СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc472670921)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc472670922)

[1.1 Описание и обоснование выбранного языка программирования 5](#_Toc472670923)

[2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 7](#_Toc472670924)

[3 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ 10](#_Toc472670925)

[3.1 Диаграмма вариантов использования 11](#_Toc472670926)

[3.2 Диаграмма классов 12](#_Toc472670927)

[3.3 Диаграмма деятельности 12](#_Toc472670928)

[3.4 Диаграмма коопераций 13](#_Toc472670929)

[3.5 Диаграмма последовательности 14](#_Toc472670930)

[3.6 Диаграмма развёртывания 14](#_Toc472670931)

[3.7 Диаграмма состояний 15](#_Toc472670932)

[3.8 Диаграмма компонентов 16](#_Toc472670933)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc472670934)

[Генерация кода 18](#_Toc472670934)

# ВВЕДЕНИЕ

Гостиничный бизнес – одна из основных компонент индустрии гостеприимства. Данная сфера обслуживания ориентирована на общение, работу с клиентами, на создание комфортных для них условий проживания. В этом случае уровень обслуживания прямо пропорционален классу гостиницы, а от этого зависит и конкурентоспособность вашего гостиничного комплекса.

Гостиничная цепь  — это определённое количество гостиниц, пользующихся одним и тем же названием и фирменным знаком, имеющих определенные общие потребительские особенности и проводящих совместную маркетинговую деятельность. Предприятие, управляющее гостиничной цепью, может быть её владельцем или совладельцем, управлять ею либо осуществлять надзор за её деятельностью на основе используемых соглашений.

В 1995 году мировая гостиничная индустрия обеспечила рабочими местами 11,3 млн человек. По числу занятых, в расчёте на один гостиничный номер, лидировала [Африка](https://ru.wikipedia.org/wiki/%D0%90%D1%84%D1%80%D0%B8%D0%BA%D0%B0) (3,3:1), за ней следовал [Ближний Восток](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%B8%D0%B6%D0%BD%D0%B8%D0%B9_%D0%92%D0%BE%D1%81%D1%82%D0%BE%D0%BA) (2,5:1), в отелях [Азии](https://ru.wikipedia.org/wiki/%D0%90%D0%B7%D0%B8%D1%8F) и [Океании](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%B5%D0%B0%D0%BD%D0%B8%D1%8F) на тот момент было занято 2,9 млн человек — четверть обслуживающего персонала мировой гостиничной индустрии. Доля Европы не превышала 24 %, и на одного работника здесь в среднем приходилось два номера (1:2,1).

В 1996 году 40 стран мира располагали гостиничным фондом, превышающим 100 тыс. мест в каждой. Средний размер отелей несколько меньше в традиционных [туристических странах](https://ru.wikipedia.org/wiki/%D0%A2%D1%83%D1%80%D0%B8%D0%B7%D0%BC) по сравнению с теми, которые лишь недавно вышли на рынок и заявили о себе как о новых туристических направлениях.

Гостиничная деятельность – деятельность юридических лиц и индивидуальных предпринимателей, обладающих или наделенных в установленном порядке имущественными правами на какое-то коллективное средство размещения, по непосредственному распоряжению и управлению им для предоставления услуг по временному размещению (проживанию) и обслуживанию граждан.

С точки зрения бизнеса гостиница представляет собой предприятие по производству и предоставлению услуг (гостиничного продукта) коммерческого гостеприимства, которое предлагает свои удобства и сервис потребителю.

Автоматизация бизнес – процессов в гостинице, отеле, санатории позволяет организовать стабильную работу предприятия и эффективно управлять им.

Целью данной работы является разработка автоматизированной информационной системы гостиницы.

Предметом работы является автоматизированная информационная система гостиничного бизнеса.

Работа является актуально по той причине, что на данный момент бизнес-процессы на данном предприятии не автоматизированы, что приводит к большому количеству ошибок в оформлении документов, и большому объему бумажного документооборота.

Автоматизация этих процессов позволит существенно сократить время подготовки необходимых документов и составления отчетов в результате использования справочников и современных технических средств.

# 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

# 

# 1.1 Описание и обоснование выбранного языка программирования

Java - объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения

Отличительной особенностью Java в сравнении с другими языками программирования общего назначения является обеспечение высокой продуктивности программирования, нежели производительность работы приложения или эффективность использования им памяти.

В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.

Три ключевых элемента объединились в технологии языка Java:

1) Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.

2) Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты.

3) Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

Огромное преимущество Java заключается в том, что на этом языке можно создавать приложения, способные работать на различных платформах. К сети Internet подключены компьютеры самых разных типов - Pentium PC, Macintosh, рабочие станции Sun и так далее. Даже в рамках компьютеров, созданных на базе процессоров Intel, существует несколько платформ, например, Microsoft Windows версии 3.1, Windows 95, Windows NT, OS/2, Solaris, различные разновидности операционной системы UNIX с графической оболочкой X­Windows. Между тем, создавая сервер Web в сети Internet, хотелось бы, чтобы им могло пользоваться как можно большее число людей. В этом случае выручат приложения Java, предназначенные для работы на различных платформах и не зависящие от конкретного типа процессора и операционной системы.

Программы, составленные на языке программирования Java, можно разделить по своему назначению на две большие группы.

К первой группе относятся приложения Java, предназначенные для автономной работы под управлением специальной интерпретирующей машины Java. Реализации этой машины созданы для всех основных компьютерных платформ.

Вторая группа - это так называемые аплеты (applets). Аплеты представляют собой разновидность приложений Java, которые интерпретируются виртуальной машиной Java, встроенной практически во все современные браузеры.

Приложения, относящиеся к первой группе - это обычные автономные программы. Так как они не содержат машинного кода и работают под управлением специального интерпретатора, их производительность заметно ниже, чем у обычных программ, составленных, например, на языке программирования C++. Программы Java без перетрансляции способны работать на любой платформе, что само по себе имеет большое значение в плане разработок для Internet.

Аплеты Java встраиваются в документы HTML, хранящиеся на сервере Web. С помощью аплетов вы можете сделать страницы сервера Web динамичными и интерактивными. Аплеты позволяют выполнять сложную локальную обработку данных, полученных от сервера Web или введенных пользователем с клавиатуры. Из соображений безопасности аплеты (в отличие от обычных приложений Java) не имеют никакого доступа к файловой системе локального компьютера. Все данные для обработки они могут получить только от сервера Web. Более сложную обработку данных можно выполнять, организовав взаимодействие между аплетами и расширениями сервера Web - приложениями CGI и ISAPI.

Для повышения производительности приложений Java в современных браузерах используется компиляция "на лету"- Just-In-Time compilation (JIT). При первой загрузке аплета его код транслируется в обычную исполнимую программу, которая сохраняется на диске и запускается. В результате общая скорость выполнения аплета Java увеличивается в несколько раз.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Так же как и библиотеки классов систем разработки приложений на языке C++, библиотеки классов Java значительно упрощают разработку приложений, представляя в распоряжение программиста мощные средства решения распространенных задач. Поэтому программист может больше внимания уделить решению прикладных задач.

Язык Java специально ориентирован на самые передовые технологии, связанные с сетью Internet. Растущая популярность Internet и, в особенности, серверов Web, создает для программистов новые возможности для реализации своих способностей.

# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Проведем формализованное описание информационной системы в терминах IDEF0 моделирования. В работе проводится моделирование с использование IDEF0(BPWin), UML (Rational Rose 2000), IDEF1x (ErWin).

1. Важная роль отводится процессу функционального проектирования.

Для регламентирования создания функциональных моделей ПС предназначен стандарт IDEF0 (Integrated Definition Function Modeling), который и реализован в пакете BpWin.

В основе IDEF0 лежит понятие блока, который реализует некую конкретную функцию. Четыре стороны блока имеют разное назначение. Слева отображаются входные данные (исходные данные). Справа – выходные данные (результат выполнения функции). Сверху – управление (управляющие воздействия на функцию). Снизу – механизм (посредством чего реализуется данная функция).

Функция – это управляемое действие над входными данными, результатом которого являются выходные данные, при этом используется некий механизм. Взаимодействие между функциями отображается в виде стрелок. Иногда стороны блока называют направлениями, а стрелки потоками. Стрелки можно подписывать. Подписи связываются с конкретной стрелкой при помощи зигзага.

В основе IDEF0 лежит три базовых принципа:

1) принцип функциональной декомпозиции – любая функция может быть разбита (декомпозирована) на более простые функции (более понятен термин детализация);

2) принцип ограничения сложности – количество блоков на диаграмме должно быть не менее двух, но не более шести (условие удобочитаемости);

3) принцип контекста – моделирование делового процесса начинается с построения контекстной диаграммы, на которой отображается только один блок – главная функция моделирующей системы, ограничивающая область границы моделирующей системы (регламентирует начальный этап построения модели).

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что является внешним воздействием на систему. Т. е. первоначально нужно определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели.

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой мы называем универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друх от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира. Примером сущности IDEF1X может быть сущность "СОТРУДНИК", которая представляет собой всех сотрудников предприятия, а один из них, скажем, Иванов Петр Сергеевич, является конкретной реализацией этой сущности.

Сase-средство BPwin значительно облегчает задачу создания информационной системы, позволяя осуществить декомпозицию сложной системы на более простые с тем, чтобы каждая из них могла проектироваться независимо, и для понимания любого уровня проектирования достаточно было оперировать с информацией о немногих ее частях.

Разработанные диаграммы выглядят следующим образом:

Цель модели – моделирование АРМ менеджера гостиницы.

Точка зрения модели – менеджер гостиницы, отвечающий за заселение и выселение граждан.

Входы модели (диаграммы):

1. Список проживающих в гостинице – содержит информацию о жильцах гостиницы;
2. План гостиницы – сведения о поведении в чрезвычайных ситуациях расположении запасных аварийных выходов;
3. Расценки– содержит расценки номеров , бронирование, количество мест и этаж, а еще расценки для Vip- гостей;
4. Время – все этапы процесса АРМ менеджера требуют затрат времени.

Процесс менеджера регламентируется порядком заселения и выселения граждан и порядком.

Выходные процессы менеджера – запись об свободных комнатах, записи их в базу и итоговая отчетность.

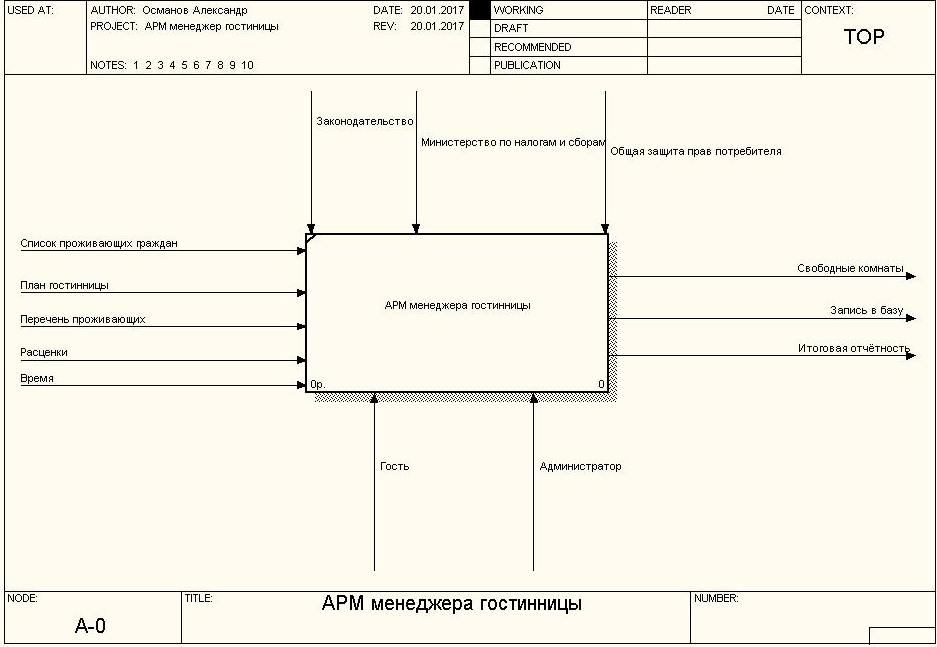


Рисунок 2.1- Контекстная диаграмма функционирования системы

Декомпозиция первого уровня контекстной диаграммы делится на 3 составных блока:

- проверка наличия мест;

- поиск комнаты;

- проверка будущих мест;

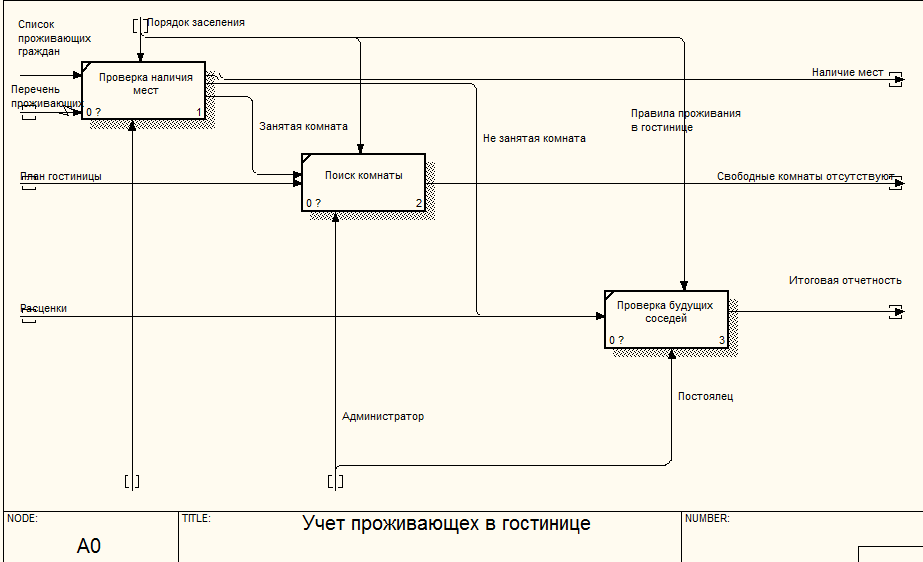


Рисунок 2.2- Декомпозиция диаграммы

После построения диаграммы декомпозиции первого уровня для указанных на ней функций строятся отдельные диаграммы (диаграммы декомпозиции второго уровня). Затем процесс декомпозиции (построения диаграмм) продолжается до тех пор, пока дальнейшая детализация функций не теряет смысла. Для каждой атомарной функции, описывающей элементарную операцию (т. е. функции, не имеющей диаграмму декомпозиции), составляется подробная спецификация, определяющая ее особенности и алгоритм реализации. В качестве дополнения к спецификации могут использоваться блок-схемы алгоритмов. Таким образом, процесс функционального моделирования заключается в постепенном выстраивании иерархии функций.

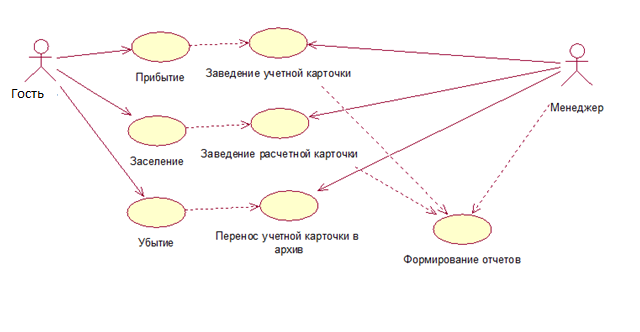
# 3 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

# 3.1 Диаграмма вариантов использования

UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

Рисунок 3.1 – Диаграмма вариантов использования

# 

# 3.2 Диаграмма классов

Диаграмма классов— диаграмма, демонстрирующая классы системы, их атрибуты, методы и взаимосвязи между ними. Входит в UML.

Существует два вида:

* Статический вид диаграммы рассматривает логические взаимосвязи классов между собой;
* Аналитический вид диаграммы рассматривает общий вид и взаимосвязи классов, входящих в систему.

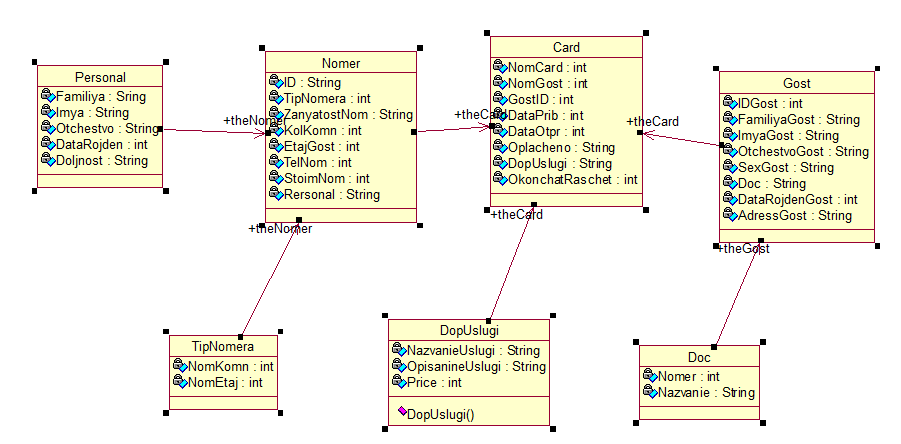
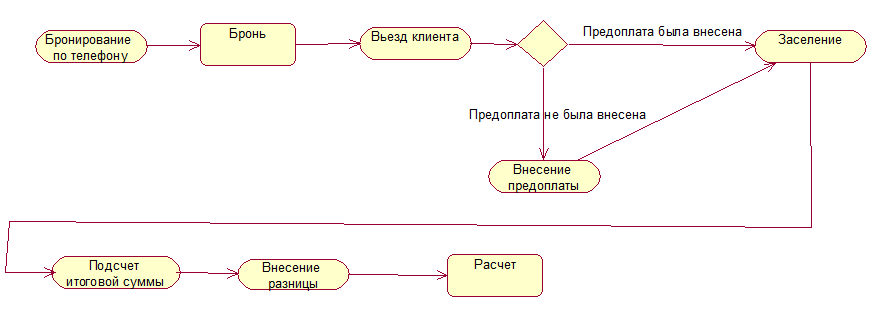


Рисунок 3.2 – Диаграмма классов

# 3.3 Диаграмма деятельности

Диаграмма деятельности — UML-диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Рисунок 3.3 – Диаграмма деятельности

# 3.4 Диаграмма коопераций

Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. На этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров. Следовательно, если необходимо явно специфицировать взаимосвязи между объектами в реальном времени, лучше это делать на диаграмме последовательности.

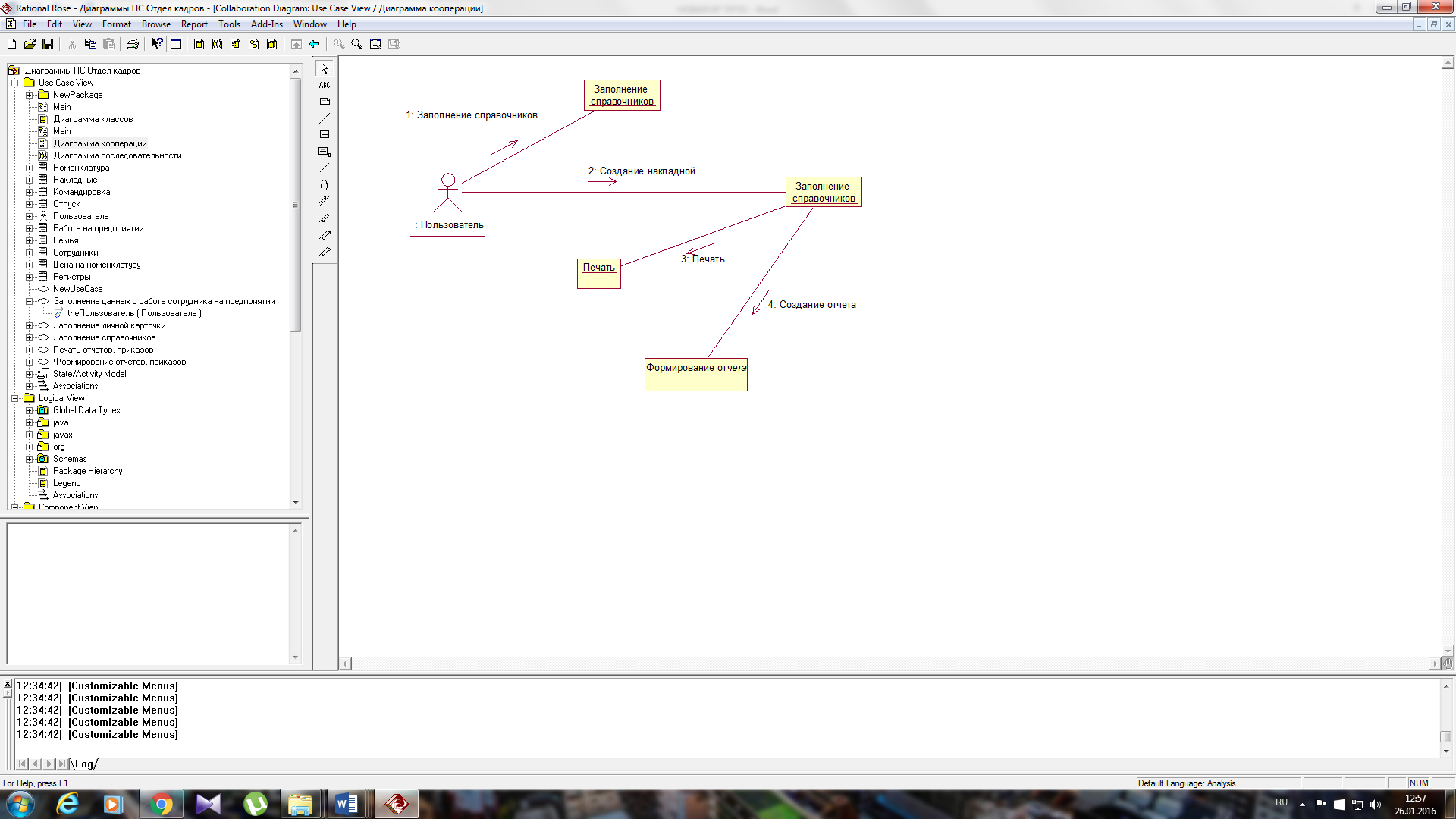


Рисунок 3.4 – Диаграмма коопераций

# 3.5 Диаграмма последовательности

Диаграмма последовательности — диаграмма, на которой показано взаимодействие объектов, упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления. Используется в языке UML.

На данной диаграмме объекты располагаются слева направо.

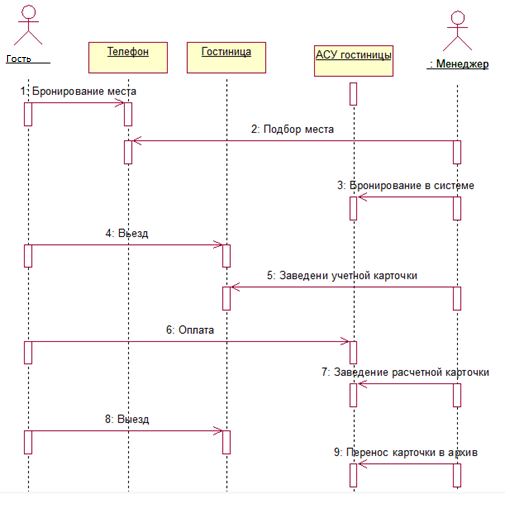


Рисунок 3.5 – Диаграмма последовательности

# 3.6 Диаграмма развёртывания

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

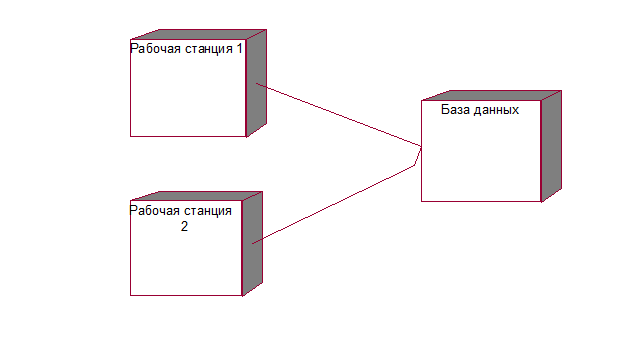


Рисунок 3.6 – Диаграмма развёртывания

# 3.7 Диаграмма состояний

Диаграмма состояний — это, по существу, диаграмма состояний из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до бизнес-процессов.

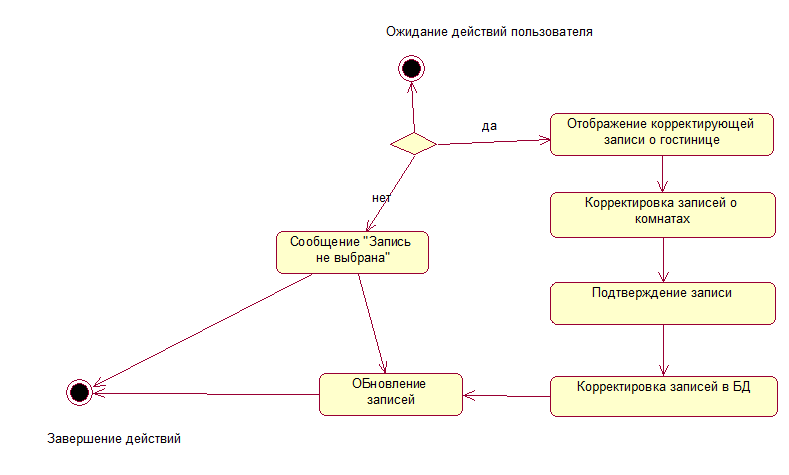


Рисунок 3.7 – Диаграмма состояний

# 3.8 Диаграмма компонентов

Диаграмма компонентов — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывается связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

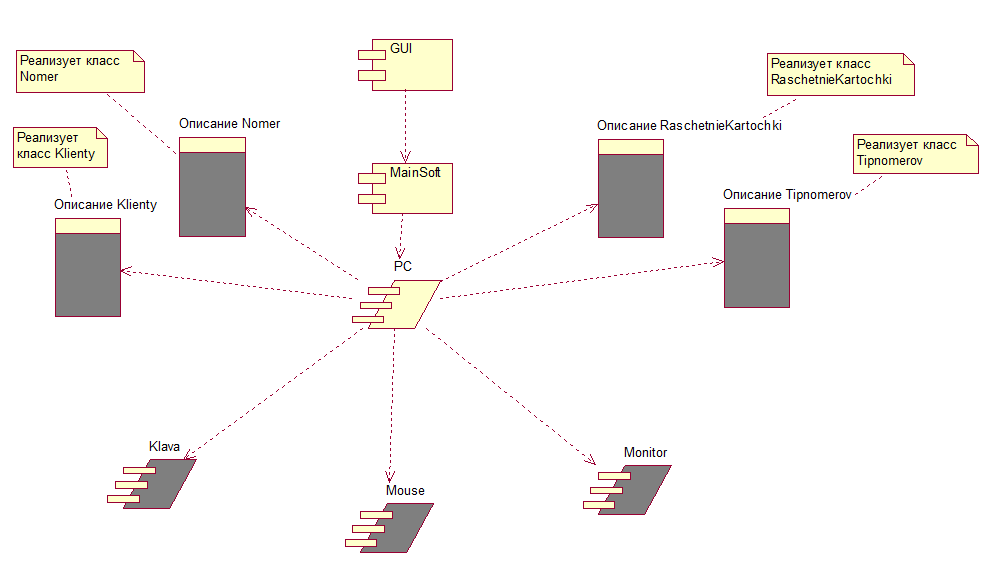


Рисунок 3.8 – Диаграмма компонентов

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Леоненков. «Самоучитель UML».

2. Б. Хичков SYBASE: Настольная книга администратора. Лори, 2000 г. 448 с.

3. Р. Мюллер. [Базы данных и UML: Проектирование](javascript:parent.descr(window,%22d160587%22)).– Лори, 2002г. 432 с.

4. Фельдман С.К. Система программирования Java без секретов: Как создать безопасное приложение с "нуля". – Новый издательский дом" , 2005 г. , 347 с.

5. Дейтел П.Дж., Дейтел Х.М. Как програмировать на Java. Книга 2. Файлы, сети, базы данных. – "Бином" · 2005 г., 672 с.

6. http://www.avacco.ru/page.asp?code=electronniy\_arhiv

7. <http://www.java.alfamoon.com/>

8. <http://www.cfin.ru/vernikov/idef/idef0.shtml>

9. http://www.cfin.ru/vernikov/idef/idef1x.shtml

10. Интернет обучение [Электронный ресурс] / Котляр Д.С. Использование CASE-средства ERwin для автоматизации проектирования и разработки базы данных – Режим доступа: http://royallib.com/read/ bezopasnost – Дата доступа: 15.12.2015.

11. Интернет обучение [Электронный ресурс] / Информационные системы и технологии – Режим доступа: <http://www.narfu.ru> – Дата доступа: 19.12.2016.

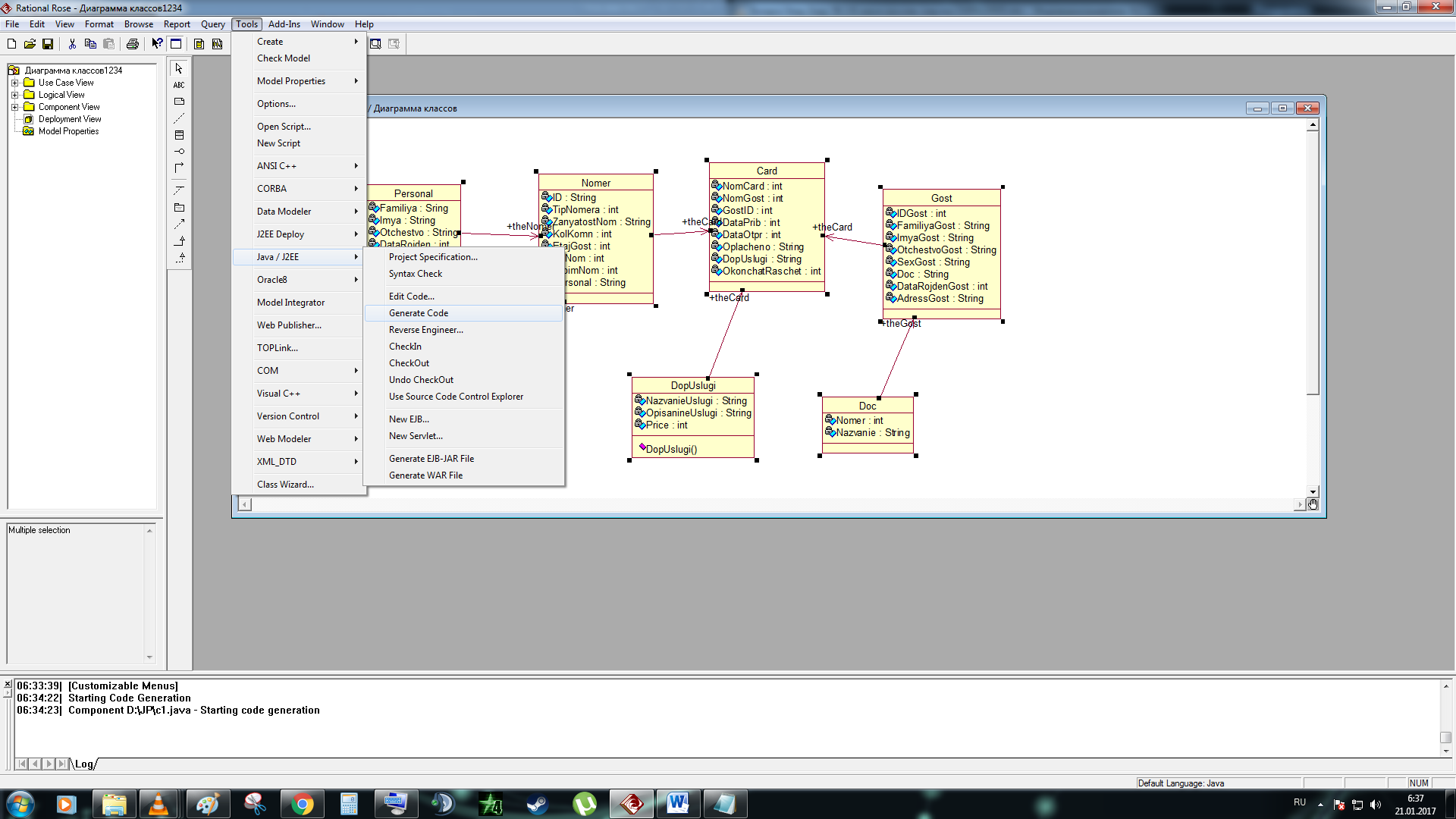
12. Интернет обучение [Электронный ресурс] / Нормализация структурны данных – Режим доступа: http://infostart.ru/public/269803/ – Дата доступа: 15.01.2017.

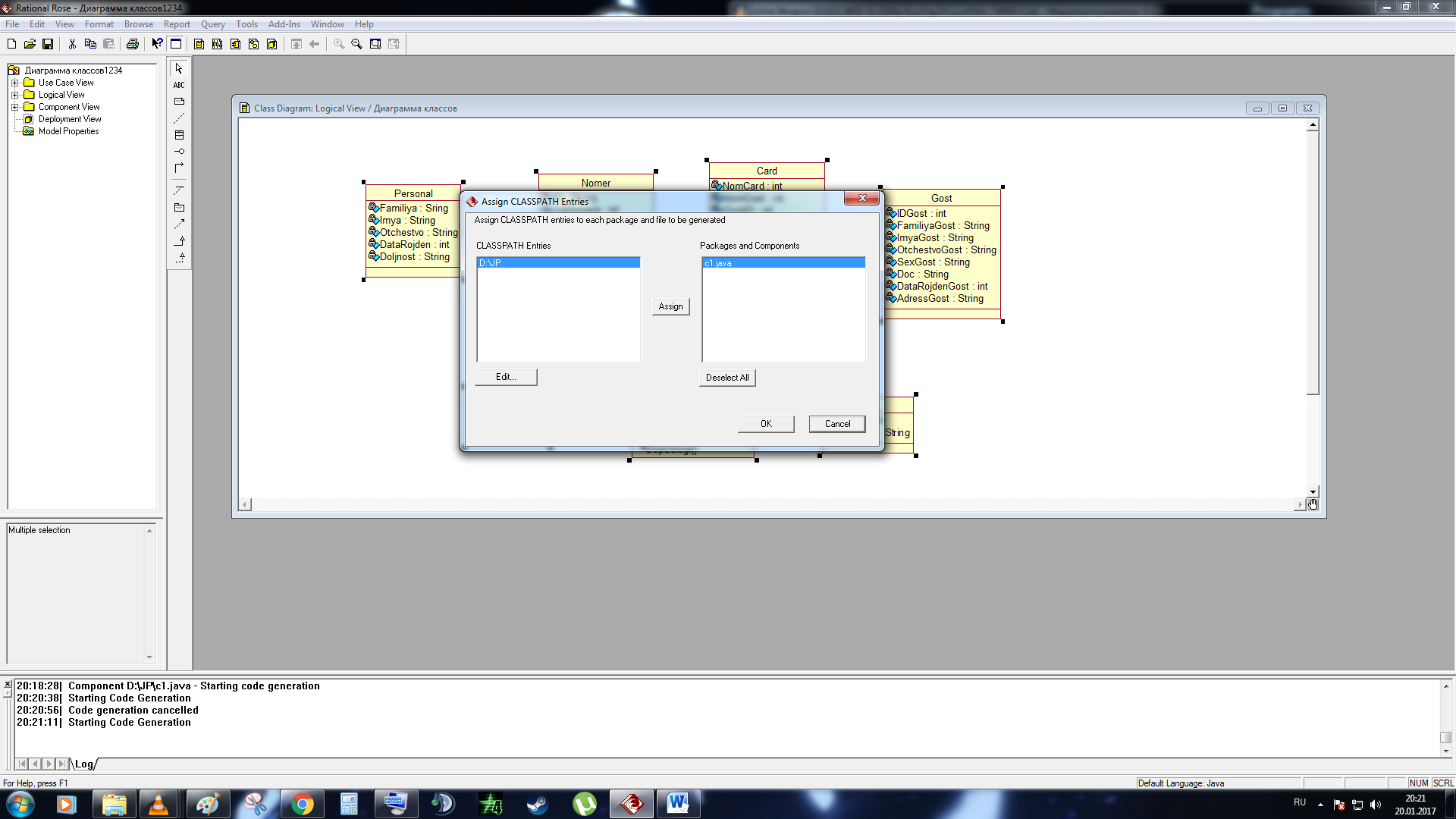
13. Интернет обучение [Электронный ресурс] / Программа компьютерного моделирования BpWin – Режим доступа: http://bourabai.kz/cm/bpwin.htm– Дата доступа: 11.11.2016.

14. Леонков, А. В. Самоучитель UML / А. В. Леонков. 2-е изд. – СПб.: БХВ - Петербург, 2007. – 596 с.

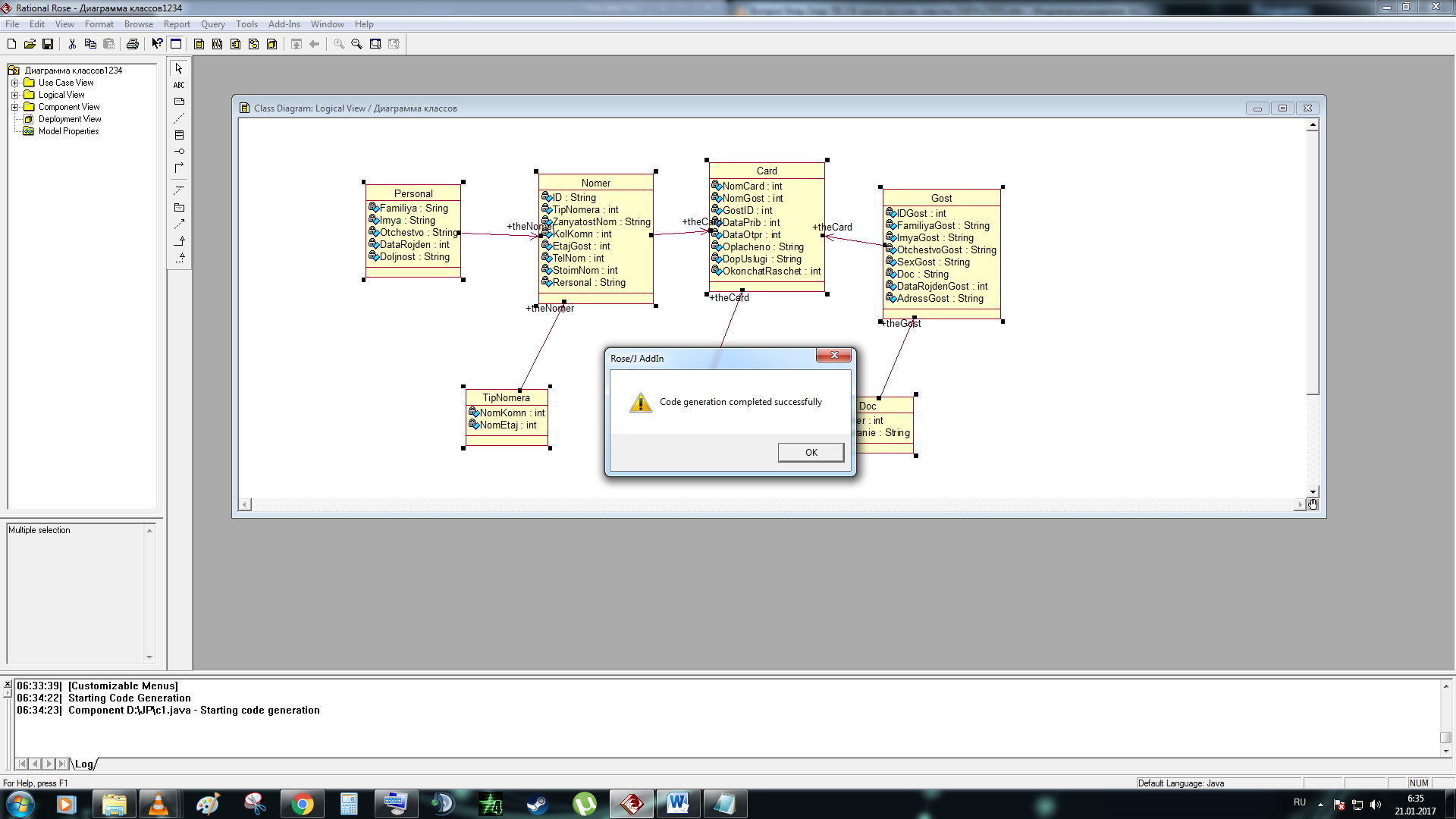
15. Хабрахабр, UML – диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – 2013. – Режим доступа: http://habrahabr.ru/post/47940/ . – Дата доступа: 15.01.2017.

**Генерация кода**





Код сгенерирован



//Source file: D:\\JP\\c1.java

private class Card

{

private int NomCard;

private int NomGost;

private int GostID;

private int DataPrib;

private int DataOtpr;

private String Oplacheno;

private String DopUslugi;

private int OkonchatRaschet;

}

private class Doc

{

private int Nomer;

private String Nazvanie;

public Gost theGost;

}

private class DopUslugi

{

private String NazvanieUslugi;

private String OpisanineUslugi;

private int Price;

/\*\*

@roseuid 588245AF0015

\*/

public DopUslugi()

{

}

}

private class Gost

{

private int IDGost;

private String FamiliyaGost;

private String ImyaGost;

private String OtchestvoGost;

private String SexGost;

private String Doc;

private int DataRojdenGost;

private String AdressGost;

public Card theCard;

}

private class Nomer

{

private String ID;

private int TipNomera;

private String ZanyatostNom;

private int KolKomn;

private int EtajGost;

private int TelNom;

private int StoimNom;

private String Rersonal;

public Card theCard;

}

private class Personal

{

private Sring Familiya;

private String Imya;

private String Otchestvo;

private int DataRojden;

private String Doljnost;

public Nomer theNomer;

}

private class TipNomera

{

private int NomKomn;

private int NomEtaj;

public Nomer theNomer;

}