# Министерство образования Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВОЙ ПРОЕКТ**

по дисциплине **«Объектно-ориентированное программирование»**

на тему **«**Сайт для швейной фирмы**»**

Выполнил:

студент группы 10702217 Салапура Н.А.

Руководитель: В.В. Иванченко

Минск 2019

Содержание

[Введение 3](#_Toc8420326)

[1. Техническое задание на предмет разработки 4](#_Toc8420327)

[2. Обоснование используемых языков и технологий](#_Toc8420328) 6

[3. Моделирование и проектирование программного обеспечения](#_Toc8420329) 11

[4. Реализация программного обеспечения 1](#_Toc8420330)3

[5. Развёртывание и тестирование программного обеспечения 1](#_Toc8420331)6

[Заключение 1](#_Toc8420332)7

[Список использованных источников 1](#_Toc8420333)8

[Приложение 1](#_Toc8420334)9

# **ВВЕДЕНИЕ**

Любой организации необходимо графическое представление, чтобы покупатели могли за короткое время ознакомиться с предоставляемыми услугами, и узнать об обновлениях или переменах компании.

Для предприятий, задачей которых является производство потребительских товаров, очень важно иметь возможность ознакомить покупателя со всей своей продукцией.

Для таких целей может выступать как сайт, так и десктоп-приложение.

Также возможность менять выпускаемую продукцию удобна для руководящих лиц.

В данном курсовом проекте на JavaFX реализовано десктоп-приложение, представляющее из себя каталог для фирмы, производящей спецодежду, с возможностью редактирования продукции.

# **Техническое задание на предмет разработки**

Основным заданием курсового проекта является разработка приложения, реализующего каталог фирмы.

Анонимный пользователь имеет такие функции:

* Просмотр каталога
* Сортировка товаров по значению
* Фильтр товаров

Режим администратора включается непосредственно авторизацией, добавляются функции:

* Добавление товара
* Удаление товара
* Редактирование товара

Процесс авторизации осуществляется в новом окне. Некорректный ввод данных сопровождается «потряхиванием» полей.

После авторизации пользователь получает фукнции администратора (появляются компоненты редактирования каталога).

Администратор может выйти из своего режима и стать обычным пользователем-гостем.

Все данные, позволяющие реализовать функциональность данного приложения, хранятся в файле формата json.

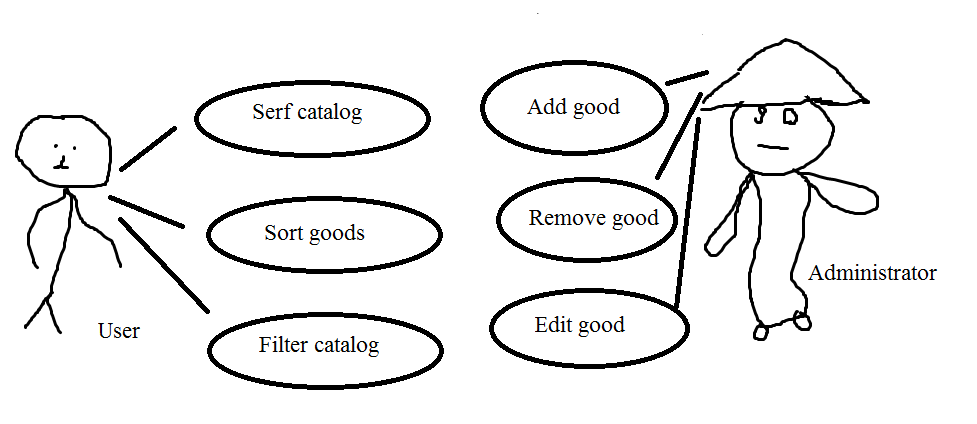


Рис. 1- use-case диаграмма

# **Обоснование используемых языков и технологий**

При разработке данного приложения был использован язык программирования Java и технология для разработки UI-приложений – JavaFX.

Java — это язык программирования общего назначения, который следует парадигме объектно-ориентированного программирования и подходу «Написать один раз и использовать везде». Java используется для настольных, сетевых, мобильных и корпоративных приложений.

Java-технологии имеют много особенностей, отличающие их от других технологий разработки программного обеспечения:

1. .**Переносимость**. Программы, написанные на языке Java, после однократной трансляции в байт-код могут быть исполнены на любой платформе, для которой реализована виртуальная Java-машина. Наиболее эффективно возможности реального компьютера может использовать только программа, написанная с использованием «родного» машинного кода.
2. **Безопасность.** Функционирование программы полностью определяется (и ограничивается) виртуальной Java-машиной. Отсутствуют указатели и другие механизмы для непосредственной работы с физической памятью и прочим аппаратным обеспечением компьютера. Дополнительные ограничения снижают возможность написания эффективно работающих Java-программ.
3. **Надежность.** В языке Java отсутствуют механизмы, потенциально приводящие к ошибкам: арифметика указателей, неявное преобразование типов с потерей точности и т.п. Присутствует строгий контроль типов, обязательный контроль исключительных ситуаций. Многие логические ошибки обнаруживаются на этапе компиляции. Наличие дополнительных проверок снижает эффективность выполнения Java-программ.
4. **Сборщик мусора.** Освобождение памяти при работе программы осуществляется автоматически с помощью «сборщика мусора», поэтому программировать с использованием динамически распределяемой памяти проще и надежнее. При интенсивной работе с динамически распределяемой памятью возможны ошибки из-за того, что «сборщик мусора» не успел освободить неиспользуемые области памяти.
5. **Стандартные библиотеки.** Многие задачи, встречающиеся при разработке программного обеспечения, уже решены в рамках стандартных библиотек. Использование объектно-ориентированного подхода позволяет легко использовать готовые объекты в своих программах. Для запуска приложения необходима установка JRE, содержащего полный набор библиотек, даже если все они не используются в приложении. Отсутствие библиотеки необходимой версии может воспрепятствовать запуску приложения.
6. **Самодокументируемый код.** Имеется механизм автоматического генерирования документации на основе комментариев, размещенных в тексте программ.
7. **Многообразие типов приложений.** На языке Java возможно реализовать абсолютно разные по способу функционированию и сфере использования программы.

**Java:**

* простой;
* объектно-ориентированный;
* распределенный;
* надежный;
* безопасный;
* не зависящий от архитектуры компьютера;
* переносимый;
* интерпретируемый;
* высокопроизводительный;
* многопоточный;

Технология **JavaFX** обеспечивает создание мощного графического интерфейса пользователя (Graphical User Interface (GUI)), 2D и 3D графику для крупномасштабных приложений, ориентированных на обработку данных, насыщенных медиа-приложений, поставляющих разнообразный медиа-контент пользователю, Mashup-приложений, объединяющих различные Web-ресурсы для пользователя, обеспечивает создание компонентов высококачественной графики и анимации для Web-сайтов, различного рода пользовательских программ, насыщенных графикой, анимацией и интерактивными элементами.

Все компоненты GUI-интерфейса являются объектами Node узлов графа сцены и характеризуются идентификатором, CSS-стилем, границами, визуальными эффектами, прозрачностью, трансформациями, обработчиками событий, состоянием, режимом наложения и участием в анимации.

**Основные возможности JavaFX:**

- Интегрированность с JRE/JDK.

- Альтернативное декларативное XML-описание GUI-интерфейса на языке FXML.

- Инструмент JavaFX Scene Builder для визуальной компоновки GUI-компонентов в GUI-интерфейс на основе языка FXML.

- За счет интегрированности с JRE/JDK и JRE-плагина Web-браузера один и тот же Java-код, созданный на базе платформы JavaFX, может запускаться как настольное приложение, которое разворачивается на клиентском компьютере автономно, может разворачиваться как Java Web Start приложение, или может отображаться в Web-браузере как JavaFX-апплет, встроенный в HTML-страничку.

- Изменение внешнего вида GUI-компонентов с использованием CSS.

- Встраивание HTML-контента в JavaFX-приложение с помощью компонента WebView с возможностью выполнения Javascript-кода и редактирование HTML-контента с помощью компонента HTMLEditor.

- Интеграция с библиотеками Swing и SWT.

- Богатый набор компонентов и компоновок компонентов для создания GUI-интерфейса.

- Встраивание аудио и видео контента в JavaFX-приложение с помощью компонентов MediaView и AudioClip.

- Использование визуальных эффектов, камеры, источников света, трансформаций и анимации.

При программирование проекта был соблюден шаблон программирования **MVC.**

**Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.**

Наиболее очевидным преимуществом модели Model-View-Controller является разделение проблем. Уровень представления, например, отвечает за представление данных пользователю. Уровни модели и контроллера не связаны с представлением данных.

Прямым преимуществом такого разделения проблем является повторное использование. Каждый из компонентов шаблона Model-View-Controller сфокусирован на конкретной задаче, что означает, что строительные блоки приложения MVC часто легко использовать повторно. Он также позволяет легко соединять эти компоненты, увеличивая их повторное использование.

Хранение всех данных приложения осуществляется в файлах формата **JSON.**

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

* Набор пар *ключ: значение*. В различных языках это реализовано как [запись](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%B8%D1%81%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), [структура](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [словарь](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), [хеш-таблица](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0), [список](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) с ключом или [ассоциативный массив](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Ключом может быть только строка (регистрозависимая: имена с буквами в разных регистрах считаются разными[[4]](https://ru.wikipedia.org/wiki/JSON#cite_note-4)), значением — любая форма.
* Упорядоченный набор *значений*. Во многих языках это реализовано как [массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [вектор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), список или [последовательность](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

Это универсальные структуры данных: как правило, любой современный язык программирования поддерживает их в той или иной форме. Они легли в основу JSON, так как он используется для обмена данными между различными языками программирования.

В качестве значений в JSON могут быть использованы:

* **Запись** — это неупорядоченное множество пар **ключ**:**значение**, заключённое в фигурные скобки *«{ }»*. Ключ описывается **строкой**, между ним и значением стоит символ *«:»*. Пары *ключ-значение* отделяются друг от друга запятыми.
* **Массив** (одномерный) — это упорядоченное множество **значений**. Массив заключается в квадратные скобки *«[ ]»*. Значения разделяются запятыми.
* **Число.**
* **Литералы** *true*, *false* и *null*.
* **Строка** — это упорядоченное множество из нуля или более символов [юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4), заключённое в двойные кавычки. Символы могут быть указаны с использованием [escape-последовательностей](https://ru.wikipedia.org/wiki/Escape-%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C), начинающихся с [обратной косой черты](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D1%81%D0%B0%D1%8F_%D1%87%D0%B5%D1%80%D1%82%D0%B0) *«\»* (поддерживаются варианты \', \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке [Unicode](https://ru.wikipedia.org/wiki/Unicode) в виде \uFFFF.

*Строка* очень похожа на одноимённый тип данных в языках [С](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) и [Java](https://ru.wikipedia.org/wiki/Java). *Число* тоже очень похоже на С- или Java-число, за исключением того, что используется только десятичный формат. Пробелы могут быть вставлены между любыми двумя синтаксическими элементами.

Также для модульного написания тестов использовался фреймворк **jUnit** версии 5.2.

jUnit применяется для модульного тестирования, которое позволяет проверять на правильность отдельные модули исходного кода программы. Преимущество данного подхода заключается в изолировании отдельно взятого модуля от других. При этом, цель такого метода позволяет программисту удостовериться, что модуль, сам по себе, способен работать корректно.

# **Моделирование и проектирование программного обеспечения**

Для реализации функциональности программы(созда ния и хранения товаров) были описаны классы-сущности Good, WorkingClothes, Suit, Footwear, ProtectionMean.

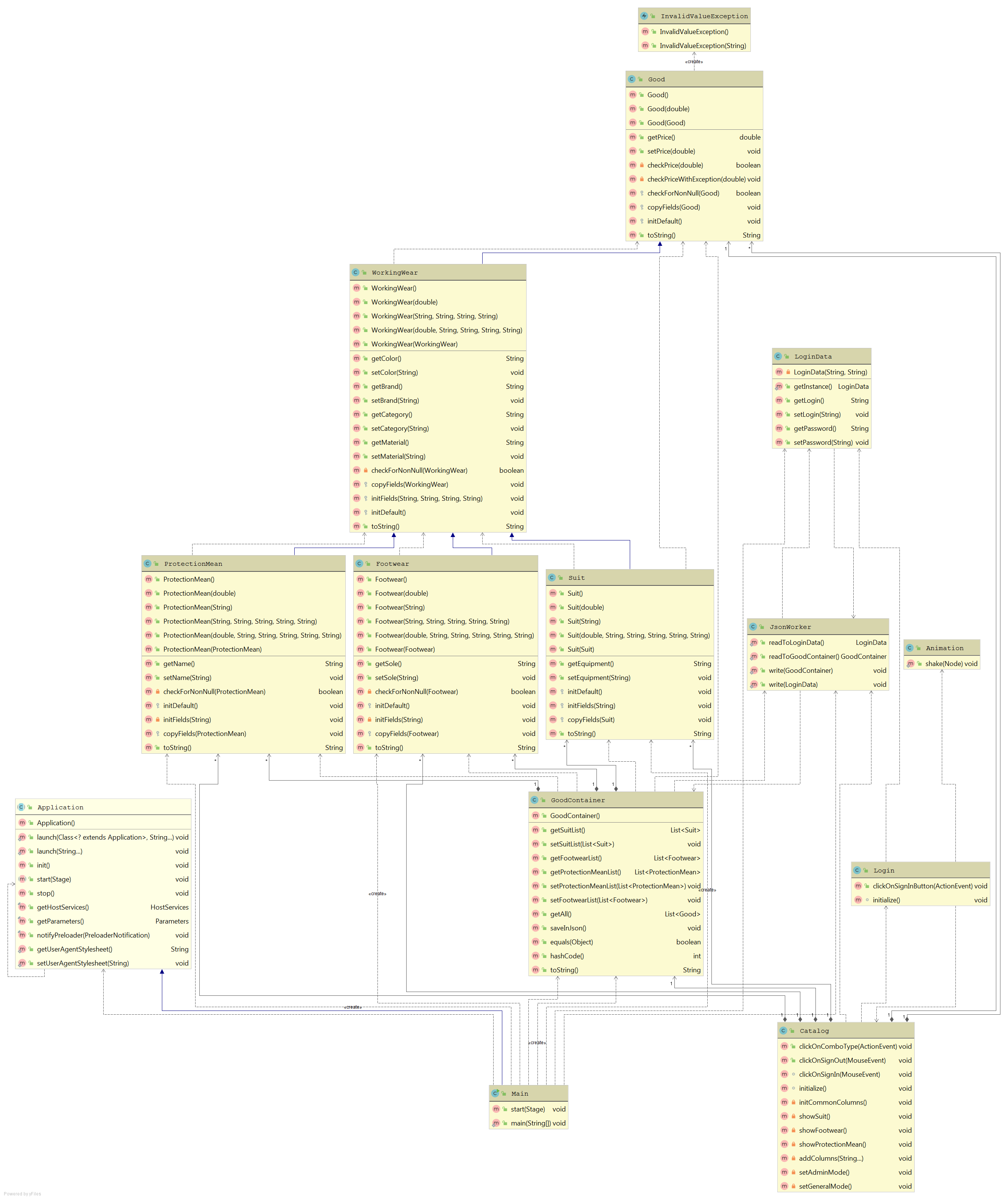


Рис. 2 - UML – диаграмма классов

# **Реализация программного обеспечения**

Стартовой точкой проекта является открытие формы и загрузки контроллера Сatalog. Пользователю предоставлено окно авторизации:

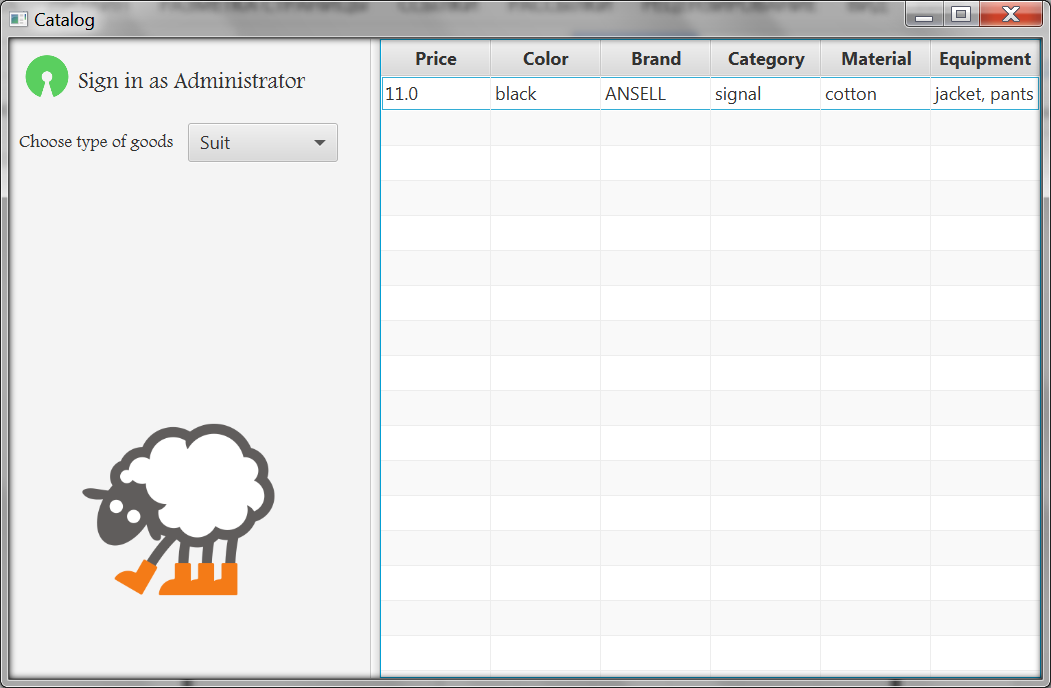


Рис. 3 – стартовое окно

Пользователь может выбирать тип продукции, а в правой части панели сразу показывается перечень таких товаров.

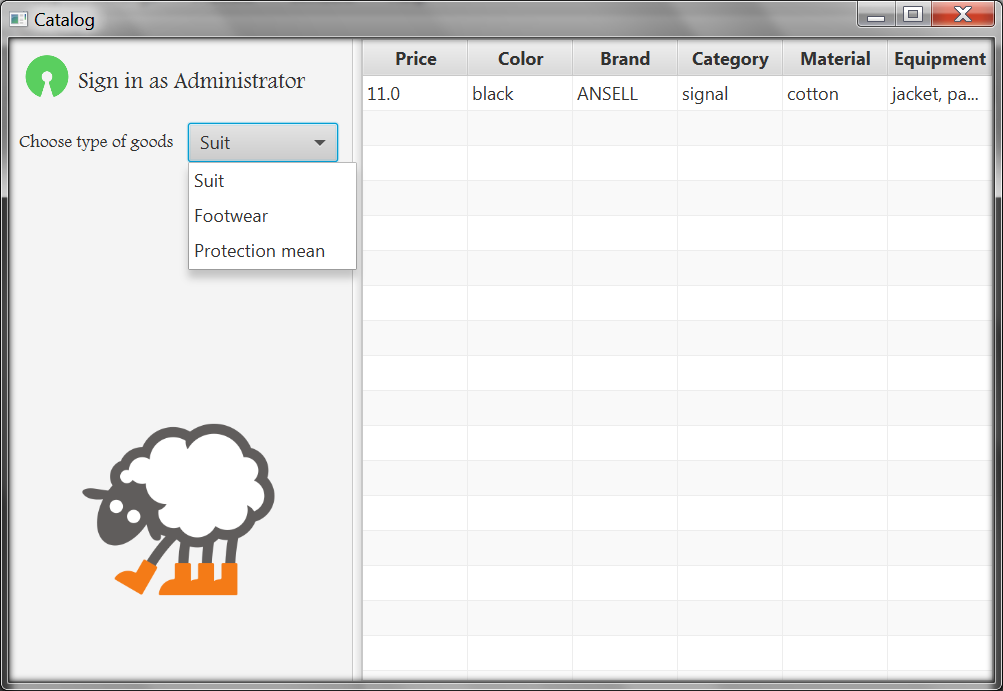


Рис.4 – Типы товаров

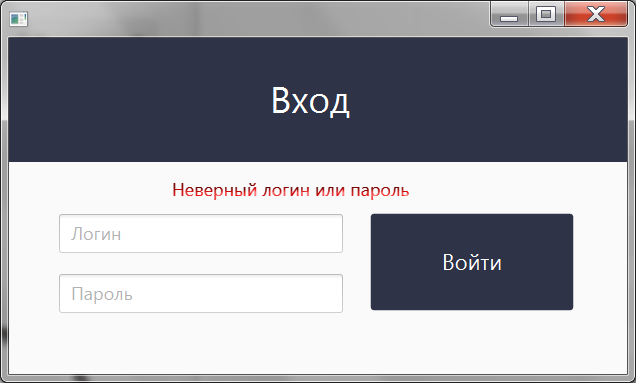


Рис. 5 – окно входа в режим администратора

При успешном прохождении авторизации пользователь в главное окно добавляется новый функционал

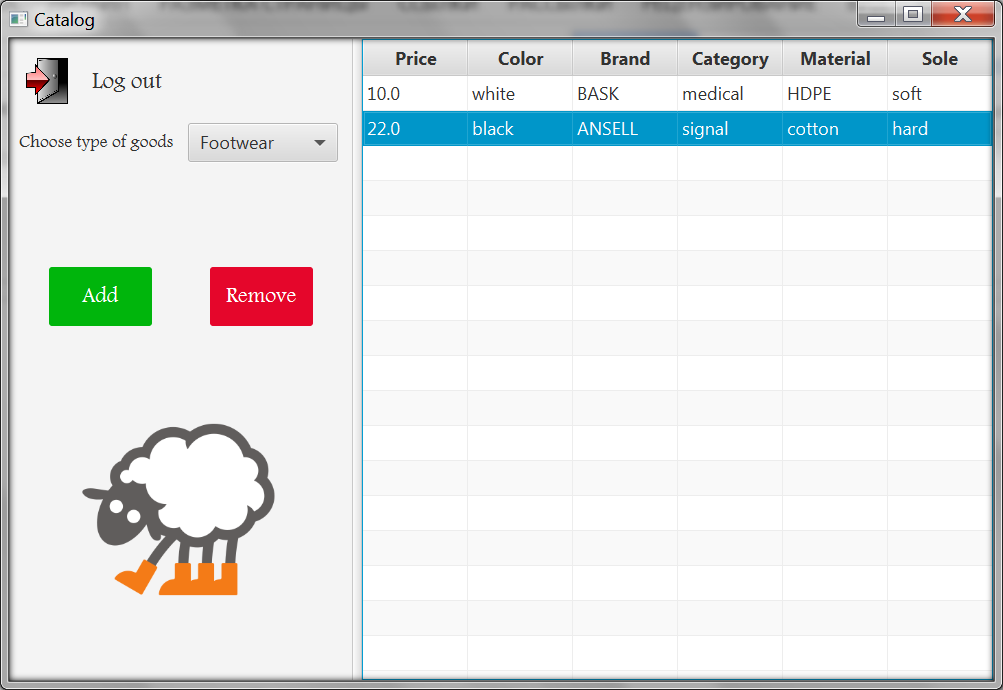


Рис. 6 – Режим администратора

Администратору лишь нужно выбрать строку и удалить, изменить в ней параметры или же добавить новый товар.

# **Развёртывание и тестирование программного обеспечения**

Развёртывание - это процесс упаковки и доставки программного обеспечения пользователю. Это важная часть процесса разработки ПО, так как здесь происходит первый контакт пользователя с приложением.

Развернуть java-приложение можно благодаря созданию исполняемого jar-архива.

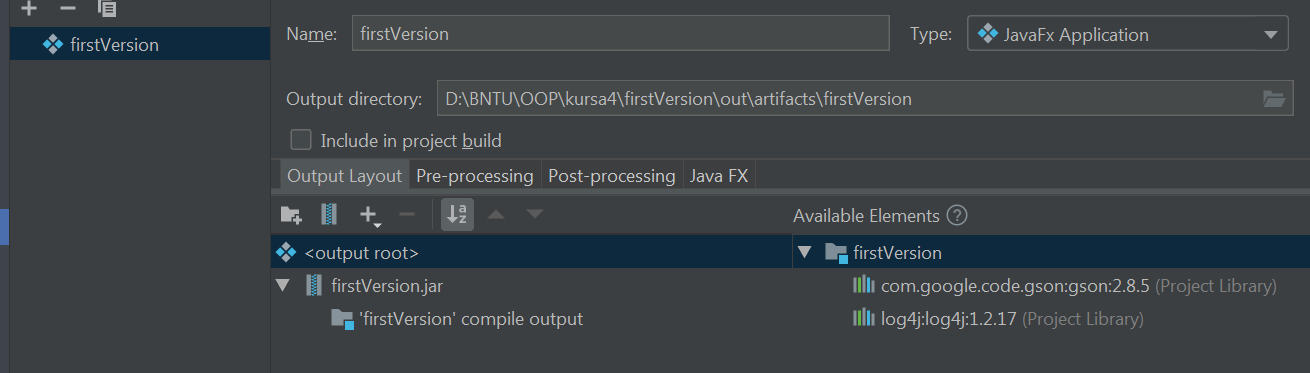


Рис. 7 – Создание jar-архива

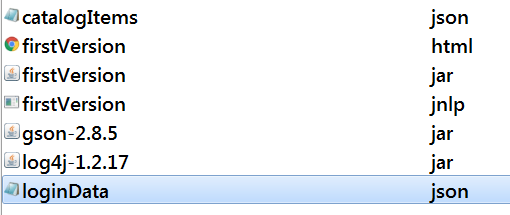


Рис. 8 - Структура аппаратного развёртывания приложения

Тестирование реализовано с использованием jUnit 4.0.

Модульное тестирование или unit testing — процесс проверки на корректность функционирования отдельных частей исходного кода программы путем запуска тестов в искусственной среде. Под частью кода в Java следует понимать исполняемый компонент. С помощью модульного тестирования обычно тестируют низкоуровневые элементы кода — такие, как методы. JUnit позволяет вне исследуемого класса создавать тесты, при выполнении которых произойдет корректное завершение программы. Кроме основного положительного сценария может выполняться проверка работоспособности системы в альтернативных сценариях, например, при генерации методом исключения как реакция на ошибочные исходные данные. Оценивая каждую часть кода изолированно и подтверждая корректность ее работы, точно установить проблему значительно проще, чем если бы элемент был частью системы. Технология позволяет и предлагает сделать более тесной связь между разработкой кода и его тестированием, а также предоставляет возможность проверить корректность работы класса, не прибегая к пробному выводу при отладке кода.

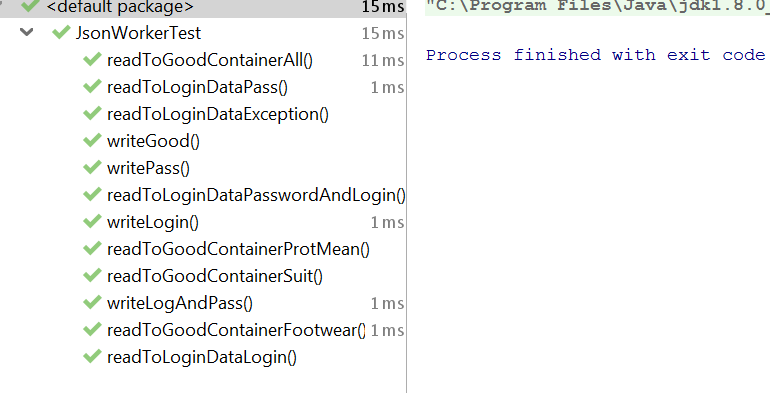


Рис 9- результат тестирования приложения

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта было спроектировано Desktop-UI-приложение, представляющее каталог организации. Были использованы и изучены json, javaFX, log4j, jUnit5, java.util.Collections и другие опции.

# **Список использованных источников**

1. <https://metanit.com/java/tutorial/3.8.php>.
2. <http://www.quizful.net/post/javafx-reloading>
3. <https://o7planning.org/ru/11079/javafx-tableview-tutorial>
4. <http://www.quizful.net/post/log4j-quickstart>
5. [http://qaru.site](http://qaru.site/)

# **Приложение**

**Class Main:**

**package** by.bntu.fitr.poisit.sleepwalker.controller;  
  
**import** by.bntu.fitr.poisit.sleepwalker.model.entity.\*;  
**import** by.bntu.fitr.poisit.sleepwalker.util.JsonWorker;  
**import** javafx.application.Application;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.stage.Stage;  
**import** org.apache.log4j.Logger;  
  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
**import** java.util.Arrays;  
  
**public class** Main **extends** Application {  
  
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception {  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"../view/fxml/mainWindow.fxml"**));  
 primaryStage.setTitle(**"Catalog"**);  
 primaryStage.setScene(**new** Scene(root));  
 primaryStage.show();  
 }  
  
 **public static void** main(String[] args) {  
  
 *launch*(args);  
 }  
}

**Class Catalog: package** by.bntu.fitr.poisit.sleepwalker.controller;  
  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
**import** java.net.URL;  
**import** java.util.\*;  
  
**import** by.bntu.fitr.poisit.sleepwalker.model.entity.\*;  
**import** by.bntu.fitr.poisit.sleepwalker.util.JsonWorker;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Node;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** javafx.scene.image.Image;  
**import** javafx.scene.image.ImageView;  
**import** javafx.scene.input.MouseEvent;  
**import** javafx.stage.Modality;  
**import** javafx.stage.Stage;  
  
**public class** Catalog {  
  
 **public static boolean** *isAdmin*;  
  
 **private** ObservableList<Suit> **suits**;  
 **private** ObservableList<Footwear> **footwears**;  
 **private** ObservableList<ProtectionMean> **protectionMeans**;  
 **private** ObservableList<Good> **goods**;  
  
 **private final** ObservableList<String> **typeList** = FXCollections  
 .*observableArrayList*(**"Suit"**, **"Footwear"**, **"Protection mean"**);  
  
 **private final** String[] **commonFields** = **new** String[]  
 {**"Price"**, **"Color"**, **"Brand"**, **"Category"**, **"Material"**};  
  
 **private** GoodContainer **goodContainer**;  
  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** ImageView **signInImage**;  
  
 @FXML  
 **private** ImageView **signOutImage**;  
  
 @FXML  
 **private** Label **logInLabel**;  
  
 @FXML  
 **private** Label **logOutLabel**;  
  
 @FXML  
 **private** Button **removeButton**;  
  
 @FXML  
 **private** Button **addButton**;  
  
 @FXML  
 **private** ComboBox<String> **comboType**;  
  
 @FXML  
 **private** TableView **goodTable**;  
  
 **private** TableColumn<Good, String>  
 **priceColumn**, **colorColumn**, **brandColumn**, **categoryColumn**, **materialColumn**,  
 **equipmentColumn**, **soleColumn**, **nameColumn**;  
  
*// private TableColumn<Good, Image> imageColumn;* **public void** clickOnComboType(ActionEvent actionEvent) {  
 **if** (**"Suit"**.equals(**comboType**.getValue())) {  
 showSuit();  
 } **else if** (**"Footwear"**.equals(**comboType**.getValue())) {  
*// addColumns("Sole");* showFootwear();  
 } **else if** (**"Protection mean"**.equals(**comboType**.getValue())) {  
*// addColumns("Name");* showProtectionMean();  
 }  
 }  
  
 @FXML  
 **public void** clickOnSignOut(MouseEvent mouseEvent) {  
 *isAdmin* = **false**;  
 initialize();  
 }  
  
 @FXML  
 **void** clickOnSignIn(MouseEvent event) {  
 FXMLLoader loader =  
 **new** FXMLLoader((getClass().getResource(**"../view/fxml/loginWindow.fxml"**)));  
 Parent root = **null**;  
 **try** {  
 root = loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 Stage stage = **new** Stage();  
  
 stage.initModality(Modality.***WINDOW\_MODAL***);  
 Node source = (Node) event.getSource();  
 stage.initOwner(source.getScene().getWindow());  
  
 stage.setScene(**new** Scene(root));  
 stage.showAndWait();  
 **if** (Login.*isAdmin*) initialize();  
 }  
  
 @FXML  
 **void** initialize() {  
 **if** (*isAdmin*) {  
 setAdminMode();  
 } **else** {  
 setGeneralMode();  
 }  
 **comboType**.setItems(**typeList**);  
 initCommonColumns();  
 }  
  
 **private void** initCommonColumns() {  
 **try** {  
 **goodContainer** = JsonWorker.*readToGoodContainer*();  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
*// priceColumn.setCellValueFactory(cellData ->  
// cellData.getValue().getPrice());* **priceColumn** = **new** TableColumn<>(**"Price"**);  
 **priceColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"price"**));  
  
 **colorColumn** = **new** TableColumn<>(**"Color"**);  
 **colorColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"Color"**));  
  
 **brandColumn** = **new** TableColumn<>(**"Brand"**);  
 **brandColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"brand"**));  
  
 **categoryColumn** = **new** TableColumn<>(**"Category"**);  
 **categoryColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"category"**));  
  
 **materialColumn** = **new** TableColumn<>(**"Material"**);  
 **materialColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"material"**));  
 **goodTable**.getColumns().addAll  
 (**priceColumn**, **colorColumn**, **brandColumn**, **categoryColumn**, **materialColumn**);  
  
 showSuit();  
  
*// for (String column : commonFields) {  
// goodTable.getColumns().add(new TableColumn<Good, String>(column));  
// }* }  
  
 **private void** showSuit() {  
 **goodTable**.getColumns().remove  
 (**commonFields**.**length**, **goodTable**.getColumns().size());  
 **equipmentColumn** = **new** TableColumn<>(**"Equipment"**);  
 **equipmentColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"equipment"**));  
 **goodTable**.getColumns().add(**equipmentColumn**);  
 **suits** = FXCollections.*observableList*(**goodContainer**.getSuitList());  
 **goodTable**.setItems(**suits**);  
 }  
  
 **private void** showFootwear() {  
 **goodTable**.getColumns().remove  
 (**commonFields**.**length**, **goodTable**.getColumns().size());  
 **soleColumn** = **new** TableColumn<>(**"Sole"**);  
 **soleColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"sole"**));  
 **goodTable**.getColumns().add(**soleColumn**);  
 **footwears** = FXCollections.*observableList*(**goodContainer**.getFootwearList());  
 **goodTable**.setItems(**footwears**);  
 }  
  
 **private void** showProtectionMean() {  
 **goodTable**.getColumns().remove  
 (**commonFields**.**length**, **goodTable**.getColumns().size());  
 **nameColumn** = **new** TableColumn<>(**"Name"**);  
 **nameColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"name"**));  
 **goodTable**.getColumns().add(**nameColumn**);  
 **protectionMeans** = FXCollections.*observableList*(**goodContainer**.getProtectionMeanList());  
 **goodTable**.setItems(**protectionMeans**);  
 }  
  
 **private void** addColumns(String... columnList) {  
 **goodTable**.getColumns().remove  
 (**commonFields**.**length**, **goodTable**.getColumns().size());  
 **for** (String column : columnList) {  
 TableColumn tableColumn = **new** TableColumn(column);  
*// tableColumn.setCellValueFactory(new PropertyValueFactory<>("price"));* **goodTable**.getColumns().add(tableColumn);  
 }  
 }  
  
 **private void** setAdminMode() {  
 **signInImage**.setVisible(**false**);  
 **logInLabel**.setVisible(**false**);  
 **signOutImage**.setVisible(**true**);  
 **logOutLabel**.setVisible(**true**);  
 **addButton**.setVisible(**true**);  
 **removeButton**.setVisible(**true**);  
 }  
  
 **private void** setGeneralMode() {  
 **signInImage**.setVisible(**true**);  
 **logInLabel**.setVisible(**true**);  
 **signOutImage**.setVisible(**false**);  
 **logOutLabel**.setVisible(**false**);  
 **addButton**.setVisible(**false**);  
 **removeButton**.setVisible(**false**);  
 }  
  
}

**Class Login:**

**package** by.bntu.fitr.poisit.sleepwalker.controller;  
  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
**import** java.net.URL;  
**import** java.util.ResourceBundle;  
  
**import** by.bntu.fitr.poisit.sleepwalker.model.entity.LoginData;  
**import** by.bntu.fitr.poisit.sleepwalker.util.Animation;  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.Label;  
**import** javafx.scene.control.PasswordField;  
**import** javafx.scene.control.TextField;  
**import** javafx.stage.Stage;  
  
**public class** Login {  
  
 **public static boolean** *isAdmin*;  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** Button **signInButton**;  
  
 @FXML  
 **private** TextField **loginField**;  
  
 @FXML  
 **private** PasswordField **passwordField**;  
  
 @FXML  
 **private** Label **wrongInputLabel**;  
  
 **public void** clickOnSignInButton(ActionEvent actionEvent) {  
 String loginText = **loginField**.getText().trim();  
 String passwordText = **passwordField**.getText().trim();  
 **try** {  
 **if** (loginText.equals(LoginData.*getInstance*().getLogin())  
 && passwordText.equals(LoginData.*getInstance*().getPassword())) {  
 Catalog.*isAdmin* = **true**;  
 *isAdmin* = **true**;  
 **signInButton**.getScene().getWindow().hide();  
  
*// FXMLLoader loader = new FXMLLoader();  
// loader.setLocation(getClass().getResource("../view/fxml/catalogEditor.fxml"));  
// try {  
// loader.load();  
// } catch (IOException e) {  
// e.printStackTrace();  
// }  
// Parent root = loader.getRoot();  
// Stage stage = new Stage();  
// stage.setScene(new Scene(root));  
// stage.showAndWait();* } **else** {  
 **wrongInputLabel**.setText(**"Wrong login or password"**);  
 Animation.*shake*(**loginField**);  
 Animation.*shake*(**passwordField**);  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 **void** initialize() {  
 *isAdmin* = **false**;  
 }  
}

**Class RegistrationController:**

package by.bntu.fitr.javaproject.ingoslingsfootsteps.controller;  
  
import by.bntu.fitr.javaproject.ingoslingsfootsteps.model.entity.Const;  
import by.bntu.fitr.javaproject.ingoslingsfootsteps.model.entity.User;  
import by.bntu.fitr.javaproject.ingoslingsfootsteps.model.logic.usersystem.RegistrationCheck;  
import by.bntu.fitr.javaproject.ingoslingsfootsteps.view.Displayer;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.scene.image.ImageView;  
import org.apache.log4j.Logger;  
  
import java.net.URL;  
import java.sql.SQLException;  
import java.util.ResourceBundle;  
  
public class RegistrationController {  
 private static final Logger *LOG*;  
 static{  
 *LOG* = Logger.*getLogger*(by.bntu.fitr.javaproject.ingoslingsfootsteps.controller.StartController.class);  
 }  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private ImageView personImage;  
  
 @FXML  
 private TextField usernameText;  
  
 @FXML  
 private ImageView passwordImage;  
  
 @FXML  
 private ImageView emailImage;  
  
 @FXML  
 private TextField emailText;  
  
 @FXML  
 private Button okButton;  
  
 @FXML  
 private PasswordField passwordText;  
  
 @FXML  
 private ImageView passwordImage1;  
  
 @FXML  
 private Label errorLabel;  
  
 @FXML  
 private PasswordField passwordText1;  
  
 @FXML  
 private Button cancelButton;  
  
 @FXML  
 void initialize() {  
 okButton.setOnAction(event -> {  
 try {  
 RegistrationCheck registration = new RegistrationCheck();  
 User user = new User();  
 if(registration.signUpNewUser(usernameText, passwordText, passwordText1, emailText, user)) {  
 Displayer.*setWindow*(Const.*PATH\_TO\_SAMPLE\_FXML*, okButton);  
 *LOG*.trace("User clicked <OK>");  
 *LOG*.info("User was registrated");  
 }  
 } catch (Exception e) {  
 errorLabel.setText(e.getMessage());  
 errorLabel.setVisible(true);  
 }  
 });  
 cancelButton.setOnAction(event -> {  
 try {  
 Displayer.*setWindow*(Const.*PATH\_TO\_SAMPLE\_FXML*, cancelButton);  
 *LOG*.trace("User clicked <CANCEL>");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 });  
 }  
}

**Class Footwear:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
  
**import** javafx.beans.property.SimpleStringProperty;  
  
**public class** Footwear **extends** WorkingWear {  
  
 **public static final** String ***DEFAULT\_SOLE*** = **"hard"**;  
  
 **protected** SimpleStringProperty **sole**;  
  
 **public** Footwear() {  
 initDefault();  
 }  
  
  
 **public** Footwear(**double** price) {  
 **super**(price);  
 initFields(***DEFAULT\_SOLE***);  
 }  
  
 **public** Footwear(String sole) {  
 **super**.initDefault();  
 initFields(sole);  
 }  
  
 **public** Footwear(String color,  
 String brand, String category, String material, String sole) {  
 **super**(color, brand, category, material);  
 initFields(sole);  
 }  
  
 **public** Footwear(**double** price, String color,  
 String brand, String category, String material, String sole) {  
 **super**(price, color, brand, category, material);  
 initFields(sole);  
 }  
  
 **public** Footwear(Footwear footwear) {  
 **if** (checkForNonNull(footwear)) {  
 copyFields(footwear);  
 } **else** {  
 initDefault();  
 }  
 }  
  
 **public** String getSole() {  
 **return sole**.get();  
 }  
  
 **public void** setSole(String sole) {  
 **this**.**sole**.set(sole);  
 }  
  
 **private boolean** checkForNonNull(Footwear footwear) {  
 **return** footwear != **null**;  
 }  
  
 **protected void** initDefault() {  
 **super**.initDefault();  
 **sole** = **new** SimpleStringProperty(***DEFAULT\_SOLE***);  
 }  
  
 **private void** initFields(String sole) {  
 **this**.**sole** = **new** SimpleStringProperty(sole);  
 }  
  
 **protected void** copyFields(Footwear footwear){  
 **super**.copyFields(footwear);  
 initFields(String.*valueOf*(footwear.**sole**));  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Footwear{"** +  
 **super**.toString() +  
 **"sole='"** + **sole** + **'\''** +  
 **'}'**;  
 }  
}

**Class Good:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
**import** by.bntu.fitr.poisit.sleepwalker.model.exception.InvalidValueException;  
**import** javafx.beans.property.SimpleDoubleProperty;  
**import** javafx.scene.image.Image;  
  
**import** javax.swing.text.Element;  
**import** javax.swing.text.html.ImageView;  
  
  
**public class** Good {  
  
 **public static final double *DEFAULT\_PRICE*** = 1;  
*// public static final String DEFAULT\_IMAGE = "D:\\BNTU\\OOP\\kursa4\\firstVersion\\src\\by\\bntu\\fitr\\poisit\\sleepwalker\\view\\images\\noPhoto.png";* **private static final** String ***INVALID\_PRICE\_MSG*** = **"Invalid price"**;  
  
 **protected** SimpleDoubleProperty **price**;  
*// protected ImageView image;* **public** Good() {  
 initDefault();  
 }  
  
 **public** Good(**double** price) {  
 **this**.**price** = **new** SimpleDoubleProperty(checkPrice(price) ? price : ***DEFAULT\_PRICE***);  
 }  
  
*// public Good(double price, ImageView image) {  
// this.price = new SimpleDoubleProperty(checkPrice(price) ? price : DEFAULT\_PRICE);  
// this.image = image;  
// }* **public** Good(Good good) {  
 **if** (checkForNonNull(good)) {  
 copyFields(good);  
 } **else** {  
 initDefault();  
 }  
 }  
  
 **public double** getPrice() {  
 **return price**.get();  
 }  
  
 **public void** setPrice(**double** price) **throws** InvalidValueException {  
 checkPriceWithException(price);  
 **this**.**price**.set(price);  
 }  
*//  
// public ImageView getImage() {  
// return image;  
// }  
//  
// public void setImage(ImageView image) {  
// this.image = image;  
// }* **private boolean** checkPrice(**double** price) {  
 **return** price > 0;  
 }  
  
 **private void** checkPriceWithException(**double** price) **throws** InvalidValueException {  
 **if** (!checkPrice(price)) {  
 **throw new** InvalidValueException(***INVALID\_PRICE\_MSG***);  
 }  
 }  
  
 **protected boolean** checkForNonNull(Good good) {  
 **return** good != **null**;  
 }  
  
 **protected void** copyFields(Good good) {  
 **price** = good.**price**;  
 }  
  
 **protected void** initDefault() {  
 **price** = **new** SimpleDoubleProperty(***DEFAULT\_PRICE***);  
*// image = new ImageView((Element) new Image(DEFAULT\_IMAGE));* }  
  
 @Override  
 **public** String toString() {  
 **return "price="** + **price** + **", "**;  
 }  
}

**Class GoodContainer:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
**import** by.bntu.fitr.poisit.sleepwalker.util.JsonWorker;  
  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.Objects;  
  
**public class** GoodContainer {  
  
  
 **private** List<Suit> **suitList**;  
 **private** List<Footwear> **footwearList**;  
 **private** List<ProtectionMean> **protectionMeanList**;  
  
 **public** GoodContainer() {  
 **suitList** = **new** ArrayList<>();  
 **footwearList** = **new** ArrayList<>();  
 **protectionMeanList** = **new** ArrayList<>();  
 }  
  
 **public** List<Suit> getSuitList() {  
 **return suitList**;  
 }  
  
 **public void** setSuitList(List<Suit> suitList) {  
 **this**.**suitList** = suitList;  
 }  
  
 **public** List<Footwear> getFootwearList() {  
 **return footwearList**;  
 }  
  
 **public** List<ProtectionMean> getProtectionMeanList() {  
 **return protectionMeanList**;  
 }  
  
 **public void** setProtectionMeanList(List<ProtectionMean> protectionMeanList) {  
 **this**.**protectionMeanList** = protectionMeanList;  
 }  
  
 **public void** setFootwearList(List<Footwear> footwearList) {  
 **this**.**footwearList** = footwearList;  
 }  
  
 **public** List<Good> getAll(){  
 List<Good> all = **new** ArrayList<>(**suitList**);  
 all.addAll(**footwearList**);  
 all.addAll(**protectionMeanList**);  
 **return** all;  
 }  
  
 **public void** saveInJson() **throws** IOException {  
 JsonWorker.*write*(**this**);  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (o == **null** || getClass() != o.getClass()) **return false**;  
 GoodContainer that = (GoodContainer) o;  
 **return** Objects.*equals*(**suitList**, that.**suitList**) &&  
 Objects.*equals*(**footwearList**, that.**footwearList**) &&  
 Objects.*equals*(**protectionMeanList**, that.**protectionMeanList**);  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return** Objects.*hash*(**suitList**, **footwearList**, **protectionMeanList**);  
 }  
  
 @Override  
 **public** String toString() {  
 **return "GoodContainer{"** +  
 **"suitList="** + **suitList** +  
 **", footwearList="** + **footwearList** +  
 **", protectionMeanList="** + **protectionMeanList** +  
 **'}'**;  
 }  
}

**Class LoginData:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
**import** by.bntu.fitr.poisit.sleepwalker.util.JsonWorker;  
  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
  
**public class** LoginData {  
  
 **public final static** String ***DEFAULT\_DATA*** = **"admin"**;  
  
 **private static** LoginData *loginData*;  
  
 **private** String **login**;  
 **private** String **password**;  
  
 **private** LoginData(String login, String password) {  
 **this**.**login** = login;  
 **this**.**password** = password;  
 }  
  
 **public static** LoginData getInstance() **throws** FileNotFoundException {  
 **if** (*loginData* == **null**) {  
 *loginData* = JsonWorker.*readToLoginData*();  
 }  
 **return** *loginData*;  
 }  
  
 **public** String getLogin() {  
 **return login**;  
 }  
  
 **public void** setLogin(String login) **throws** IOException {  
 **this**.**login** = login;  
 JsonWorker.*write*(*loginData*);  
 }  
  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public void** setPassword(String password) **throws** IOException {  
 **this**.**password** = password;  
 JsonWorker.*write*(*loginData*);  
 }  
}

**Class ProtectionMean:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
**import** javafx.beans.property.SimpleStringProperty;  
  
**public class** ProtectionMean **extends** WorkingWear {  
 **public static final** String ***DEFAULT\_NAME*** = **"helmet"**;  
  
 **protected** SimpleStringProperty **name**;  
  
 **public** ProtectionMean() {  
 initDefault();  
 }  
  
 **public** ProtectionMean(**double** price) {  
 **super**(price);  
 initFields(***DEFAULT\_NAME***);  
 }  
  
 **public** ProtectionMean(String name) {  
 **super**.initDefault();  
 initFields(name);  
 }  
  
 **public** ProtectionMean(String color,  
 String brand, String category, String material, String name) {  
 **super**(color, brand, category, material);  
 initFields(name);  
 }  
  
 **public** ProtectionMean(**double** price, String color,  
 String brand, String category, String material, String name) {  
 **super**(price, color, brand, category, material);  
 initFields(name);  
 }  
  
 **public** ProtectionMean(ProtectionMean protectionMean) {  
 **if** (checkForNonNull(protectionMean)) {  
 copyFields(protectionMean);  
 } **else** {  
 initDefault();  
 }  
 }  
  
 **public** String getName() {  
 **return name**.get();  
 }  
  
 **public void** setName(String name) {  
 **this**.**name**.set(name);  
 }  
  
 **private boolean** checkForNonNull(ProtectionMean protectionMean) {  
 **return** protectionMean != **null**;  
 }  
  
 **protected void** initDefault() {  
 **super**.initDefault();  
 **name** = **new** SimpleStringProperty(***DEFAULT\_NAME***);  
 }  
  
 **private void** initFields(String name) {  
 **this**.**name** = **new** SimpleStringProperty(name);  
 }  
  
  
 **protected void** copyFields(ProtectionMean protectionMean){  
 **super**.copyFields(protectionMean);  
 initFields(String.valueOf(protectionMean.name));  
 }  
  
 @Override  
 **public** String toString() {  
 **return "ProtectionMean{"** +  
 **super**.toString() +  
 **"name='"** + name + **'\''** +  
 **'}'**;  
 }  
}

**Class Suit:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
**import** by.bntu.fitr.poisit.sleepwalker.model.exception.InvalidValueException;  
**import** javafx.beans.property.SimpleStringProperty;  
  
**public class** Suit **extends** WorkingWear {  
  
 **public static final** String ***DEFAULT\_EQUIPMENT*** = **"jacket, pants"**;  
  
 **protected** SimpleStringProperty equipment;  
  
 **public** Suit() {  
 initDefault();  
 }  
  
 **public** Suit(**double** price) {  
 **super**(price);  
 initFields(***DEFAULT\_EQUIPMENT***);  
 }  
  
 **public** Suit(String equipment) {  
 **super**.initDefault();  
 initFields(equipment);  
 }  
  
 **public** Suit(**double** price, String color,  
 String brand, String category, String material, String equipment) {  
 **super**(price, color, brand, category, material);  
 initFields(equipment);  
 }  
  
 **public** Suit(Suit suit){  
 **if** (checkForNonNull(suit)){  
 copyFields(suit);  
 } **else**{  
 initDefault();  
 }  
 }  
  
 **public** String getEquipment() {  
 **return** equipment.get();  
 }  
  
 **public void** setEquipment(String equipment) {  
 **this**.equipment.set(equipment);  
 }  
  
 **protected void** initDefault() {  
 **super**.initDefault();  
 equipment = **new** SimpleStringProperty(DEFAULT\_EQUIPMENT);  
 }  
  
 **protected void** initFields(String equipment) {  
 **this**.equipment = **new** SimpleStringProperty(equipment);  
 }  
  
 **protected void** copyFields(Suit suit){  
 **super**.copyFields(suit);  
 initFields(String.valueOf(suit.equipment));  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Suit{"** +  
 **super**.toString() +  
 **"equipment="** + equipment +  
 **'}'**;  
 }  
}

**Class WorkingWear:**

**package** by.bntu.fitr.poisit.sleepwalker.model.entity;  
  
**import** javafx.beans.property.SimpleStringProperty;  
  
  
**public class** WorkingWear **extends** Good {  
  
 **public static final** String ***DEFAULT\_COLOR*** = **"black"**;  
 **public static final** String ***DEFAULT\_BRAND*** = **"ANSELL"**;  
 **public static final** String ***DEFAULT\_CATEGORY*** = **"signal"**;  
 **public static final** String ***DEFAULT\_MATERIAL*** = **"cotton"**;  
  
 **protected** SimpleStringProperty **color**;  
 **protected** SimpleStringProperty **brand**;  
 **protected** SimpleStringProperty **category**;  
 **protected** SimpleStringProperty **material**;  
  
 **public** WorkingWear() {  
 initDefault();  
 }  
  
 **public** WorkingWear(**double** price) {  
 **super**(price);  
 }  
  
 **public** WorkingWear(String color,  
 String brand, String category, String material) {  
 **super**.initDefault();  
 initFields(color, brand, category, material);  
 }  
  
 **public** WorkingWear(**double** price, String color,  
 String brand, String category, String material) {  
 **super**(price);  
 initFields(color, brand, category, material);  
 }  
  
 **public** WorkingWear(WorkingWear workingWear) {  
 **if** (checkForNonNull(workingWear)) {  
 copyFields(workingWear);  
 } **else** {  
 initDefault();  
 }  
 }  
  
 **public** String getColor() {  
 **return color**.get();  
 }  
  
 **public void** setColor(String color) {  
 **this**.**color**.set(color);  
 }  
  
 **public** String getBrand() {  
 **return brand**.get();  
 }  
  
 **public void** setBrand(String brand) {  
 **this**.**brand**.set(brand);  
 }  
  
 **public** String getCategory() {  
 **return category**.get();  
 }  
  
 **public void** setCategory(String category) {  
 **this**.**category**.set(category);  
 }  
  
 **public** String getMaterial() {  
 **return material**.get();  
 }  
  
 **public void** setMaterial(String material) {  
 **this**.**material**.set(material);  
 }  
  
 **private boolean** checkForNonNull(WorkingWear workingWear) {  
 **return** workingWear != **null**;  
 }  
  
 **protected void** copyFields(WorkingWear workingWear) {  
 **super**.copyFields(workingWear);  
 **this**.**color** = workingWear.**color**;  
 **this**.**brand** = workingWear.**brand**;  
 **this**.**category** = workingWear.**category**;  
 **this**.**material** = workingWear.**material**;  
 }  
  
 **protected void** initFields(String color,  
 String brand, String category, String material) {  
 **this**.**color** = **new** SimpleStringProperty(color);  
 **this**.**brand** = **new** SimpleStringProperty(brand);  
 **this**.**category** = **new** SimpleStringProperty(category);  
 **this**.**material** = **new** SimpleStringProperty(material);  
 }  
  
  
 **protected void** initDefault() {  
 **super**.initDefault();  
 initFields(***DEFAULT\_COLOR***,***DEFAULT\_BRAND***,***DEFAULT\_CATEGORY***,***DEFAULT\_MATERIAL***);  
 }  
  
 @Override  
 **public** String toString() {  
 **return super**.toString() +  
 **"color='"** + **color** + **'\''** +  
 **", brand='"** + **brand** + **'\''** +  
 **", category='"** + **category** + **'\''** +  
 **", brand='"** + **brand** + **"\', "**;  
 }  
}

**Class InvalidValueException:**

**package** by.bntu.fitr.poisit.sleepwalker.model.exception;  
  
**public class** InvalidValueException **extends** Exception {  
 **public** InvalidValueException() {  
 **super**();  
 }  
  
 **public** InvalidValueException(String message) {  
 **super**(message);  
 }  
}

**Class Animation:**

**package** by.bntu.fitr.poisit.sleepwalker.util;  
  
**import** javafx.animation.TranslateTransition;  
**import** javafx.scene.Node;  
**import** javafx.util.Duration;  
  
**public class** Animation {  
  
 **public static void** shake(Node node){  
 TranslateTransition tt = **new** TranslateTransition(Duration.*millis*(70), node);  
 tt.setFromX(0f);  
 tt.setByX(10f);  
 tt.setCycleCount(4);  
 tt.setAutoReverse(**true**);  
 tt.playFromStart();  
 }  
}

**Class JsonWorker:**

**package** by.bntu.fitr.poisit.sleepwalker.util;  
  
**import** by.bntu.fitr.poisit.sleepwalker.model.entity.GoodContainer;  
**import** by.bntu.fitr.poisit.sleepwalker.model.entity.LoginData;  
**import** com.google.gson.Gson;  
**import** com.google.gson.stream.JsonReader;  
  
**import** java.io.FileNotFoundException;  
**import** java.io.FileReader;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
  
**public class** JsonWorker {  
  
 **private final static** String ***PATH\_TO\_FILE\_OF\_GOOD\_CONTAINER*** = **"d:\\BNTU\\OOP\\kursa4\\firstVersion\\catalogItems.json"**;  
  
 **private final static** String ***PATH\_TO\_FILE\_OF\_LOGIN\_DATA*** = **"d:\\BNTU\\OOP\\kursa4\\firstVersion\\loginData.json"**;  
  
 **public static** LoginData readToLoginData() **throws** FileNotFoundException {  
 JsonReader reader = **new** JsonReader(**new** FileReader(***PATH\_TO\_FILE\_OF\_LOGIN\_DATA***));  
 **return new** Gson().fromJson(reader, LoginData.**class**);  
 }  
  
 **public static** GoodContainer readToGoodContainer() **throws** FileNotFoundException {  
 JsonReader reader = **new** JsonReader(**new** FileReader(***PATH\_TO\_FILE\_OF\_GOOD\_CONTAINER***));  
 **return new** Gson().fromJson(reader, GoodContainer.**class**);  
 }  
  
 **public static void** write(GoodContainer context) **throws** IOException {  
 Gson gson = **new** Gson();  
 FileWriter writer = **new** FileWriter(***PATH\_TO\_FILE\_OF\_GOOD\_CONTAINER***);  
 String string = gson.toJson(context);  
 writer.write(string);  
 writer.close();  
 }  
  
 **public static void** write(LoginData loginData) **throws** IOException {  
 Gson gson = **new** Gson();  
 FileWriter writer = **new** FileWriter(***PATH\_TO\_FILE\_OF\_LOGIN\_DATA***);  
 String string = gson.toJson(loginData);  
 writer.write(string);  
 writer.close();  
 }  
}

**Class log4j:**

*# Уровень логирования***log4j.rootLogger**=**ALL, file***# Апендер для работы с файлами***log4j.appender.file**=**org.apache.log4j.RollingFileAppender***# Путь где будет создаваться лог файл***log4j.appender.file.File**=**d:\\BNTU\\OOP\\kursa4\\firstVersion\\logger.log***# Указываем максимальный размер файла с логами***log4j.appender.file.MaxFileSize**=**20MB***# Конфигурируем шаблон вывода логов в файл***log4j.appender.file.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern**=**%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n**

**Class mainWindow.fxml:**

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.ComboBox***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.SplitPane***?>  
<?***import javafx.scene.control.TableView***?>  
<?***import javafx.scene.effect.InnerShadow***?>  
<?***import javafx.scene.image.Image***?>  
<?***import javafx.scene.image.ImageView***?>  
<?***import javafx.scene.layout.AnchorPane***?>  
<?***import javafx.scene.layout.VBox***?>  
<?***import javafx.scene.text.Font***?>*<**VBox prefHeight="644.0" prefWidth="985.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="by.bntu.fitr.poisit.sleepwalker.controller.Catalog"**>  
 <**children**>  
 <**SplitPane dividerPositions="0.35300101729399797" prefHeight="664.0" prefWidth="977.0"**>  
 <**items**>  
 <**AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="642.0" prefWidth="366.0"**>  
 <**children**>  
 <**Label layoutX="9.0" layoutY="84.0" prefHeight="39.0" prefWidth="162.0" text="Choose type of goods" AnchorPane.leftAnchor="9.0" AnchorPane.rightAnchor="171.0"**>  
 <**font**>  
 <**Font name="Andalus" size="18.0"** />  
 </**font**></**Label**>  
 <**ComboBox fx:id="comboType" layoutX="178.0" layoutY="84.0" onAction="#clickOnComboType" prefWidth="150.0" promptText="Suit" AnchorPane.rightAnchor="14.0"** />  
 <**ImageView fx:id="signInImage" fitHeight="79.0" fitWidth="46.0" layoutX="14.0" layoutY="14.0" onMouseClicked="#clickOnSignIn" pickOnBounds="true" preserveRatio="true"**>  
 <**image**>  
 <**Image url="@../images/login.png"** />  
 </**image**>  
 </**ImageView**>  
 <**Label fx:id="logInLabel" layoutX="82.0" layoutY="23.0" text="Log in as Administrator"**>  
 <**font**>  
 <**Font name="Andalus" size="23.0"** />  
 </**font**></**Label**>  
 <**ImageView fitHeight="343.0" fitWidth="723.0" layoutX="-1.0" layoutY="299.0" pickOnBounds="true" preserveRatio="true"**>  
 <**image**>  
 <**Image url="@../images/logo.png"** />  
 </**image**>  
 </**ImageView**>  
 <**Button fx:id="removeButton" layoutX="215.0" layoutY="228.0" mnemonicParsing="false" prefHeight="58.0" prefWidth="103.0" style="-fx-background-color: #e5062b;" text="Remove" textFill="WHITE" visible="false" AnchorPane.rightAnchor="39.0"**>  
 <**font**>  
 <**Font name="Andalus" size="22.0"** />  
 </**font**>  
 </**Button**>  
 <**Button fx:id="addButton" layoutX="39.0" layoutY="228.0" mnemonicParsing="false" prefHeight="58.0" prefWidth="103.0" style="-fx-background-color: #00b50c;" text="Add" textFill="WHITE" visible="false" AnchorPane.leftAnchor="39.0"**>  
 <**font**>  
 <**Font name="Andalus" size="22.0"** />  
 </**font**>  
 </**Button**>  
 <**ImageView fx:id="signOutImage" fitHeight="46.0" fitWidth="46.0" layoutX="16.0" layoutY="19.0" onMouseClicked="#clickOnSignOut" pickOnBounds="true" preserveRatio="true" visible="false"**>  
 <**image**>  
 <**Image url="@../images/logout.png"** />  
 </**image**>  
 </**ImageView**>  
 <**Label fx:id="logOutLabel" layoutX="82.0" layoutY="24.0" text="Log out" visible="false"**>  
 <**font**>  
 <**Font name="Andalus" size="23.0"** />  
 </**font**>  
 </**Label**>  
 </**children**>  
 </**AnchorPane**>  
 <**AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="494.0" prefWidth="459.0"**>  
 <**children**>  
 <**TableView fx:id="goodTable" layoutX="47.0" layoutY="25.0" prefHeight="642.0" prefWidth="631.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0"**>  
 <**columnResizePolicy**>  
 <**TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY"** />  
 </**columnResizePolicy**></**TableView**>  
 </**children**></**AnchorPane**>  
 </**items**>  
 <**effect**>  
 <**InnerShadow** />  
 </**effect**>  
 </**SplitPane**>  
 </**children**>  
</**VBox**>

**Class loginWindow:**

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.Cursor***?>  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.PasswordField***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.effect.Light.Distant***?>  
<?***import javafx.scene.effect.Lighting***?>  
<?***import javafx.scene.effect.Shadow***?>  
<?***import javafx.scene.layout.AnchorPane***?>  
<?***import javafx.scene.text.Font***?>*<**AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="336.0" prefWidth="618.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="by.bntu.fitr.poisit.sleepwalker.controller.Login"**>  
 <**children**>  
 <**AnchorPane layoutX="1.0" layoutY="124.0" prefHeight="223.0" prefWidth="666.0" style="-fx-background-color: #fafafa;" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="124.0"**>   
 <**children**>  
 <**TextField fx:id="loginField" layoutX="50.0" layoutY="52.0" prefHeight="36.0" prefWidth="284.0" promptText="Логин" AnchorPane.leftAnchor="50.0" AnchorPane.rightAnchor="284.0" AnchorPane.topAnchor="52.0"** />  
 <**PasswordField fx:id="passwordField" layoutX="50.0" layoutY="112.0" prefHeight="36.0" prefWidth="284.0" promptText="Пароль" AnchorPane.leftAnchor="50.0" AnchorPane.rightAnchor="284.0" AnchorPane.topAnchor="112.0"** />  
 <**Button fx:id="signInButton" layoutX="362.0" layoutY="52.0" mnemonicParsing="false" onAction="#clickOnSignInButton" prefHeight="96.0" prefWidth="202.0" style="-fx-background-color: #2E3348;" text="Войти" textFill="WHITE" AnchorPane.rightAnchor="54.0" AnchorPane.topAnchor="52.0"**>  
 <**font**>  
 <**Font size="22.0"** />  
 </**font**></**Button**>  
 <**Label fx:id="wrongInputLabel" layoutX="163.0" layoutY="14.0" prefHeight="27.0" prefWidth="292.0" textAlignment="CENTER" textFill="RED" AnchorPane.leftAnchor="163.0" AnchorPane.rightAnchor="163.0" AnchorPane.topAnchor="14.0"**>  
 <**effect**>  
 <**Lighting diffuseConstant="1.1" specularConstant="0.53" specularExponent="24.05" surfaceScale="6.84"**>  
 <**bumpInput**>  
 <**Shadow** />  
 </**bumpInput**>  
 <**light**>  
 <**Light.Distant** />  
 </**light**>  
 </**Lighting**>  
 </**effect**>  
 </**Label**>  
 </**children**>  
 </**AnchorPane**>  
 <**Label layoutX="261.0" layoutY="35.0" text="Вход" textFill="WHITE" AnchorPane.rightAnchor="276.0" AnchorPane.topAnchor="35.0"**>  
 <**font**>  
 <**Font size="37.0"** />  
 </**font**>  
 </**Label**>  
 </**children**>  
 <**cursor**>  
 <**Cursor fx:constant="DEFAULT"** />  
 </**cursor**>  
</**AnchorPane**>