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: «Торговая платформа Forex»

Студент-заочник 2 курса

Группы № 581072

ФИО: Шакола Александр Юрьевич

Адрес г. Минск,

Ул.Одинцова

д. 22, к. 1, кв.9

Тел. +375 29 344 99 31

Проверила: Бакунова О.М.

Минск, 2017

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 ПОСТАНОВКА ЗАДАЧИ 4

1.1 Анализ предметной области 4

1.2 Цели и задачи проектирования 4

2 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ. 6

3 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ 9

3.1 Диаграмма вариантов использования 9

3.2 Диаграмма классов 10

3.3 Диаграмма деятельности 12

3.4 Диаграмма коопераций 14

3.5 Диаграмма последовательности 15

3.6 Диаграмма развёртывания 17

3.7 Диаграмма компонентов 18

4 ГЕНЕРАЦИЯ КОДА 20

ЗАКЛЮЧЕНИЕ 30

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31

**ВВЕДЕНИЕ**

Современная жизнь немыслима без эффективного управления. Важной категорией являются системы обработки информации, от которых во многом зависит эффективность работы любого предприятия или учреждения.

Для принятия обоснованных и эффективных решений в производственной деятельности, в управлении экономикой и в политике современный специалист должен уметь с помощью компьютеров и средств связи получать, накапливать, хранить и обрабатывать данные, представляя результат в виде наглядных документов.

Темой контрольной работы является разработка программного продукта «Торговая платформа Forex». Рынок Forex – это глобальный международный рынок, товаром на котором выступают валюты. Его отличительной особенностью является то, что цены формируются на основе соглашения между участниками и зависят только от спроса и предложения на ту или иную валюту. Валютный рынок Форекс позволяет зарабатывать в любой ситуации благодаря колебанию курсов. Если спрос на какую-то валюту падает (соответственно, данная валюта дешевеет), это говорит о том, что какая-то другая валюта становится более востребованной, а ее цена растет. Это подводит нас к определению одного из основных понятий рынка Форекс — курса валютной пары. Котировки курсов валютной пары меняются постоянно, каждую секунду. Для нормальной и оптимальной работы пользователя без потери в средствах необходимо программное средство позволяющее отслеживать котировки рынков в реальном времени.

Для реализации поставленной цели необходимо решение следующих задач:

* разработка логической модели программного средства;
* создание простого и понятного интерфейса;
* создание единой базы данных для работы программы;
* обеспечение целостности данных при работе с базой данных;
* возможность редактирования, изменение, добавление данных в программу;
* предоставление возможности работы с базой данной простому пользователю, не являющимся его администратором;

# **1 ПОСТАНОВКА ЗАДАЧИ**

## **1.1 Описание предметной области**

Основной оценкой качества обслуживания рынка forex является актуальность отображаемой котировки что позволит пользователю не терять денежные средства. Также важна оптимизация приложения, что позволит клиенту быстро открывать новые позиции. В дополнении важно иметь несколько удобных инструментов позволяющих пользователю управлять позициями автоматически (без его участия).

В настоящее время потребность использовать автоматизированные средства для торговле на рынке резко возросла. Обстановка в мире меняется ежеминутно. Игроки теряют огромные вложения лишь за счёт того что не успевают контролировать поток средств в условиях высокой динамики рынков.

Программное средство (ПС), должно обеспечить возможность ведения учета информации о котировках рынков и балансе пользователя, позволяющие клиенту нейтрализовать потери в средствах в связи с быстро меняющейся обстановкой в мире.

**1.2 Цели и задачи проектирования**

Требуется создать программное средство, которое бы обеспечивала автоматизированную систему для игры на рынках в реальном времени. Суть данной программы в том, чтобы постоянно отображать актуальную информацию о котировках а также автоматизировать управление средствами клиента во время игры.

Необходимо предоставить возможность добавления, редактирования, удаления и поиска данных по базе. Данные, которые зависят друг от друга, должны быть корректно сопоставлены.

Разрабатываемая программа должна обеспечивать следующее:

* точность информации о котировка и истории рынка;
* возможность доступа к требуемой информации и статистике;
* ведение базы данных.

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:

* котировки биржевых рынков;
* средства клиентов.

Выходными данными являются:

* актуальная информация о рынках;
* актуальная и точечная статистика;
* корректные и максимально подробные сообщения о сбоях в управлении средствами;
* средства заработанные клиентами во время игры

К постоянным данным относятся:

* список доступных рынков.

Разработанное программное обеспечение позволит быстро и легко как вести базу данных, так и формировать отчеты для дальнейшего развития программного средства. Выбор СУБД и языка программирования позволили создать компактную реляционную модель базы данных отвечающей последним инновационным требованиям. Внешний дизайн каждой формы должен быть ориентирован на интуитивный интерфейс и позволяет легко и быстро получать интересующую в данный момент информацию.

1. **ФУКНЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ**

Субъектом моделирования является программное средство «Торговая платформа Forex», цель – описать функциональность работы программного средства, точка зрения– клиент, управляющий.

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).

В основе IDEF0 методологии лежит понятие блока, который отображает некоторую бизнес-функцию. Четыре стороны блока имеют разную роль: левая сторона имеет значение "входа", правая - "выхода", верхняя - "управления", нижняя - "механизма" (рисунок 2.1)

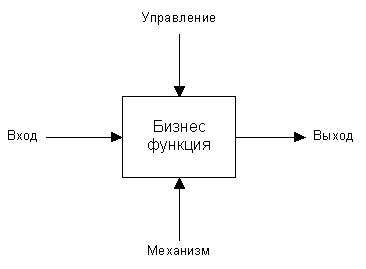


Рисунок 2.1 – Стрелки управления функционального блока

В IDEF0 реализованы три базовых принципа моделирования процессов:

* принцип функциональной декомпозиции;
* принцип ограничения сложности;
* принцип контекста.

Принцип контекстной диаграммы. Моделирование делового процесса начинается с построения контекстной диаграммы. На этой диаграмме отображается только один блок - главная бизнес-функция моделируемой системы. Главная бизнес-функция разрабатываемой системы – ТОРГОВАЯ ПЛАТФОРМА FOREX, так как программное средство реализует почти все функции торговой платформы: открытие позиций, закрытие позиций (в том числе автоматически по ограничениям), обновление котировок, вывод и внесение денежных средств. Входные данные:

* информация о клиентах (ФИО, платежная информация, возраст, пол и др.);
* информация о рынках (название, котировка, дозволенные мультипликаторы)
* авторизация.

Управление:

* Международный Валютный Фонд (МВФ);
* Национальный банк Республики Беларусь;
* Министерство по налогам и сборам Республики Беларусь
* Международные фондовые биржы Forex.

Механизмы:

* клиент;
* управляющий.

Выходные данные:

* баланс клиента;
* статистика и история изменения котировок;
* информация об ошибках.

Контекстная диаграмма модели представлена на рисунке 2.2:

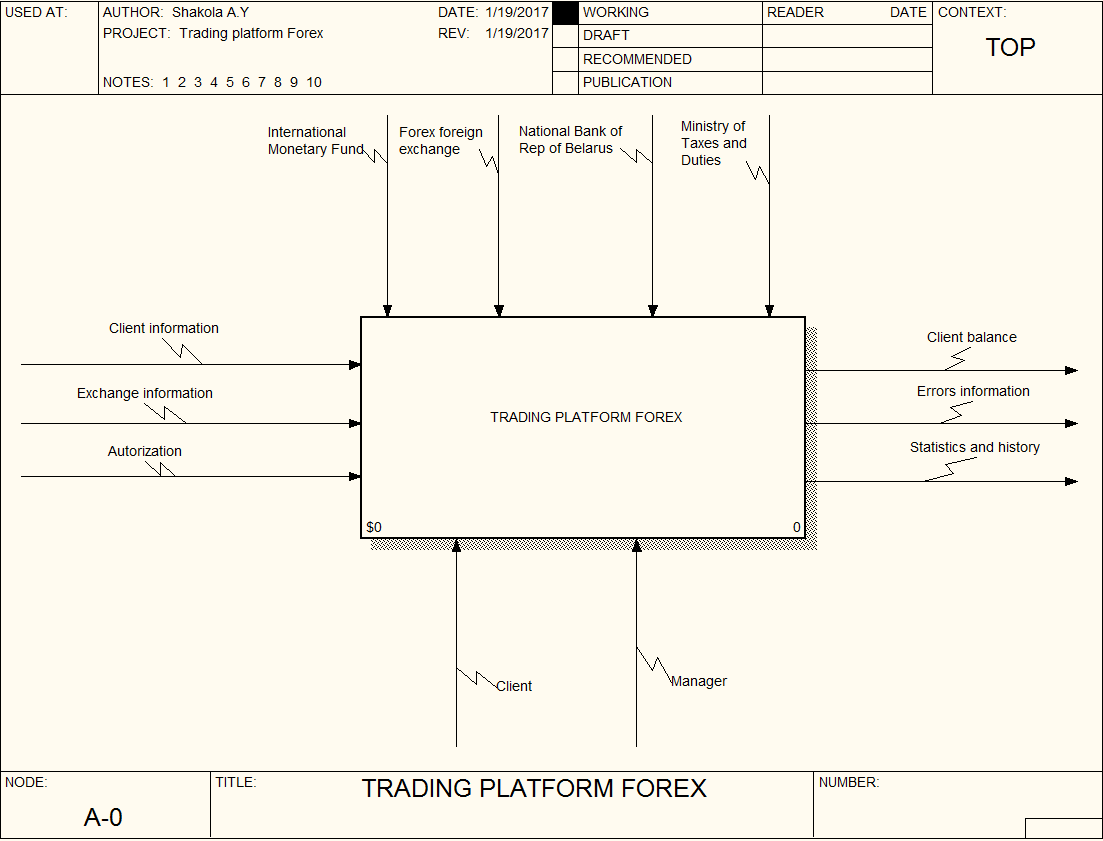


Рисунок 2.2 – Контекстная диаграмма

Принцип ограничения сложности. При работе с IDEF0 диаграммами существенным является условие их разборчивости и удобочитаемости. Суть принципа ограничения сложности состоит в том, что количество блоков на диаграмме должно быть не менее двух и не более шести. Практика показывает, что соблюдение этого принципа приводит к тому, что функциональные процессы, представленные в виде IDEF0 модели, хорошо структурированы, понятны и легко поддаются анализу.

Принцип функциональной декомпозиции представляет собой способ моделирования типовой ситуации, когда любое действие, операция, функция могут быть разбиты (декомпозированы) на более простые действия, операции, функции. Другими словами, сложная бизнес-функция может быть представлена в виде совокупности элементарных функций. Представляя функции графически, в виде блоков, можно как бы заглянуть внутрь блока и детально рассмотреть ее структуру и состав.

С помощью принципа декомпозиции главная бизнес функция была разбита на 4 активные составляющие: вход БД, получить список рынков, подсчёт баланса, вывод баланса. Эта последовательность в общей сложности и представляет из себя игру на рынке Forex. Декомпозиция контекстной диаграммы функциональной модели показана на рисунке 2.3:

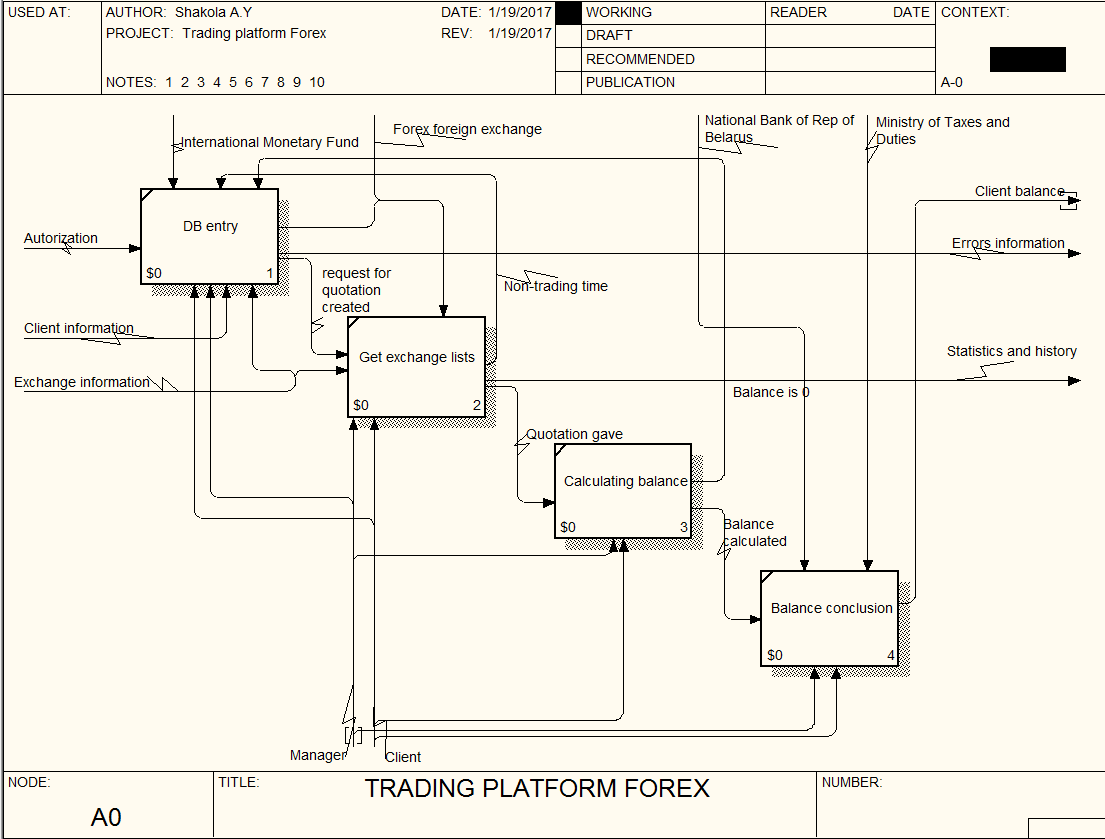


Рисунок 2.3 - Декомпозиция контекстной диаграммы

Для построения модели использовался продукт BPwin 7.3.3.1773 фирмы Computer Associates.

1. **ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ**

Процесс разработки программного средства начинается с этапа проектирования диаграмм, отражающих процесс функционирования приложения. Для проектирования диаграмм используется унифицированный язык моделирования UML. Он определяет следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграмма объектов, диаграмма последовательностей, диаграмма взаимодействия, диаграмма состояний, диаграмма активности, диаграмма развертывания.

### **Диаграмма вариантов использования**

Функционирование приложения можно отразить с помощью диаграммы вариантов использования, диаграммы деятельности, диаграммы последовательности и диаграммы классов.

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе.

Важно понимать, что диаграммы вариантов использования не предназначены для отображения проекта и не могут описывать внутреннее устройство системы. Диаграммы вариантов использования предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы. Другими словами, диаграммы вариантов использования говорят о том, что система должна делать, не указывая сами применяемые методы.

Диаграмма вариантов использования представлена на рисунке 3.1

В качестве актеров данной программы выступают: пользователь (или клиент), терминал (автоматическая система – сервер – транслирующий информацию прямым каналом с мировых бирж)

Как клиент, так и терминал для работы с системой должен авторизоваться в базе данных, а уже потом взаимодействовать с основными функциями программы. Пользователю предоставляются функции работающие как API после чего команда будет послана терминалу задача которого обработать запрос и выдать пользователю ответ, а также осветить событие в базе данных (в том числе записать статистику и историю).

Вариант использования «Создать инвестицию» инициирует отправку команды терминалу для открытия позиции на выбранный рынок, взамен терминал вернёт пользователю подтверждение и выбрав вариант использование «Активы» пользователь сможет наблюдать изменения баланса своей инвестиции в реальном времени, которую терминал передаёт после каждого прохода через вариант использования «Расчёт».

Также пользователю доступна возможность просмотра рынков через вариант использования «Рынки» и доступных управляющих (вариант использование «Управляющие»). Вся информация постоянно обновляется терминал с помощью варианта использование «Информация об управляющих» и «Информация о рынках».

Управляющие это – клиент которые достигли значительных успехов на платформе и могут предложить услуги по управлению средства другим пользователям что связано для них с риском потери денежных средств в случае проигрыша управляющего но освобождает его от бремени следить за ситуацией на рынках

Диаграмма вариантов использования показана на рисунке 3.1:

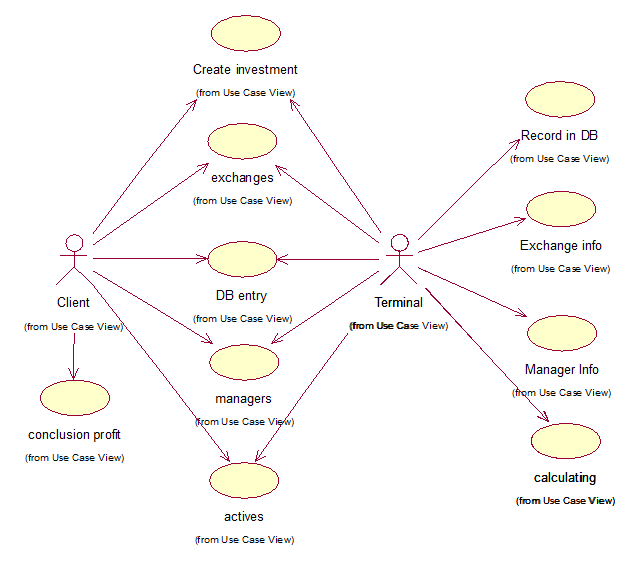


Рисунок 3.1 – Диаграмма вариантов использования

### **3.2 Диаграмма классов**

Центральное место в объектно-ориентированном программировании занимает разработка логической модели системы в виде диаграммы классов. Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений.

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений. Диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи.

Существует два вида:

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

Классы History и Statistics наследует свойства класса Active. Класс менеджеров представляет собой специфически класс Client и наследует его свойства. В остальном же случае классы связаны элементом отношения.

Диаграмма классов представлена на рисунке 3.2:

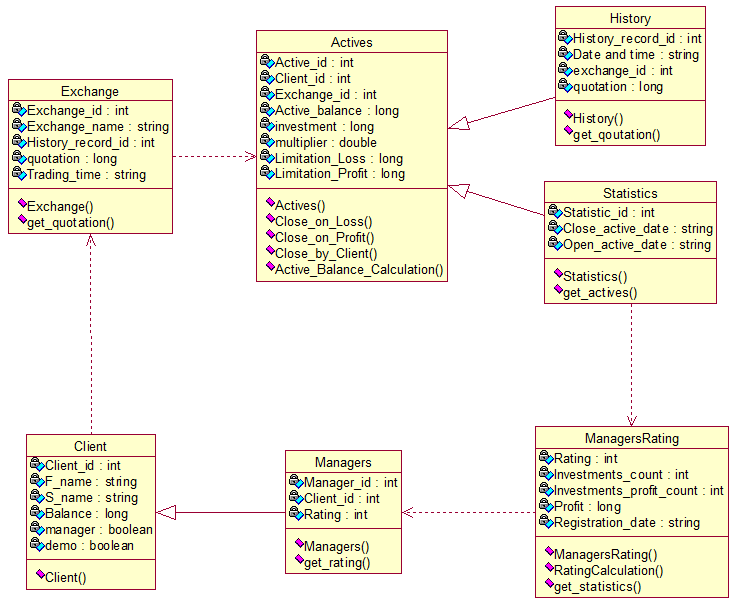


Рисунок 3.2 – Диаграмма классов

**3.3 Диаграмма деятельности**

При моделировании поведения системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы 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.

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности.

Компонентами диаграммы деятельности являются:

* состояния действия;
* переходы;
* дорожки;
* объекты.

Диаграмма деятельности для реализуемого программного средства начинается с начального узла, далее подключение к БД и авторизация, а затем идет разделение действий, в зависимости от того какие рынки хочет рассмотреть пользователь для последующей инвестиции.

Далее клиент выбирает рынок для инвестиции и пытается открыть позицию. Пользователь может получить отказ в случае если выбранный рынок находится в не торговом времени в данный момент. В таком случае пользователь будет возвращён в выбор типа рынка. Если позиция будет успешно открыта пользователю будет показан список его позиций. Даллее пользователь может вернутся или к выбору типа рынка или выводу средств из приложения.

Остальные действия выполняются последовательно.

Диаграмма деятельности представлена на рисунке 3.3:

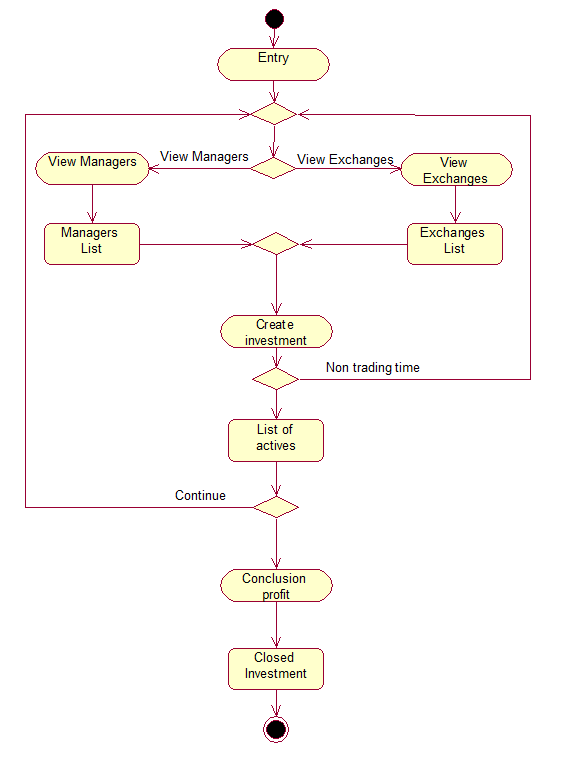


Рисунок 3.3 – Диаграмма деятельности

**3.4 Диаграмма развёртывания**

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Если разрабатывается программа, выполняющаяся локально на компьютере пользователя и не использующая периферийных устройств и ресурсов, то в разработке дополнительных диаграмм нет необходимости. При разработке же корпоративных приложений наличие таких диаграмм может быть крайне полезным для решения задач рационального размещения компонентов в целях эффективного использования распределенных вычислительных и коммуникационных ресурсов сети, обеспечения безопасности и других. Для представления общей конфигурации и топологии распределенной программной системы в UML предназначены диаграммы развертывания.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации. Разработка диаграммы развертывания, как правило, является последним этапом спецификации модели программной системы.

При разработке диаграммы развертывания преследуют следующие цели:

– 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.

Диаграмма развёртывания представлена на рисунке 3.4:

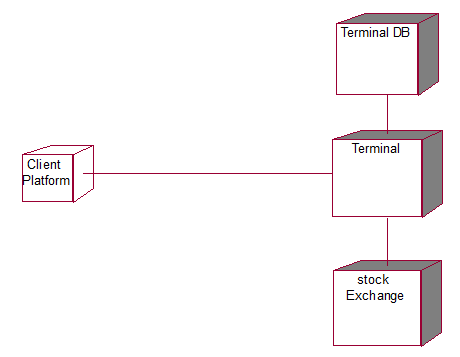


Рисунок 3.4 – Диаграмма развёртывания

**3.5 Диаграмма последовательности**

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Говоря об этих диаграммах, имеют в виду два аспекта взаимодействия. Во-первых, взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности. Временной аспект поведения может иметь существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели в языке UML используются диаграммы 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 компонентами диаграммы последовательности являются:

* объекты;
* линия жизни;
* фокусы управления;
* сообщения;
* ветвления.

В данной диаграмме представлена последовательность действий системы при открытии позиции пользователя. Обновление котировок рынков. Действия системы в случае отказа в открытии в позиции в связи с нахождением рынка в неторговой моменте, последовательность действий при выводе клиентом баланса на свой личный счёт

Диаграмма последовательности представлена на рисунке 3.5:

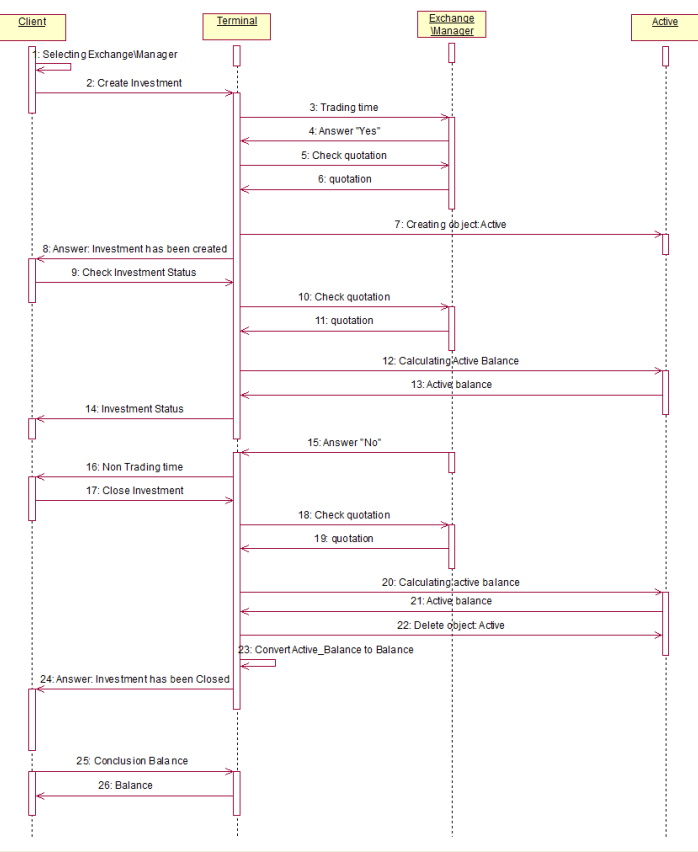


Рисунок 3.5 – Диаграмма последовательности

### **3.6 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 3.6:

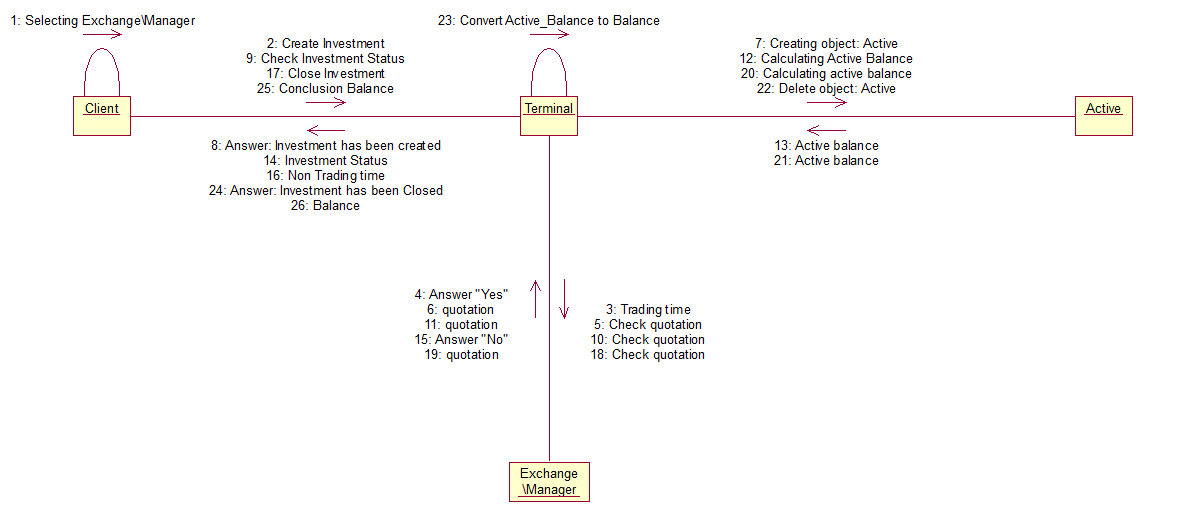


Рисунок 3.6 – Диаграмма коопераций

### **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.

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, 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.

Далее проектируется логическая структура системы с помощью диаграммы компонентов. На данном этапе выделяются компоненты исходного кода, формирующие структуру системы, а также компоненты, реализующие некий набор операций, способствующий достижению целей в рамках выбранного прецедента.

Данный вид диаграмм отражает следующие аспекты проектируемой системы:

* обмен сообщениями между объектами (в том числе в рамках обмена сообщениями со сторонними системами)
* ограничения, накладываемые на взаимодействие объектов
* события, инициирующие взаимодействия объектов.

В диаграмме представлена визуализация общей структуры исходного кода программной системы

Компонент «Actives module» - отвечает за отображение пользовательских активов (позиций), содежит в себе все полученные параметры при открытии позиции. Постоянно получает информацию от компонента «International data module» о текущих котировка на основании чего высчитывает и обновляет баланс пользователя в реальном времени. Также отвечат за закрытие пользовательской позиции.

Компонент «Client Module» - отвечает за создание пользователей, перевод пользователей в управляющих, поддержание актуального баланса пользователя. Важно, то что через компонент «International data module» постоянно обновляет баланс пользователя при инвестии в пользователя (если текущий пользователь - управляющий).

Компонент «Exchange module» - постоянно поддерживает список доступных рынков в открытом виде. Постоянно получает актуальную информацию о текущих котировках рынках от компонента «International Data Module». При получении неизвестной котировки, запрашивает информацию о ней и добавляет новый рынок.

Компонент «International Data Module» - постоянно высалает актуальную информацию другим компонентам для их корректной работы.

Компонент «Main» - это главный модуль рабочей системы.

Диаграмма компонентов представлена на рисунке 3.7:

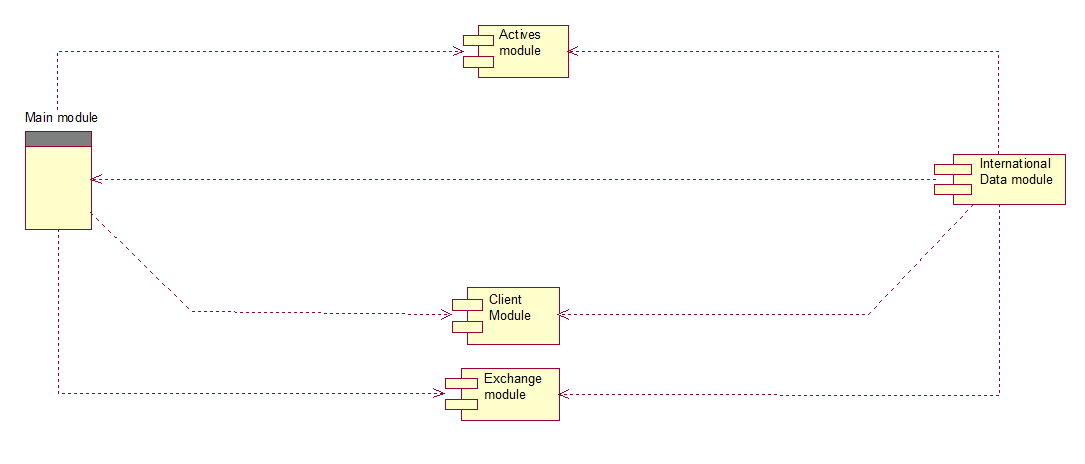


Рисунок 3.7 – Диаграмма компонентов

**4 ГЕНЕРАЦИЯ КОДА**

Одним из наиболее важных свойств программы IBM Rational Rose является возможность генерации программного кода на нескольких языках программирования, которая может быть использована разработчиком после построения модели.

Общая последовательность действий, которые необходимо выполнить для генерации программного кода в среде IBM Rational Rose, состоит из следующих этапов:

* проверка модели на отсутствие ошибок;
* создание компонентов для реализации классов;
* отображение классов на компоненты;
* выбор языка программирования для генерации текста программного кода;
* установка свойств генерации программного кода;
* выбор класса, компонента или пакета;
* генерация программного кода.

В качестве языка программирования был выбран язык Java.

Процесс генерации кода представлен на рисунках 4.1 – 4.4.

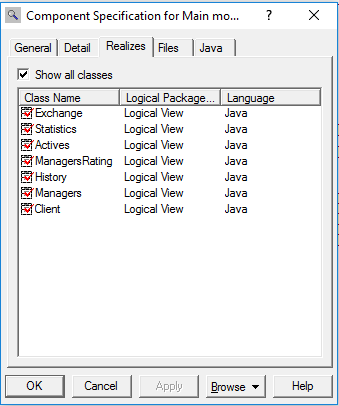


Рисунок 4.1 – Спецификация главного компонента программы

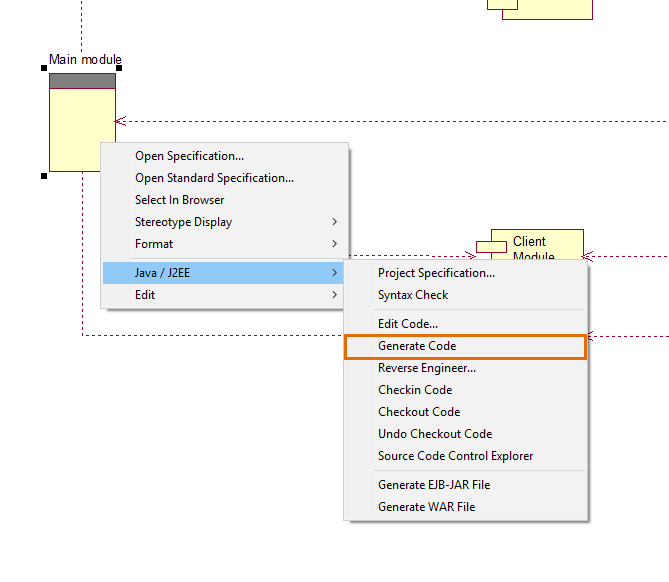


Рисунок 4.2 – Выбор языка программирования и начало процесса генерации

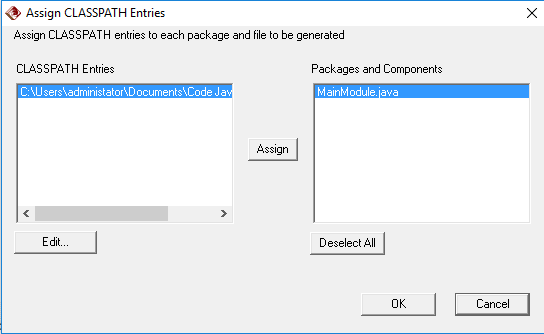


Рисунок 4.3 – Соотношение компонентов и мест их хранения на дисковом пространстве

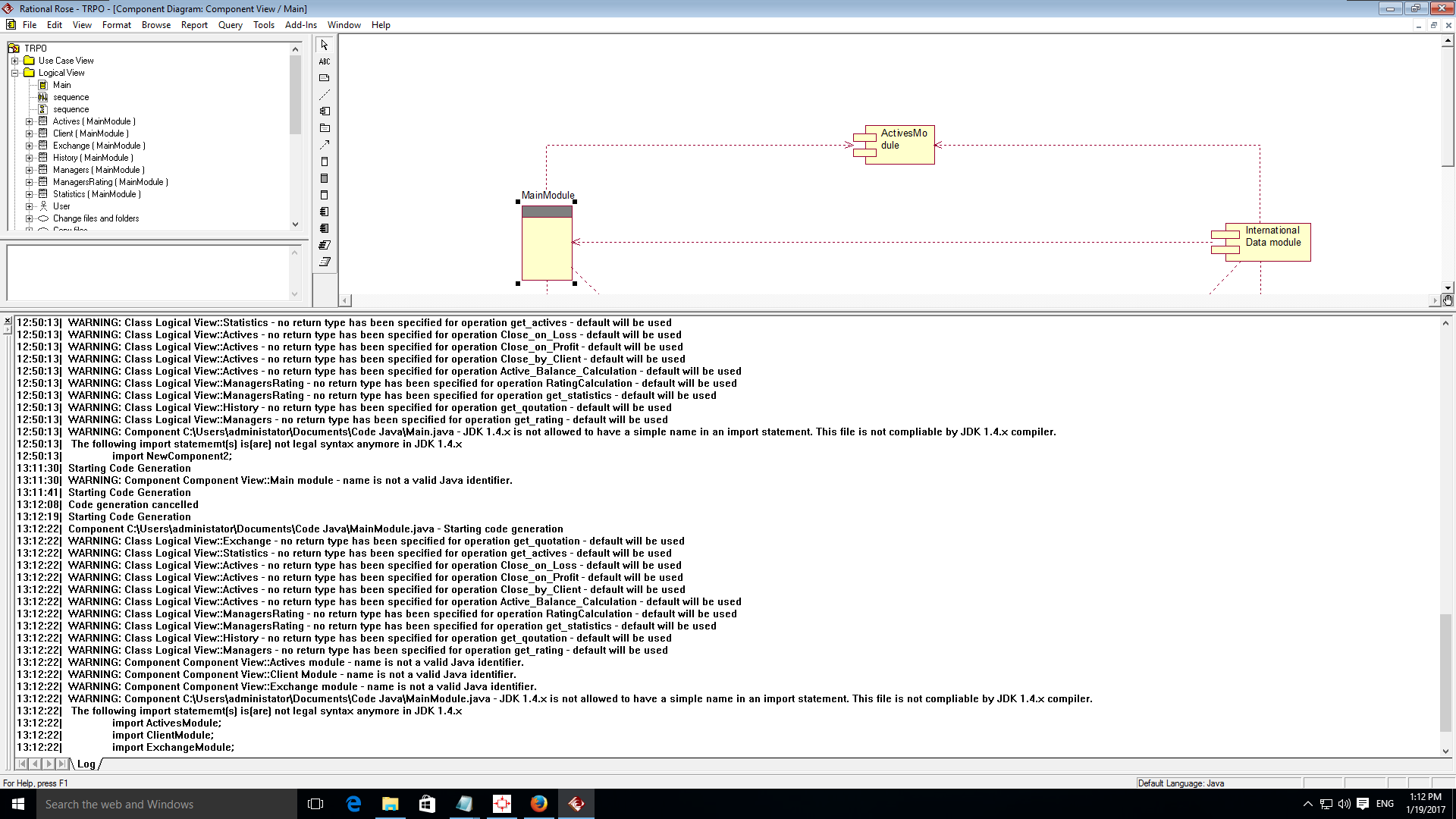


Рисунок 4.4 –Процесс генерации кода

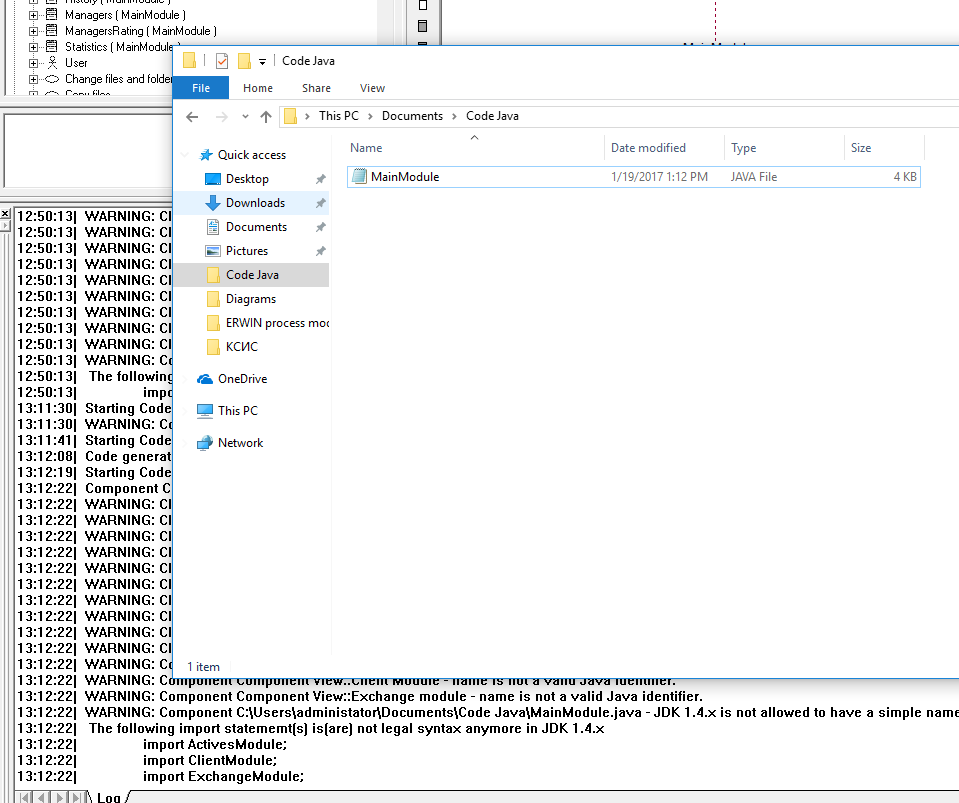


Рисунок 4.5 – Сгенерированный код

//Source file: C:\\Users\\administator\\Documents\\Code Java\\Main.java

import NewComponent2;

**Класс рынков:**

private class Exchange

{

private int Exchange\_id;

private string Exchange\_name;

private int History\_record\_id;

private long quotation;

private string Trading\_time;

/\*\*

\* @roseuid 587F64860176

\*/

public Exchange()

{

}

/\*\*

\* @roseuid 587F649103BB

\*/

public void get\_quotation()

{

}

}

**Класс активов:**

private class Statistics extends Actives

{

private int Statistic\_id;

private string Close\_active\_date;

private string Open\_active\_date;

/\*\*

\* @roseuid 587F6BCE004E

\*/

public Statistics()

{

}

/\*\*

\* @roseuid 587F6BE000C4

\*/

public void get\_actives()

{

}

}

private class Actives

{

private int Active\_id;

private int Client\_id;

private int Exchange\_id;

private long Active\_balance;

private long investment;

private double multiplier;

private long Limitation\_Loss;

private long Limitation\_Profit;

/\*\*

\* @roseuid 587E90AE009F

\*/

public Actives()

{

}

/\*\*

\* @roseuid 587E90BB0069

\*/

public void Close\_on\_Loss()

{

}

/\*\*

\* @roseuid 587E90C40258

\*/

public void Close\_on\_Profit()

{

}

/\*\*

\* @roseuid 587E90CE0278

\*/

public void Close\_by\_Client()

{

}

/\*\*

\* @roseuid 587E9132026D

\*/

public void Active\_Balance\_Calculation()

{

}

}

private class ManagersRating

{

private int Rating;

private int Investments\_count;

private int Investments\_profit\_count;

private long Profit;

private string Registration\_date;

/\*\*

\* @roseuid 587F680D0068

\*/

**Класс Рейтинга Менеджеров:**

public ManagersRating()

{

}

/\*\*

\* @roseuid 587F68220079

\*/

public void RatingCalculation()

{

}

/\*\*

\* @roseuid 587F693A02DA

\*/

public void get\_statistics()

{

}

}

**Класс истории:**

private class History extends Actives

{

private int History\_record\_id;

private string DateAndTime;

private int exchange\_id;

private long quotation;

/\*\*

\* @roseuid 587F65E00109

\*/

public History()

{

}

/\*\*

\* @roseuid 587F65F00289

\*/

public void get\_qoutation()

{

}

}

**Класс управляющих:**

private class Managers

{

private int Manager\_id;

private int Client\_id;

private int Rating;

/\*\*

\* @roseuid 587F666901BE

\*/

public Managers()

{

}

/\*\*

\* @roseuid 587F66740236

\*/

public void get\_rating()

{

}

}

**Класс клиентов:**

private class Client

{

private int Client\_id;

private string F\_name;

private string S\_name;

private long Balance;

private boolean manager;

private boolean demo;

/\*\*

\* @roseuid 587E915C0043

\*/

public Client()

{

}

}

**ЗАКЛЮЧЕНИЕ**

Предметной областью является программное средство «Информационная система кинотеатра». Разработанное программное средство оптимизирует и автоматизирует работу кинотеатра.

Контрольная работа состоит из следующих разделов:

1. Постановка задачи. В данном разделе была поставлена задача по разработке программного средства «Информационная система кинотеатра».

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

3. Описание моделей информационной системы с использованием языка UML.

4 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. Трофимов, С.А. CASE-технологии. Практическая работа в Rational Rose / С. А. Трофимов - Бином-Пресс 2002 . – 288 с.
11. Хабрахабр, UML – диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – 2013. – Режим доступа: http://habrahabr.ru/post/47940/ . – Дата доступа: 15.01.2017.
12. Сайтмонитор, Язык UML. Руководство пользователя [Электронный ресурс]. – 2013. – Режим доступа: http://sitemonitor.ru/doc/UML\_HTM/ . – Дата доступа: 15.01.2017.