Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Специальность: «Программное обеспечение информационных технологий»

**КОНТРОЛЬНАЯ РАБОТА**

По курсу: «Технология разработки программного обеспечения»

На тему: «Автоматизация работы салона кинопроката»

Студент-заочник 2 курса

Группы № 581072

ФИО: Яковлев Дмитрий Владимирович

Адрес г. Щучин,

ул. Пушкина д.3, кв. 44

Тел. 8-044-72-92-152

Проверила: Бакунова О.М.

Минск, 2017

**СОДЕРЖАНИЕ**

Введение 3

1 Анализ предметной области 4

1.1 Постановка задачи 4

1.2 Описание и обоснование выбранного языка программирования 5

2 Системное проектирование программого средтва. 7

3 Логическое моделирование 10

3.1 Диаграмма вариантов использования 10

3.2 Диаграмма классов 11

3.3 Диаграмма деятельности 12

3.4 Диаграмма коопераций 13

3.5 Диаграмма последовательности 14

3.6 Диаграмма развёртывания 14

3.7 Диаграмма компонентов 15

3.7 Генерация кода 17

Заключение 23

Список использованных источников 24

**ВВЕДЕНИЕ**

В сегодняшнем технологическом мире все больше увеличивается значительность информации в самом широком смысле этого слова и связанных с ней информационных технологий. Для больших объемов информации главным качеством является их структурированность, так как именно от характера структуры данных зависит скорость обработки поиска информации.

База данных на сегодняшний день – это самый распространенный «повод» для написания программ. Все современные языки программирования содержат в себе встроенные возможности для быстрого и удобного создания СУБД.

СУБД – это, конечно, нечто большее, чем просто набор структурированных данных. СУБД можно назвать «умной» средой, с помощью которой можно обрабатывать, искать, передавать, хранить данные. От качества СУБД зависит эффективность работы с базой данных в целом.

Специализированные СУБД создаются для управления базами данных конкретного назначения — бухгалтерскими, складскими, банковскими и т. д. Заказные СУБД требуют существенных затрат, а их подготовка к работе и отладка занимают значительный период времени (от нескольких месяцев до нескольких лет). Однако, в отличие от промышленных, заказные СУБД в максимальной степени учитывают специфику работы заказчика (того или иного предприятия), их интерфейс обычно интуитивно понятен пользователям и не требует от них специальных знаний.

## В настоящее время в разных сферах деятельности все более нуждаются в средствах, позволяющих быстро и безошибочно перерабатывать большое количество информации. Применение таких средств позволяет существенно снизить затраты и повысить эффективность работы.

## Например, в каждом видео прокате большой поток данных (клиенты, фильмы, режиссеры и т.п.) и чтобы снизить время обработки информации и облегчить работу сотрудникам заведений создается информационная система, которая может это позволить. Информационные технологии в настоящее время являются средством совершенствования и развития управленческой деятельности*.* Внедрение информационных систем способствуют более качественному управлению заведением, уменьшают расходы и повышают эффективность принимаемых управленческих решений.

Цель контрольной работы заключается в разработке информационной системы для удобства работы сотрудников видеопроката. Информационная система будет включать в себя базу данных и приложение для более функциональной работы. Для успешного создания работоспособной, отказоустойчивой и удобной информационной системы необходимо тщательное изучение предметной области, сбор исходных данных и их анализ.

## **1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Постановка задачи**

## Существенной частью современного общества являются разнообразные системы доступа и хранения информации, которые являются неотъемлемой составляющей современного научно-технического прогресса. Существует много веских причин перевода существующей информации на компьютерную основу, т.к. более быстрая обработка данных и централизация их хранения с использованием клиент/серверных технологий позволяют сберечь значительные средства, а главное и время для получения необходимой информации, а также упрощает доступ и ведение.

## Сущностью проекта является создание программы, предоставляющей услуги проката аудио- и видеокассет, DVD и CD дисков.. Видеопрокат относится к материальным услугам и характеризуется следующими качествами:

* неосязаемостью (их невозможно транспортировать, хранить, упаковывать или изучать до покупки, нельзя увидеть, услышать, потрогать, можно лишь верить в результат, можно лишь почувствовать эффект, который появится в результате получения услуги); - невозможностью хранения услуг;
* неотделимостью от источника (контакт с потребителем - неотъемлемая часть предоставления услуги видеопроката);
* изменчивостью, что означает непостоянства качества услуги; в данном виде деятельности качество удовлетворения потребности видеолюбителя зависит от квалификации продавца-консультанта, от менеджеров, формирующих ассортимент видеопродукции.

Программное средство (ПС) должно быть разработано для функционирования под управлением операционной системы Windows. Оно должно обладать интуитивно-понятным интерфейсом и интерактивным руководством, помогающим пользователю разобраться с управлением в процессе выполнения программы, так же требует наличие специальной поисковой системы для быстрого и эффективного поиска необходимой информации.

ПС должно обеспечить возможность эффективной работы с данными о товаре – мебели, находящейся в распоряжении магазина. Необходимо обеспечить возможность ведения учета информации о мебели, поступившей на склад и проданной, а также ведения базы данных о покупателях, поставщиках и продавцах. Также должна быть возможность добавления, редактирования, удаления и поиска данных по базе. Данные, которые зависят друг от друга, должны быть корректно сопоставлены.

## Требования, предъявляемые к данной программе:

## организация удобного пользовательского интерфейса;

## возможность постоянного обновления основных таблиц базы данных;

## возможность удаления записей из основных таблиц;

## возможность быстрой сортировки и поиска данных.

## К входным данным относятся следующие пункты:

## сведения о фильмах;

## сведения о работниках;

## заказы.

## Выходными данными являются:

## информация о сданных фильмах в прокат;

## кассовый чек (номер заказа, код товара, наименование товара, цена, количество, сумма и дата начала проката).

## К постоянным данным относятся:

## сведения о списке фильмов.

**1.2 Описание и обоснование выбранного языка программирования**

Java – [объектно-ориентированный язык программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный компанией [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems). Приложения Java обычно [компилируются](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой [виртуальной Java-машине](http://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) независимо от [компьютерной архитектуры](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0).

Основныевозможности:

* [автоматическое управление памятью](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0);
* расширенные возможности обработки исключительных ситуаций;
* богатый набор средств фильтрации ввода/вывода;
* набор стандартных коллекций, таких как [массив](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), [список](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29), [стек](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) и т.п.;
* наличие простых средств создания сетевых приложений (в том числе с использованием [протокола](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [RMI](http://ru.wikipedia.org/wiki/RMI));
* наличие классов, позволяющих выполнять [HTTP](http://ru.wikipedia.org/wiki/HTTP)-запросы и обрабатывать ответы;
* встроенные в язык средства создания многопоточных приложений;
* унифицированный доступ к [базам данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85):
* на уровне отдельных [SQL](http://ru.wikipedia.org/wiki/SQL)-запросов - на основе [JDBC](http://ru.wikipedia.org/wiki/JDBC), [SQLJ](http://ru.wikipedia.org/wiki/SQLJ);
* на уровне концепции объектов, обладающих способностью к хранению в базе данных - на основе [Java Data Objects](http://ru.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1) ([англ.](http://en.wikipedia.org/wiki/Java_Data_Objects) ) и [Java Persistence API](http://ru.wikipedia.org/w/index.php?title=Java_Persistence_API&action=edit&redlink=1) ([англ.](http://en.wikipedia.org/wiki/Java_Persistence_API) );
* поддержка шаблонов (начиная с версии 1.5);
* параллельное выполнение программ.

Внутри Java существуют несколько основных семейств технологий:

* [J2EE](http://ru.wikipedia.org/wiki/J2EE) или Java EE (начиная с v1.5) - Java Enterprise Edition, для создания программного обеспечения уровня предприятия;
* [J2SE](http://ru.wikipedia.org/wiki/J2SE) или Java SE (начиная с v1.5) - Java Standard Edition, для создания пользовательских приложений, в первую очередь - для настольных систем;
* [J2ME](http://ru.wikipedia.org/wiki/J2ME), Java ME или Java Micro Edition, для использования в устройствах, ограниченных по вычислительной мощности, в том числе [мобильных телефонах](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [КПК](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), встроенных системах;
* [Java Card](http://ru.wikipedia.org/wiki/Java_Card) для использования в устройствах без собственного человекомашинного интерфейса, например в [смарт-картах](http://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82-%D0%BA%D0%B0%D1%80%D1%82%D0%B0).

Самыми популярными считаются серверные технологии семейства J2EE.

Для реализации поставленных задач было выбрано J2SE.

1. **СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

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.

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило — наиболее важная функция находится в верхнем левом углу, кроме того есть правило стороны:

* стрелка входа приходит всегда в левую кромку активности,
* стрелка управления — в верхнюю кромку,
* стрелка механизма — нижняя кромка,
* стрелка выхода — правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-проектов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия:

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде прямоугольника (см. рис. 2.1) и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, “производить услуги”, а не “производство услуг”).

Декомпозиция контекстной диаграммы функциональной модели показана на рис. 2.2. Контекстная диаграмма модели показана на рисунке 2.1.

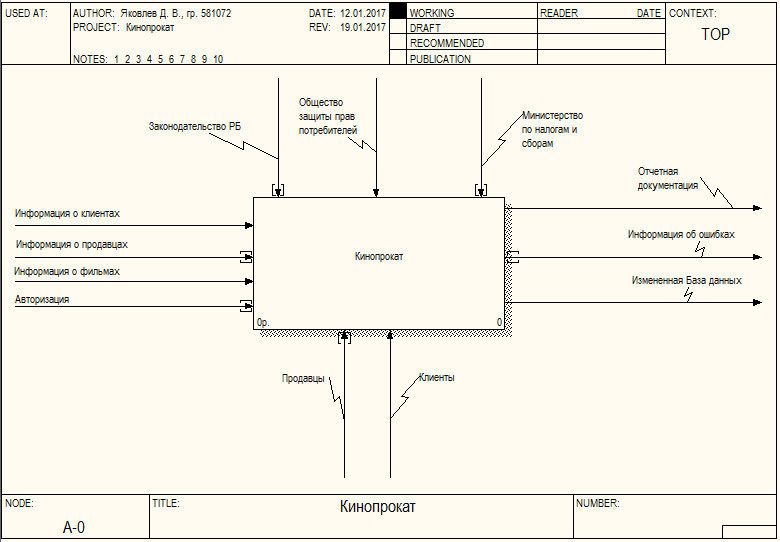


Рисунок 2.1 – Контекстная диаграмма модели

Входные данные:

* информация о клиентах (личный идентификатор, предпочтения, платежная информация, возраст, пол и др.);
* информация о продавцах (личный идентификатор, возраст, пол и др.);
* авторизация.

Управление:

* законодательство РБ;
* министерство торговли РБ
* общество защиты прав потребителей.

Механизмы:

* продавец;
* клиент.

Выходные данные:

* измененная БД;
* отчетная документация;
* информация об ошибках.

Контекстная диаграмма модели представлена на рисунке 2.2

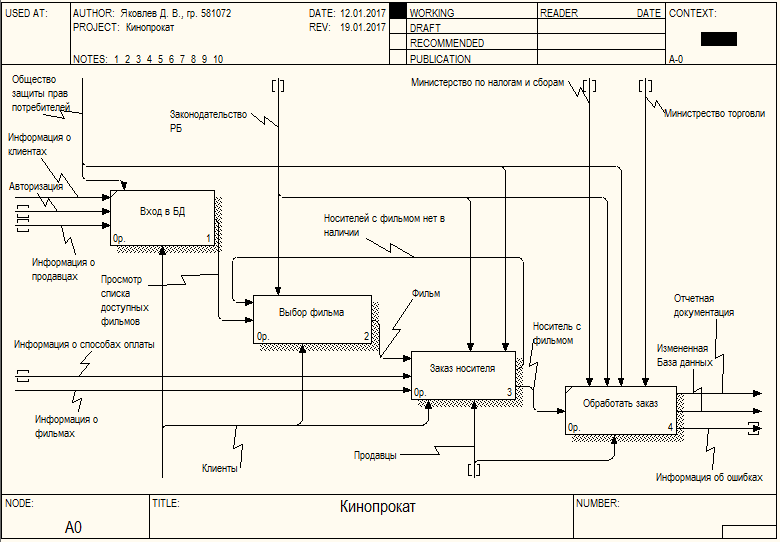


Рисунок 2.2– Декомпозиция контекстной диаграммы

функциональной модели

Для построения модели использовался продукт BPwin 4.0 фирмы Computer Associates.

Субъектом моделирования является программное средство «Администрирование кинопроката», цель – описать функциональность работы программного средства, точка зрения– пользователь, продавец.

1. **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**

Модель вариантов использования предназначается для определения требований к системе. Она включает в себя актеров, варианты использования и связи между ними.

Разработка диаграммы вариантов использования преследует цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.



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.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

* концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
* точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
* точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма классов представлена на рисунке 3.2.



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.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграмма деятельности представлена на рисунке 3.3.



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.

Прежде всего, на диаграмме кооперации в виде прямоугольников изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи - потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

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.

Диаграмма коопераций представлена на рисунке 3.4.



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.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Диаграмма последовательности представлена на рисунке 3.5.



Рисунок 3.5 – Диаграмма последовательности

### **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).

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь под узлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

* узел устройства;
* узел среды выполнения.

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Диаграмма развёртывания представлена на рисунке 3.6.



Рисунок 3.6 – Диаграмма развёртывания

### **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 внутренн яюю 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.

Диаграмма компонентов представлена на рисунке 3.7.



Рисунок 3.7 – Диаграмма компонентов

### **ГЕНЕРАЦИЯ КОДА**

На рисунках 4.1 — 4.5 представлен процесс генерации кода на языке Java, непосредственно сам код расположен ниже.

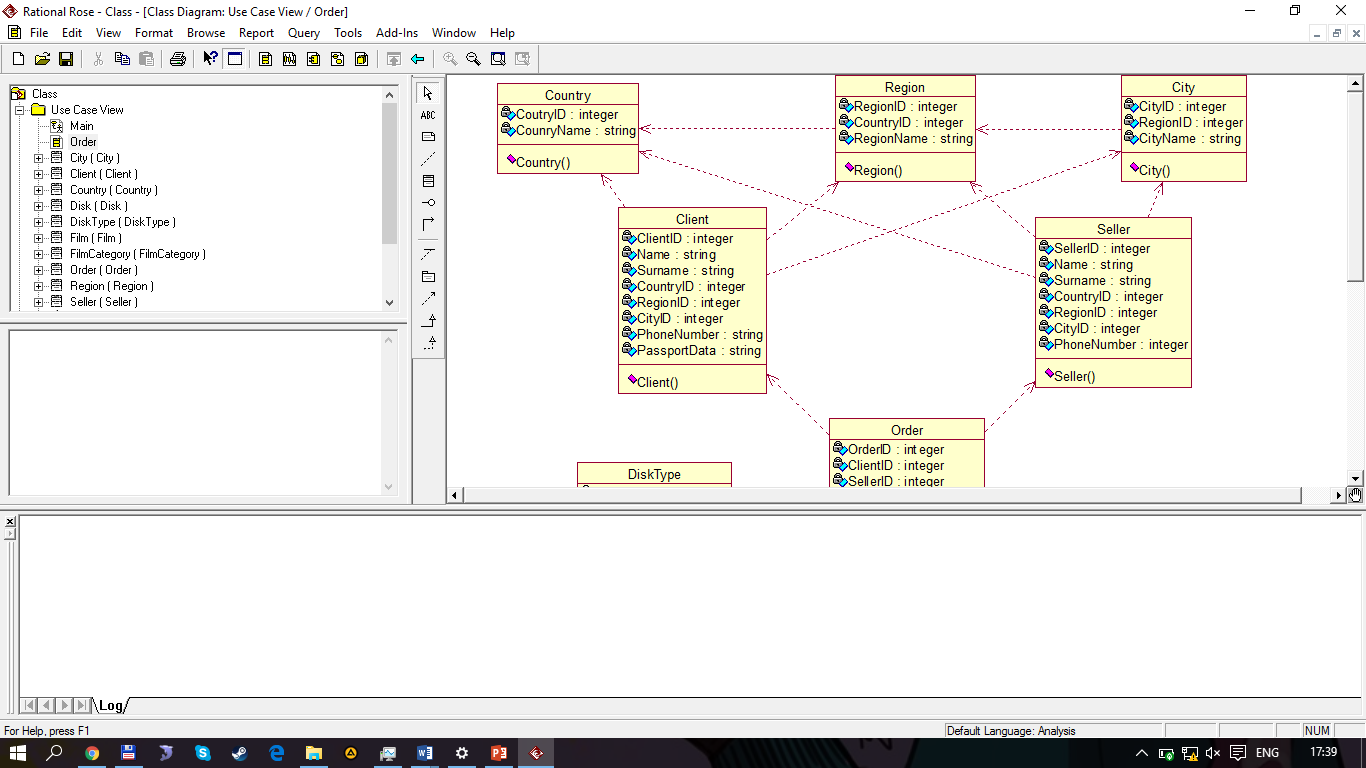


Рисунок 4.1 – Процесс генерации кода

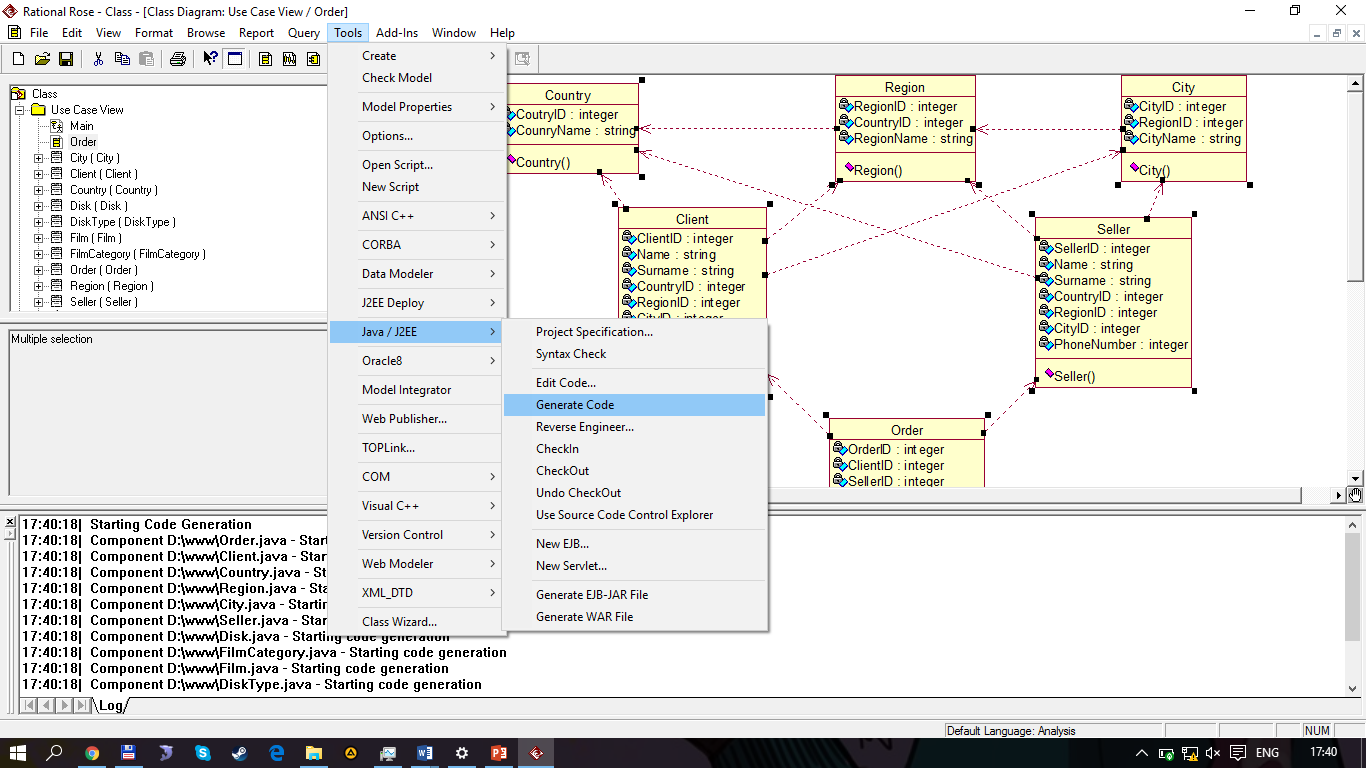


Рисунок 4.2 – Процесс генерации кода

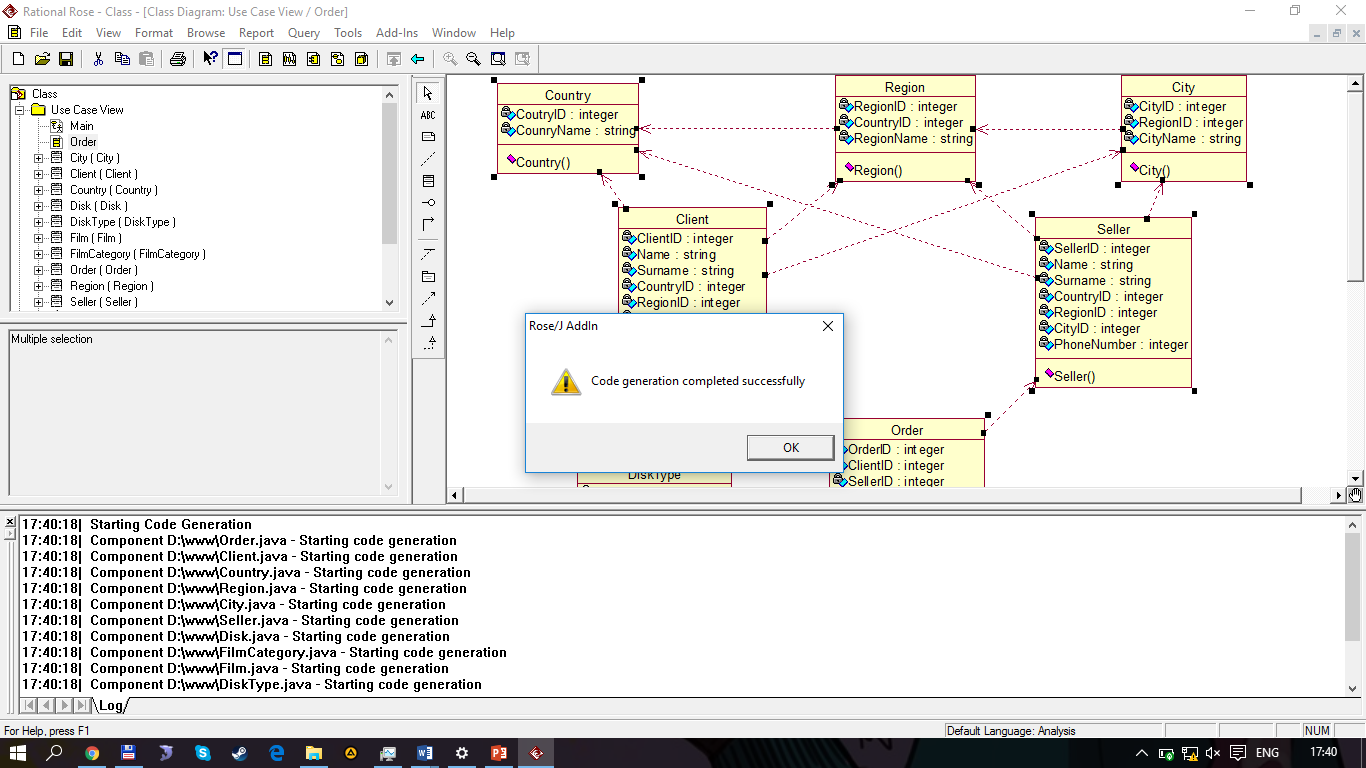


Рисунок 4.3 – Процесс генерации кода

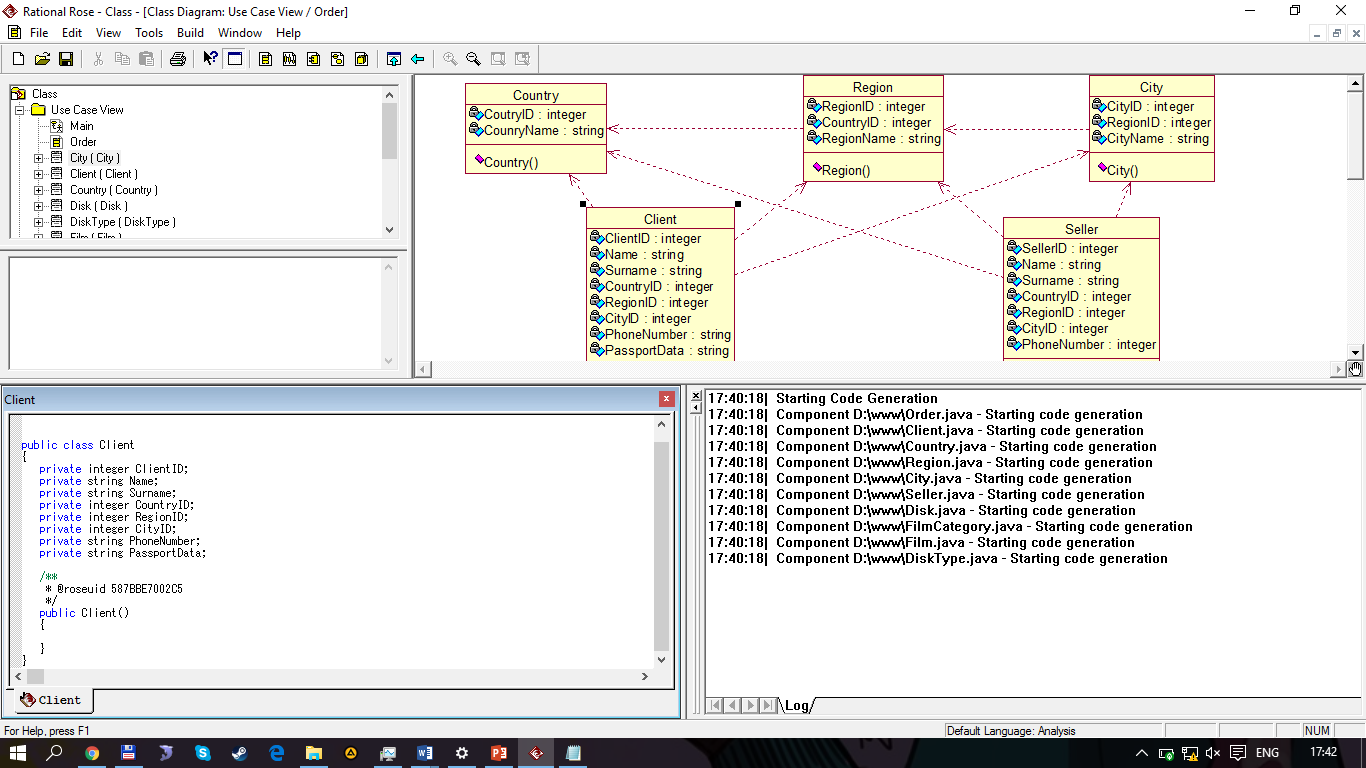


Рисунок 4.4 – Процесс генерации кода

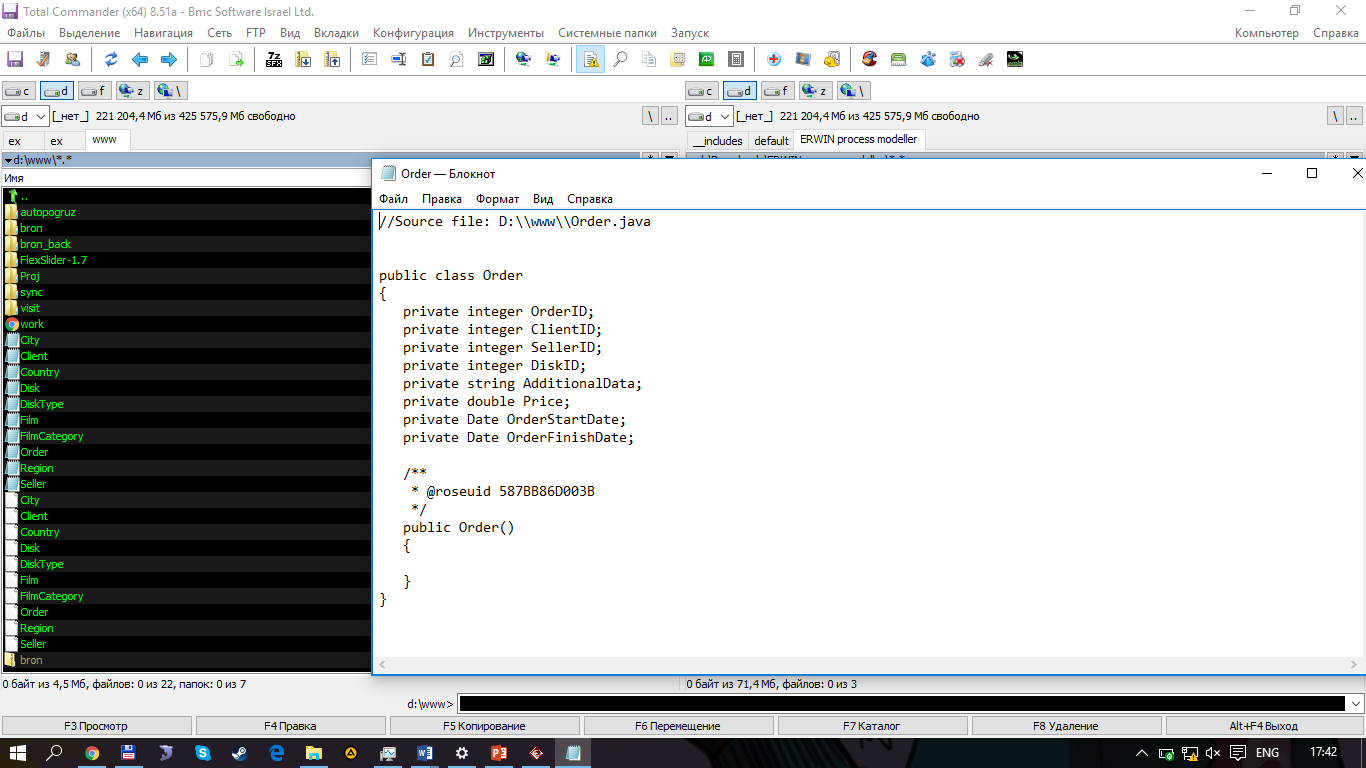


Рисунок 4.5 – Процесс генерации кода

Сгенерированный код:

//Source file: D:\\www\\City.java

public class City

{

private integer CityID;

private integer RegionID;

private string CityName;

/\*\*

\* @roseuid 587C8EFC031D

\*/

public City()

{

}

//Source file: D:\\www\\Client.java

public class Client

{

private integer ClientID;

private string Name;

private string Surname;

private integer CountryID;

private integer RegionID;

private integer CityID;

private string PhoneNumber;

private string PassportData;

/\*\*

\* @roseuid 587BBE7002C5

\*/

public Client()

{

}

}

//Source file: D:\\www\\Country.java

public class Country

{

private integer CoutryID;

private string CounryName;

/\*\*

\* @roseuid 587C9177004A

\*/

public Country()

{

}

}

}

//Source file: D:\\www\\Disk.java

public class Disk

{

private integer DiskID;

private integer DiskTypeID;

private integer FilmID;

private Date CreationDate;

/\*\*

\* @roseuid 587C8EDD012F

\*/

public Disk()

{

}

}

//Source file: D:\\www\\DiskType.java

public class DiskType

{

private integer DiskTypeID;

private string DiskTypeName;

private Time ServiseTime;

/\*\*

\* @roseuid 587C917700CE

\*/

public DiskType()

{

}

}

//Source file: D:\\www\\Film.java

public class Film

{

private integer FilmID;

private string Name;

private integer FilmCategoryID;

private integer Raiting;

private string Description;

/\*\*

\* @roseuid 587C8ED401CD

\*/

public Film()

{

}

}

//Source file: D:\\www\\FilmCategory.java

public class FilmCategory

{

private integer FilmCategoryID;

private string CategoryName;

/\*\*

\* @roseuid 587C8EB802DF

\*/

public FilmCategory()

{

}

}

//Source file: D:\\www\\Order.java

public class Order

{

private integer OrderID;

private integer ClientID;

private integer SellerID;

private integer DiskID;

private string AdditionalData;

private double Price;

private Date OrderStartDate;

private Date OrderFinishDate;

/\*\*

\* @roseuid 587BB86D003B

\*/

public Order()

{

}

}

//Source file: D:\\www\\Region.java

public class Region

{

private integer RegionID;

private integer CountryID;

private string RegionName;

/\*\*

\* @roseuid 587C8F040263

\*/

public Region()

{

}

}

//Source file: D:\\www\\Seller.java

public class Seller

{

private integer SellerID;

private string Name;

private string Surname;

private integer CountryID;

private integer RegionID;

private integer CityID;

private integer PhoneNumber;

/\*\*

\* @roseuid 587C8EF500D5

\*/

public Seller()

{

}

}

**ЗАКЛЮЧЕНИЕ**

Предметной областью контрольной работы является программное средство «Администрирование кинопроката». Разработанное программное существенно упрощает и автоматизирует работу.

Пояснительная записка состоит из следующих разделов:

1. Постановка задачи. В данном разделе была поставлена задача по разработке программного средства «Администрирование кинопроката,0».

2. Описание основного процесса предметной области с использованием стандарта IDEF0.

3. Описание информационной модели с использованием стандарта IDEF1.

4. Описание моделей информационной системы с использованием языка UML.

5 Генерация кода на основе моделей Rational Rose.

В ходе написания контрольной работы были закреплены теоретические знания по таким разделам как составление функциональной модели (IDEF0), проектирование системы с помощью [унифицированного языка моделирования](http://edu.dvgups.ru/METDOC/GDTRAN/YAT/ITIS/PROEK_INF_SIS/METOD/UMK_DO/frame/UMK_DO/M6/L11.htm#11_1) (UML).

Контрольная работа выполнялась в соответствии с методическими требованиями и указаниями.

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. ГОСТ 2.105-95 ЕСКД. Введ. 1996–07–01. – М.: Изд-во стандартов, 1996.
2. ГОСТ 19.104–78 – Единая система программной документации. Основные надписи.
3. Маклаков С.В. BPwin и ERwin: CASE - средства для разработки информационных систем.
4. Федотов Д.Э., Семенов Ю.Д., Чижик К.Н. Практикум для высших учебных заведений. CASE-технологии.-157с.
5. 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.
6. Интернет обучение [Электронный ресурс] / Информационные системы и технологии – Режим доступа: <http://www.narfu.ru> – Дата доступа: 19.12.2016.
7. Интернет обучение [Электронный ресурс] / Нормализация структурны данных – Режим доступа: http://infostart.ru/public/269803/ – Дата доступа: 15.01.2017.
8. Интернет обучение [Электронный ресурс] / Программа компьютерного моделирования BpWin – Режим доступа: http://bourabai.kz/cm/bpwin.htm– Дата доступа: 11.11.2016.
9. Леонков, А. В. Самоучитель UML / А. В. Леонков. 2-е изд. – СПб.: БХВ - Петербург, 2007. – 596 с.
10. Хабрахабр, UML – диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – 2013. – Режим доступа: http://habrahabr.ru/post/47940/ . – Дата доступа: 15.01.2017.