Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по курсу:

«Системный анализ и проектирование информационных систем

(с модулем Разработка Веб-приложений)»

на тему:

**«**Программное средство для поиска наилучшей альтернативы на основе принципа Кондорсе

(на примере фермы)**»**

Студент Купреева С. И.

группа 872302

Руководитель Ассистент кафедры

ЭИ, Унучек Е. Н.

Минск 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc41468034)

[1 ОПИСАНИЕ ФЕРМЫ КРУПНОГО РОГАТОГО СКОТА 4](#_Toc41468035)

[3 ПОСТРОЕНИЕ БАЗОВОЙ АНАЛИТИЧЕСКОЙ МОДЕЛИ 12](#_Toc41468036)

[4 МОДЕЛИ ПРЕДСТАВЛЕНИЯ 15](#_Toc41468037)

[5 ОБОСНОВАНИЕ ВЫБОРА ПРОГРАММНЫХ СРЕДСТВ РАЗРАБОТКИ 18](#_Toc41468038)

[6 ОПИСАНИЕ СТРУКТУРЫ РАЗРАБАТЫВАЕМОГО СРЕДСТВА 19](#_Toc41468039)

[7 АЛГОРИТМ РАБОТЫ ПРОГРАММЫ 22](#_Toc41468040)

[8 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ 24](#_Toc41468041)

[9 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 26](#_Toc41468042)

[10 КОНТРОЛЬНЫЙ ПРИМЕР 38](#_Toc41468043)

[ЗАКЛЮЧЕНИЕ 40](#_Toc41468044)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 41](#_Toc41468045)

[ПРИЛОЖЕНИЕ А (Обязательное) Блок-схемы алгоритмов к разделу 7 42](#_Toc41468046)

[ПРИЛОЖЕНИЕ Б (Обязательное) Листинг кода 44](#_Toc41468047)

# ВВЕДЕНИЕ

Несмотря на то, что во многих странах мира фермерство уже не так популярно, как раньше, сегодня по-прежнему существуют функционирующие сельскохозяйственные фермы. Одни фермы специализируются на выращивании фруктов и овощей, другие выращивают домашний скот и дают сырьё для разного рода молочной продукции, третьи разводят рыбу.

Каждая ферма – организация, целью которой, как и любой другой организации, является получение прибыли. Соответственно от действий руководства фермы зависит и количество прибыли, и качество производимого продукта.

Но не всегда грамотное и верное решение, приносящее прибыль является очевидным и простым, особенно когда руководство – это не один единственный человек или не двое компаньонов, а больше десятка или двух десятков. Ведь у каждого из руководителей свое мнение о том или ином действии, каждый действует по-своему.

Возникает вопрос: как же выбрать из всех предложенных вариантов тот, который окажется самым эффективным и удовлетворять мнению большинства руководителей?

Как раз для этого и разрабатывается удобное приложение для оптимизации выбора наилучшей альтернативы из представленных методом Кондорсе.

Целью данного курсового проекта является разработка программы, позволяющей обеспечить увеличение прибыли фермы крупного рогатого скота и оптимизировать работу клиентов с данными.

Поставленная цель требует решения следующих задач:

– исследовать процесс работы фермы;

– определить функциональные требования к системе;

– разработать алгоритмы работы функций;

– выполнить программную реализацию;

– предусмотреть обработку исключительных ситуаций.

# ОПИСАНИЕ ФЕРМЫ КРУПНОГО РОГАТОГО СКОТА

1.1 Описание деятельности изучаемой предметной области

В прошлом семейные фермы обеспечивали доход своим владельцам и являлись источником продуктов питания для всей семьи. Несмотря на то, что частные фермы сегодня уже не так распространены, крупные коммерческие предприятия по выращиванию сельскохозяйственных зерновых культур можно найти практически в любой стране. Крупный фермерский бизнес может специализироваться как на выращивании какой-то одной культуры, так и на нескольких видах фруктов и овощей одновременно.

Наряду с фермами, выращивающими разного рода сельскохозяйственные культуры, существуют животноводческие фермы, занимающиеся разведением домашнего и крупного рогатого скота. Наиболее распространены свиноводческие фермы, а также фермы по разведению коров и коз. Обычно агрофирмы такого типа выращивают скот для его последующего забоя и разделки, обеспечивая рынок поставками свежего мяса.

Животноводческие фермы и комплексы состоят из нескольких разных по назначению и объему зданий, взаимное расположение которых обеспечивает производство продукции при минимальных затратах труда, средств и материалов и создает благоприятные условия для работы обслуживающего персонала. Различают животноводческие здания основных видов: коровники, телятники для молодняка крупного рогатого скота, свинарники- маточники, свинарники-откормочники и др. Они возводятся, в основном по типовым проектам. На практике применяют застройки следующих видов: отдельными зданиями (павильонная); зданиями, объединенными в блок (блокированная); укрупненными (моноблоками) и многоэтажными зданиями. Объединение под одной крышей производственных (подсобных и вспомогательных) помещений позволяет значительно снизить капитальные вложения по сравнению с павильонной застройкой за счет уменьшения площади ограждающих конструкций и сокращения протяженности инженерных и транспортных путей [2].

Животноводческая ферма - это подразделение сельскохозяйственного предприятия, в основных и вспомогательных постройках которого выращивают поголовье животных того или иного вида. По функциональным признакам животноводческие фермы и комплексы делят на основные производственные и обслуживающего назначения. К последним относят подсобные производственные, складские и вспомогательные помещения. Животноводческие фермы бывают племенные и товарные. Племенные фермы предназначены для улучшения существующих и выведения новых пород скота и птицы. На товарных фермах производят продукцию для народного потребления и удовлетворения нужд промышленности.

Животноводческий комплекс - это предприятие, предназначенное для равномерного круглогодового производства высококачественной продукции на основе применения промышленной технологии, научной организации труда, высокого уровня концентрации и специализации производства на базе комплексной механизации, автоматизации и поточной организации производственных процессов. По подчиненности комплексы бывают республиканского и местного значения; по форме собственности - государственные, государственно-колхозные, межколхозные, колхозные; по источникам поступления кормов - на привозных кормах из государственных ресурсов, на кормах собственного производства; по основной специализации - по производству молока, говядины, свинины, шерсти, баранины, яиц и мяса птицы и др.; по уровню специализации - с законченным технологическим циклом, специализированные на отдельных стадиях технологического цикла; по размерам - мелкие, средние, крупные. На фермах и комплексах применяют комплексную механизацию и частичную автоматизацию производственных процессов [3].

Животноводческие фермы как правило имеют разнообразную механизацию, которая уменьшает долю ручного труда при кормлении животных, их дойке, уборке навоза. На закате СССР, ввиду перекосов в экономике фермы в колхозах и совхозах механизировались слабо. Имевшаяся механизация использовалась неудовлетворительно, зачастую выходила из строя и подолгу не ремонтировалась. Промышленностью производились доильные аппараты, измельчители кормов, кормодробилки, корнерезки, соломосилосорезки, навозные транспортёры. По назначению фермы делятся на племенные и товарные: первые воспроизводят скот для улучшения стада вторых, которые целенаправленно выращивают и откармливают скот для производства мяса, молока, кожи, шерсти [4].

Разведение крупного рогатого скота требует, в первую очередь, наличия грамотно устроенного помещения для его содержания, где должны учитываться основные факторы, от которых зависит здоровье и продуктивность животных. Современный каркасно-тентовый коровник позволяет создать наилучшие условия для того, чтобы наши кормилицы чувствовали себя комфортно.

Сегодня широко используется метод «холодного содержания» телят на откорме. При таком способе выращивания животные становятся более закаленными, а риск инфекционных заболеваний сводится к минимуму. Освещение, воздухообмен и температурный режим позволяют молодняку получать природную энергию для здорового развития организма. Телята содержатся в таких помещениях до достижения шестимесячного возраста. Откорм производится по технологии беспривязного содержания. Каждый проект телятника индивидуален и учитывает природно-климатические особенности региона. Размеры должны определяться, в первую очередь, от количества голов [5].

Также для любой фермы важно соблюдать перечень требований, предписанных законодательством страны. Например, в Республике Беларусь существует закон Республики Беларусь от 2 июля 2010 г. № 161-З «О ветеринарной деятельности», в котором прописаны следующие пункты правил содержания дойных животных [6]:

* на молочно-товарные фермы могут поступать только здоровые животные, что должно быть подтверждено ветеринарным свидетельством, выданным в порядке, установленном Положением о порядке и условиях выдачи ветеринарных документов, утвержденным постановлением Совета Министров Республики Беларусь от 29 апреля 2017 г. № 319;
* комплектование молочно-товарных ферм крупного рогатого скота осуществляется обезроженными животными;
* перемещение животных, в том числе формирование отдельных групп, без осмотра специалиста в области ветеринарии этой сельскохозяйственной организации не допускается;
* корма для дойных животных должны быть качественными и безопасными в ветеринарно-санитарном отношении;
* дойные животные должны содержаться в чистоте;
* чистку кожного покрова и обмывание тазовых конечностей животных осуществляют по мере их загрязнения;
* при стойловом содержании животных организуют их моцион.

Или же следующие предписания по ветеринарно-санитарным мероприятиям по профилактике заболеваний крупного рогатого скота на племенных заводах (фермах), прописанные в постановлении министра сельского хозяйства и продовольствия Республики Беларусь от 22 декабря 2008 г. № 93 [7]:

* посещение специализированных племенных заводов (ферм), элеверов и госплемпредприятий разрешается при наличии письменного допуска Главного управления ветеринарии Минсельхозпрода Республики Беларусь;
* вход на территорию осуществляется через санитарный пропускник с обязательной заменой верхней одежды и обуви на спецодежду и спецобувь;
* специальные транспортные средства на племенные заводы (фермы), элеверы и госплемпредприятия пропускают только через дезбарьер;
* обслуживающий персонал животноводческих предприятий обеспечивается двумя комплектами санитарной одежды, смену, стирку и дезинфекцию которых необходимо проводить не реже одного раза в неделю.

1.2 Описание внутренних бизнес-процессов

Стратегическая карта подробно показывает цели организации и пути их достижения. Стратегическая карта фермы крупного рогатого скота представлена на рисунке 1.1.

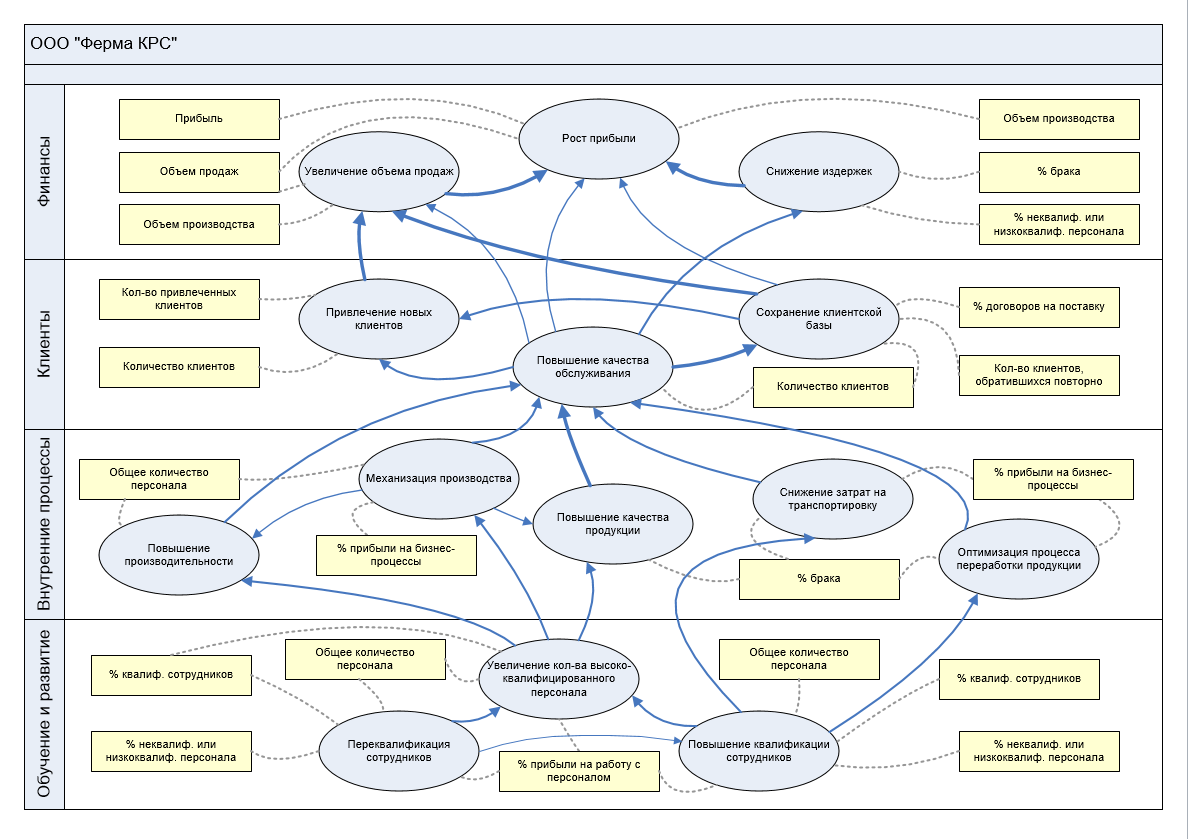


Рисунок 1.1 – Стратегическая карта

IDEF0 — методология функционального моделирования.

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

Методология IDEF0 предписывает построение иерархической системы диаграмм — единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция — система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

На рисунках 1.2, 1.3, 1.4, 1.5 представлена функциональная модель IDEF0.

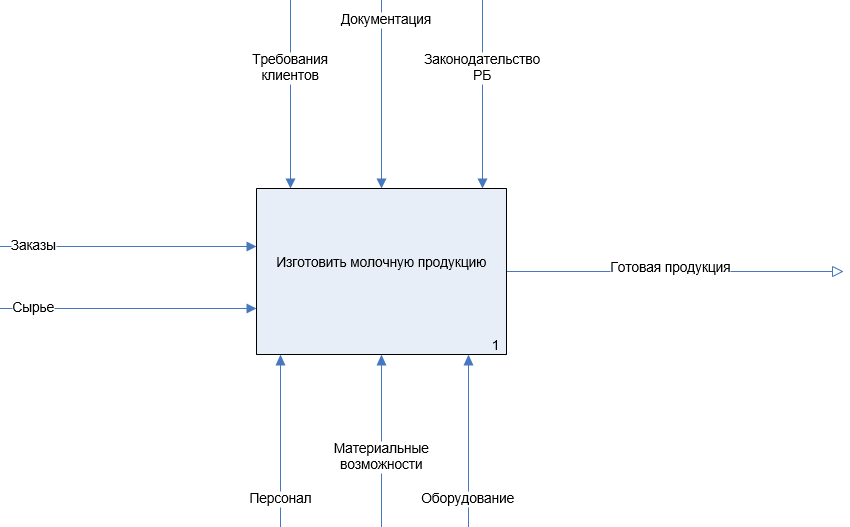


Рисунок 1.2 – Контекстная диаграмма верхнего уровня

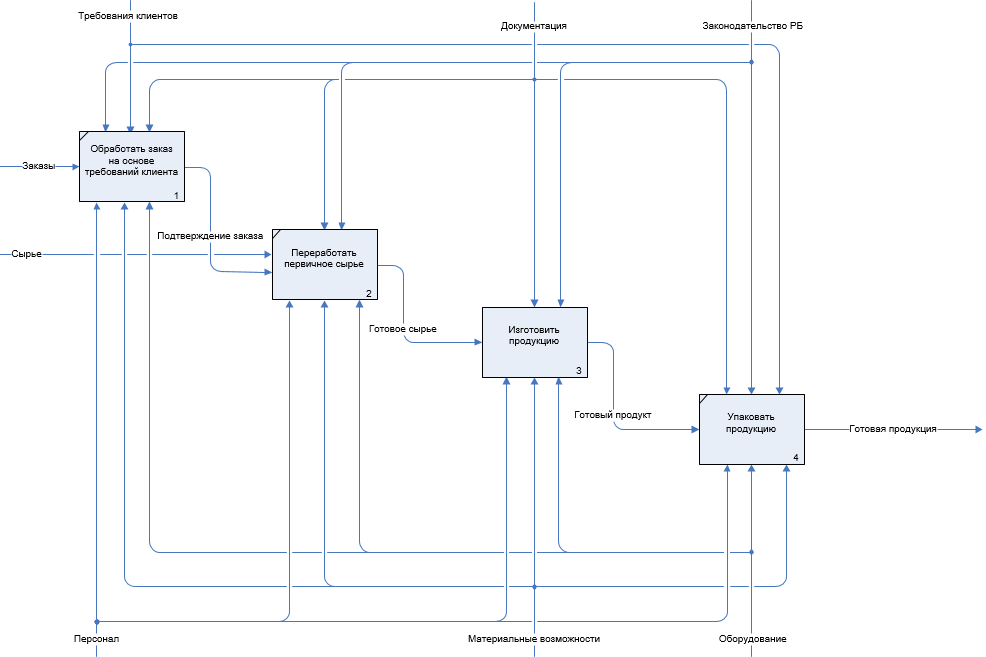


Рисунок 1.3 – Декомпозиция контекстной диаграммы верхнего уровня

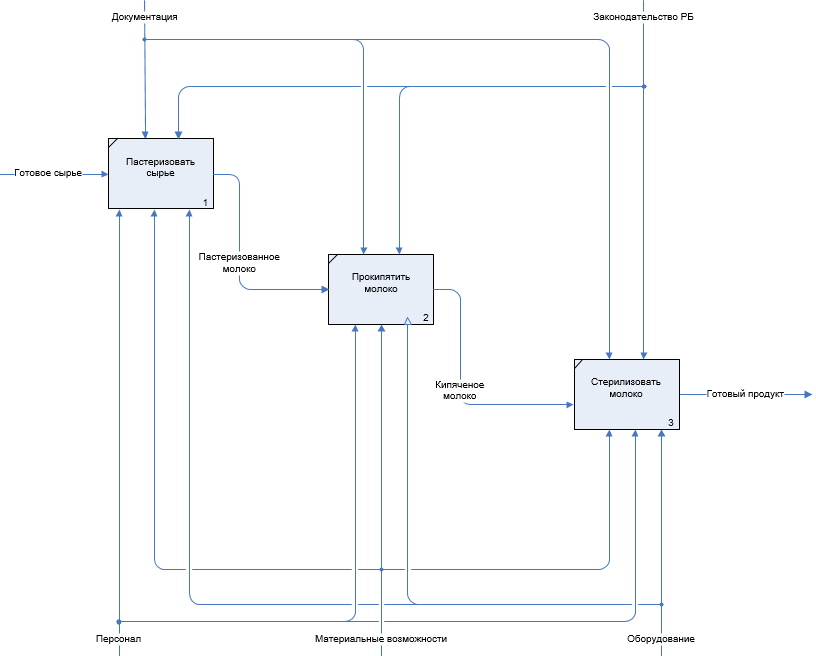


Рисунок 1.4 – Декомпозиция контекстной диаграммы второго уровня

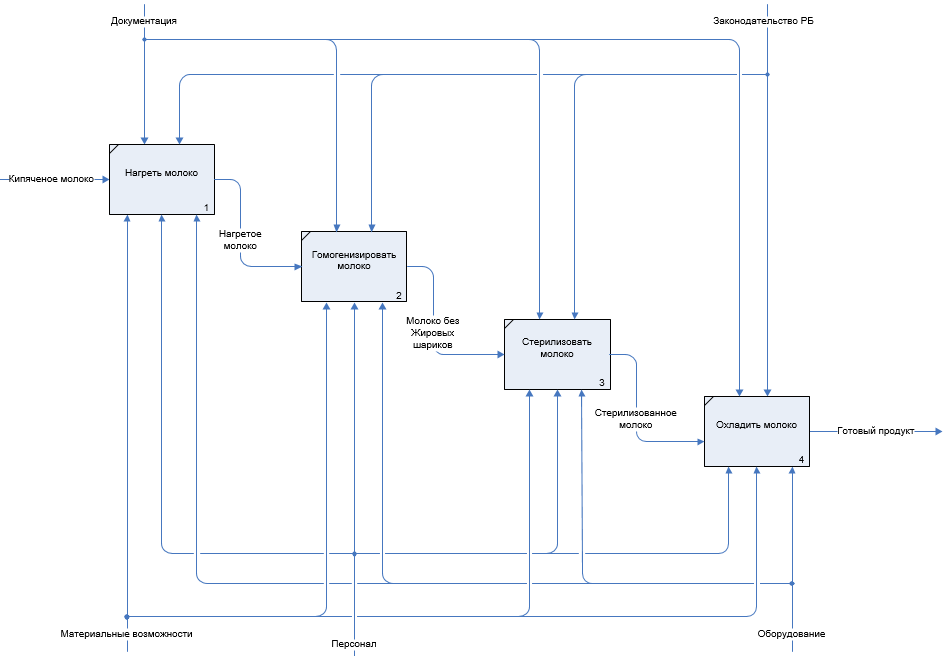


Рисунок 1.5 – Декомпозиция контекстной диаграммы третьего уровня

1. ПОСТАНОВКА ЗАДАЧИ НА РАЗРАБОТКУ ПРОГРАММНОГО СРЕДСТВА

Разрабатываемая система должна автоматизировать процессы, которые могут упростить работу пользователю во время поиска наилучшей альтернативы из множества предложенных. Кроме того, программа помогает руководителям фермы отслеживать объемы производства продукции, прослеживать состав поголовья скота, просчитать примерную стоимость фермы и благодаря этому руководители смогут увеличить прибыль, приносимую фермой. Также программа должна иметь удобный для пользователя интерфейс, который будет оптимизирован для работы с любыми исходными данными, и защищена от ошибок.

Для обеспечения успешного выполнения поставленной задачи, система должна предоставлять ряд возможностей для работы администратора: ввод необходимых данных, их редактирование, удаление, вывод на экран, также предоставление отчётной информации о проделанной работе с данными и управление пользователями.

Пользователи (гости) в свою очередь должны иметь возможность просмотра информации о продукции фермы, работы с ней, также возможности просмотра состава поголовья скота, имеющегося на предприятии, и отчета о расчете стоимости фермы с учетом состава скота и общей площади организации.

# ПОСТРОЕНИЕ БАЗОВОЙ АНАЛИТИЧЕСКОЙ МОДЕЛИ

Системный анализ используют для выяснения причин существующих сложностей, постановки целей, выработки методов и вариантов устранения проблем. Он выступает в роли организатора и координатора. Опирается на междисциплинарный подход, с помощью которого эффективно объединяет и концентрирует усилия группы специалистов на решении конкретной проблемы. Системное объединение достижений различных областей знаний, позволяет решать такие проблемы, которые не могут быть разрешены в рамках отдельных дисциплин и частных подходов.

Иногда выбор наилучшей альтернативы – сложная и запутанная процедура. При этом невозможно предугадать последствия принятия того или иного решения, особенно когда выбирает не один человек, а несколько. В такие моменты одним из лучших вариантов принятия решения является теория общественного выбора.

Теория общественного выбора представляет экономический подход к изучению нерыночного принятия решений, т.е. дополняет теорию рыночного обмена соответствующей теорией функционирования политических рынков. Принадлежность теории общественного выбора к экономической науке определяется тем, что она опирается на строгие экономические допущения о человеке как рациональном и максимизирующем полезность субъекте. Индивид соглашается отдать исключительную власть, поскольку считает, что она будет служить их личным интересам, для чего принимают определённый набор правил – конституцию [8].

К теории общественного выбора относится такое явление, как «Парадокс Кондорсе» – возможная не транзитивность (противоречивость) коллективного выбора избирателей при транзитивности выбора каждого избирателя. Таким образом, волеизъявления разных групп избирателей, каждая из которых представляет большинство, могут вступать в парадоксальное противоречие друг с другом. Парадокс возможен в случае трёх и более вариантов выбора [9].

Также Кондорсе определил правило, по которому сравнение выбираемых альтернатив (кандидатов) производится с учётом полной ординалистской информации о предпочтениях избирателей.

Согласно принципу Кондорсе, для определения истинной воли большинства необходимо, чтобы каждый голосующий проранжировал всех кандидатов в порядке их предпочтения. После этого для каждой пары кандидатов определяется, сколько голосующих предпочитает одного кандидата другому — формируется полная матрица попарных предпочтений избирателей.

Для заданной таблицы результатов голосования (таблицы предпочтений) победителем по Кондорсе называется кандидат, который побеждает любого другого кандидата при парном сравнении по правилу большинства. Если парные сравнения образуют цикл, то победителя по Кондорсе нет, и говорят, что имеет место так называемый парадокс Кондорсе.

Политологи считали парадокс Кондорсе редким явлением. Однако, результаты математического моделирования голосования по методу Кондорсе, приведенные в таблице 3.1, показывают, что это не так [1].

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число |  |  | Число избирателей | | | |  |  |
|  |  |  |  |  |  |  |  |  |
| кандидатов | 3 | 5 |  | 7 | 9 |  | 11 | 1 |
| 3 | 0.050 | 0.069 |  | 0.075 | 0.078 |  | 0.080 | 0.088 |
|  |  |  |  |  |  |  |  |  |
| 4 | 0.111 | 0.139 |  | 0.150 | 0.156 |  | 0.160 | 0.176 |
|  |  |  |  |  |  |  |  |  |
| 5 | 0.160 | 0.200 |  | 0.215 | 0.230 |  | 0.251 | 0.251 |
|  |  |  |  |  |  |  |  |  |
| 6 | 0.202 | 0.255 |  | 0.258 | 0.284 |  | 0.294 | 0.315 |
|  |  |  |  |  |  |  |  |  |
| 7 | 0.239 | 0.299 |  | 0.305 | 0.342 |  | 0.343 | 0.369 |
|  |  |  |  |  |  |  |  |  |

Таблица 3.1 – Пример вероятности реализации парадокса Кондорсе при соответствующем количестве избирателей и кандидатов

На базе этой матрицы, используя транзитивность отношения предпочтения, можно попытаться построить коллективную ранжировку кандидатов.

Приведём численный пример из работы Кондорсе: введём для краткости обозначение: A≻B≻C будет означать, что голосующий предпочитает кандидата A кандидату B, а кандидата B — кандидату С.

Пусть 60 голосующих дали следующие предпочтения:

23 человека: A≻C≻B

19 человек: B≻C≻A

16 человек: C≻B≻A

2 человека: C≻A≻B

При сравнении A с B имеем: 23 + 2 = 25 человек за то, что A≻B, и 19 + 16 = 35 человек за то, что B≻A.

По принципу Кондорсе мнение большинства состоит в том, что В лучше А. Сравнивая А и С, будем иметь: 23 человека за A≻C и 37 человек за C≻A. Отсюда, по Кондорсе, заключаем, что большинство предпочитает кандидата С кандидату А. Аналогично (19 человек за B≻C, 41 человек за C≻B) С более предпочтителен, чем B.

Таким образом, по Кондорсе воля большинства выражается в виде трех суждений: C≻B; B≻A; C≻A, которые можно объединить в одно отношение предпочтения C > B > A и если необходимо выбрать одного из кандидатов, то, согласно принципу Кондорсе, следует предпочесть кандидата С [10].

Рассмотрим еще один пример более подробно, базируясь на результатах частных ранжированиях альтернатив: a1, a2, a3, a4, a5.

1. Эксперты осуществляют ранжирование альтернатив:

Э1 = (a1, a3, a2, a5, a4)

Э2 = (a1, a2, a4, a3, a5)

Э3 = (a1, a2, a5, a3, a4)

Э4 = (a2, a3, a1, a5, a4)

Э5 = (a2, a4, a3, a1, a5)

1. Находятся оценки mik, характеризующих предпочтение альтернатив в парных предпочтениях

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **mik** | **a1** | **a2** | **a3** | **a4** | **a5** |
| **a1** |  | 3 | 3 | 4 | 5 |
| **a2** | 2 |  | 4 | 5 | 5 |
| **a3** | 2 | 1 |  | 3 | 4 |
| **a4** | 1 | 0 | 2 |  | 2 |
| **a5** | 0 | 0 | 1 | 3 |  |

1. Выполняются проверки согласно принципу Кондорсе: наилучшей является альтернатива ai, если mik ≥ mki для всех k ≠ i

К = 4; m14≥m41; 4>1 — выполняется, т.е. правилу Кондорсе удовлетворяет только альтернатива a1.

1. Выбирается альтернатива Кондорсе. Это a1.

# МОДЕЛИ ПРЕДСТАВЛЕНИЯ

4.1 Диаграмма вариантов использования

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом, актором) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

Диаграмма вариантов использования данного проекта представлена на рисунке 4.1.

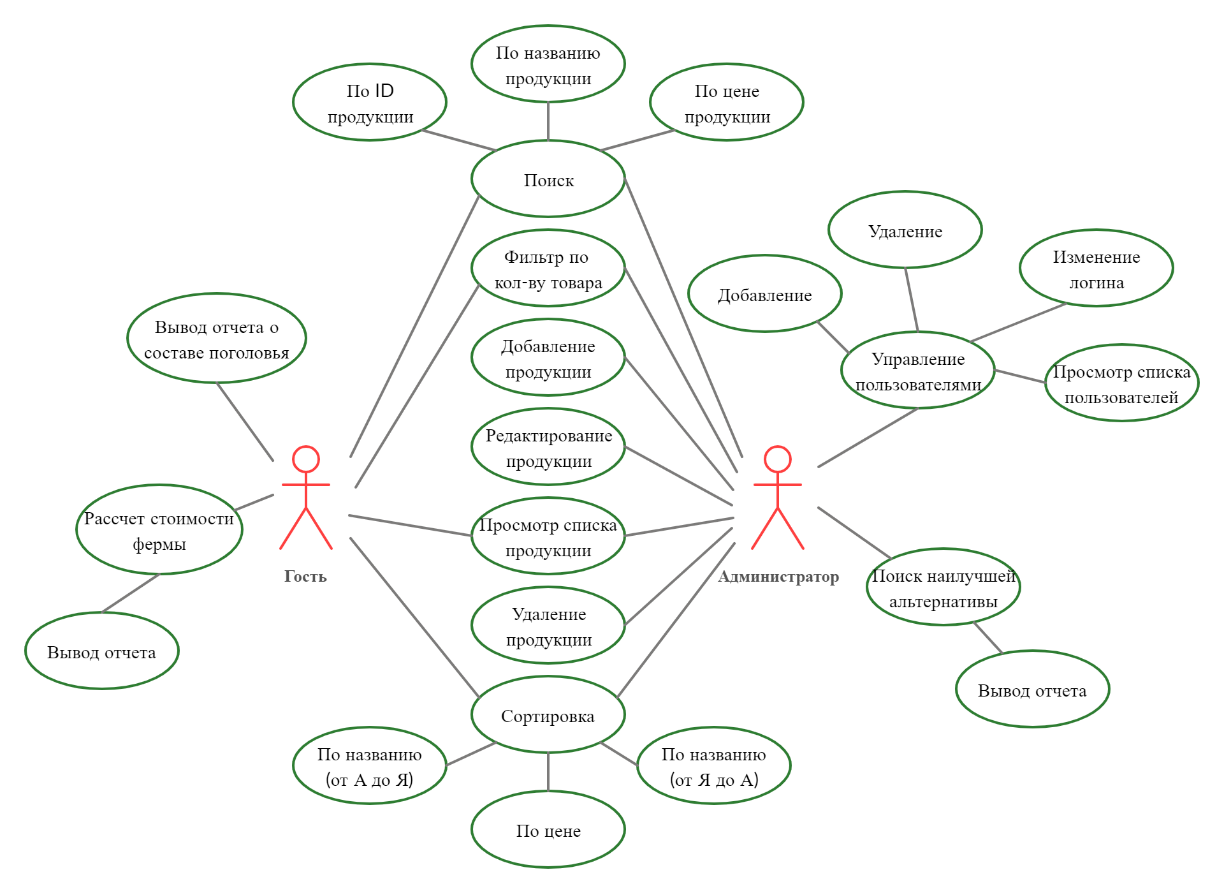


Рисунок 4.1 – Диаграмма вариантов использования

4.2 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Так как приложение является клиент-серверным, в нем две диаграммы классов для серверной и клиентской части, которые представлены на рисунках 4.2 и 4.3 соответственно.

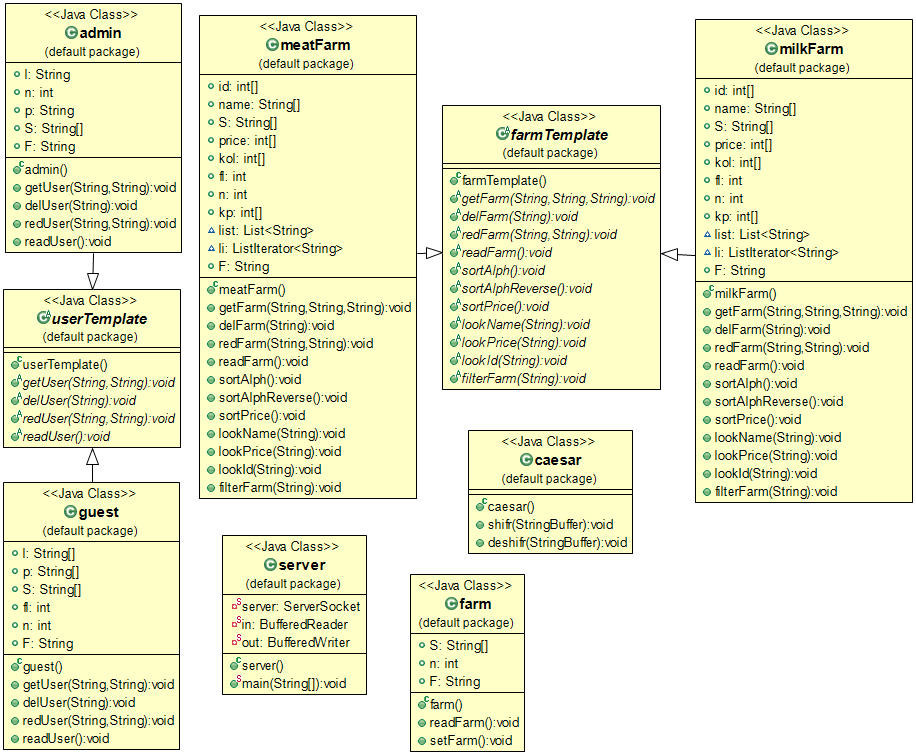


Рисунок 4.2 – Диаграмма классов серверной части программы

Серверная часть состоит из 9 классов, 2 из которых – абстрактные. В самом классе server происходит подключение, обработка информации. Класс farmTemplate и его наследники – milkFarm и meatFarm – отвечают за работу с молочным и мясным отделами фермы соответственно. userTemplate и его наследники – guest и admin – осуществляют работу с пользователем (гостем) и администратором соответственно. Caesar – класс для шифровки и дешифровки информации шифром Цезаря с ключом 3. Класс farm работает с общей информацией о ферме.

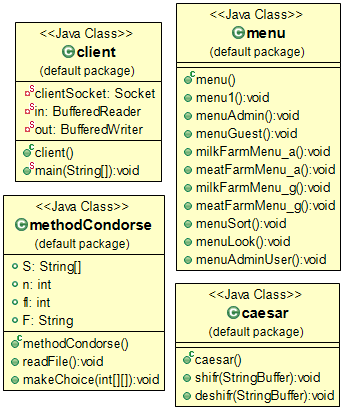


Рисунок 4.3 – Диаграмма классов клиентской части программы

В классе client происходит подключение к серверу и работа с сокетами, также в нем осуществляется ввод данных и вывод на экран результатов действий серверной части. Класс menu предназначен для вывода всех меню программы на экран. В классе methodCondorse происходит вся работа по поиску наилучшей альтернативы методом Кондорсе. Класс caesar предназначен для шифровки и дешифровки данных шифром Цезаря с ключом 3.

# ОБОСНОВАНИЕ ВЫБОРА ПРОГРАММНЫХ СРЕДСТВ РАЗРАБОТКИ

В рамках курсового проекта было принято решение использовать такие инструменты как среда разработки IntelliJ IDEA Community Edition 2020.1.1 и язык программирования Java SE Development Kit 13.

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Kotlin, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. Программа содержит полный набор необходимых для создания полноценных приложений компонент: редактор, среда компиляции и выполнения, а также отладчик.

Данная программа помогает специалисту экономить время вследствие глубокого анализа контекста и удаления неподходящих вариантов. Эта и другие детали обеспечивают повышение уровня продуктивности пользователя, одновременно позволяя ему получать больше удовольствия от деятельности.

Java — это достаточно универсальный язык программирования, который часто используется для веб-разработки и для разработки под Android.

Также Java — это объектно-ориентированный язык программирования.

Программы на Java транслируются в байт-код, который затем выполняется виртуальной машиной Java (JVM). JVM — это программа, которая обрабатывает байтовый код и передает инструкции оборудованию как интерпретатор. Достоинством подобной реализации является независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует JVM.

# ОПИСАНИЕ СТРУКТУРЫ РАЗРАБАТЫВАЕМОГО СРЕДСТВА

В данном курсовом проекте все данные хранятся в соответствующих текстовых файлах, к которым есть доступ только у сервера. В проекте используются следующие текстовые файлы:

1. Admin – здесь хранятся логины и пароли администраторов в зашифрованном виде (Рисунок 6.1).

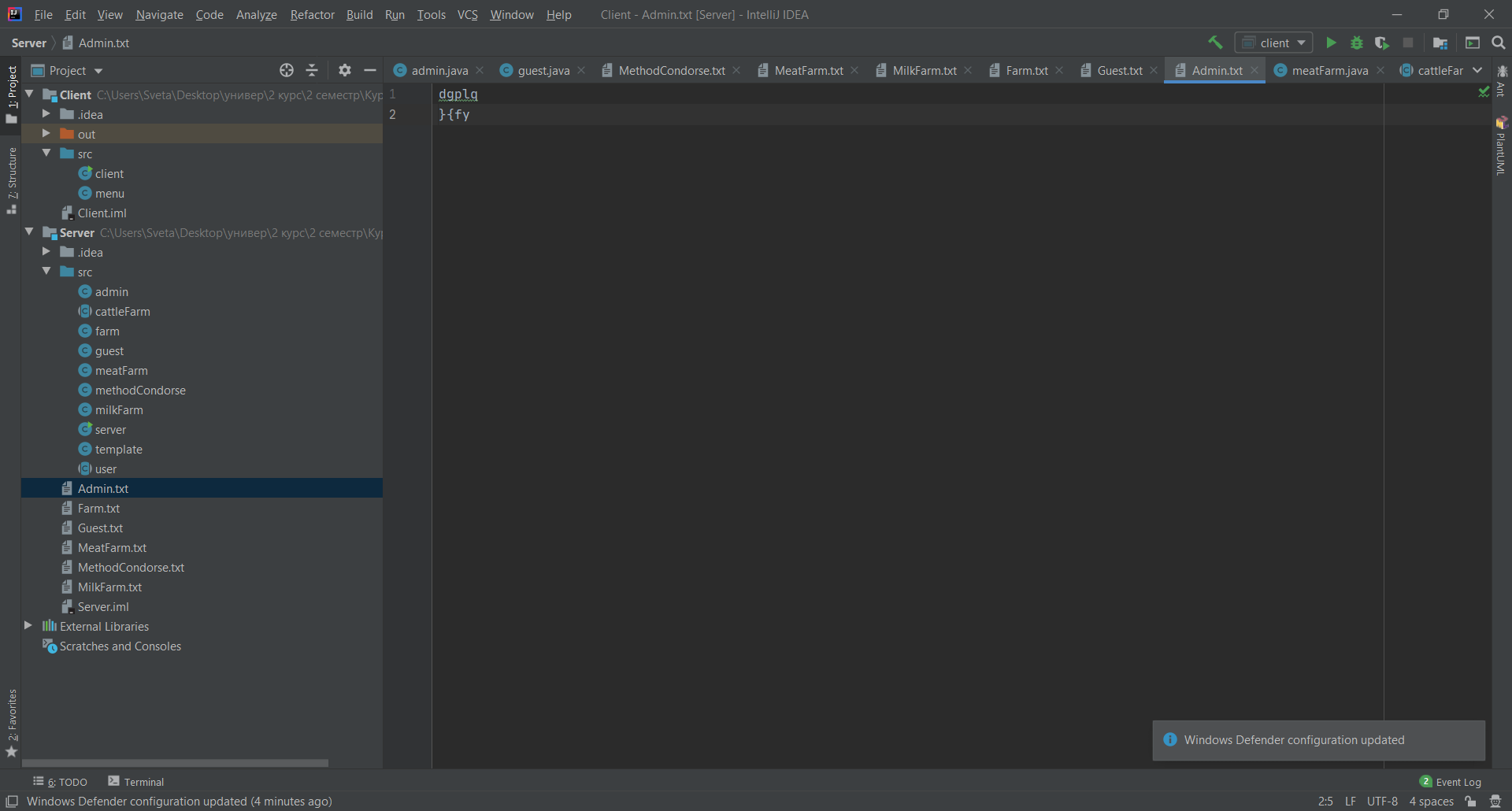


Рисунок 6.1 – Содержимое файла Admin.txt

Данные, имеющиеся в файле:

1. логин администратора;
2. пароль администратора.
3. Guest – здесь хранятся логины и пароли пользователей (гостей) в зашифрованном виде (Рисунок 6.2).

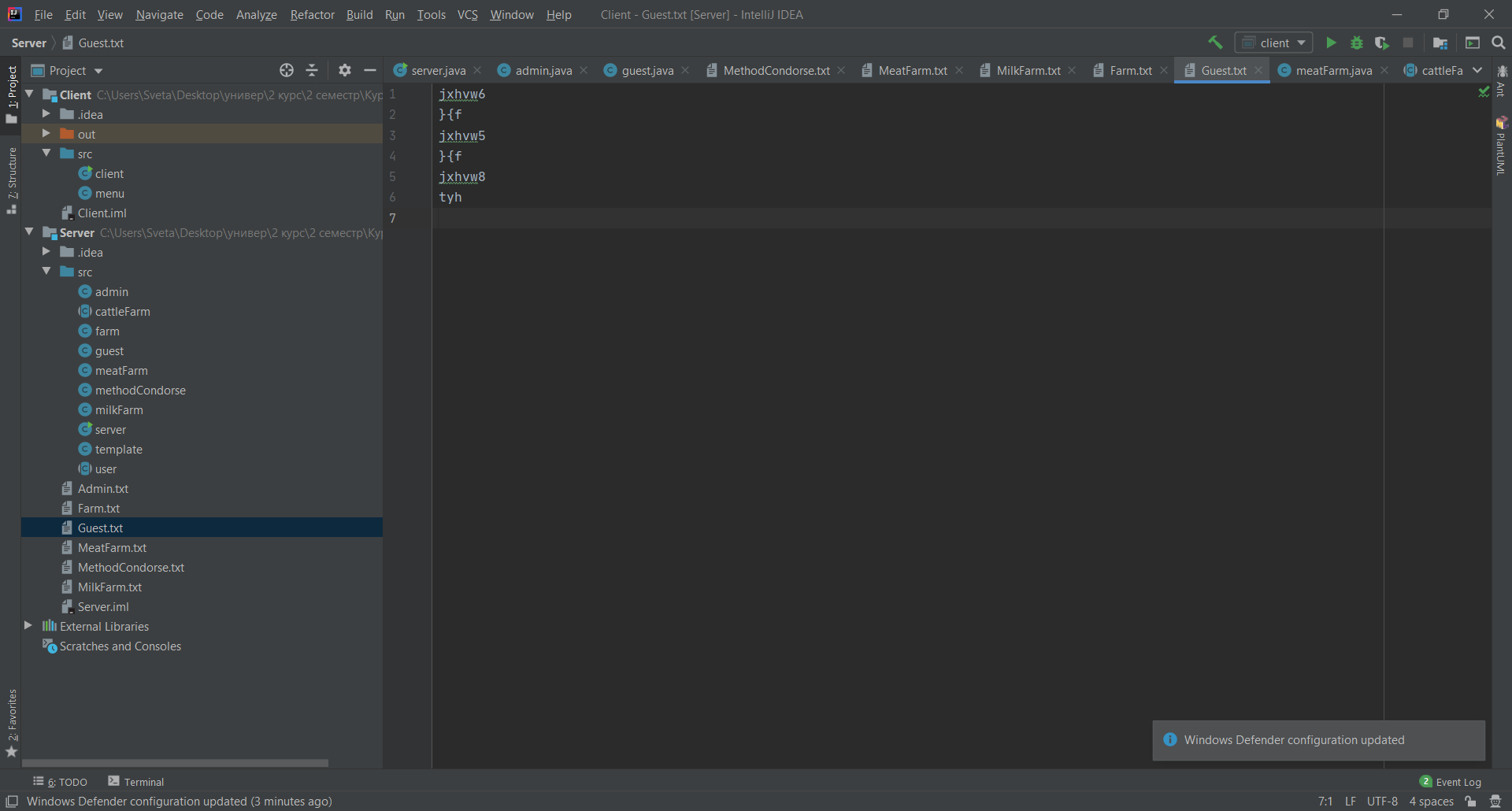


Рисунок 6.2 – Содержимое файла Guest.txt

Данные, имеющиеся в файле:

1. логин пользователя (гостя);
2. пароль пользователя (гостя).
3. Farm – здесь хранится полная информация о поголовье скота в виде таблицы и данные для примерного расчета стоимости фермы (Рисунок 6.3).

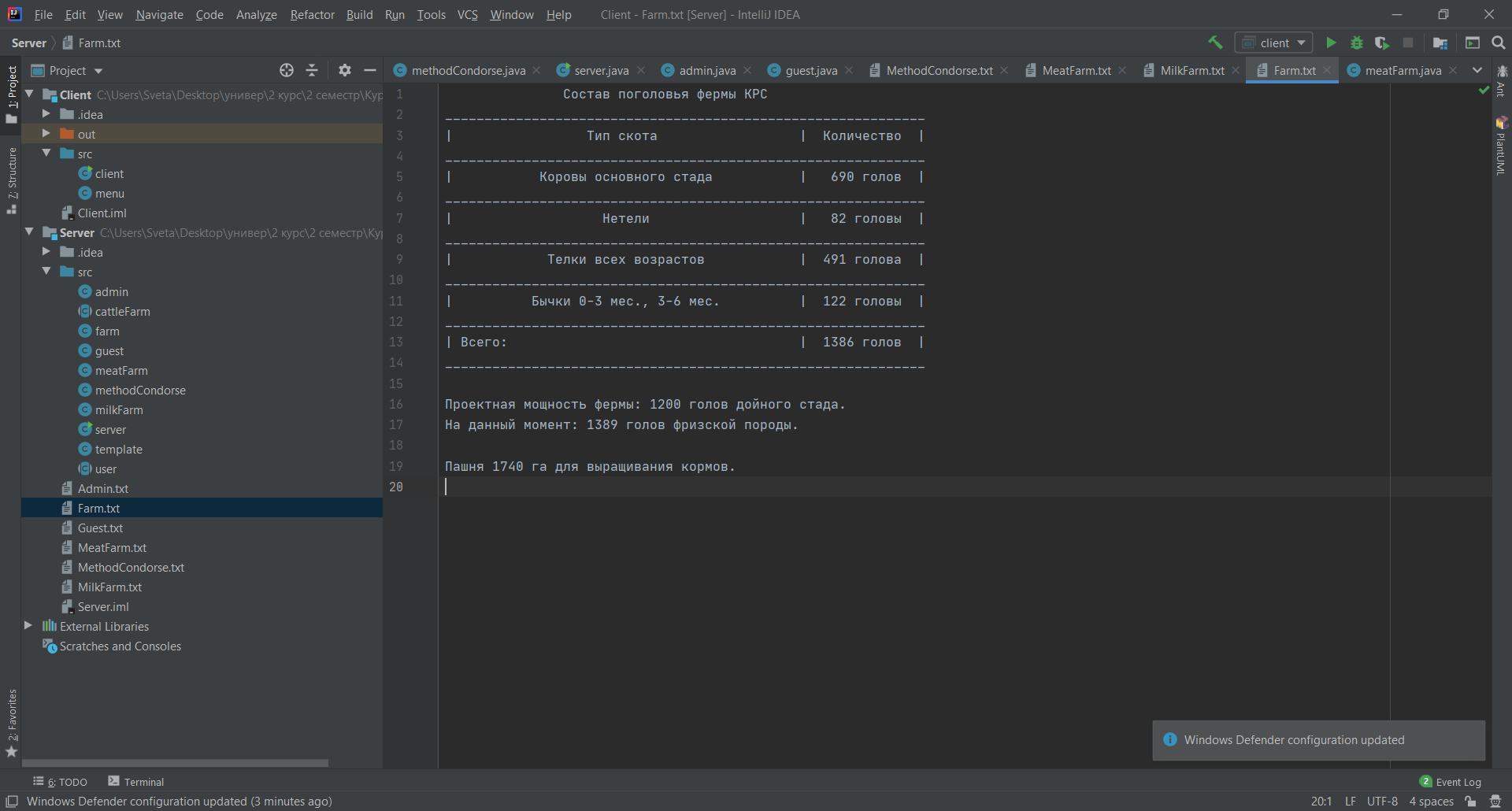


Рисунок 6.3 – Содержимое файла Farm.txt

Данные, имеющиеся в файле:

1. таблица «Состав поголовья скота»;
2. дополнительная информация о составе фермы.
3. MilkFarm – здесь хранится вся информация о продукции молочного отдела фермы (Рисунок 6.4).

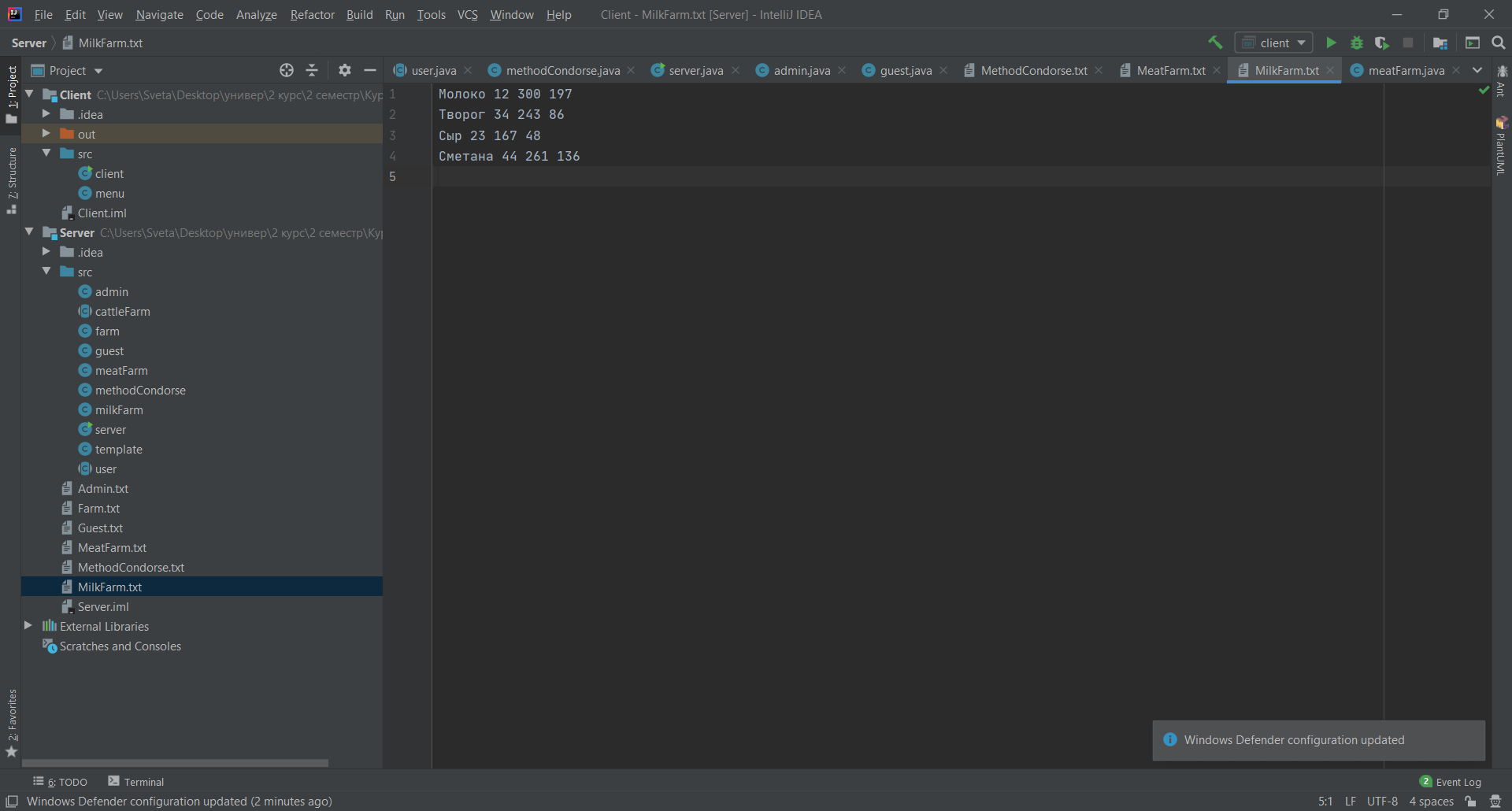


Рисунок 6.4 – Содержимое файла MilkFarm.txt

Данные, имеющиеся в файле:

1. название товара;
2. цена товара;
3. количество товара на складе;
4. ID товара.
5. MeatFarm – здесь хранится вся информация о продукции мясного отдела фермы (Рисунок 6.5).

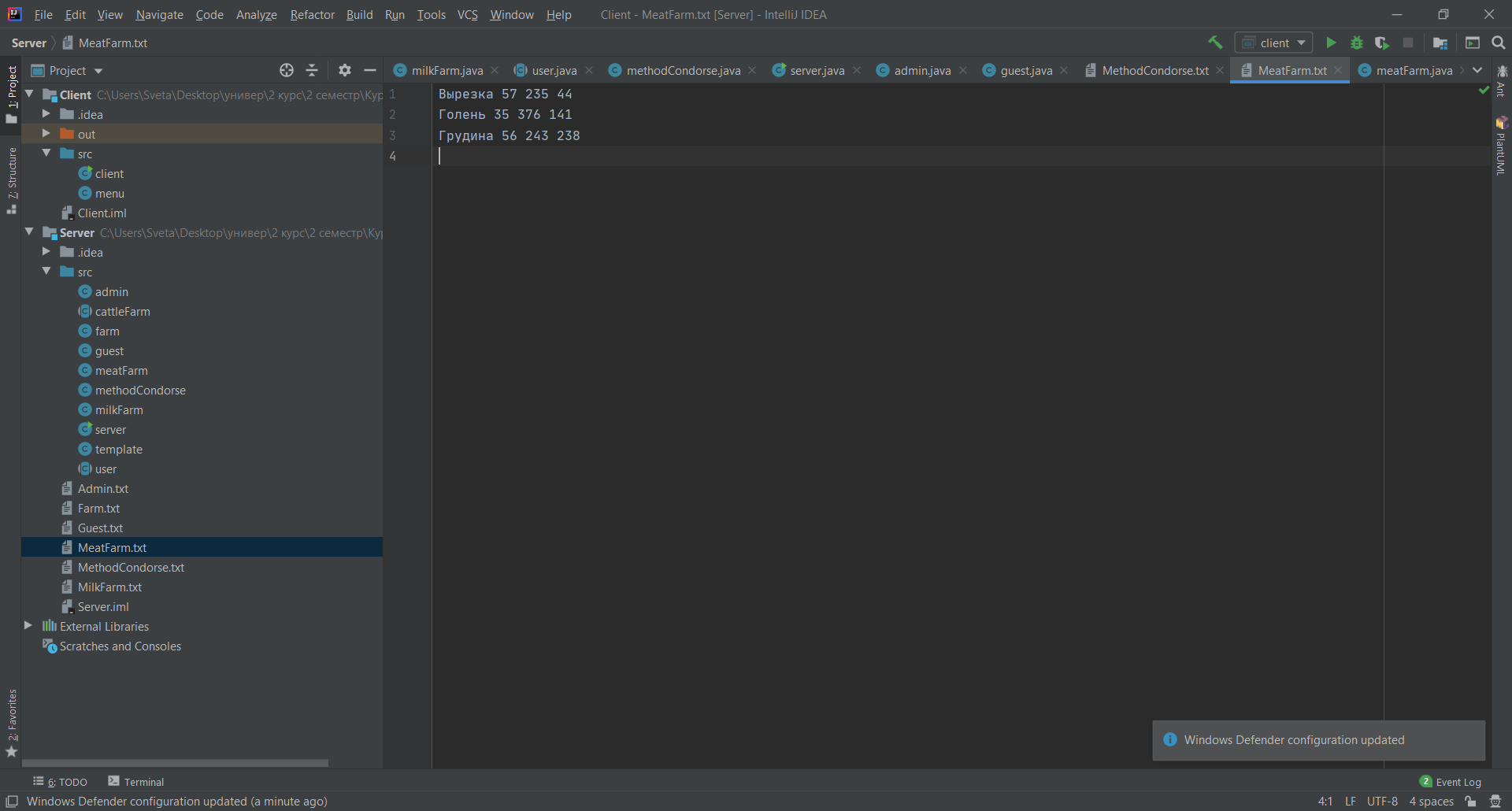


Рисунок 6.5 – Содержимое файла MeatFarm.txt

Данные, имеющиеся в файле:

1. название товара;
2. цена товара;
3. количество товара на складе;
4. ID товара.
5. MethodCondorse – здесь хранится информация об альтернативах, матрица ранжирования предпочтений альтернатив и данные для последующего поиска наилучшей альтернативы методом Кондорсе (Рисунок 6.6).

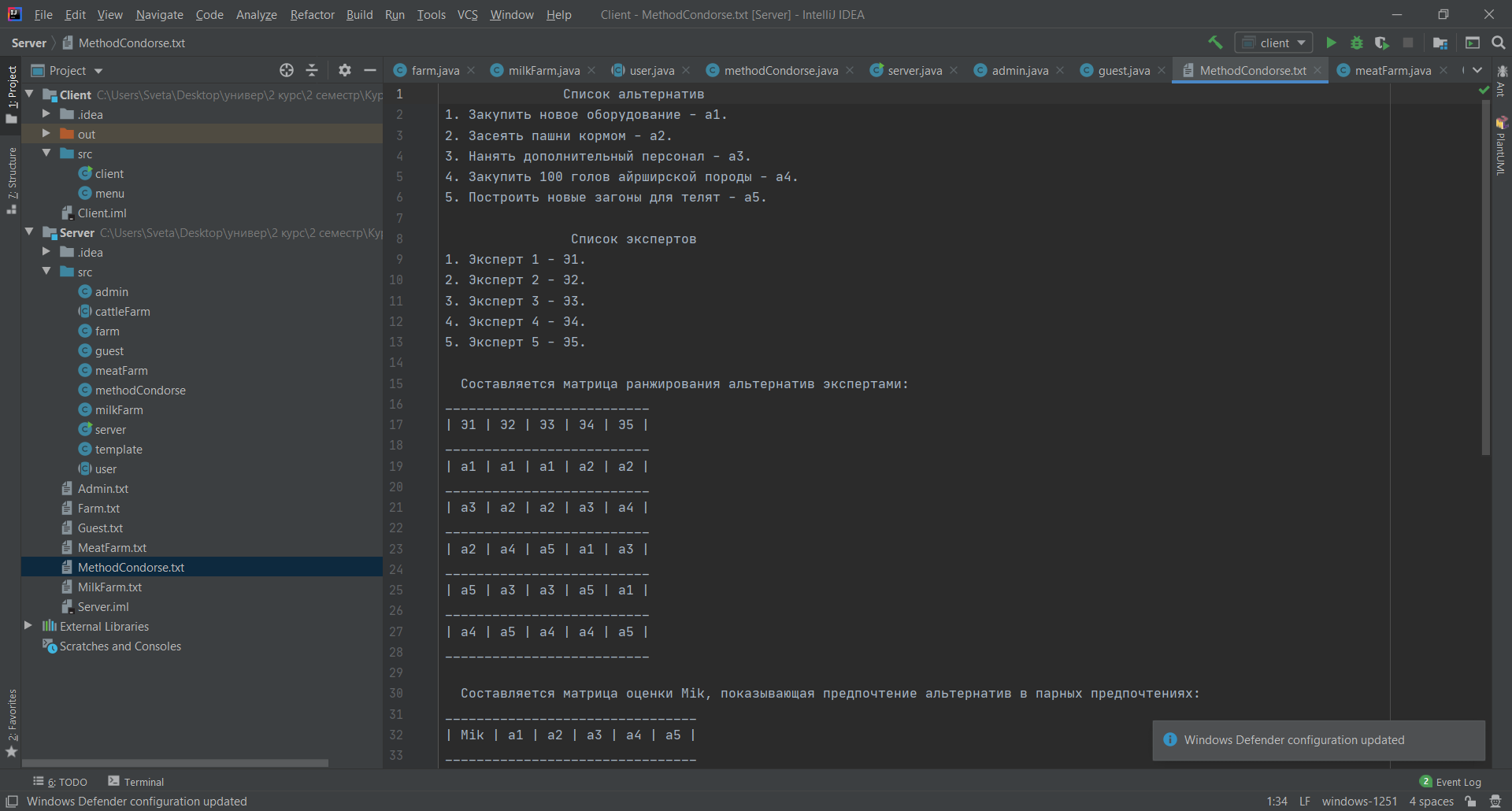


Рисунок 6.6 – Содержимое файла MethodCondorse.txt

Данные, имеющиеся в файле:

1. список альтернатив;
2. список экспертов;
3. матрица распределения предпочтений.

Для записи и считывания информации из файлов используются методы классов FileWriter и FileReader соответственно, хранящиеся в библиотеке java.io\*.

# АЛГОРИТМ РАБОТЫ ПРОГРАММЫ

Задачей данного проекта является поиск наилучшей альтернативы методом Кондорсе. Метод Кондорсе предполагает использование мнений независимых экспертов, выполняющих оценку списка альтернатив по их предпочтительности. Необходимо заполнить матрицу предпочтений, на основании этой матрицы вычисляется альтернатива, имеющая наибольшее количество голосов среди экспертов. Алгоритм представлен на рисунке А.1.

Чтобы клиент мог взаимодействовать с сервером, он должен успешно подключиться к серверу. Алгоритм подключения показан на рисунке А.2.

Пример кода Java:

import java.io.\*;  
import java.net.Socket;  
import java.util.Scanner;  
  
public class client {  
 private static Socket *clientSocket*;  
 private static BufferedReader *in*;  
 private static BufferedWriter *out*;  
  
 public static void main(String[] args) {  
 menu m = new menu();  
 int k = 3;  
 Scanner scan = new Scanner(System.*in*);  
 try{  
 try {  
 *clientSocket* = new Socket("localhost", 4004);  
 *out* = new BufferedWriter(new OutputStreamWriter(*clientSocket*.getOutputStream()));  
 *in* = new BufferedReader(new InputStreamReader(*clientSocket*.getInputStream()));  
  
 System.*out*.println("Подключение к серверу закончено (успешно).");  
 System.*out*.println("\nЗдравствуйте.\n");  
 }finally {  
 System.*out*.println("Работа Клиента завершена.");  
 *clientSocket*.close();  
 *in*.close();  
 *out*.close();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

# РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ

Приложение написано на языке программирования Java. Для его установки на свой ПК нужно установить пакет языка Java версии 13.02. Для этого требуется перейти на официальный сайт компании Oracle, перейти в раздел «Скачать Java SE13» (или выше), выбрать нужную ОС и скачать архив (Рисунок 8.1). После чего извлечь файлы из архива и установить Java SE13 (или выше) на свой ПК [11].

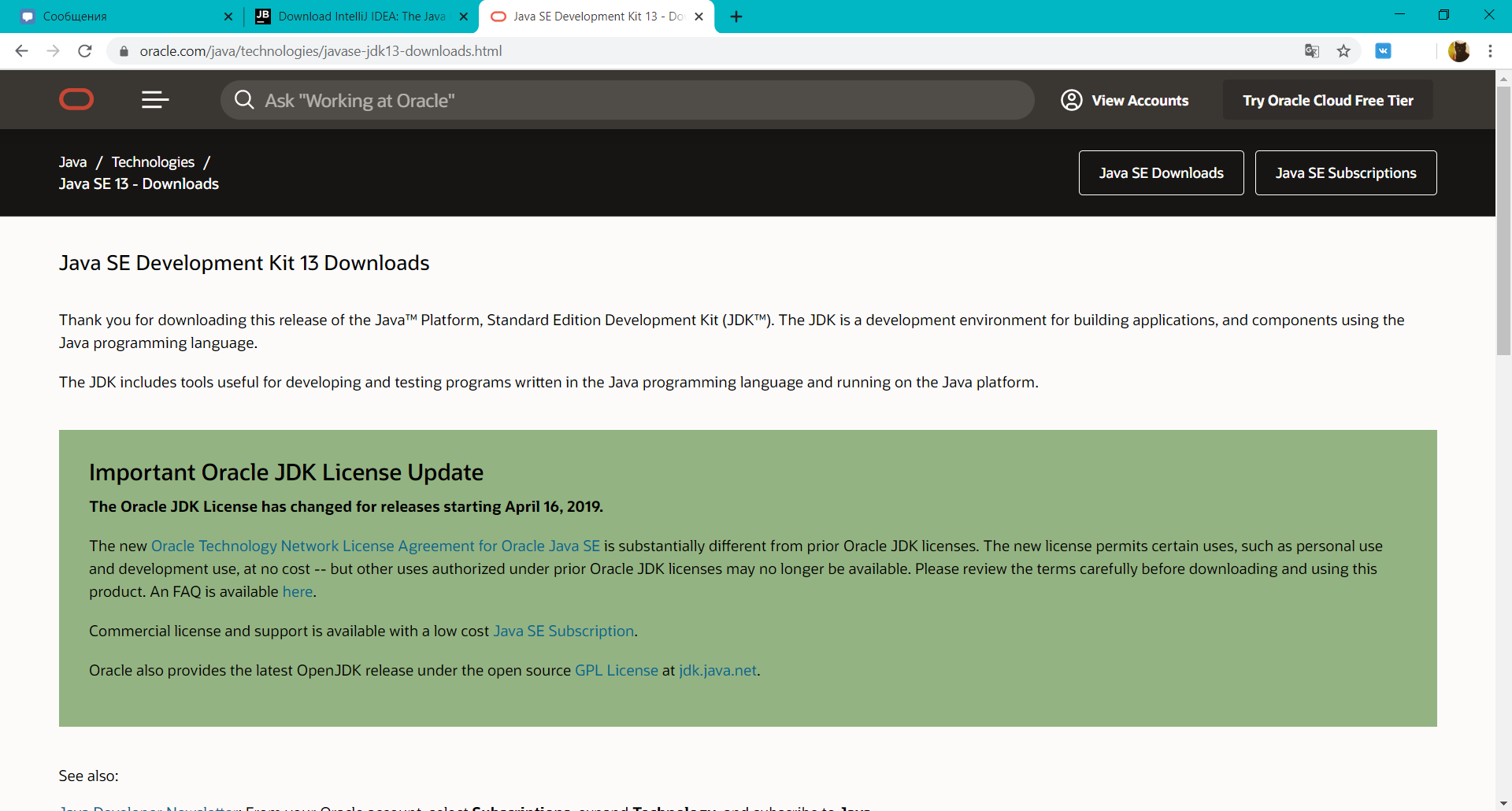


Рисунок 8.1 – Скачивание Java SE13

После окончания установки заходим в командную строку и устанавливаем русскую раскладку через команду «chcp 1251». После переходим в папку модуля Server, затем в папку «src», копируем путь к данной папке. Возвращаемся в окно командной строки и через команду «cd» вставляем путь к папке, после чего выполняем команду. Далее необходимо скомпилировать проект. Для этого вводим команду «javac –encoding UTF-8 server.java», выполняем. Далее запускаем проект с помощью команды «java server». После запуска сервера в открывшемся окне можно увидеть сообщение об успешном запуске сервера (Рисунок 8.3).

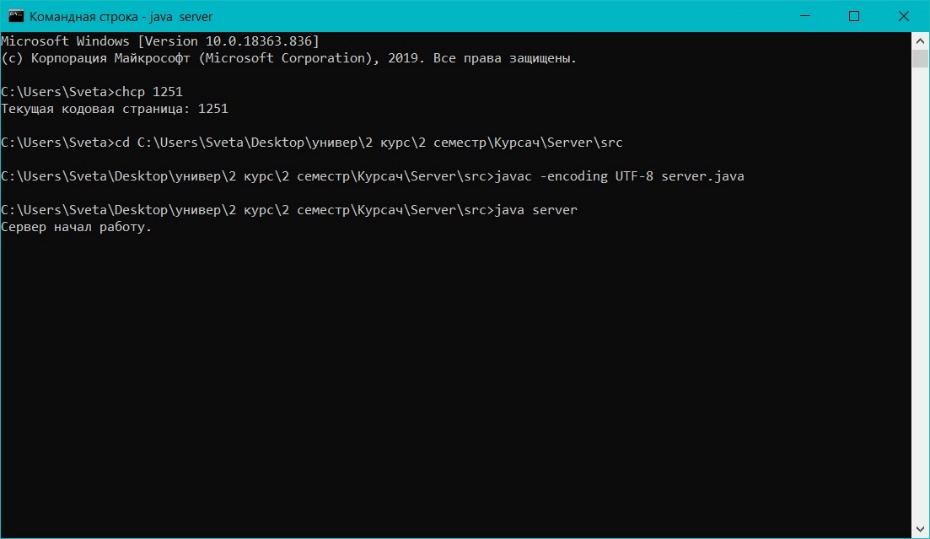


Рисунок 8.3 – Запуск серверной части программы

Для нормальной работы программы в проекте должны присутствовать оба модуля, так как курсовой проект – это клиент-серверное приложение. Поэтому повторяем действия, описанные выше и для модуля Client. После успешного подключения клиента к серверу выводится на экран соответствующее сообщение и пользователь переходит в меню первого порядка (Рисунок 8.4).

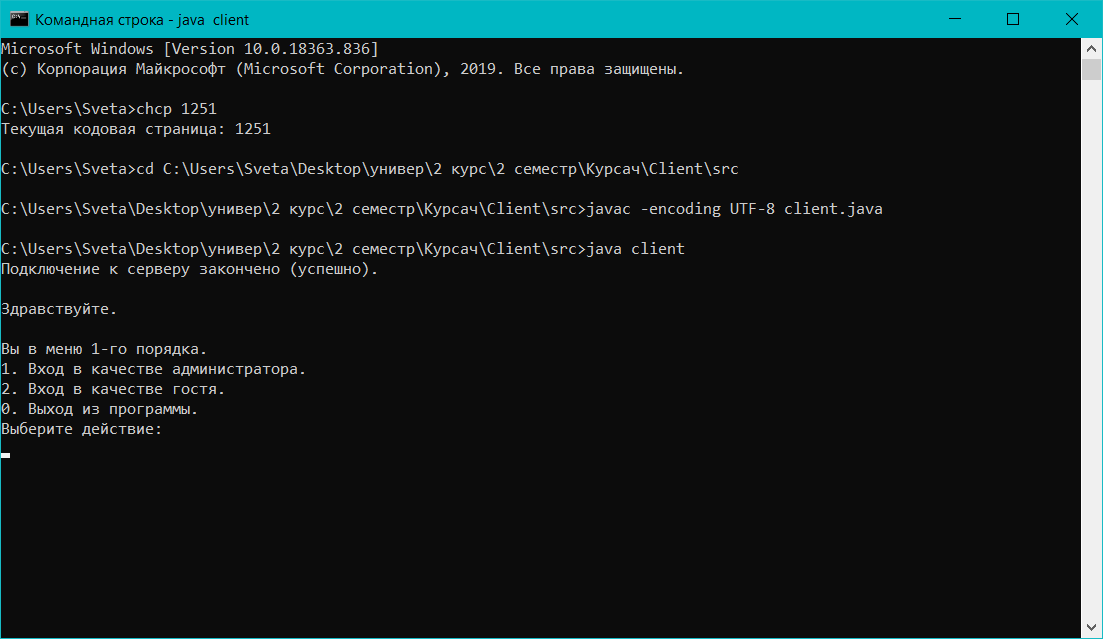


Рисунок 8.4 – Запуск клиентской части программы

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

После запуска и серверной, и клиентской частей приложения в окне запуска отображается меню первого порядка. Здесь пользователь должен выбрать вход в систему в качестве администратора, пользователя (гостя) или выход из программы (Рисунок 9.1).

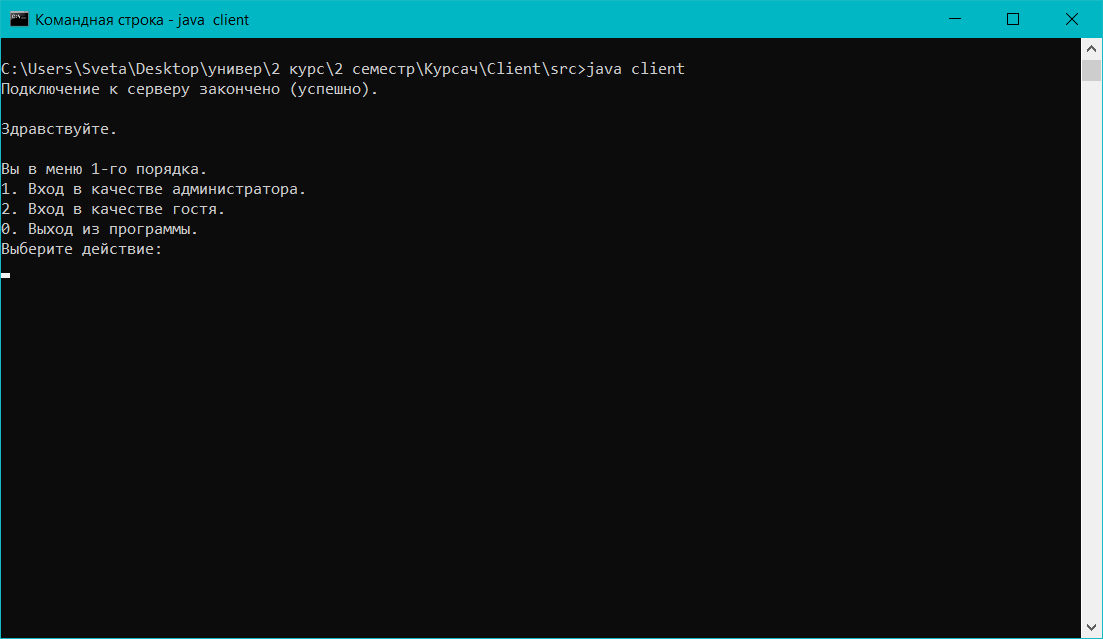


Рисунок 9.1 – Меню первого порядка

Выбор пункта меню происходит нажатием клавиш «1», «2» или «0». При вводе другой информации приложение выводит сообщение об ошибке и просит пользователя ввести информацию повторно до тех пор, пока не будут введены верные данные (Рисунок 9.2).

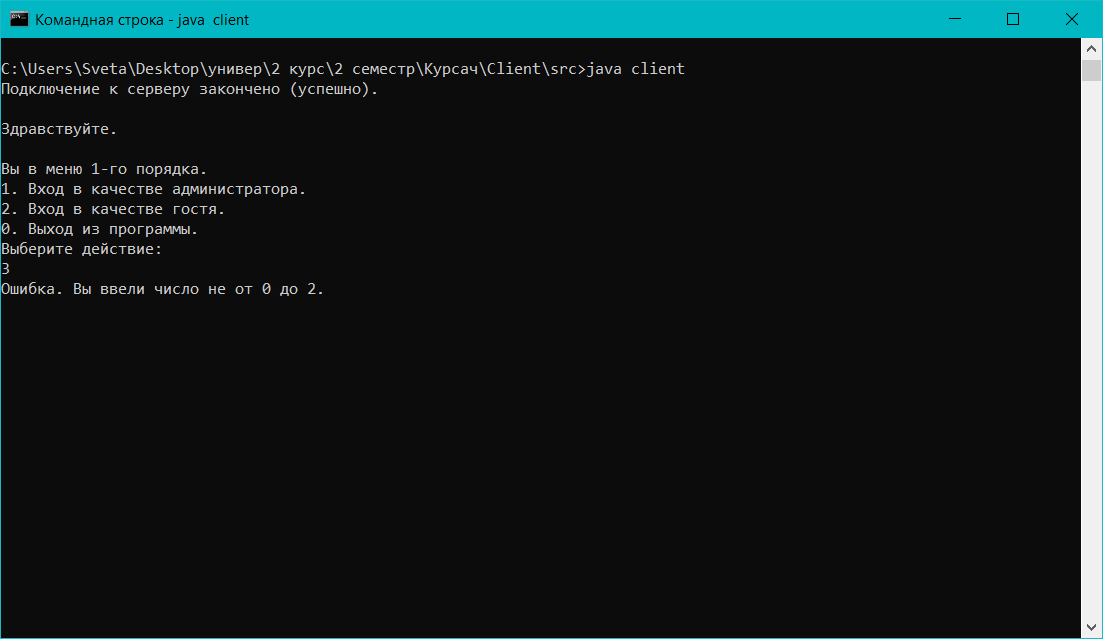


Рисунок 9.2 – Вывод сообщения об ошибке

При выборе пункта меню «Вход в качестве администратора» программа просит ввести логин и пароль. После правильного ввода логина и пароля, пользователь переходит в меню, состоящее из 5 пунктов (рисунок 9.3).

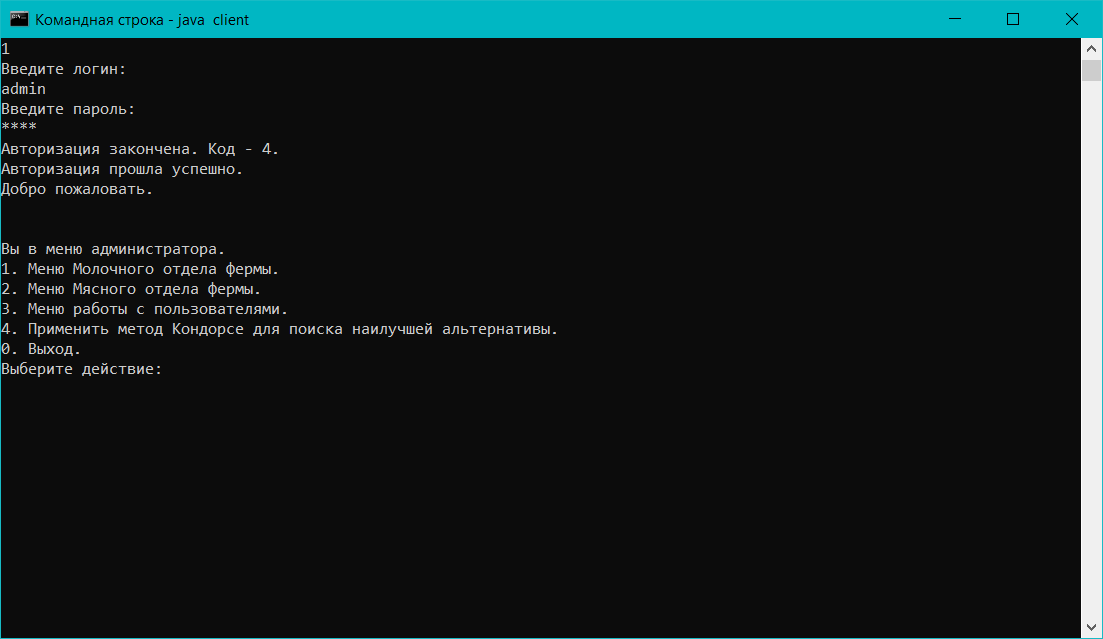


Рисунок 9.3 – Меню администратора

Если же логин или пароль были введены неправильно, то программа выводит сообщение об ошибке и завершает работу (Рисунок 9.4)

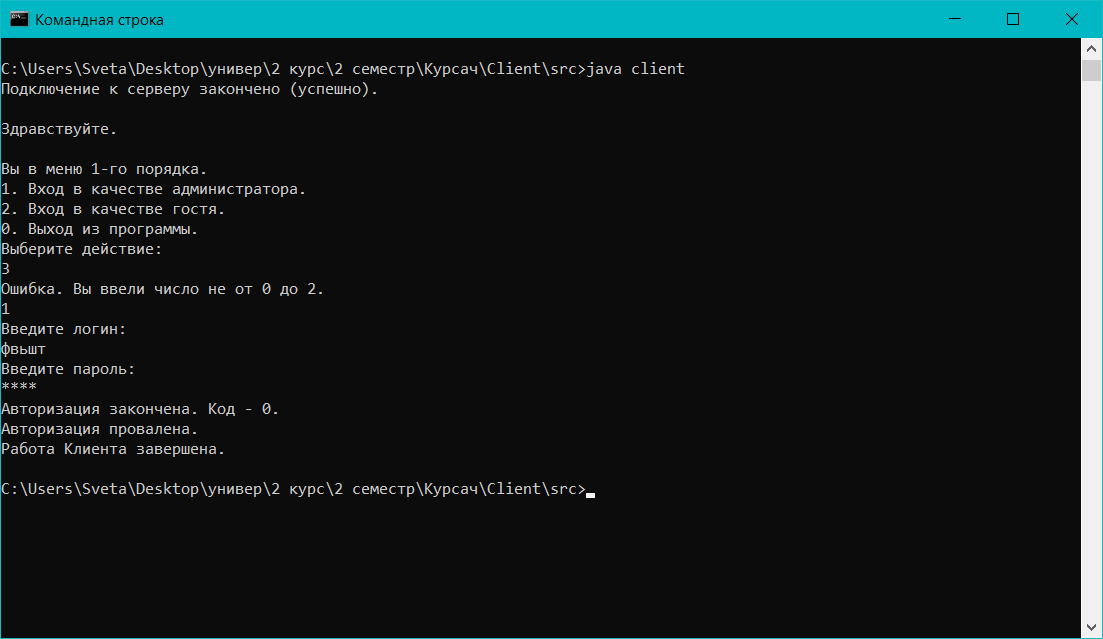


Рисунок 9.4 – Неправильный ввод пароля или логина

1. Выбрав пункт меню «Меню Молочного отдела фермы», мы попадаем в меню молочного отдела, состоящего из 8 пунктов (Рисунок 9.5).

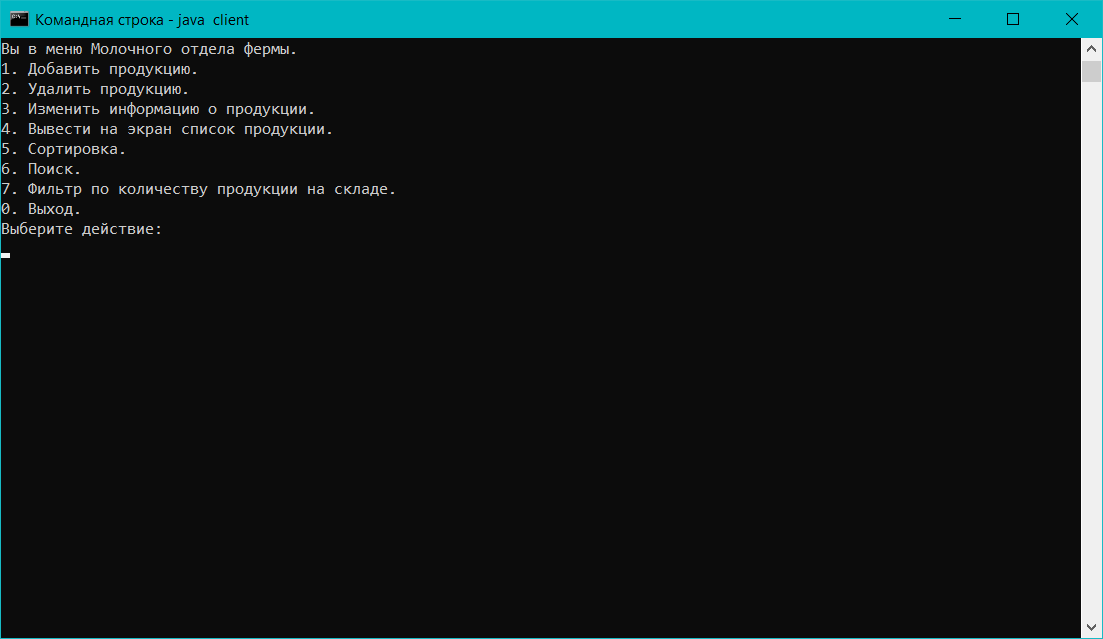


Рисунок 9.5 – Меню молочного отдела фермы

* 1. При выборе пункта «Добавить продукцию» на сервере вызывается добавление продукции. После этого нам необходимо ввести запрашиваемые программой данные для добавления нового товара (Рисунок 9.6).

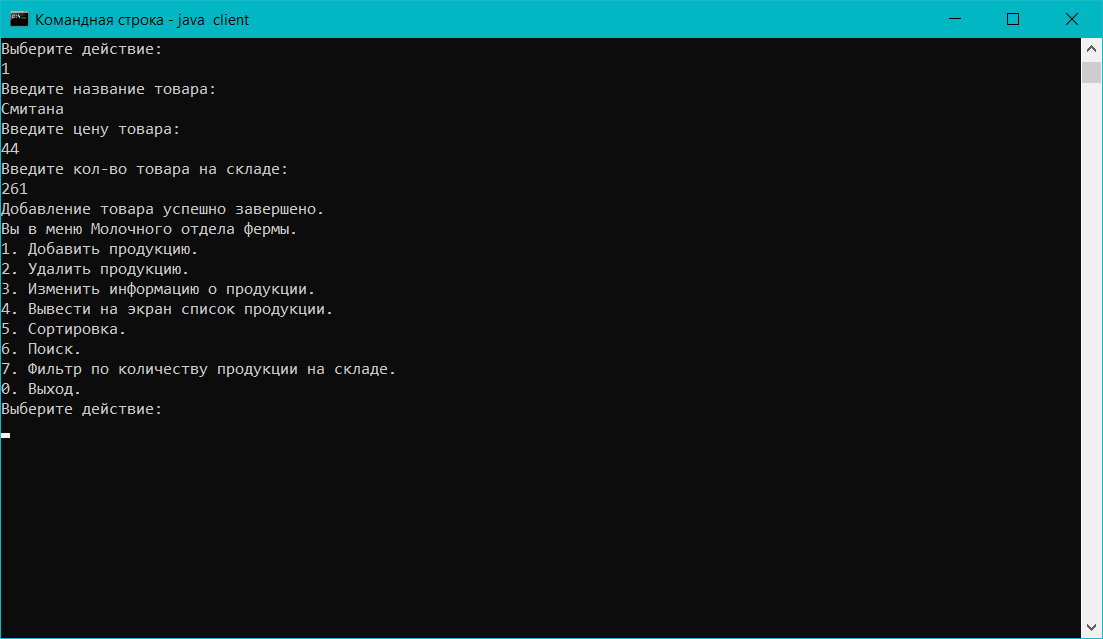


Рисунок 9.6 – Добавление нового товара молочного отдела фермы

* 1. При выборе пункта «Удалить продукцию» на сервере вызывается удаление продукции. После этого на экран выводится список товаров и нам необходимо ввести запрашиваемые программой данные для удаления товара (Рисунок 9.7).

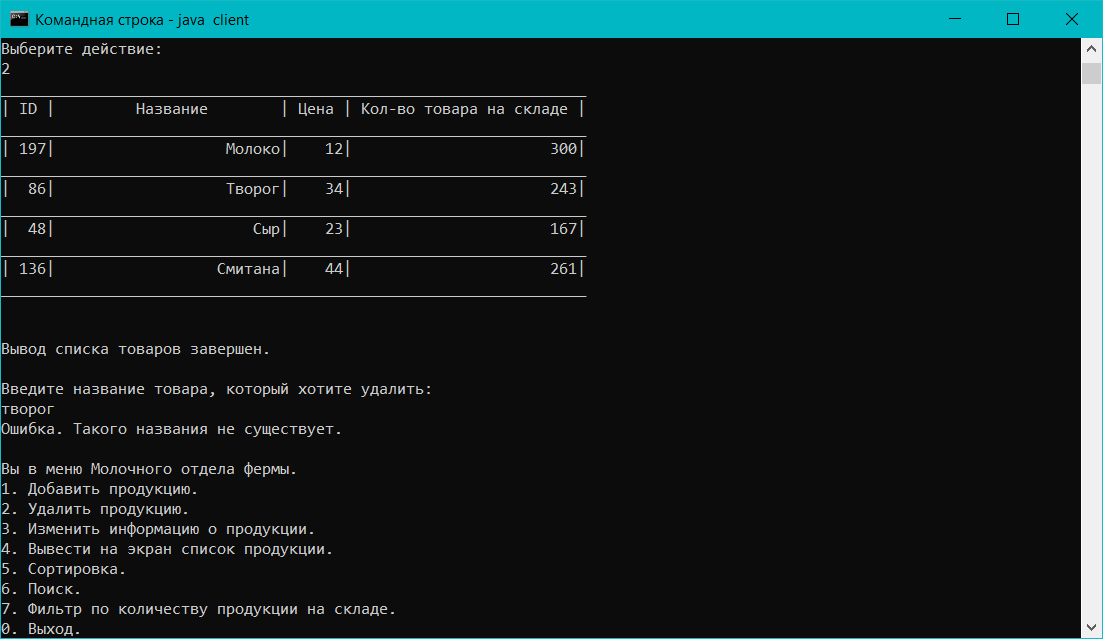


Рисунок 9.7 – Провальное удаление товара молочного отдела фермы

* 1. При выборе пункта «Редактировать продукцию» на сервере вызывается изменение данных о продукции. После этого на экран выводится список товаров и нам необходимо ввести запрашиваемые программой данные для редактирования товара (Рисунок 9.8).

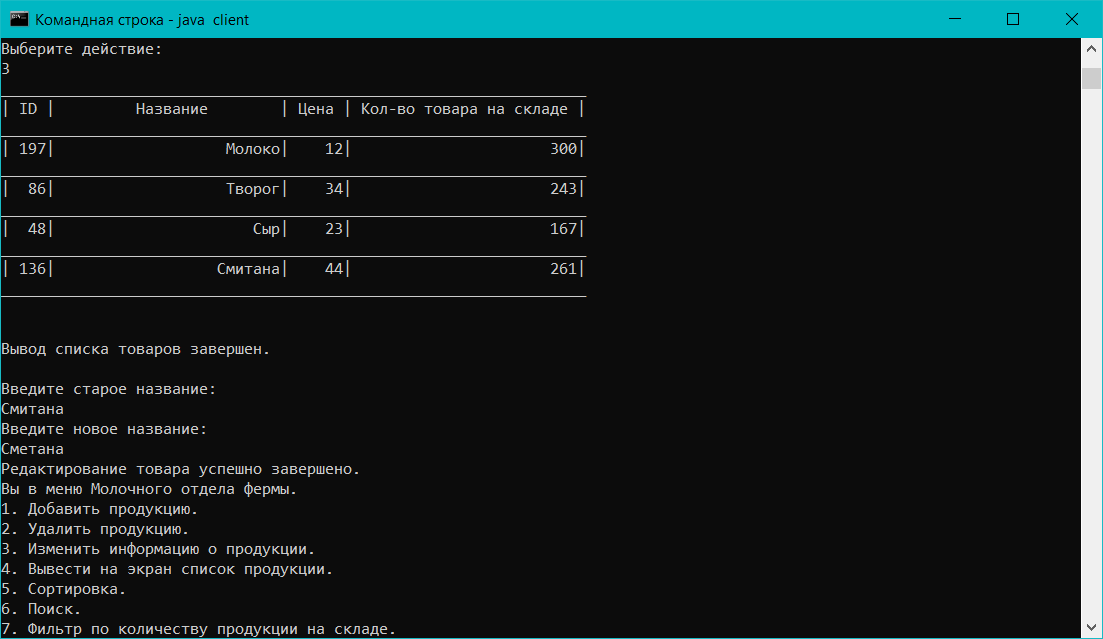


Рисунок 9.8 – Редактирование товара молочного отдела фермы

* 1. При выборе пункта «Вывести на экран список продукции» на экран выводится таблица товаров молочного отдела (Рисунок 9.9).

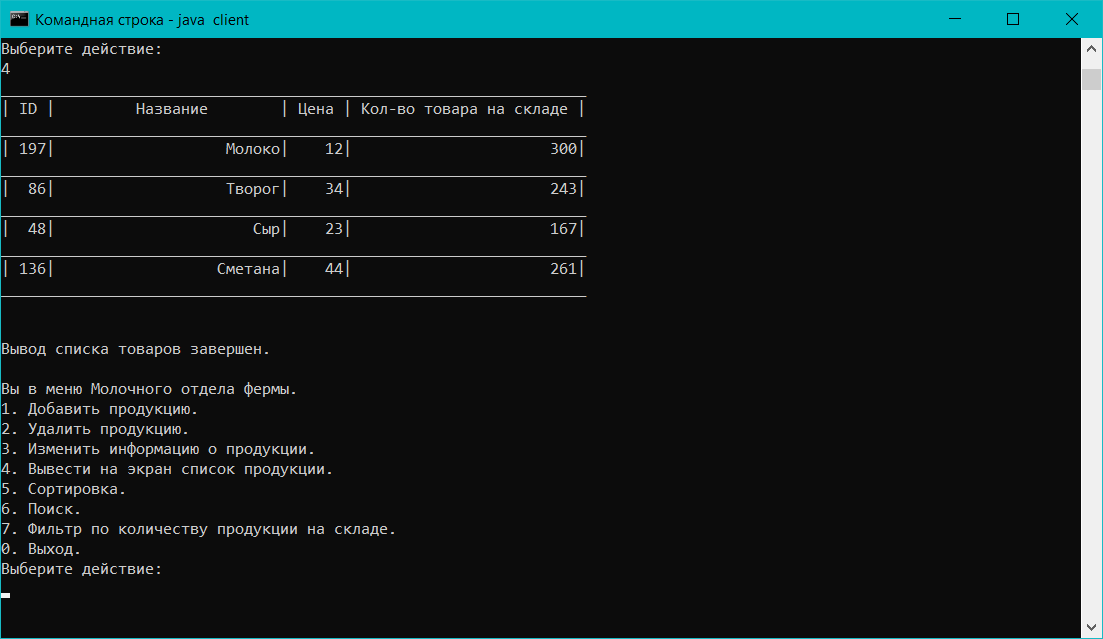


Рисунок 9.9 – Вывод списка товаров молочного отдела фермы на экран

* 1. При выборе пункта «Сортировка» переходим в меню сортировки (Рисунок 9.10).

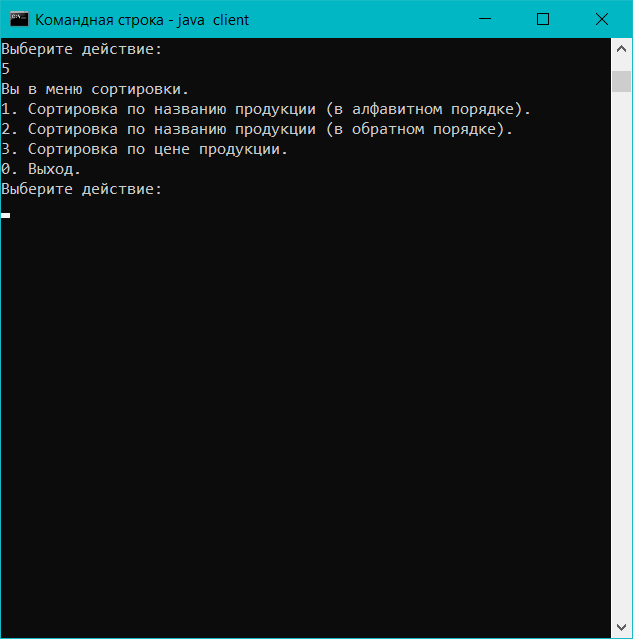


Рисунок 9.10 – Меню сортировки

* + 1. При выборе пункта «Сортировка по названию продукции (в алфавитном порядке)» на экран выводится успешно отсортированная таблица товаров либо, если сортировка не прошла успешно, выводится сообщение об ошибке (Рисунок 9.11).

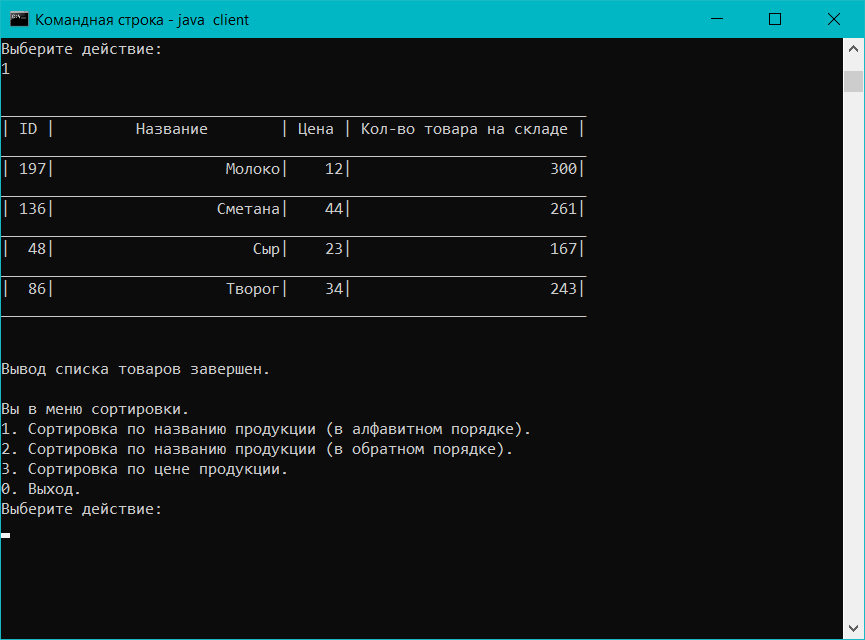


Рисунок 9.11 – Вывод успешной сортировки по названию продукции (в алфавитном порядке) на экран

* + 1. При выборе пункта «Сортировка по названию продукции (в обратном порядке)» на экран выводится успешно отсортированная таблица товаров либо, если сортировка не прошла успешно, выводится сообщение об ошибке (Рисунок 9.12).

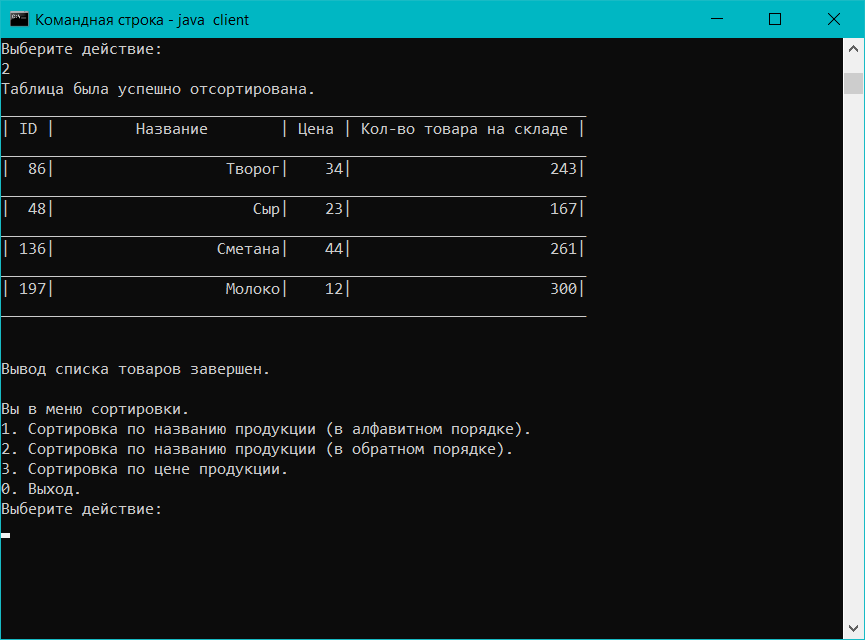


Рисунок 9.12 – Вывод успешной сортировки по названию продукции (в обратном порядке) на экран

* + 1. При выборе пункта «Сортировка по цене продукции» на экран выводится успешно отсортированная таблица товаров либо, если сортировка не прошла успешно, выводится сообщение об ошибке (Рисунок 9.13).

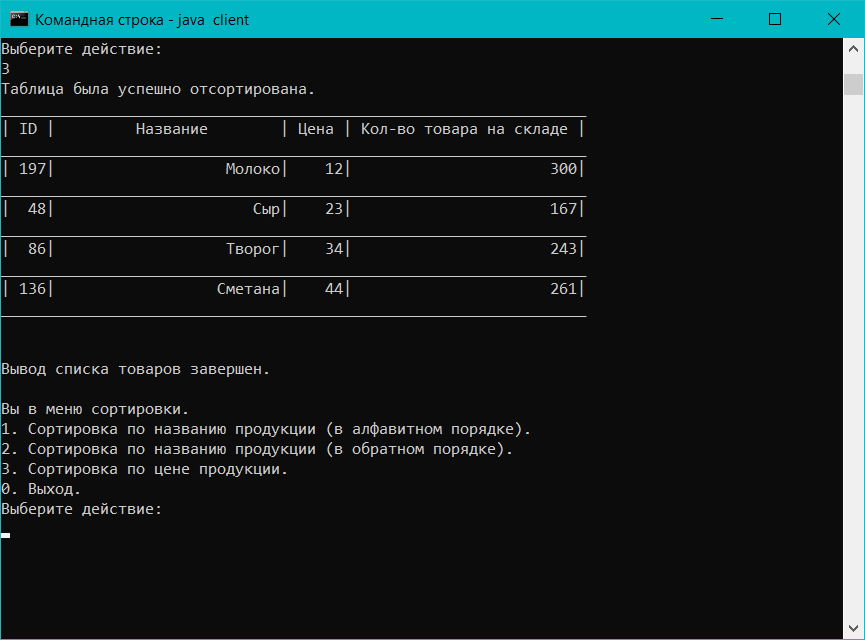


Рисунок 9.13 – Вывод успешной сортировки по цене продукции на экран

* 1. При выборе пункта «Поиск» переходим в меню поиска (Рисунок 9.14).

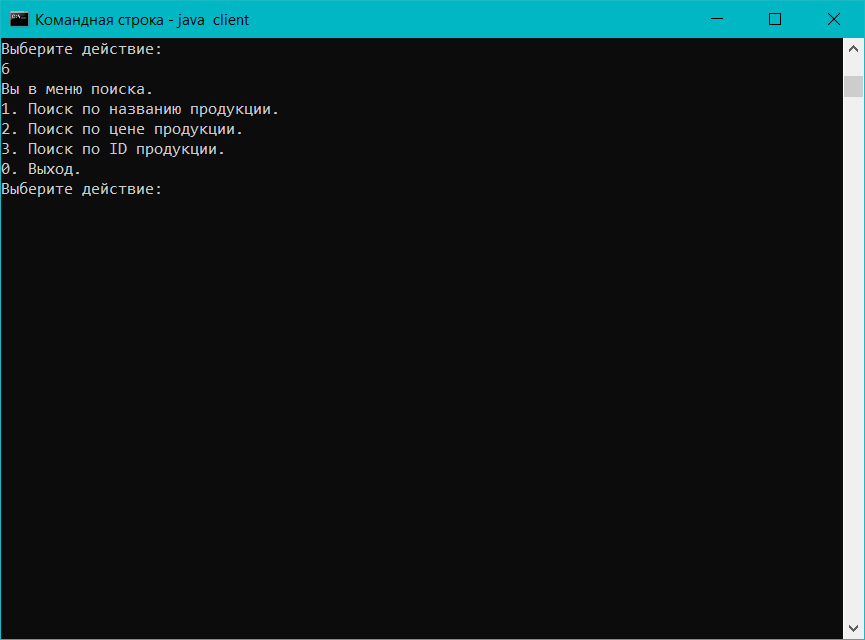


Рисунок 9.14 – Меню поиска

* + 1. При выборе пункта «Поиск по названию продукции» программа запрашивает данные для поиска, если такой товар найден, то на экран выводится информация о товаре, в противном случае выводится сообщение об ошибке (Рисунок 9.15).

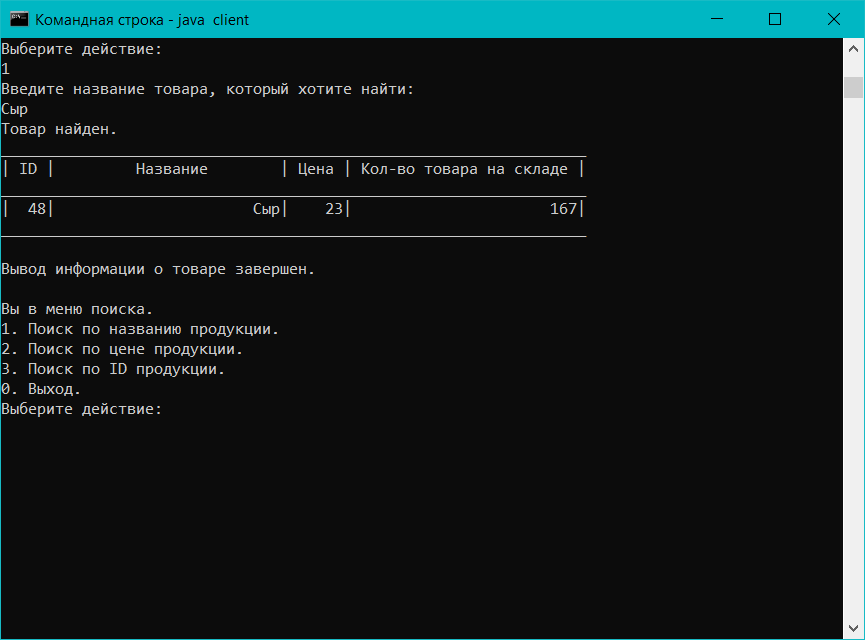


Рисунок 9.15 – Успешный поиск по названию продукции

* + 1. При выборе пункта «Поиск по цене продукции» программа запрашивает данные для поиска, если такой товар найден, то на экран выводится информация о товаре, в противном случае выводится сообщение об ошибке (Рисунок 9.16).

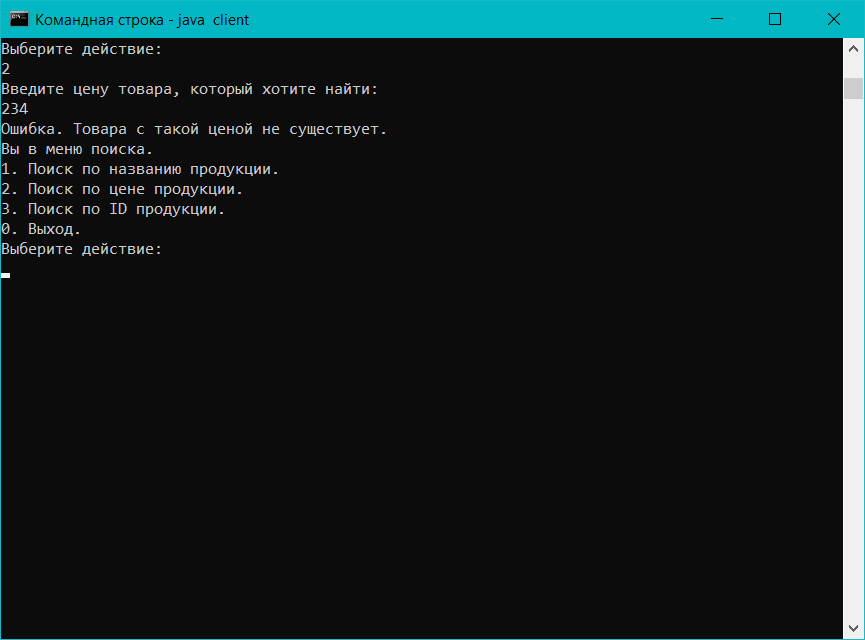


Рисунок 9.16 – Провальный поиск по цене продукции

* + 1. При выборе пункта «Поиск по ID продукции» программа запрашивает данные для поиска, если такой товар найден, то на экран выводится информация о товаре, в противном случае выводится сообщение об ошибке (Рисунок 9.17).

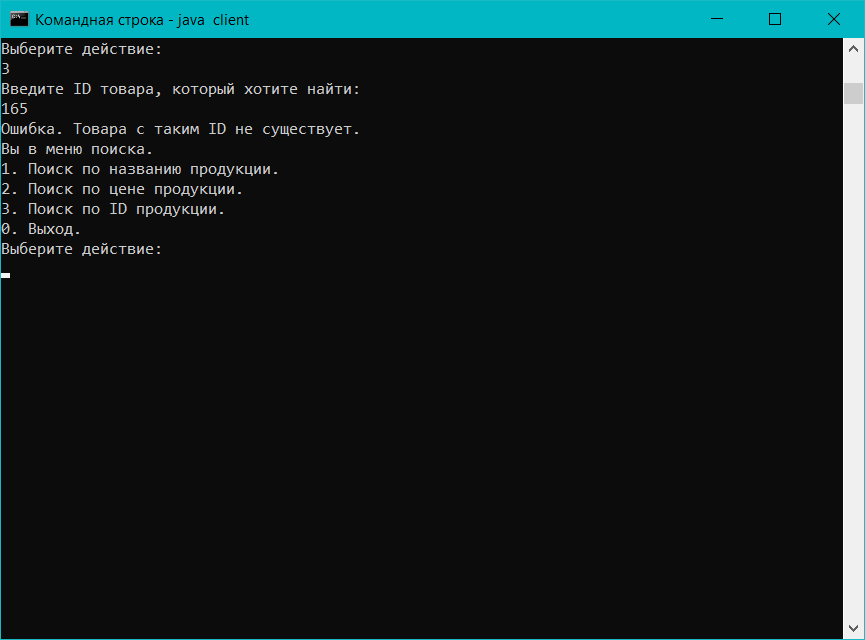


Рисунок 9.17 – Провальный поиск по ID продукции

* 1. При выборе пункта «Фильтр по количеству товара на складе» программа просит ввести данные для фильтрации. Если данные найдены, то на экран выводится список товаров, подходящих под введенные данные, в противном случае на экран выводится сообщение об ошибке (Рисунок 9.18).

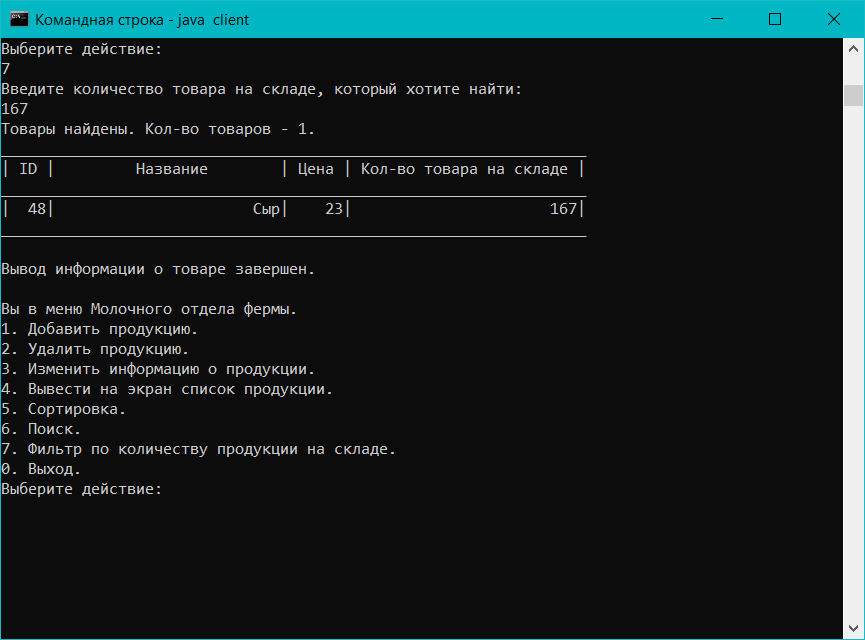


Рисунок 9.18 – Фильтр по количеству товара на складе молочного отдела фермы

1.8 При выборе пункта «Выход» переходим в меню администратора.

1. Выбрав пункт меню «Меню Мясного отдела фермы», мы попадаем в меню мясного отдела, состоящего из 8 пунктов (Рисунок 9.19).

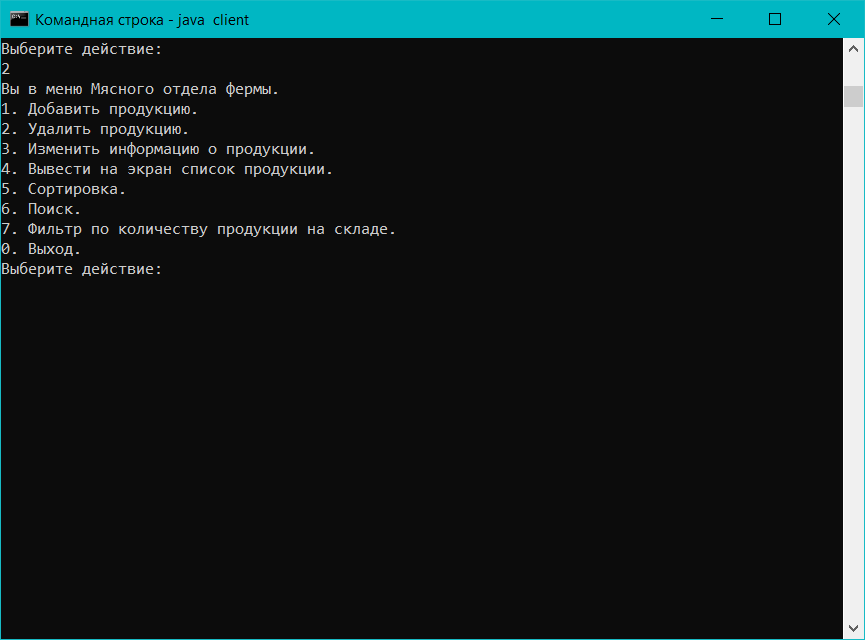


Рисунок 9.19 – Меню мясного отдела фермы

Функционал и действия в данном пункте такие же, как и в пункте «Меню Молочного отдела фермы», за исключением данных. Здесь используются данные из другого текстового файла.

1. Выбрав пункт меню «Меню работы с пользователями», мы попадаем в меню работы с пользователями, состоящего из 5 пунктов (Рисунок 9.20).

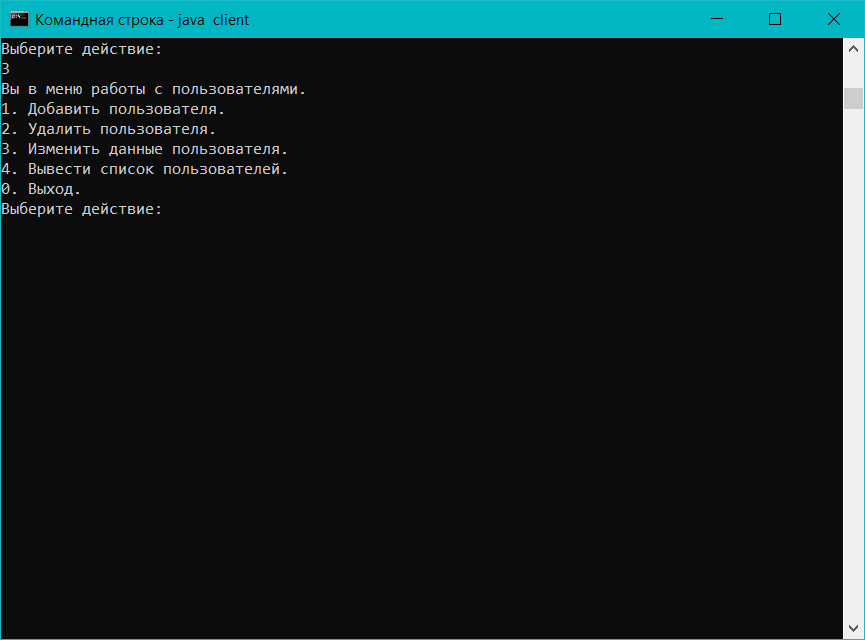


Рисунок 9.20 – Меню работы с пользователями

* 1. При выборе пункт меню «Добавить пользователя» программа запрашивает необходимые данные для добавления нового пользователя. После на сервере происходит добавление пользователя в файл. Если добавление прошло успешно, то выводится соответствующее сообщение. В противном случае выводится сообщение об ошибке (Рисунок 9.21).

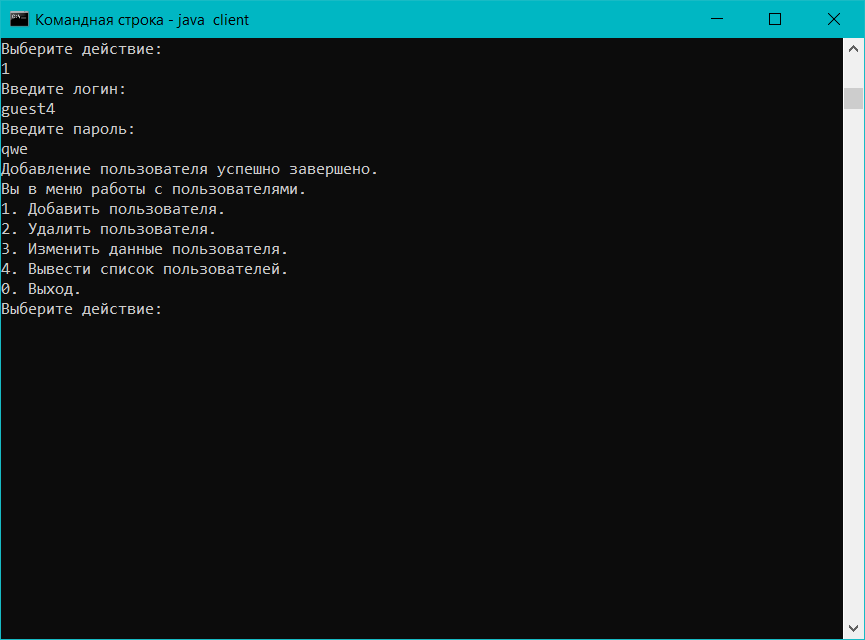


Рисунок 9.21 – Успешное добавление нового пользователя

* 1. При выборе пункт меню «Удалить пользователя» программа выводит на экран список всех пользователей и запрашивает необходимые данные для удаления. После на сервере происходит удаление пользователя. Если удаление прошло успешно, то выводится соответствующее сообщение. В противном случае выводится сообщение об ошибке (Рисунок 9.22).

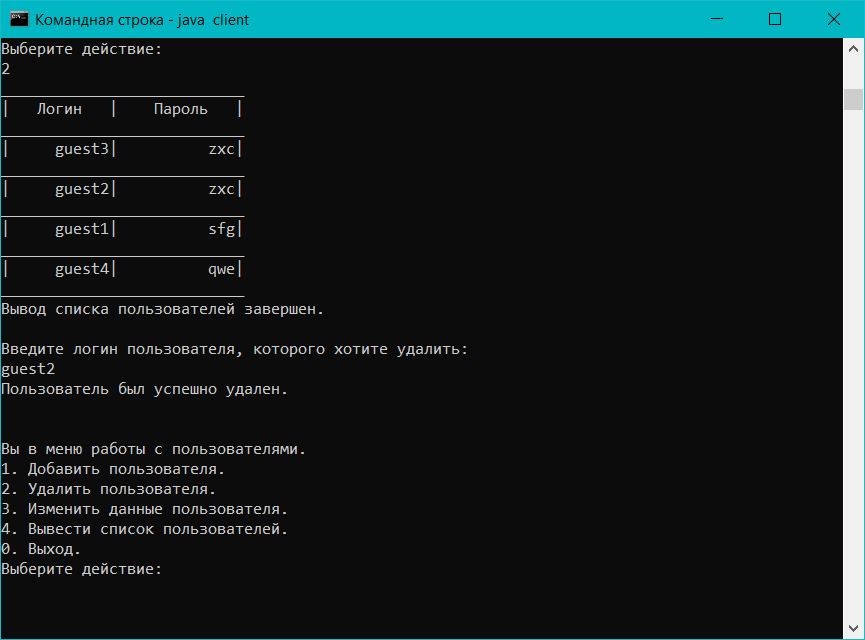


Рисунок 9.22 – Успешное удаление пользователя

* 1. При выборе пункт меню «Изменить данные пользователя» программа выводит на экран список всех пользователей и запрашивает необходимые данные для редактирования. После на сервере происходит изменение данных о пользователе. Если удаление прошло успешно, то выводится соответствующее сообщение. В противном случае выводится сообщение об ошибке (Рисунок 9.23).

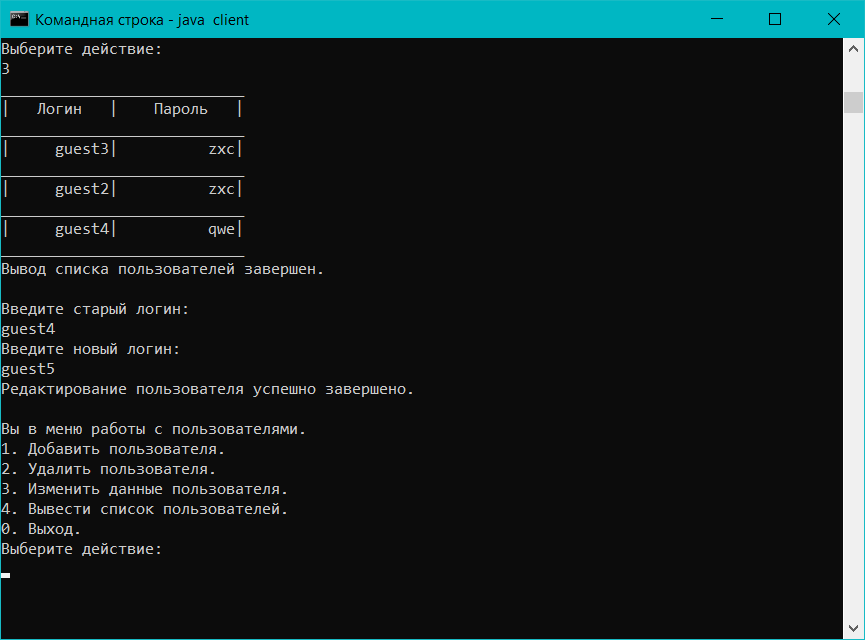


Рисунок 9.23 – Успешное редактирование пользователя

* 1. При выборе пункт меню «Вывести список пользователей» программа выводит на экран список всех пользователей \, если при выводе произошла какая-либо ошибка, то выводится сообщение об ошибке (Рисунок 9.22).

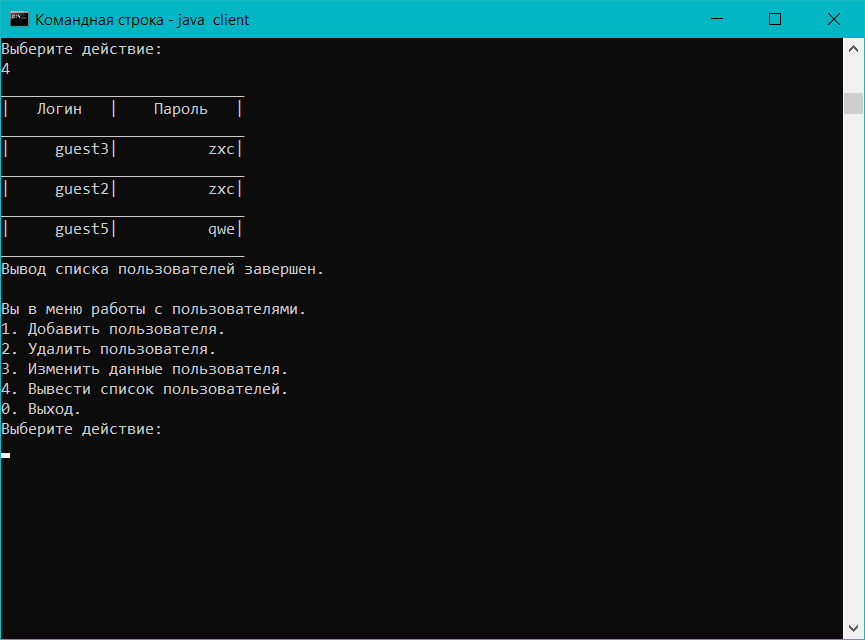


Рисунок 9.22 – Вывод списка пользователей на экран

* 1. При выборе пункта «Выход» переходим в меню администратора.

1. Выбрав пункт меню «Применить метод Кондорсе для поиска наилучшей альтернативы», программа запрашивает необходимые данные для расчета, затем выводит список альтернатив и список экспертов, матрицу предпочтений, ход поиска и его результат (Рисунок 9.23 и Рисунок 9.24).

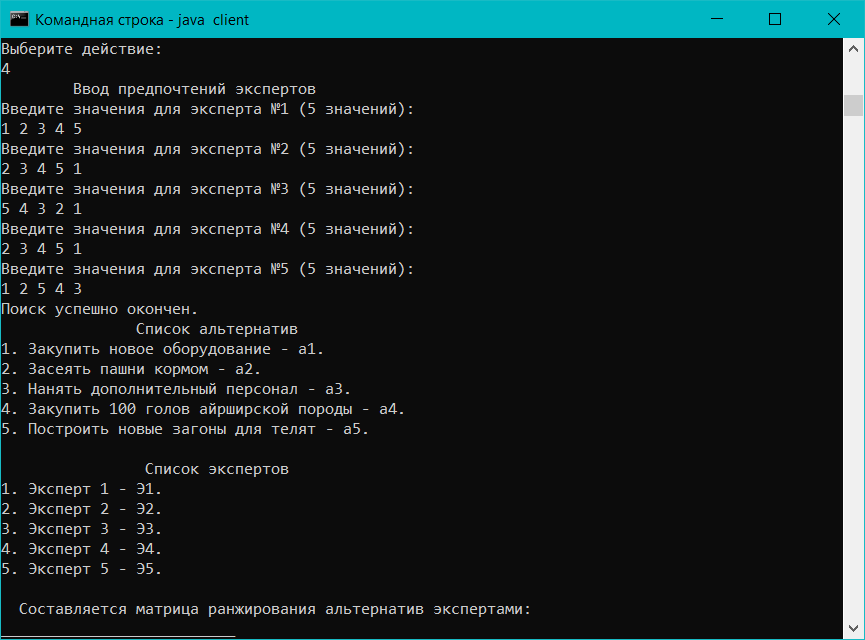


Рисунок 9.23 – Ввод исходных данных для поиска наилучшей альтернативы методом Кондорсе

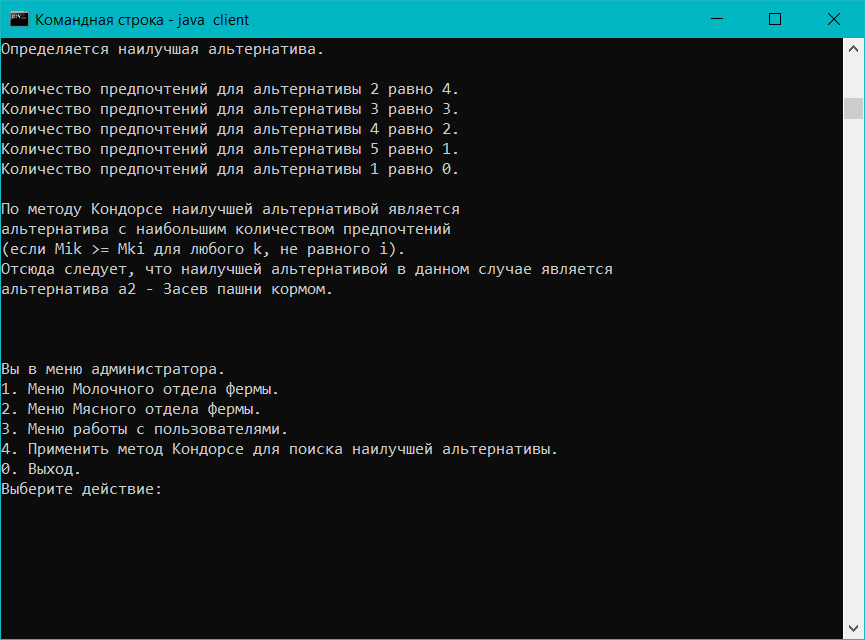


Рисунок 9.24 – Результат поиска наилучшей альтернативы методом Кондорсе

1. Выбрав пункт меню «Выход», переходим в меню первого порядка.

При выборе пункта меню «Вход в качестве гостя» программа просит ввести логин и пароль. После правильного ввода логина и пароля, пользователь переходит в меню, состоящее из 5 пунктов (рисунок 9.25).

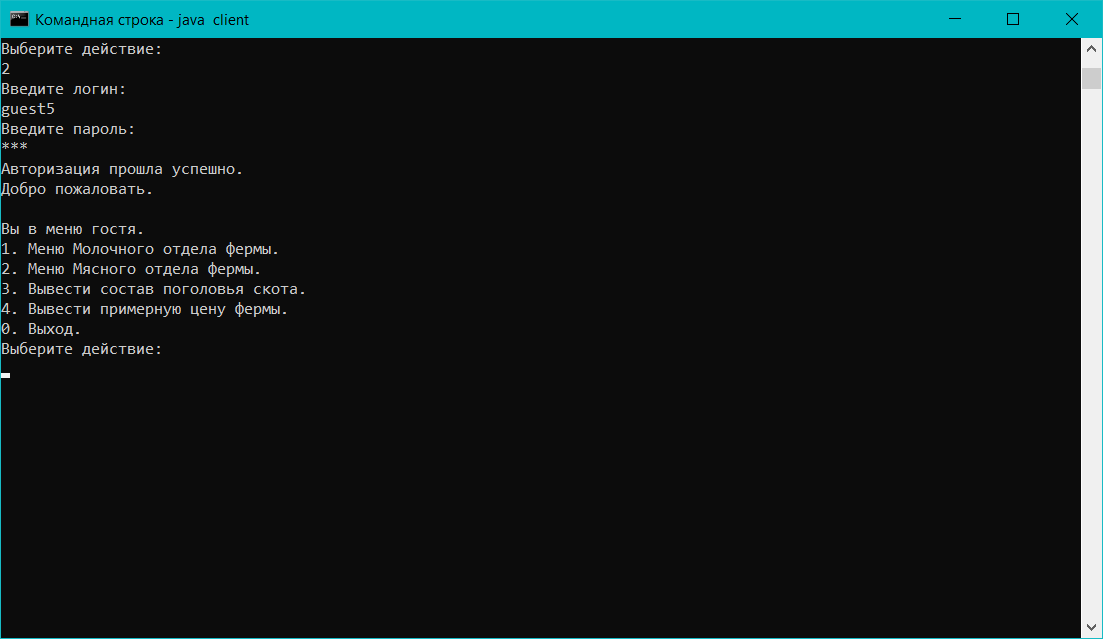


Рисунок 9.25 – Меню пользователя (гостя)

1. Выбрав пункт «Меню Молочного отдела» попадаем в меню молочного отдела фермы, которое содержит 5 пунктов (Рисунок 9.26).

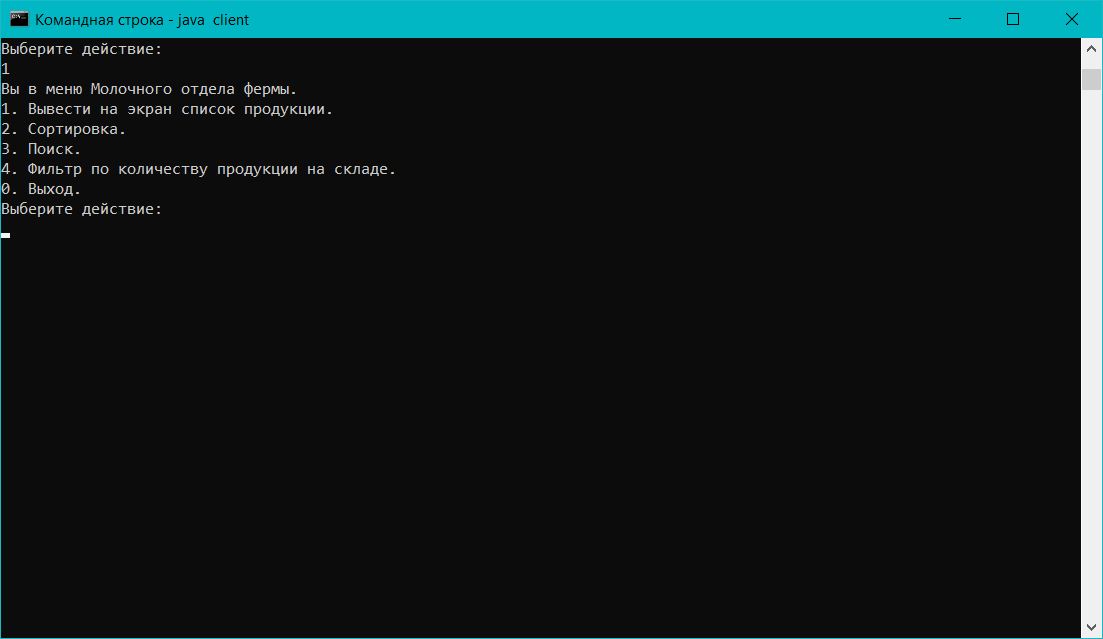


Рисунок 9.26 – Меню молочного отдела фермы

Все пункты меню расписаны в меню администратора (см. выше).

1. Выбрав пункт «Меню Мясного отдела» попадаем в меню мясного отдела фермы, которое содержит 5 пунктов (Рисунок 9.27).

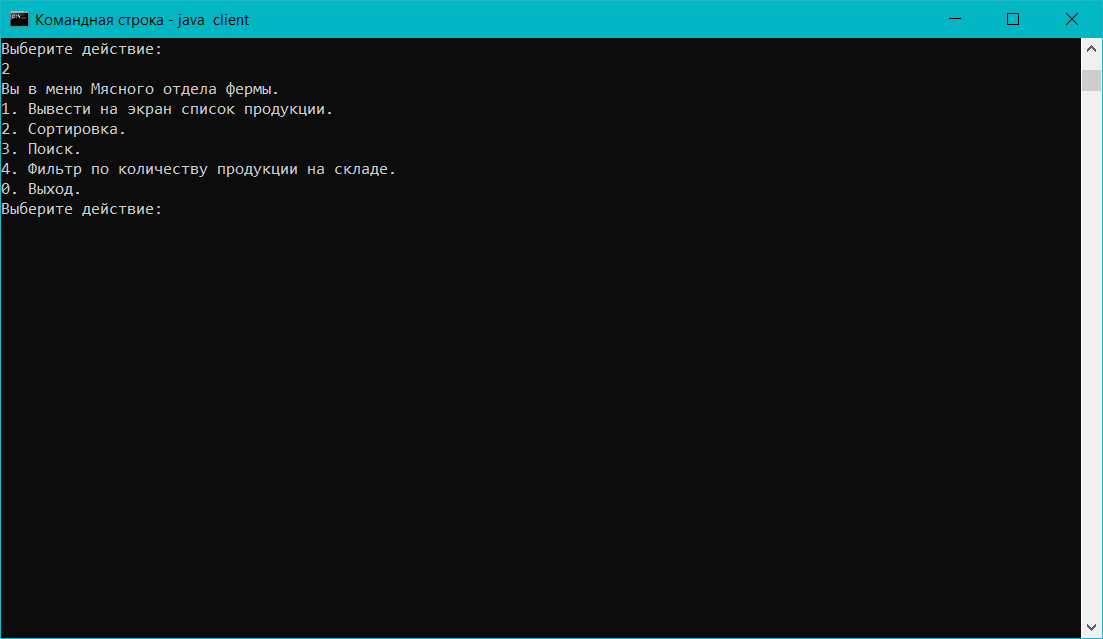


Рисунок 9.27 – Меню мясного отдела фермы

Все пункты меню расписаны в меню администратора (см. выше).

1. При выборе пункта «Вывести состав поголовья скота» программа выводит из файла информацию о составе поголовья скота фермы на экран (Рисунок 9.28).



Рисунок 9.28 – Вывод состава поголовья скота

1. При выборе пункта «Вывести примерную цену фермы» программа выводит на экран состав поголовья скота и площадь пашенной территории фермы, после сервер просчитывает примерную стоимость фермы. Далее расчет стоимости и результат выводится на экран (Рисунок 9.29).

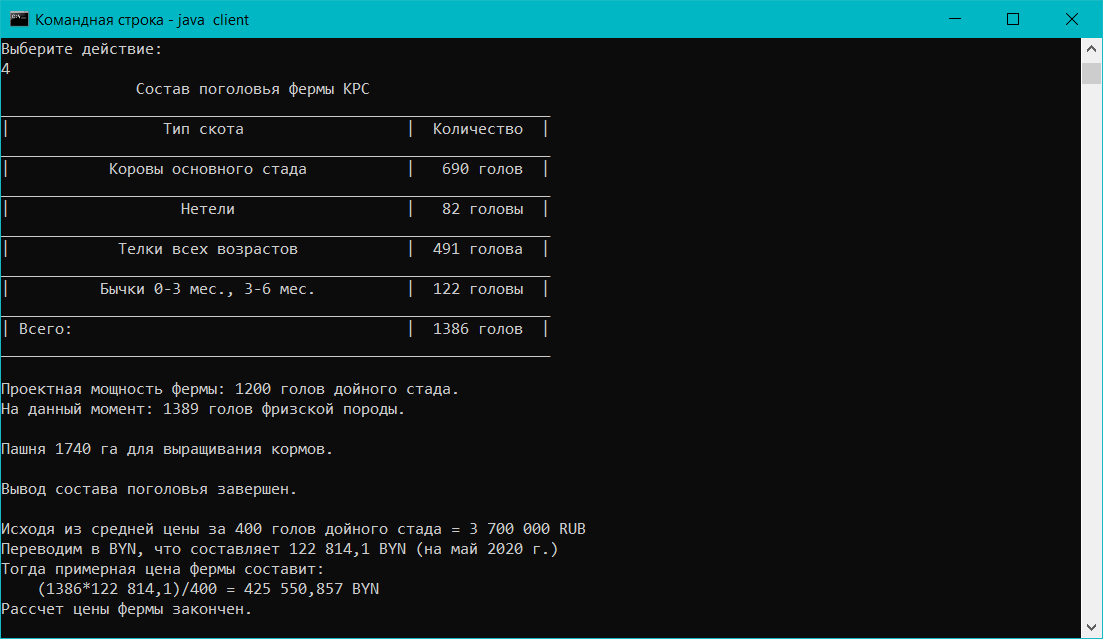


Рисунок 9.29 – Вывод примерной цены фермы

1. При выборе пункта «Выход» переходим в меню первого порядка.

# КОНТРОЛЬНЫЙ ПРИМЕР

По умолчанию в программе существуют 5 экспертов. Все, что нужно указать пользователю – это предпочтения экспертов касательно 5 имеющихся альтернатив. Пусть ранжирование альтернатив будет следующим:

Э1 = (a1, a2, a3, a4, a5)

Э2 = (a2, a4, a5, a3, a1)

Э3 = (a5, a1, a2, a4, a3)

Э4 = (a3, a4, a2, a1, a5)

Э5 = (a3, a5, a1, a4, a2)

Находятся оценки Mik, характеризующих предпочтение альтернатив в парных предпочтениях и составляется следующая матрица:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mik | A1 | A2 | A3 | A4 | A5 |
| A1 |  | 3 | 2 | 3 | 2 |
| A2 | 2 |  | 3 | 3 | 3 |
| A3 | 3 | 2 |  | 3 | 3 |
| A4 | 2 | 2 | 2 |  | 3 |
| A5 | 3 | 2 | 2 | 2 |  |

Выполняются проверки согласно принципу Кондорсе: наилучшей является альтернатива Ai, если Mik ≥ Mki для всех k ≠ i

K2 = K3 = 3, т.е. правилу Кондорсе удовлетворяют альтернативы A2  и A3.

Выбираются альтернативы Кондорсе. Это A2  и A3.

На рисунках 10.1 – 10.9 приведены результаты работы программы с заданными значениями.

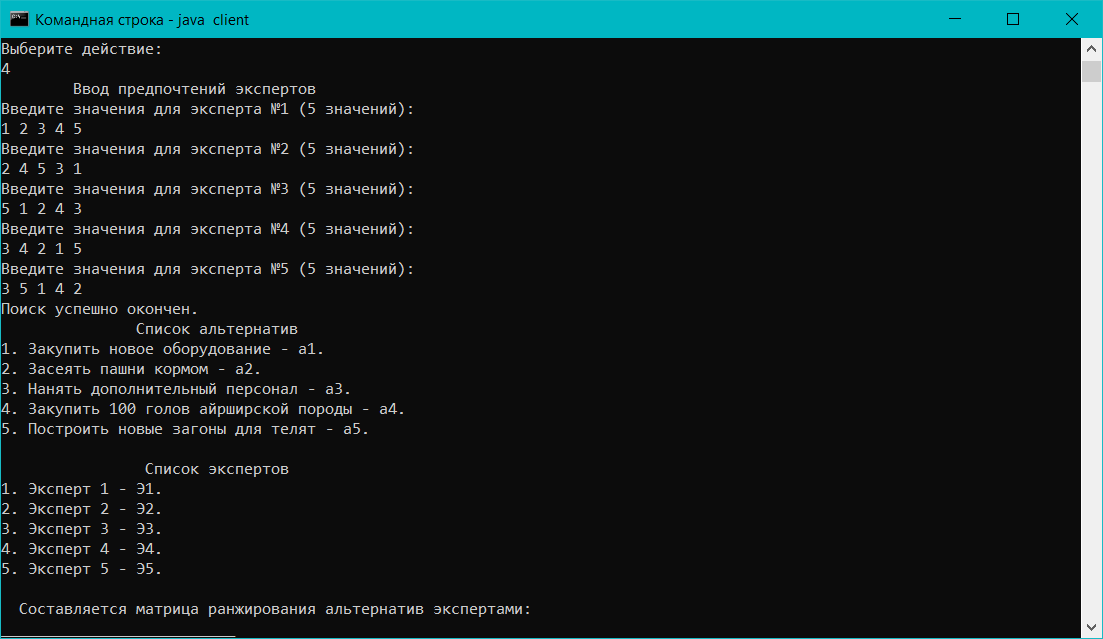


Рисунок 10.1 – Ввод оценок экспертов

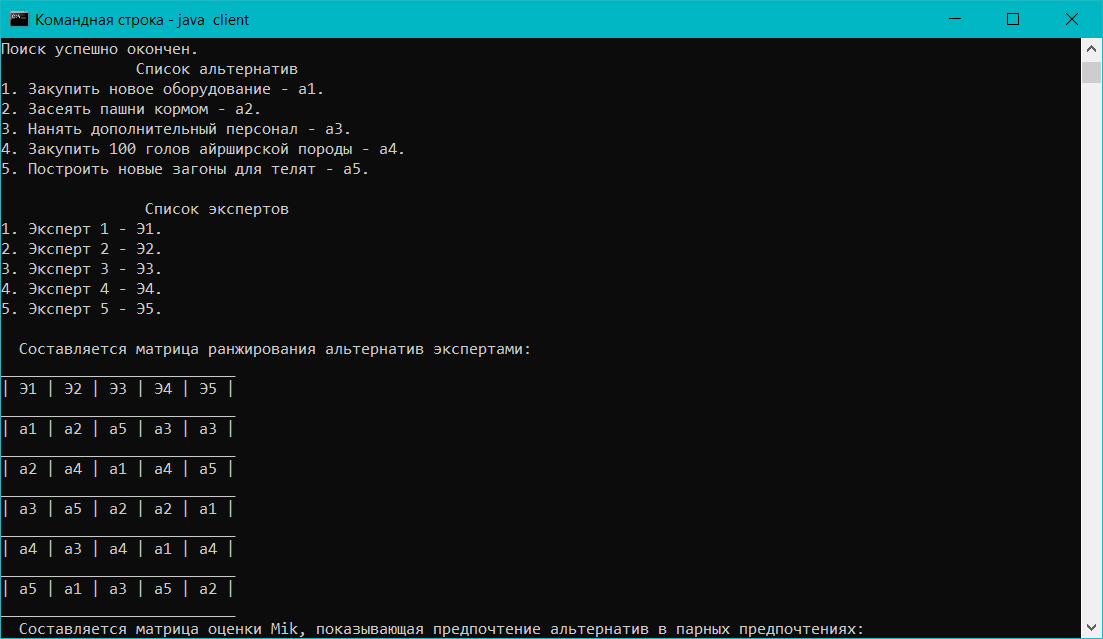


Рисунок 10.2 – Вывод списка альтернатив и экспертов

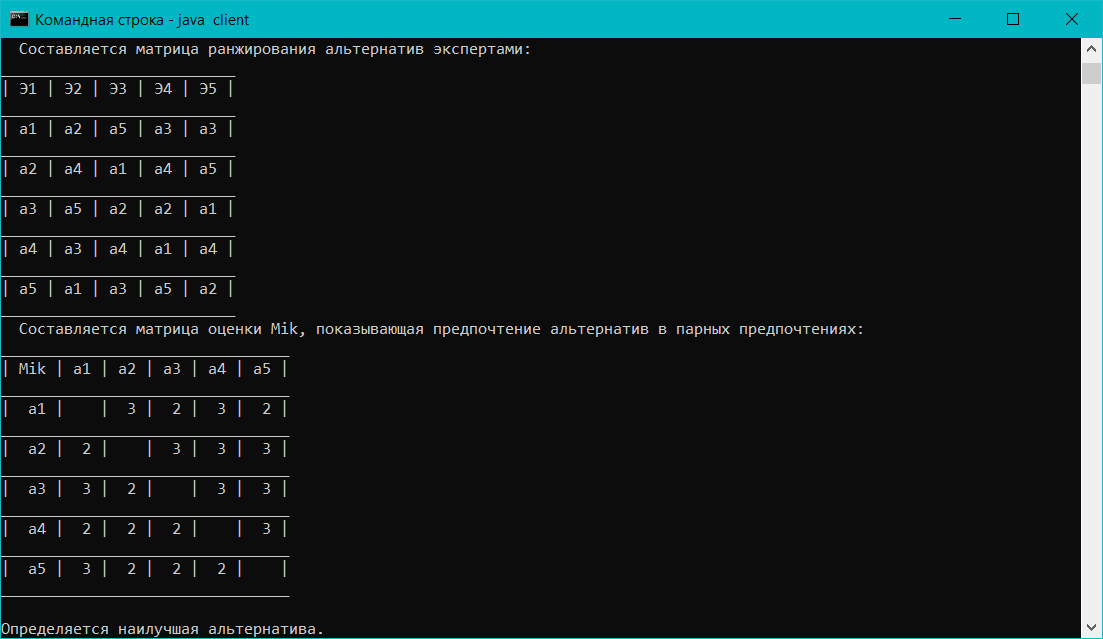


Рисунок 10.3 – Вывод матрицы ранжирования альтернатив экспертами и матрицы оценки Mik на экран

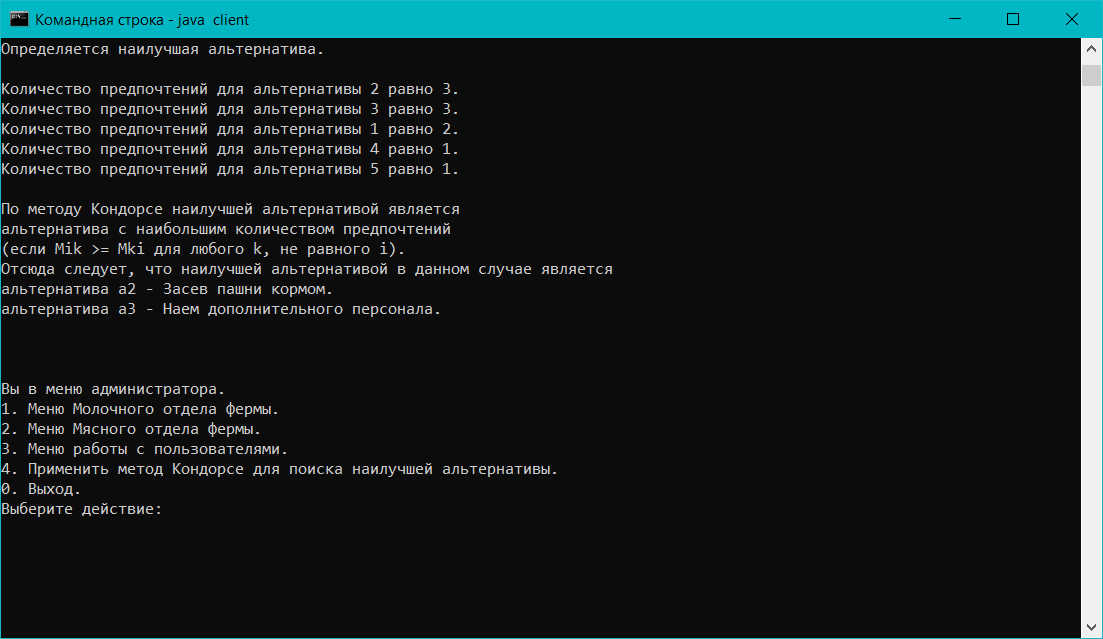


Рисунок 10.4 – Вывод результатов поиска наилучшей альтернативы

# ЗАКЛЮЧЕНИЕ

При выполнении данного курсового проекта была исследована предметная область «Ферма», ее организация в нашей стране, сравнение организации в нашей стране с организациями фермерских хозяйств в соседних странах и процессы, производимые внутри данной области.

Приложение удовлетворяет основным требованиям, которые были заявлены. Оно удобно в эксплуатации, целостно, конкретизировано в рамках заданной предметной области. Были разработаны пользовательские функции приложений: функции для работы с файлами, функции поиска, фильтрации и сортировки данных, функции авторизации, редактирования, удаления, проверки вводимых данных и функции бизнес логики.

Были разработаны алгоритмы работы следующих функций: меню первого порядка и последующих меню программы, функции поиска товаров по одному критерию, сортировки данных по имени и цене товаров, фильтрации по количеству товара на складе, вывод отчета по расчету стоимости фермы, вывод отчета о составе поголовья скота, вывод отчета по результатам поиска наилучшей альтернативы методом Кондорсе и т. д.

Была предусмотрена обработка исключительных ситуаций с помощью функций проверки вводимой информации, функции проверки наличия информации в файлах и функций проверки соединения клиента с сервером.

Итогом данной работы является функционирующее приложение, являющееся помощником в поиске наилучшей альтернативы в сфере фермерского хозяйства. В данном приложении реализованы базовые принципы объектно-ориентированного программирования, использованы стандартные и пользовательские функции, виртуальные функции, контейнеры и итераторы, механизмы наследования и шаблонные методы. Для использования данной программы не требуются высококвалифицированные специалисты, так как работа с ней не требует наличия сертифицированной компьютерной программы для оформления.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] Методичка – Теория принятия решений. Раздел: теория голосования – О. Д. Кичмаренко, А. П. Огуленко, ОНУ имени И.И.Мечникова, 2012 г.;

[2] Электронный ресурс: https://www.agroprodmash-expo.ru/ru/ui/17145/;

[3] Электронный ресурс: http://www.mesport.ru/index.php/business-and-economy/280-tipy selskokhozyaystvennykh-ferm;

[4] Электронный ресурс: https://ru.wikipedia.org/wiki/;

[5] Электронный ресурс: https://www.agro-ferma.ru/dayatelnost/ferma-krs/;

[6] Электронный ресурс: https://mshp.gov.by/documents/technical-acts/cca7d9e25148a014.html;

[7] Электронный ресурс: https://mshp.gov.by/documents/technical-acts/cb559517cff17d41.html;

[8] Электронный ресурс: https://bookonlime.ru/lecture/tema-xi-teoriya-obshchestvennogo-vybora;

[9] Электронный ресурс: https://baguzin.ru/wp/paradoks-kondorse-teorema-errou-ili-k/;

[10] Электронный ресурс: https://pages.fandom.com/ru/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%BE%D0%BA%D1%81\_%D0%9A%D0%BE%D0%BD%D0%B4%D0%BE%D1%80%D1%81%D0%B5;

[11] Электронный ресурс: https://www.oracle.com/java/technologies/javase-jdk13-downloads.html.

# ПРИЛОЖЕНИЕ А (Обязательное) Блок-схемы алгоритмов к разделу 7



Рисунок А.1 – Алгоритм решения задачи

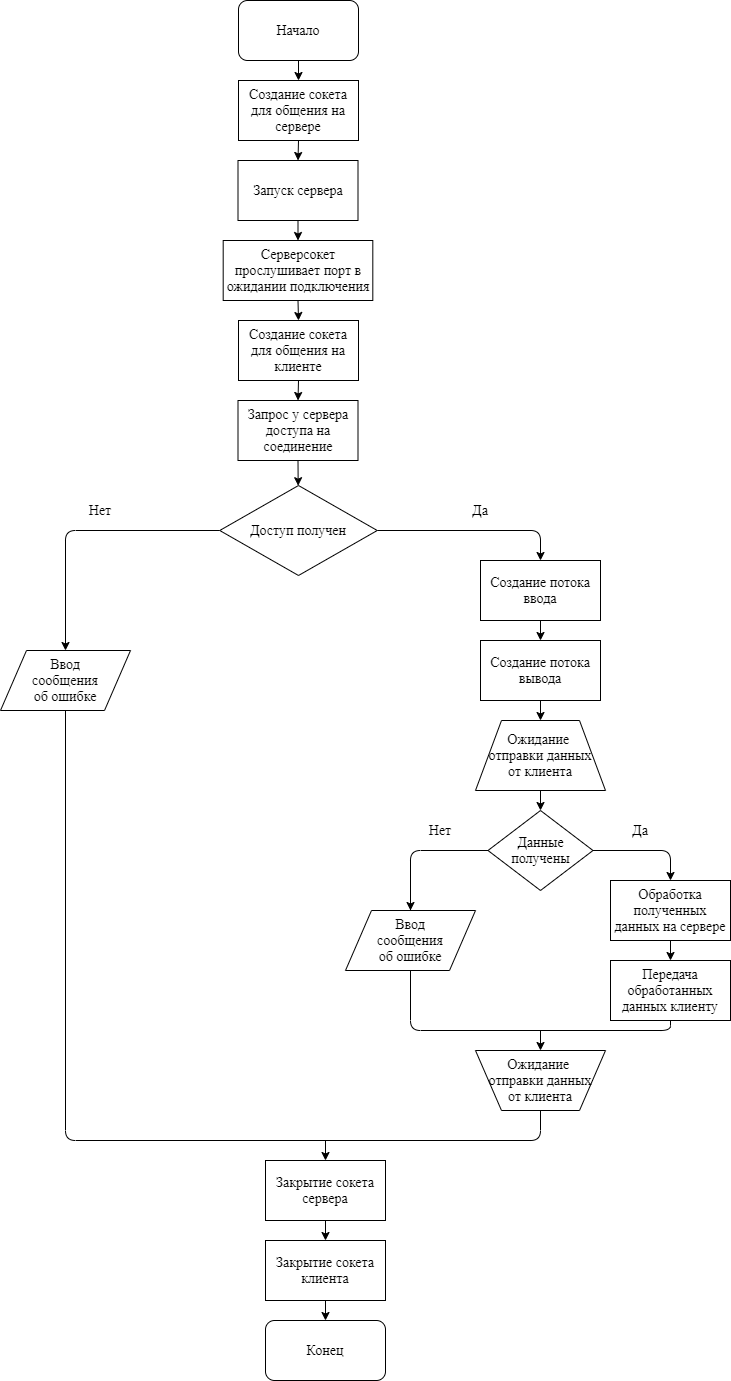


Рисунок А.2 – Алгоритм подключения

# ПРИЛОЖЕНИЕ Б (Обязательное) Листинг кода

Код класса «caesar», отвечающего за шифровку и дешифровку данных:

public class caesar {

public void shifr(StringBuffer str) {

String s = str.toString();

char[] sc = s.toCharArray();

for(int i = 0; i < s.length(); i++) {

int k = (int) sc[i];

k+=3;

sc[i] = (char) k;

}

str.delete(0, str.length());

str.append(sc);

}

public void deshifr(StringBuffer str) {

String s = str.toString();

char[] sc = s.toCharArray();

for(int i = 0; i < s.length(); i++) {

int k = (int) sc[i];

k-=3;

sc[i] = (char) k;

}

str.delete(0, str.length());

str.append(sc);

}

}

Код класса «methodCondorse», отвечающего за поиск наилучшей альтернативы методом Кондорсе:

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

public class methodCondorse {

public String[] S = new String[255];

public int n;

public int fl;

public String F = "C:\\Users\\Sveta\\Desktop\\универ\\2 курс\\2 семестр\\Курсач\\Server\\MethodCondorse.txt";

public void readFile() throws Exception{

FileReader fr = new FileReader(F);

BufferedReader reader = new BufferedReader(fr);

String line = reader.readLine();

n = 0;

if(line == null)

fl = 3;

while(line != null){

S[n] = line;

n++;

line = reader.readLine();

}

fr.close();

}

public void makeChoice(int E[][]) throws Exception{

fl = 0;

FileWriter fw = new FileWriter(F);

fw.write(" Список альтернатив\n");

fw.write("1. Закупить новое оборудование - a1.\n");

fw.write("2. Засеять пашни кормом - а2.\n");

fw.write("3. Нанять дополнительный персонал - а3.\n");

fw.write("4. Закупить 100 голов айрширской породы - а4.\n");

fw.write("5. Построить новые загоны для телят - а5.\n\n");

fw.write(" Список экспертов\n");

fw.write("1. Эксперт 1 - Э1.\n");

fw.write("2. Эксперт 2 - Э2.\n");

fw.write("3. Эксперт 3 - Э3.\n");

fw.write("4. Эксперт 4 - Э4.\n");

fw.write("5. Эксперт 5 - Э5.\n\n");

fw.write(" Составляется матрица ранжирования альтернатив экспертами:\n");

fw.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

fw.write("| Э1 | Э2 | Э3 | Э4 | Э5 |\n");

fw.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

String[][] a = new String[5][5];

for(int i = 0; i < 5; i++){

for(int j = 0; j < 5; j++) {

switch(E[i][j]){

case 1: a[i][j] = "a1";

break;

case 2: a[i][j] = "a2";

break;

case 3: a[i][j] = "a3";

break;

case 4: a[i][j] = "a4";

break;

case 5: a[i][j] = "a5";

break;

}

}

}

for(int i = 0; i < 5; i++){

for(int j = 0; j < 5; j++) {

fw.write("| "+a[i][j] + " ");

}

fw.write("|\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

fw.write(" Составляется матрица оценки Mik, показывающая предпочтение альтернатив в парных предпочтениях:\n");

fw.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

fw.write("| Mik | a1 | a2 | a3 | a4 | a5 |\n");

int[][] p = new int[5][5];

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

p[i][j] = 0;

for (int k = 0; k < 5; k++)

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

if (E[j][i] == (k + 1))

p[k][i] = j + 1;

int[][] m = new int[5][5];

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

m[i][j] = 0;

for (int k = 0; k < 5; k++){

for (int i = 0; i < 5; i++){

for (int j = 0; j < 5; j++){

if (p[k][j] < p[i][j] && i != k)

m[k][i]++;

}

}

}

fw.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

for (int i = 0; i < 5; i++){

fw.write("| а"+(i+1)+" |");

for (int j = 0; j < 5; j++){

if (i != j)

fw.write(" " + m[i][j] + " |");

else

fw.write(" |");

}

fw.write("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

fw.write("\n");

int[][] n = new int[5][2];

for (int i = 0; i < 5; i++){

for (int j = 0; j < 5; j++){

if (m[i][j] >= m[j][i] && i != j) {

n[i][0]++;

n[i][1] = i;

}

}

}

fw.write("Определяется наилучшая альтернатива.\n\n");

for (int j = 0; j < 5; j++)

{

for (int i = 4; i > j; i--)

{

if(n[i-1][0] < n[i][0]){

int[][] l = {{n[i-1][0], n[i-1][1]}};

n[i-1][0] = n[i][0];

n[i-1][1] = n[i][1];

n[i][0] = l[0][0];

n[i][1] = l[0][1];

}

}

}

for(int i = 0; i < 5; i++){

fw.write("Количество предпочтений для альтернативы "+ (n[i][1]+1) + " равно " + n[i][0] + ".\n");

}

fw.write("\nПо методу Кондорсе наилучшей альтернативой является \n" +

"альтернатива с наибольшим количеством предпочтений \n" +

"(если Mik >= Mki для любого k, не равного i).\n" +

"Отсюда следует, что наилучшей альтернативой в данном случае является\n");

fl = 1;

for(int i = 0; i < 5; i++){

if(n[0][0] == n[i][0]){

switch(n[i][1]+1){

case 1: fw.write("альтернатива a1 - Закупка нового оборудования.\n");

break;

case 2: fw.write("альтернатива a2 - Засев пашни кормом.\n");

break;

case 3: fw.write("альтернатива a3 - Наем дополнительного персонала.\n");

break;

case 4: fw.write("альтернатива a4 - Закупка 100 голов айрширской породы.\n");

break;

case 5: fw.write("альтернатива a5 - Постройка новых загонов для телят.\n");

break;

default: fl = 2;

break;

}

}

}

fw.write("\n\n\n");

fw.flush();

fw.close();

}

}

Код класса «guest», отвечающего за работу с пользователями (гостями):

import java.io.\*;

public class guest extends userTemplate{

public String[] l = new String[125];

public String[] p = new String[125];

public String[] S = new String[250];

public int fl;

public int n;

public String F = "C:\\Users\\Sveta\\Desktop\\универ\\2 курс\\2 семестр\\Курсач\\Server\\Guest.txt";

public guest() throws Exception{

File f = new File(F);

if(f.createNewFile()) {

System.out.println("Файл Guest успешно создан.");

}

readUser();

}

@Override

public void getUser(String login, String password) throws Exception {

readUser();

System.out.println("Происходит добавление логина и пароля нового пользователя в файл.");

fl = 0;

for(int i = 0; i < n/2; i++) {

if (login.equals(l[i])) {

fl = 2;

break;

}

}

if(fl == 0) {

FileWriter fw = new FileWriter(F, true);

fw.write(login + "\n" + password + "\n");

fw.flush();

fw.close();

fl = 1;

System.out.println("Новый пользователь добавлен.");

}

}

@Override

public void delUser(String login) throws Exception{

readUser();

fl = 0;

System.out.println("Происходит проверка логина пользователя на существование (удаление).");

for(int i = 0; i < n; i++) {

if(login.equals(S[i])){

File f = new File(F);

FileWriter fw = new FileWriter(F);

if(f.exists()){

System.out.println("Происходит удаление пользователя.");

for(int j = 0; j < n; j++) {

if(!S[i].equals(S[j]))

fw.write(S[j] + "\n");

}

fl = 1;

System.out.println("Пользователь успешно удален.");

fw.flush();

fw.close();

break;

}

}

else

fl = 2;

}

}

@Override

public void redUser(String login1, String login2) throws Exception {

readUser();

fl = 0;

System.out.println("Происходит проверка логина пользователя на существование (редактирование).");

for(int i = 0; i < n; i++) {

if(login1.equals(S[i])){

File f = new File(F);

FileWriter fw = new FileWriter(F);

if(f.exists()){

S[i] = login2;

System.out.println("Происходит редактирование пользователя.");

for(int j = 0; j < n; j++)

fw.write(S[j] + "\n");

fl = 1;

System.out.println("Пользователь успешно отредактирован.");

fw.flush();

fw.close();

break;

}

}

else

fl = 2;

}

}

@Override

public void readUser() throws Exception {

FileReader fr = new FileReader(F);

BufferedReader reader = new BufferedReader(fr);

String line = reader.readLine();

n = 0;

while(line != null){

S[n] = line;

n++;

line = reader.readLine();

}

int j = 0;

for(int i = 0; i < n; i++){

if(i%2 == 0) {

l[j] = S[i];

j++;

}

else

p[j] = S[i];

}

fr.close();

}

}

Код класса «milkFarm», отвечающего за работу с молочным отделом фермы:

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.util.\*;

public class milkFarm extends farmTemplate {

public int[] id = new int[255];

public String[] name = new String[255];

public String[] S = new String[255];

public int[] price = new int[255];

public int[] kol = new int[255];

public int fl;

public int n;

public int[] kp = new int[255];

List<String> list = new ArrayList<String>();

ListIterator<String> li = list.listIterator();

public String F = "C:\\Users\\Sveta\\Desktop\\универ\\2 курс\\2 семестр\\Курсач\\Server\\MilkFarm.txt";

public milkFarm() throws Exception{

File f = new File(F);

if(f.createNewFile()) {

System.out.println("Файл MilkFarm успешно создан.");

}

readFarm();

}

@Override

public void getFarm(String Name, String Price, String Kol) throws Exception{

System.out.println("Происходит добавление нового товара в файл.");

fl = 0;

for(int i = 0; i < n/2; i++) {

if (Name.equals(name[i])) {

fl = 2;

break;

}

if (Integer.parseInt(Price) == price[i]) {

fl = 3;

break;

}

}

if(fl == 0) {

int a = 0, b = 255;

int Id = a + (int) (Math.random() \* b);

for (int i = 0; i < n; i++) {

if (Id == id[i]) {

Id++;

i = 0;

}

}

FileWriter fw = new FileWriter(F, true);

fw.write(Name + " " + Price + " " + Kol + " " + Id + "\n");

fw.flush();

fw.close();

fl = 1;

System.out.println("Новый товар добавлен.");

}

}

@Override

public void delFarm(String Name) throws Exception{

fl = 0;

System.out.println("Происходит проверка названия товара на существование (удаление).");

for(int i = 0; i < n; i++) {

if(Name.equals(name[i])){

File f = new File(F);

FileWriter fw = new FileWriter(F);

if(f.exists()){

System.out.println("Происходит удаление товара.");

int j = 0;

while(li.hasNext()){

if(i != j)

fw.write(li.next() + "\n");

j++;

}

//for(int j = 0; j < n; j++) {

// if(i != j)

// fw.write(S[j] + "\n");

//}

fl = 1;

System.out.println("Товар успешно удален.");

fw.flush();

fw.close();

break;

}

}

else

fl = 2;

}

}

@Override

public void redFarm(String Name1, String Name2) throws Exception{

fl = 0;

System.out.println("Происходит проверка названия товара на существование (редактирование).");

for(int i = 0; i < n; i++) {

if(Name1.equals(name[i])){

File f = new File(F);

FileWriter fw = new FileWriter(F);

if(f.exists()){

name[i] = Name2;

System.out.println("Происходит редактирование названия товара.");

while(li.hasNext())

fw.write(li.next() + "\n");

fl = 1;

System.out.println("Имя товара успешно отредактировано.");

fw.flush();

fw.close();

break;

}

}

else

fl = 2;

}

}

@Override

public void readFarm() throws Exception{

list.clear();

FileReader fr = new FileReader(F);

BufferedReader reader = new BufferedReader(fr);

String line = reader.readLine();

n = 0;

while(line != null){

S[n] = line;

n++;

line = reader.readLine();

}

for (int j = 0; j < n; j++) {

String[] subS = S[j].split(" ");

name[j] = subS[0];

price[j] = Integer.parseInt(subS[1]);

kol[j] = Integer.parseInt(subS[2]);

id[j] = Integer.parseInt(subS[3]);

}

fr.close();

list.addAll(Arrays.asList(S).subList(0, n));

}

@Override

public void sortAlph() throws Exception{

Collections.sort(list);

System.out.println("Таблица была успешно отсортирована.\n");

}

@Override

public void sortAlphReverse() throws Exception{

Collections.sort(list, Collections.reverseOrder());

System.out.println("Таблица была успешно отсортирована.\n");

}