Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа программной инженерии

Курсовая работа  
по дисциплине «Технологии разработки качественного программного обеспечения»

Тема: Фитнес-трекер

Выполнил  
студент гр. в3530904/70321 Т.Д. Кирсанов

Руководитель Н.Г. Смирнов

Санкт-Петербург

2020

Оглавление

[**Задание** 3](#_Toc41705449)

[**Фитнес-Трекер** 3](#_Toc41705450)

[**Применённые Архитектуры** 4](#_Toc41705451)

[**Scene Builder** 8](#_Toc41705452)

[**Реализация программы** 9](#_Toc41705453)

[**Вывод** 14](#_Toc41705454)

[**Список литературы** 14](#_Toc41705455)

[**Код программы** 15](#_Toc41705456)

## **Задание**

Приложение позволяет указать вид тренировки: отжимание, скакалка, приседания.  
Доступны команды: начать тренировку (запускается таймер тренировки),  
закончить тренировку (таймер останавливается).  
За отработанное время высчитывается количество потраченных  
калорий по формуле K\*t=cal, где K - количество калорий в час, затрачиваемое  
на определенный вид тренировки, t - время, засеченное трекером.  
От запуска к запуску программы данные должны сохраняться и общее количество калорий - суммироваться.  
Персистенция данных приложения с помощью ObjectOutputStream +5 баллов или JAXB + 10 баллов  
Поддержка нескольких профилей пользователей приложением + 5 баллов

## **Фитнес-Трекер**

Фитнес трекер также известный как Трекер активности, это устройство или приложение предназначенное для мониторинга показателей связанных с фитнесом. Таких как: пройденное расстояние, потребление калорий, показатели сердечного ритма и качества сна. Является одним из видов носимого компьютера. В настоящее время этот термин чаще всего используется для умных часов, которые синхронизируются со смартфоном или компьютером для отслеживания данных, и выполняют роль трекера активности. Также существуют мобильные приложения для отслеживания фитнес активности. Некоторые исследователи утверждают, что использование таких устройств не приводит к значительному уменьшению веса человека.

## **Применённые Архитектуры**

**JavaFX**

JavaFX — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing. Платформа JavaFX конкурирует с Microsoft Silverlight, Adobe Flash и аналогичными системами.

Начиная с версии Java 11 больше не входит в Java SE и не разрабатывается компанией Oracle (как отдельный модуль поддерживается компанией Gluon)[1]. Но Oracle будет вносить необходимые изменения до 2022 года как для части Java SE 8[2].

Версия 1.0

Технология JavaFX была впервые продемонстрирована корпорацией Sun Microsystems на конференции JavaOne в мае 2007 года. 4 декабря 2008 года вышла версия 1.0 платформы, содержащая следующие компоненты:

Средства разработки — компилятор и среда исполнения JavaFX, язык программирования JavaFX Script, а также графические, медийные и веб-библиотеки для создания RIA-приложений для настольных компьютеров, веб-сайтов и мобильных устройств.

Интегрированная среда разработки NetBeans IDE (версии 6.\*) — средства для кодирования и отладки приложений, написанных на JavaFX Script. В редакторе JavaFX Script есть возможность быстрого добавления объектов JavaFX с уже готовыми геометрическими фигурами, компонентами интерфейса пользователя, средствами преобразования и анимацией.

Production Suite — набор инструментов и плагинов для импорта графических объектов в приложения JavaFX. Включает следующие компоненты:

Плагины для графических редакторов Adobe Photoshop CS3, CS4 и Adobe Illustrator CS3, CS4. С помощью плагинов можно экспортировать графические объекты из этих приложений в код JavaFX Script.

Media Factory: набор инструментов для конвертирования SVG-графики в код JavaFX и просмотра графических объектов, импортированных в JavaFX из других форматов. Также включает примеры приложений, учебные курсы, статьи, API-документацию и примеры кода.

Windows 7 x86 и x86-64 SP1 2.0

Mac OS X 64-бит 10.7.3+ 2.2

Для создания приложений для мобильных устройств разработчикам предлагается мобильный эмулятор, входящий в состав SDK.

**История изменений**

**JavaFX 1.2**

Дата выпуска — 2 июня 2009 года.

Значительно переработана предыдущая версия. API и синтаксис были изменены настолько, что стали несовместимы с предыдущими версиями. Из этого следует, что для работы программ, скомпилированных для предыдущих версий, для работы с текущей необходима их перекомпиляция.

Улучшена поддержка CSS.

Улучшены элементы графического интерфейса предыдущей версии, добавлены новые.

Улучшена производительность JavaFX Runtime.

В плагине к среде разработки NetBeans 6.8 — улучшена поддержка разработки приложений для мобильных устройств.

В плагине к среде разработки NetBeans 6.8 улучшен редактор кода.

В JavaFX Production Suite добавлены опции.

**JavaFX 1.3**

Вышел 22 апреля 2010 года.

Добавлены новые элементы графического интерфейса.

Улучшена поддержка CSS.

Улучшены элементы графического интерфейса предыдущей версии.

Улучшена производительность JavaFX Runtime.

Добавлена поддержка разработки приложений для TV (в плагин к среде разработки NetBeans 6.9 включён соответствующий эмулятор).

В плагине к среде разработки NetBeans 6.9 улучшена поддержка разработки приложений для мобильных устройств. На Mac OC теперь есть возможность использовать эмулятор мобильных устройств.

В плагине к среде разработки NetBeans 6.9 улучшен редактор кода (расширен список возможного рефакторинга).

В JavaFX Production Suite добавлены опции.

**JavaFX 1.3.1**

Выпущен 21 августа 2010 года.

Ускорен запуск приложений JavaFX.

Новый процесс запуска JavaFX-апплетов и Web-приложений, которые могут быть настроены разработчиками.

JavaFX 2.0

Вышел 10 октября 2011 года.

Медиа-движок, поддерживающий воспроизведение мультимедийного контента.

Веб-компонент, позволяющий встраивать HTML в приложения JavaFX.

Обновлённый плагин для браузера, который осуществляет загрузку апплетов JavaFX на основе Prism.

Набор элементов управления пользовательского интерфейса, такие, как диаграммы, таблицы, меню и панели.

Для декларативного описания пользовательских интерфейсов применён XML подобный язык разметки — FXML.

Примеры приложений, демонстрирующие технологии JavaFX 2.0.

**JavaFX 2.1**

Дата выпуска 27 апреля 2012 года

Поддержка H.264 и AAC.

Официальная поддержка Mac OS X.

LCD text.

Улучшены компоненты пользовательского интерфейса: Combo box, диаграмма и панель меню.

Компонент Webview теперь позволяет посредством JavaScript вызывать Java-методы.

**JavaFX 2.2**

Дата выпуска 14 августа 2012 года

Поддержка воспроизведения H.264 и AAC.

Поддержка Linux (включая плагин для браузера и WebStart-приложения).

Canvas.

Новые компоненты управления: Color Picker, Pagination.

Поддержка HTTP Live Streaming (не будет работать в Windows XP).

Сенсорные события и жесты.

API для манипуляции над изображением.

**JavaFX 8**

Дата выпуска 18 марта 2014 года

JavaFX стал частью пакетов JRE/JDK вместе с версией Java 8.

Поддержка 3D графики.

Поддержка сенсорного ввода.

Поддержка печати.

## **Scene Builder**

Scene Builder - это интерактивный инструмент проектирования графического интерфейса для JavaFX .

Программное обеспечение доступно в двух версиях: одна (8.x) для JavaFX 8 и другая (9.0 и +) для JavaFX 9 и более.

**История**

Программное обеспечение было впервые выпущено на 14 августа 2012 г.по Oracle как JavaFX Scene Builder 1 .

От 3 декабря 2013 г.программное обеспечение публикуется с открытым исходным кодом, как часть проекта OpenJFX 2 .

Наконец, Gluon объявляет о поддержке непрерывности разработки программного обеспечения 3 и публикует первую версию8 марта 2015 г.на Github .

**Операция**

Программное обеспечение используется с техникой перетаскивания: боковая панель слева от окна позволяет пользователю выбирать и размещать компонент в иерархии компонентов или в центральном виде, отображающем рендеринг.

Боковая панель, расположенная справа от окна, позволяет вам определить характеристики компонента, выбранного пользователем.

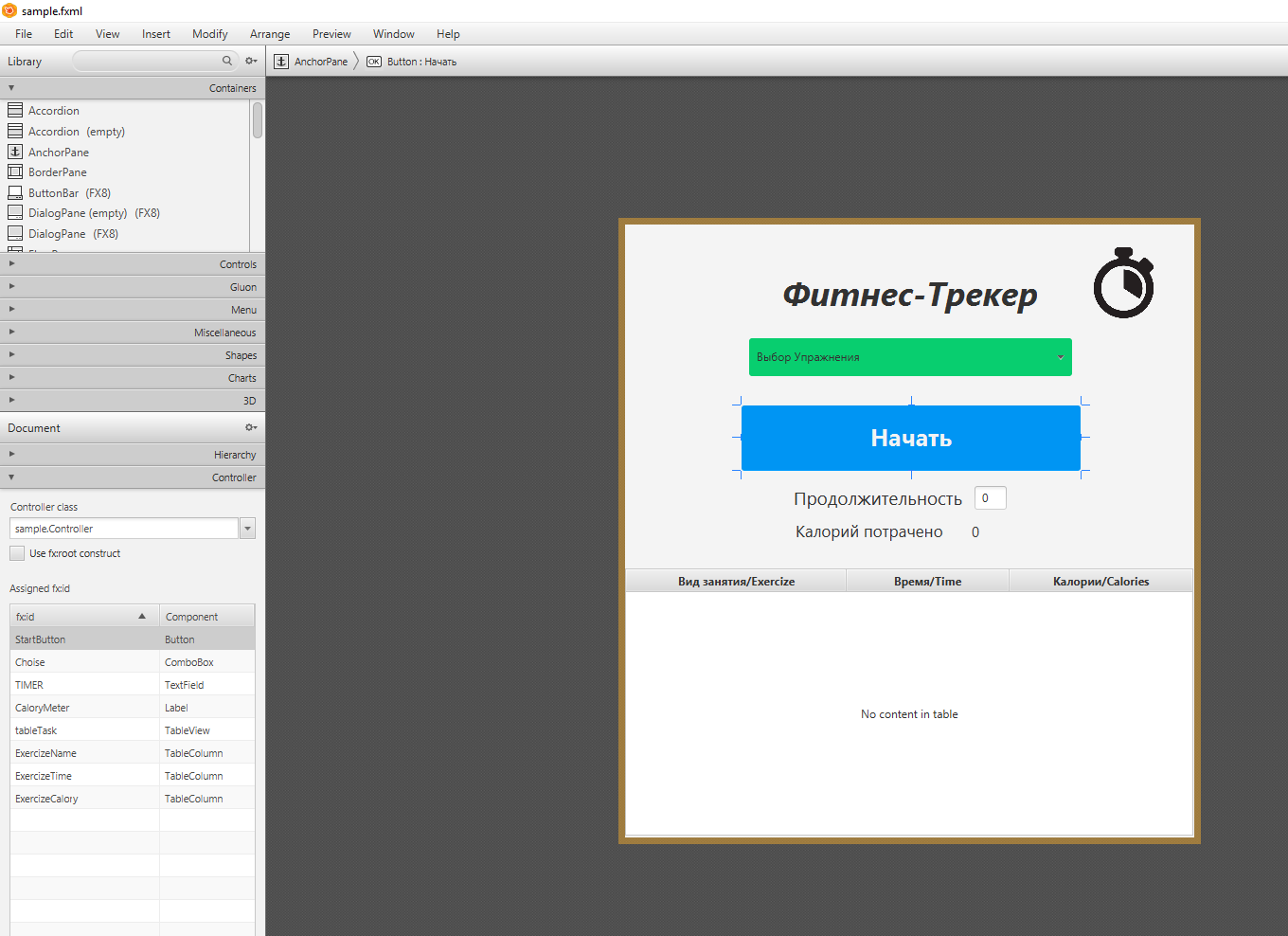
Интерфейсы сохраняются в файлах в формате FXML , которые затем считываются и загружаются в память программой, а затем отображаются на экране.

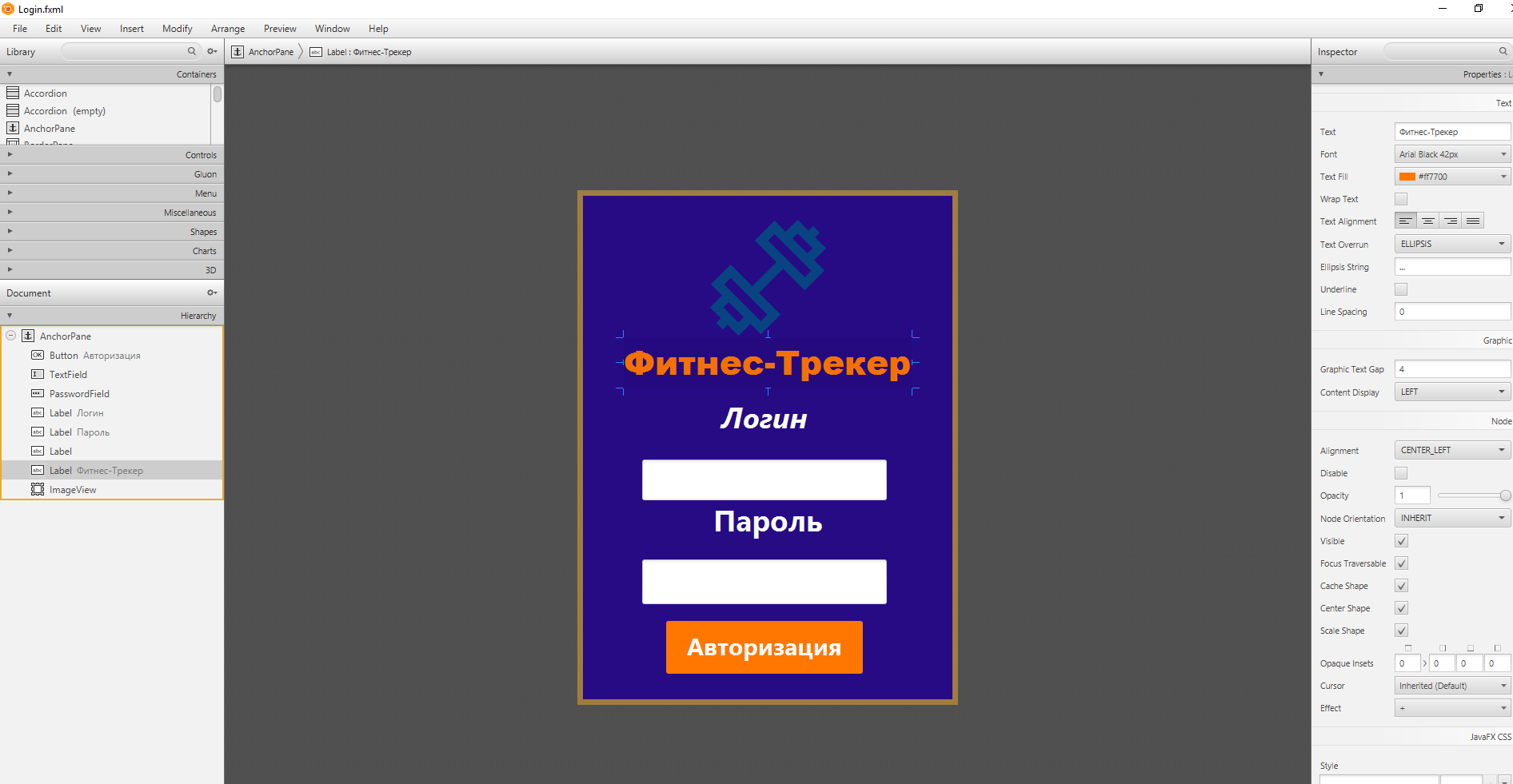
## **Реализация программы**

В ходе выполнения задания применялась архитектура JavaFX и Scene Builder.

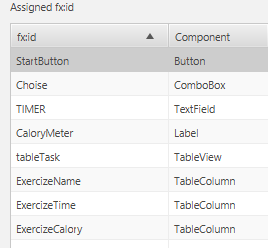
В первую очередь создаются необходимые FXML Файлы. Через них будет осуществляться создание интерфейса при помощи Scene Builder.

В моём случае было разработано два интерфейса. Один для основного окна, а другой для авторизации.

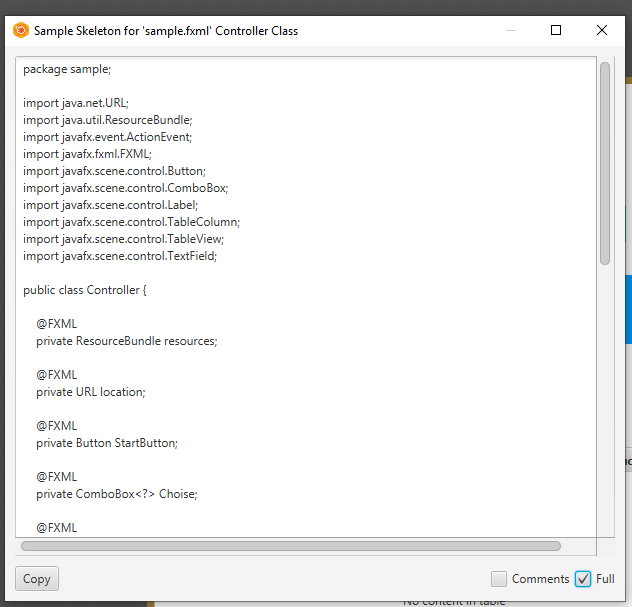


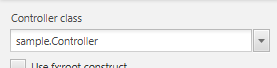


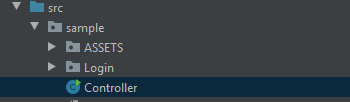
Под эти интерфейсы создаётся определённый функционал.



И на основе этого функционала в нашей программе можно создать классы, в которых мы сможем его инициализировать и прописать для интерфейсов дальнейшую логику.

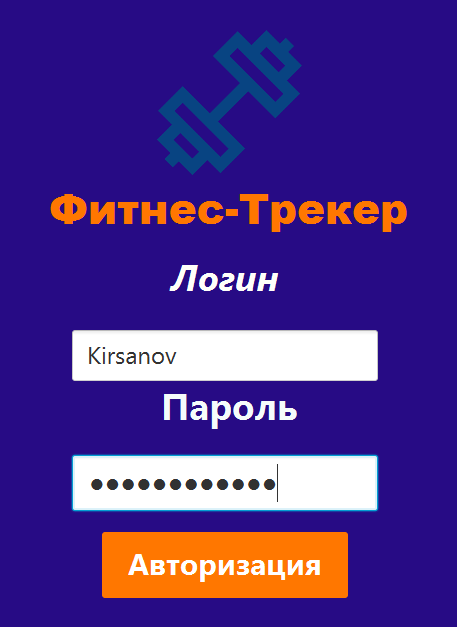


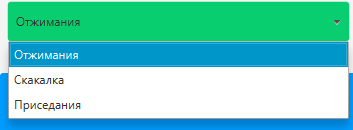




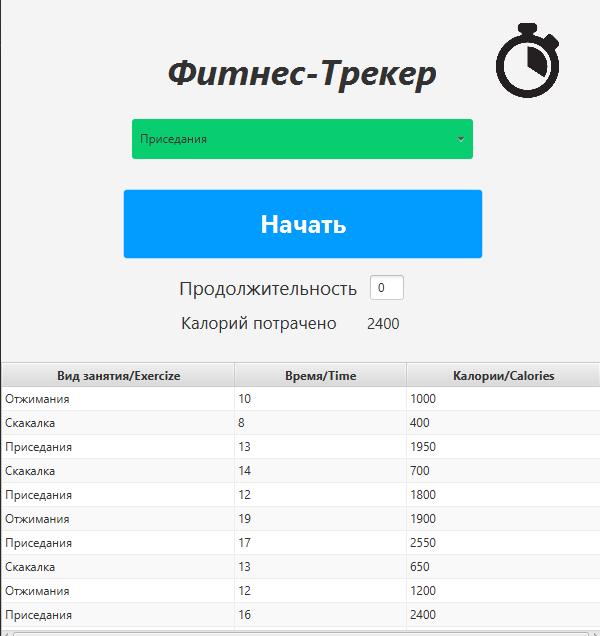
К примеру в основном классе Controller будет реализовываться основной интерфейс. Все изменения в sample.FXML будут непосредственно сказываться на основном классе.

В конечном итоге было реализовано окно авторизации, через которое пользователь может перейти к основному окну тренировок.





В основном окне можно выбирать один из трёх видов тренировок, которые и требовались для нашего задания.



Пройденные тренировки сохраняются посредством сериализации и десериализации.

## **Вывод**

В ходе данной курсовой работы были приобретены практические навыки в создании интерфейсов на базе JavaFX с использованием Scene Builder, их инициализации непосредственно в классах и прописывании их программной логики.

## **Список литературы**

docs.oracle.com [электронный ресурс]– -<https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html>

ru.wikipedia.org [электронный ресурс] - <https://ru.wikipedia.org/wiki/JavaFX>

metanit.com [электронный ресурс] - <https://metanit.com/java/javafx/1.1.php>

www.oracle.com [электронный ресурс] - https://www.oracle.com/java/technologies/javase/javafx-overview.html

javarush.ru [электронный ресурс] https://javarush.ru/groups/posts/2560-vvedenie-v-java-fx

## **Код программы**

package sample;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
  
  
import javax.xml.bind.annotation.XmlAttribute;  
import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlRootElement;  
  
import java.net.URL;  
import java.util.ResourceBundle;  
import javafx.fxml.Initializable;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.control.cell.PropertyValueFactory;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.io.OutputStream;  
import java.io.FileReader;  
import javax.print.attribute.standard.JobOriginatingUserName;  
import javax.xml.bind.JAXBContext;  
import javax.xml.bind.Marshaller;  
import javax.xml.bind.Unmarshaller;  
import javax.swing.\*;  
import java.awt.\*;  
  
import java.io.File;  
import java.io.FileReader;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.List;  
import java.io.File;  
import javax.xml.bind.JAXBContext;  
import javax.xml.bind.JAXBException;  
import javax.xml.bind.Marshaller;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.util.ArrayList;  
import java.util.List;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import java.util.Timer;  
import java.util.jar.JarException;  
import java.io.\*;  
import java.io.Serializable;  
import java.util.stream.Collectors;  
  
import javax.xml.bind.annotation.\*;  
  
  
/\*  
\*  
\* //Инициализация инструментария  
\*  
\* \*/  
  
public class Controller implements Initializable {  
 @FXML  
 public ComboBox<String> Choise;  
 @FXML  
 private Button StartButton;  
 @FXML  
 private TextField TIMER;  
 @FXML  
 private Label CaloryMeter;  
 @FXML  
 private TableView<Journal> tableTask;  
 @FXML  
 private TableColumn<Journal, String> ExercizeName;  
 @FXML  
 private TableColumn<Journal, Integer> ExercizeTime;  
 @FXML  
 private TableColumn<Journal, Integer> ExercizeCalory;  
  
 public String BT;  
  
 public String TM;  
  
 public String CBV;  
  
 public int FTimer;  
  
 public int FCAl;  
  
 public int ALLCAL;  
  
  
  
 ObservableList<Journal> data = FXCollections.*observableArrayList*();  
 public ObservableList<Journal> listfromfile;  
  
 @Override  
 public void initialize(URL url, ResourceBundle resources) {  
  
  
 TaskDeserialized();  
 CaloryMeter.setText(String.*valueOf*(ALLCAL));  
  
  
 ExercizeName.setCellValueFactory(  
 new PropertyValueFactory<Journal, String>("nameTask")  
  
 );  
  
 ExercizeTime.setCellValueFactory(  
 new PropertyValueFactory<Journal, Integer>("timeTask")  
 );  
  
 ExercizeCalory.setCellValueFactory(  
 new PropertyValueFactory<Journal, Integer>("kkalTask")  
 );  
  
 Choise.getItems().setAll  
  
  
 ("Отжимания", "Скакалка", "Приседания");  
  
  
 Choise.setValue("Отжимания");  
  
  
 ObservableList<Journal> listfromfile = *read*();  
 System.*out*.println("Lists equal? "+ listfromfile.equals(data));  
  
  
  
  
 //ReadData();  
 tableTask.setItems(listfromfile);  
 }  
  
 public class TASK implements java.io.Serializable  
  
 {  
  
  
  
 public int CALORY;  
 private static final long *serialVersionUID* = 1L;  
  
  
  
 public TASK(int Cal)  
 {  
  
 CALORY=Cal;  
 }  
 }  
  
 @XmlRootElement  
 public class JM {  
  
 Journal JData;  
  
 @XmlElement(name="journal")  
 public Journal getJData() { return JData; }  
  
 public void setJData(Journal JData) { this.JData = JData; }  
  
 }  
  
  
  
 public void buttonClicked(javafx.event.ActionEvent actionEvent) throws IOException {  
 BT = StartButton.getText();  
  
 if (BT == "Конец Упражнений") { StartButton.setText("Начать");  
  
  
 stop();  
  
 TM = TIMER.getText();  
  
 FTimer = Integer.*parseInt*(TM);  
  
 reset();  
  
  
 FCAl = FTimer \* getCbTextKkal();  
  
 ALLCAL =+ FCAl;  
  
 CaloryMeter.setText(  
 String.*valueOf*(ALLCAL)  
 );  
 TaskSerialized();  
 data.add(new Journal(  
  
 Choise.getValue(), FTimer, FCAl)  
 );  
  
 tableTask.setItems(data);  
 *write*(data);  
  
 } else { StartButton.setText("Конец Упражнений");  
 start();  
 }  
 }  
 public static void main(String[] args){}  
  
 Runnable run = new Runnable() {  
 @Override  
 public synchronized void run() {  
 while (true) {  
 try {  
 Thread.*sleep*(1000);  
 if (STOP == false) {  
 sec++;TIMER.setText(""+sec);  
 }  
 } catch (InterruptedException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
 };  
  
 public void start() {  
  
 Thread thread = new Thread(run);  
 thread.start();  
  
 STOP = false;  
 }  
 public void reset() {  
  
 sec = 0;  
  
  
 TIMER.setText(""+sec);  
 }  
 public void stop() {  
  
 STOP = true;  
 }  
 int sec = 0;  
 boolean STOP = false;  
  
  
 public String NameOfFile = "src/sample/file.dat";  
 public void TaskSerialized()  
 {  
 TASK object = new TASK(ALLCAL);  
  
 try  
 {  
  
  
  
 FileOutputStream file = new FileOutputStream(NameOfFile);  
  
 ObjectOutputStream Output = new ObjectOutputStream(file);  
  
  
 Output.writeObject(object);  
 Output.close();  
 file.close();  
 }  
 catch(IOException ex)  
 {  
 System.*out*.println("IOException is caught");  
 }  
 }  
  
  
 public void TaskDeserialized()  
 {  
 TASK object1 = null;  
 try  
 {  
  
  
  
 FileInputStream file;  
  
 file = new FileInputStream(NameOfFile);  
  
 ObjectInputStream input;  
  
 input = new ObjectInputStream(file);  
  
  
 object1 = (TASK)input.readObject();  
  
 input.close();  
 file.close();  
 ALLCAL = object1.CALORY;  
 }  
 catch(IOException ex)  
  
 { System.*out*.println("IOException is caught"); } catch(ClassNotFoundException ex)  
 { System.*out*.println("ClassNotFoundException is caught"); }  
 }  
  
 public int getCbTextKkal() {  
 int CalVal = 0;  
 CBV = Choise.getValue();  
 if (CBV=="Отжимания") { CalVal = 100; } else if(CBV=="Приседания") { CalVal = 150; }  
 else CalVal = 50;  
 return CalVal;  
 }  
  
 private static void write(List<Journal> Member ) {  
 try {  
 Path file = Paths.*get*("src/sample/journal.ser");  
  
 OutputStream FS;  
  
 FS = Files.*newOutputStream*(file);  
  
 ObjectOutputStream Output;  
  
 Output = new ObjectOutputStream(FS);  
  
 List<JournalDto> dtos = Member.stream().map(d -> new JournalDto(  
  
 d.getNameTask(),  
  
 d.getTimeTask(),  
  
 d.getKkalTask()))  
 .collect(Collectors.*toList*());  
  
 Output.writeObject(dtos);  
  
 Output.close();  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 private static ObservableList<Journal> read() {  
 try {  
  
  
 Path file = Paths.*get*("src/sample/journal.ser");  
  
 InputStream input;  
  
 input = Files.*newInputStream*(file);  
  
 ObjectInputStream ObjectInput;  
  
 ObjectInput = new ObjectInputStream(input);  
  
  
 List<Journal> data = ((List<JournalDto>) ObjectInput.readObject()).stream()  
 .map(o -> new Journal(  
 o.getNameTask(),  
 o.getTimeTask(),  
 o.getKkalTask()))  
 .collect(Collectors.*toList*());  
  
 return FXCollections.*observableList*(data);  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return FXCollections.*emptyObservableList*();  
 }  
}

package sample.Login;  
  
import javafx.application.Application;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Login extends Application {  
  
 @Override  
 public void start(Stage primaryStage) throws Exception{  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("Login.fxml"));  
 Parent root = loader.load();  
 Scene scene = new Scene(root);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
  
}

. package sample.Login;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.fxml.Initializable;  
import javafx.scene.Node;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.scene.control.Label;  
import javafx.stage.Stage;  
import java.awt.\*;  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
  
  
import java.awt.\*;  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
  
public class LoginController implements Initializable {  
 @FXML  
 private TextField LoginWrite;  
 @FXML  
 private PasswordField PasswordWrite;  
 @FXML  
 private Button SignUp;  
 @FXML  
 private Label ErrorWarning;  
  
 @Override  
 public void initialize(URL url, ResourceBundle resources) {  
  
 }  
  
 public void ButtonClick(ActionEvent actionEvent) throws IOException {  
  
  
  
 if(LoginWrite.getText().equals("admin") && PasswordWrite.getText().equals("admin"))  
 {  
 Parent parent = FXMLLoader.*load*(getClass().getResource("sample.fxml"));  
 Stage stage = new Stage();  
 Scene scene = new Scene(parent);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
 else{ ErrorWarning.setText("Неверный логин/пароль"); }  
 }  
}