оформлена на клиента карта брони, т. е. зарезервировано ли ему место в гостинице. Поиск карты брони администратор осуществляет по фамилии клиента. Если карта брони существует, то на ее основе администратор создает карту визита, вносит недостающие данные и выдает клиенту ключ от номера. Если номер не был заранее забронирован, то необходимо проверить наличие свободных мест в гостинице на интересующие даты. Если места есть, то администратор оформляет карту визита и выдает ключ клиенту. Если же мест в гостинице нет, то администратор сообщает об этом клиенту.

ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ И ДИАГРАММА КООПЕРАЦИИ

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется **диаграмма последовательности**.

Кроме того, можно рассматривать структурные особенности взаимодействия объектов. Для представления структурных особенностей передачи и приема сообщений между объектами используется **диаграмма кооперации**.

На диаграмме последовательности изображаются исключительно те **объекты**, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет два измерения. Одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни. Внутри прямоугольника записываются подчеркнутые имя объекта и имя класса, разделенные двоеточием.

Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия. Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности образуют некоторый порядок, определяемый началом активности этих объектов при взаимодействии друг с другом.

Второе измерение диаграммы последовательности – вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством **сообщений**, которые посылаются одними объектами

другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения, и также образуют порядок по времени своего возникновения.

Линия жизни объекта изображается пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях.

Отдельные объекты могут быть **уничтожены**, чтобы освободить занимаемые ими ресурсы. Для таких объектов линия жизни обрывается в момент его уничтожения. Для обозначения момента уничтожения объекта в языке UML используется специальный символ в форме латинской буквы «X». Ниже этого символа пунктирная линия не изображается, поскольку соответствующего объекта в системе уже нет, и этот объект должен быть исключен из всех последующих взаимодействий.

В процессе функционирования одни объекты могут находиться в активном состоянии, непосредственно выполняя определенные действия или в состоянии пассивного ожидания сообщений от других объектов. Чтобы явно выделить подобную активность объектов, в языке UML применяется специальное понятие, получившее название **фокуса управления**.

Фокус управления изображается в форме вытянутого узкого прямоугольника, верхняя сторона которого обозначает начало получения фокуса управления объектом (начало активности), а ее нижняя сторона – окончание фокуса управления (окончание активности).

В отдельных случаях инициатором взаимодействия в системе может быть **актер**. В этом случае актер изображается на диаграмме последовательности самым первым объектом слева со своим фокусом управления.

Каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой. **Сообщение** представляет собой законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение

определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

Сообщения не только передают некоторую информацию, но и требуют или предполагают от принимающего объекта выполнения ожидаемых действий. Сообщения могут инициировать выполнение операций объектом соответствующего класса, а параметры этих операций передаются вместе с сообщением. На диаграмме последовательности все сообщения упорядочены по времени своего возникновения в моделируемой системе.

Каждое сообщение имеет направление от объекта, который инициирует и отправляет сообщение, к объекту, который его получает. Иногда отправителя сообщения называют клиентом, а получателя – сервером. При этом сообщение от клиента имеет форму запроса некоторого сервиса, реакция сервера на запрос после получения сообщения может быть связана с выполнением определенных действий или передачи клиенту необходимой информации тоже в форме сообщения.

На диаграммах последовательности и кооперации могут встречаться несколько разновидностей сообщений, каждое из которых имеет свое графическое изображение.

Сплошной стрелкой с закрашенным наконечником обозначается **вызов**. Вызов является наиболее распространенным сообщением и используется для вызова процедур, выполнения операций или обозначения отдельных вложенных потоков управления.

Сплошной стрелкой с не закрашенным наконечником обозначается **асинхронное сообщение**, которое передается в произвольный момент. Пунктирной стрелкой обозначается сообщение **возврата** из вызова процедуры.

Для изображения ветвления рисуются две или более стрелки, выходящие из одной точки фокуса управления объекта. При этом соответствующие условия должны быть явно указаны рядом с каждой из стрелок в форме сторожевого условия.

В языке UML предусмотрены некоторые стандартные действия, выполняемые в ответ на получение соответствующего сообщения. Тогда они указыва-

ются рядом с сообщением, к которому они относятся. Существуют следующие стереотипы:

- «call» (вызвать) – сообщение, требующее вызова операции или процедуры принимающего объекта;
- «return» (возвратить) – сообщение, возвращающее значение выполненной операции или процедуры вызвавшему ее объекту;
- «create» (создать) – сообщение, требующее создания другого объекта для выполнения определенных действий;
- «destroy» (уничтожить) – сообщение с явным требованием уничтожить соответствующий объект;
- «send» (послать) – обозначает посылку другому объекту некоторого сигнала, который асинхронно инициируется одним объектом и принимается (перехватывается) другим. Отличие сигнала от сообщения заключается в том, что сигнал должен быть явно описан в том классе, объект которого инициирует его передачу.

Кроме стереотипов, сообщения могут иметь собственное обозначение операции, вызов которой они инициируют у принимающего объекта. В этом случае рядом со стрелкой записывается имя операции с круглыми скобками, в которых могут указываться параметры или аргументы соответствующей операции.

Диаграмма последовательности обычно строится для каждого сценария диаграммы сценариев, т. е. описывает взаимодействия объектов с помощью сообщений, которыми они обмениваются в рамках одного прецедента.

На рис. 32 изображена диаграмма последовательности для сценария «Распечатать счет».

Дежурный администратор выбирает пункт меню «Визиты» на главной форме. Происходит открытие формы «Карты визита». По запросу «Получить визиты» класса Визит происходит передача данных для формы, содержащей список текущих карт визита.

Далее администратор выбирает нужную карту визита из списка и нажимает на нее. При этом происходит переход на форму с выбранной картой визита.

Все поля формы заполняются по запросу получения информации по визиту класса «Карта визита».

Затем администратор выбирает в меню пункт «Выписать счет за проживание». После этого происходит открытие формы просмотра счета за проживание. Все поля формы заполняются по запросу получения информации класса «Счет за проживание».

После этого администратор просматривает счет за проживание в открывшемся окне и нажимает кнопку «Печать» на форме, по которой запускается метод печати счета за проживание класса «Счет за проживание».

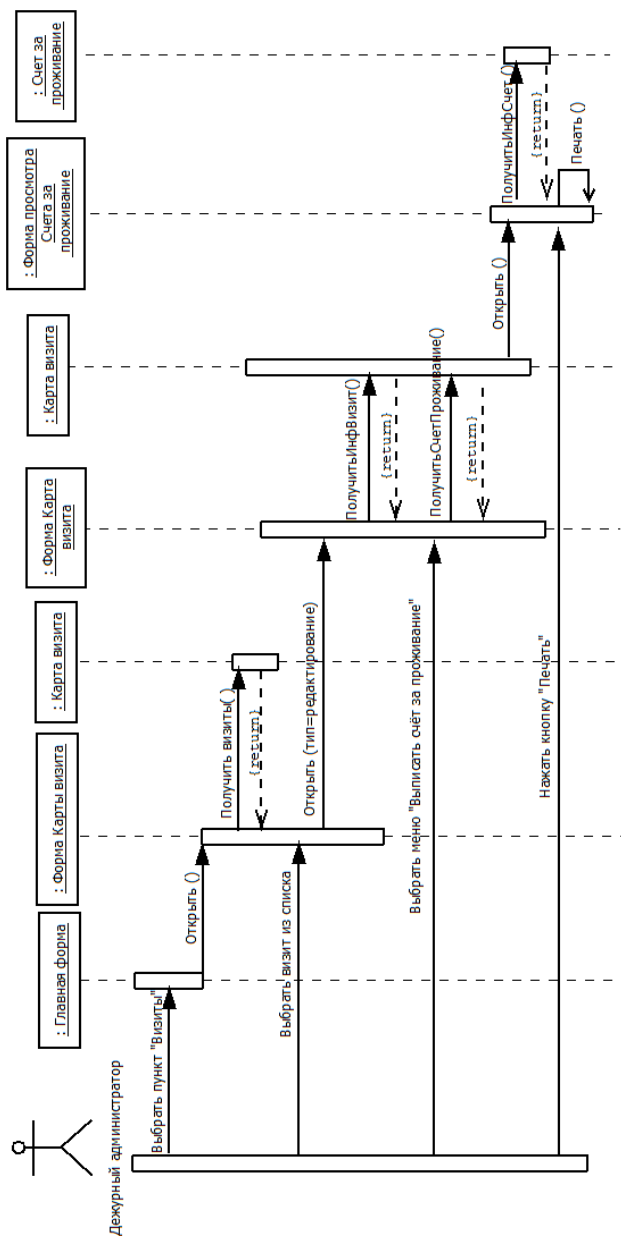


Рис. 32. Диаграмма последовательности «Распечатать счет за проживание»

На рис. 33 приведена диаграмма последовательности для процедуры авторизации дежурного администратора.

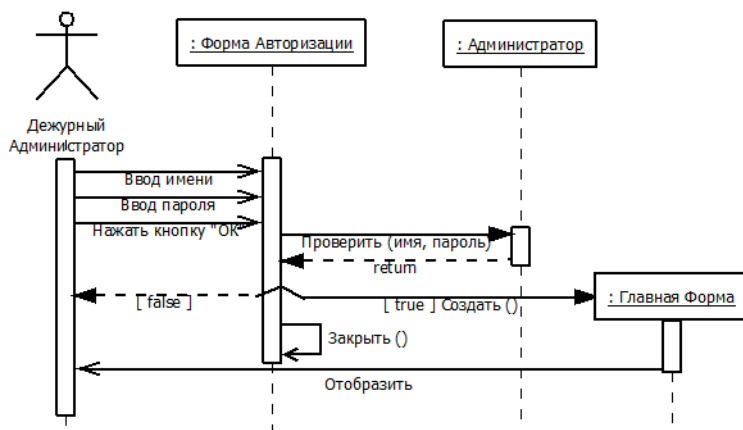


Рис. 33. Диаграмма последовательности «Авторизация администратора»

Дежурный администратор вводит имя и пароль на форме «Авторизации», нажимает кнопку «ОК». При нажатии кнопки форма вызывает метод «Проверить ()» с параметрами «имя» и «пароль» класса «Администратор».

Метод «Проверить ()» выполняет проверку имени и пароля, возвращает логическое значение. Если метод возвращает значение «Истина» (авторизация успешна), происходит переход на «Главную форму». При этом «Форма авторизации» закрывается. Если метод возвращает значение «Ложь», то переход на главную форму не происходит.

Диаграмма кооперации описывает динамическое поведение на базе обмена сообщениями. На диаграмме кооперации в виде прямоугольников изображаются участвующие во взаимодействии объекты, указываются ассоциации между объектами в виде различных соединительных линий. Дополнительно могут быть изображены динамические связи – потоки сообщений, которые представляются в виде соединительных линий между объектами, над которыми

располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

На рис. 34 приведена диаграмма кооперации для процедуры авторизации администратора.

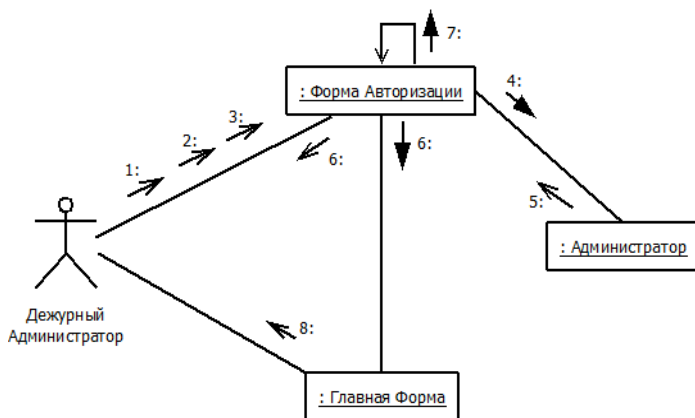


Рис. 34. Диаграмма кооперации «Авторизация администратора»

Передаваемые сообщения помечены номерами. Сообщение 1: ввод имени; 2: ввод пароля; 3: нажать кнопку «ОК»; 4: Проверить (имя, пароль); 5: возврат; 6: если «истина», то «Создать»; 7: Закреть (); 8: Отобразить. В отличие от диаграммы кооперации, указаны ассоциации между объектами.

ДИАГРАММА КОМПОНЕНТОВ

В языке UML для физического представления моделей систем используются так называемые диаграммы реализации: диаграммы компонентов и диаграммы топологии.

Диаграмма компонентов описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ.

Компонентом может быть любой крупно модульный объект: общая подсистема, бинарный исполняемый файл, готовая к использованию система, объектно-ориентированное приложение. Диаграмма компонентов похожа на диаграмму классов.

Поскольку компонент как элемент физической реализации модели представляет отдельный модуль кода, иногда его комментируют с указанием дополнительных графических символов, иллюстрирующих конкретные особенности его реализации.

Каждый компонент внутри диаграммы будет документирован с помощью более детальной диаграммой компонент, диаграммой сценариев или диаграммой классов.

На диаграмме компонентов могут быть представлены отношения зависимости между компонентами и реализованными в них классами. Эта информация имеет важное значение для обеспечения согласования логического и физического представлений модели системы.

Диаграмма компонентов разрабатывается для следующих целей:

- визуализации общей структуры исходного кода программной системы;
- спецификации исполнимого варианта программной системы;

- обеспечения многократного использования отдельных фрагментов программного кода;
- представления концептуальной и физической схем баз данных.

До начала разработки диаграммы компонентов 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. После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных.

На рис. 35 приведена общая диаграмма компонентов для разрабатываемой информационной системы, автоматизирующей деятельность дежурного администратора.

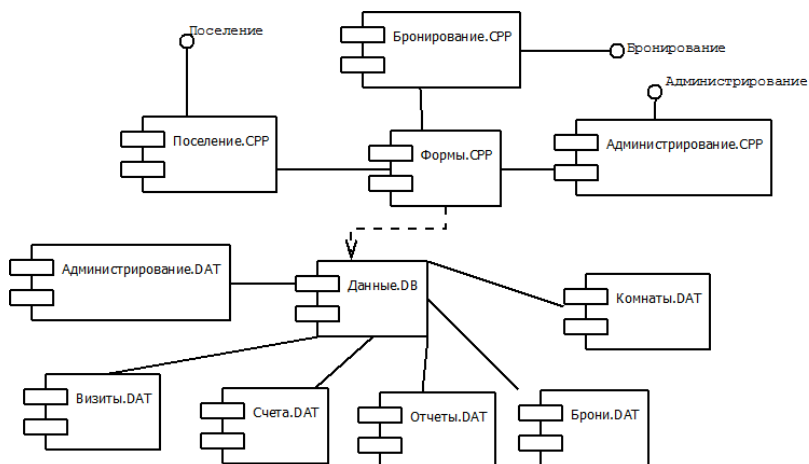


Рис. 35. Общая диаграмма компонентов

На диаграмме компонентов выделены два основных компонента, один из которых содержит базу данных, другой содержит компонент с экранными формами.

Компонент «Формы» объединяет три модуля приложения: компонент, отвечающий за логику приложения при поселении клиента в гостиницы (связан с интерфейсом «Поселение»); компонент, отвечающий за бронирование (связан с интерфейсом «Бронирование»); компонент администрирования (связан с интерфейсом «Администрирование»).

Компонент «Данные» объединяет блоки данных: компонент для хранения данных администрирования, информации о визитах, данных о счетах, об отчетах, о бронированиях, о комнатах.

На рис. 36 приведена часть диаграммы компонентов, на которой изображено распределение классов, описывающих данные по компонентам базы данных.

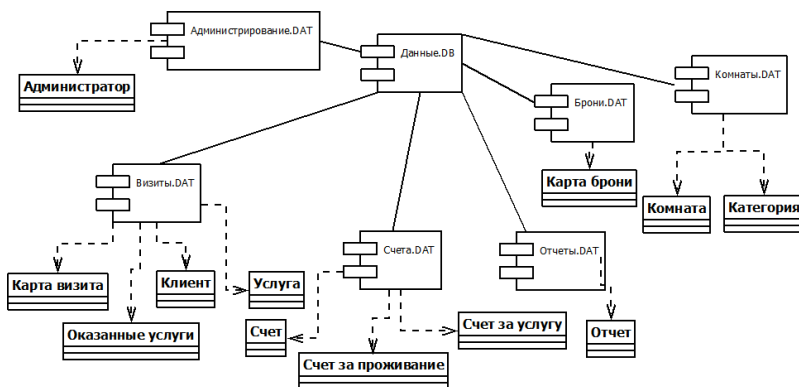


Рис. 36. Зависимости компонентов с данными от классов

Компонент «Администрирование» содержит класс «Администратор».

Компонент «Визиты» содержит классы «Карта визита», «Оказанные услуги», «Клиент», «Услуга».

Компонент «Счета» содержит классы «Счет», «Счет за проживание», «Счет за услугу».

Компонент «Отчеты» содержит класс «Отчет».

Компонент «Брони» содержит класс «Карта брони».

Компонент «Комната» содержит классы «Комната», «Категория».

На рис. 37 приведена часть диаграммы компонентов, на которой изображено распределение классов, описывающих экранные формы, по компонентам форм.

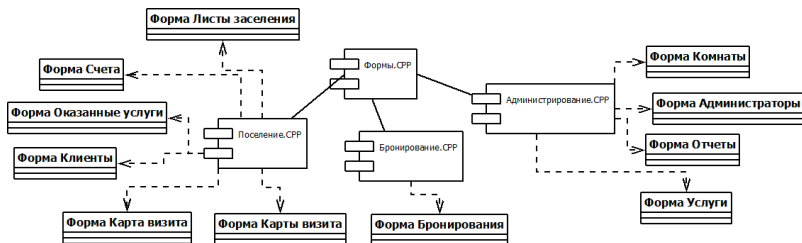


Рис. 37. Зависимости компонентов с формами от классов

Компонент «Поселение» содержит классы «Форма Листы заселения», «Форма Счета», «Форма Оказанные услуги», «Форма Клиенты», «Форма Карта визита», «Форма Карты визита».

Компонент «Бронирование» содержит класс «Форма бронирования».

Компонент «Администрирование» содержит классы «Форма Комнаты», «Форма Администраторы», «Форма Отчеты», «Форма Услуги».

ДИАГРАММА ТОПОЛОГИИ

Диаграммы топологии (развертывания) показывают конфигурацию выполняющих блоков или частей системы, включая аппаратное и программное обеспечение, которое на них выполняется. Она применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма топологии показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Перечислим цели разработки диаграммы топологии:

- определить распределение компонентов системы по ее физическим узлам;
- показать физические связи между всеми узлами реализации системы на этапе ее исполнения;
- выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Узел представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом. Узлом может быть компьютер, сервер, датчик, принтер, модем, цифровая камера, сканер и т. д. Графически на диаграмме развертывания узел изображается в форме трехмерного куба.

Диаграмма топологии содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма топологии является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации.

На рис. 38 приведена диаграмма топологии для разрабатываемой информационной системы, автоматизирующей деятельность дежурного администратора.

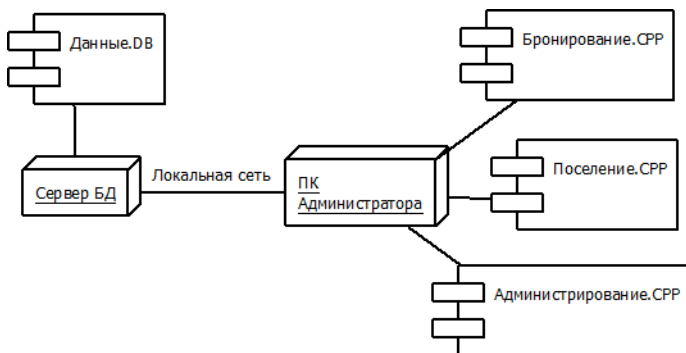


Рис. 38. Диаграмма топологии

Диаграмма топологии содержит узел «Сервер БД», где расположена база данных, узел «ПК Администратора», который содержит компоненты приложения. Узлы соединены локальной сетью.

ЗАКЛЮЧЕНИЕ

В учебном пособии представлен пример проектирования информационной системы на основе объектно-ориентированного подхода с построением объектной базы данных. Система полностью описана диаграммами языка UML.

Если все эти диаграммы реализовать с помощью соответствующего программного инструментария, например Rational Rose, то программный код на заданном языке программирования может быть сформирован автоматически, что значительно упрощает разработку информационной системы.

С помощью языка UML может быть также описана информационная система, включающая реляционную базу данных. Возможности языка никак не ограничивают применение любых средств разработки, наоборот, в большинстве случаев ускоряют ее и повышают эффективность работ.

В то же время объектно-ориентированный подход может быть применен совместно или в комбинации с другими подходами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Щеголева, Л. В.* Проектирование информационной системы : структурный подход : учеб. пособие / Л. В. Щеголева, А. Н. Кириленко. – Петрозаводск : Изд-во ПетрГУ, 2013. – 133 с. : ил.
2. *Буч, Г.* Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон. – Москва : ДМК, 2000. – 432 с. : ил.
3. *Буч, Г.* Язык UML: специальный справочник / Г. Буч, Д. Рамбо, А. Джекобсон. – Санкт-Петербург : Питер, 2001. – 656 с. : ил.
4. *Буч, Г.* Введение в UML от создателей языка / Г. Буч, Д. Рамбо, А. Якобсон. – Москва : ДМК Пресс, 2015. – 496 с. : ил.
5. *Вендров, А. М.* CASE-технологии : современные методы и средства проектирования информационных систем / А. М. Вендров. – Москва : Финансы и статистика, 1998. – 176 с. : ил.
6. *Скотт, К.* UML : основные концепции / К. Скотт. – Москва : Вильямс, 2002. – 144 с. : ил.
7. *Леоненков, А. В.* Самоучитель UML / А. В. Леоненков. – Санкт-Петербург : БХВ-Петербург, 2006. – 432 с. : ил.
8. *Орлов, С. А.* Технологии разработки программного обеспечения : учебник / С. А. Орлов. – Санкт-Петербург : Питер, 2002. – 464 с. : ил.
9. *Шмуллер Д.* Освой самостоятельно UML за 24 часа / Д. Шмуллер. – Москва : Вильямс, 2005. – 416 с. : ил.

СОДЕРЖАНИЕ

Введение	3
Описание предметной области «Гостиница»	4
Диаграмма сценариев	9
Диаграмма классов	15
Построение классов-данных	18
<i>Построение классов-данных для «Гостиницы»</i>	19
<i>Построение классов экранных форм</i>	40
Диаграмма состояния	48
Диаграмма деятельности.	52
Диаграмма последовательности и диаграмма кооперации	57
Диаграмма компонентов	65
Диаграмма топологии	69
Заключение.	71
Библиографический список.	72

Учебное электронное издание

ВОРОНОВА Анна Михайловна
ЩЕГОЛЕВА Людмила Владимировна

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ: ОБЪЕКТНЫЙ ПОДХОД

Учебное электронное пособие для студентов

Редактор *И. И. Куроптева*
Оригинал-макет,
компьютерная верстка,
и оформление обложки *Н. Н. Осипов*

Подписано к изготовлению 26.12.2016.

1 CD-R. Изд. № 413

Федеральное государственное бюджетное образовательное
учреждение высшего образования
ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

185910, г. Петрозаводск, пр. Ленина, 33

<https://petrsu.ru>
Тел. (8142) 71 10 01

Изготовлено в Издательстве ПетрГУ

185910, г. Петрозаводск, пр. Ленина, 33

<https://press.petrsu.ru/UNIPRESS/UNIPRESS.html>

Тел./факс (8142) 78 15 40
nvpahomova@yandex.ru

ISBN 978-5-8021-1864-1



9 785802 118641