|  |  |
| --- | --- |
| Федеральное агентство железнодорожного транспорта  Омский государственный университет путей сообщения  Кафедра «Автоматика и системы управления»  Разработка приложений на платформе Java FX  Курсовая работа  по теме «Программирование» | |
|  | Студент гр. 78-ИС                           Швед Т.Ю.  «    »                  2021 г.  Руководитель –                           Альтман Е. А.  «    »                  2021 г. |
| Омск 2020 | |

**Задание**

В курсовой работе разрабатывается приложение на платформе JavaFX. Приложение разрабатывается по индивидуальному заданию.

Основные этапы выполнения курсовой работы совпадают с изученными в лабораторных работах элементами платформы JavaFX. Необходимо разработать интерфейс приложения и создать «fxml» файлы для всех окон. Далее разрабатываются нестандартные элементы графического интерфейса. Следующим шагом разрабатываются контроллеры окон.

В дополнении к пройденному на лабораторных работах материалу самостоятельно изучается: технология «JAXB» для сохранения данных введенных пользователем. Эти библиотека и технология необходима для использования в курсовой работе.

Реферат

УДК 000.713

Курсовой проект содержит 52 страниц, 37 рисунков, 8 источника, 4 приложения.

Объектом исследования являются приложения на платформе JavaFX.

Цель работы – закрепить навыки разработки приложений на платформе JavaFX, на изучаемом языке программирования Java и изучение применения их для типовых задач.

В процессе работы была написана программа которая реализует нашу цель.

Курсовая работа оформлена в текстовом редакторе Microsoft Word.

**Содержание**

[Введение 5](#_Toc69596918)

[Основная часть 6](#_Toc69596919)

[1. Техническое задание 6](#_Toc69596920)

[2. Средства разработки 7](#_Toc69596921)

[3. Структура программы 8](#_Toc69596922)

[4. Реализация программы 9](#_Toc69596923)

[4.1. Создание интерфейса 9](#_Toc69596924)

[4.2. Отображение интерфейса 12](#_Toc69596925)

[4.3. Обработка данных 17](#_Toc69596926)

[4.4. Сохранение и загрузка данных 22](#_Toc69596927)

[4.5. Удаление и добовление данных 25](#_Toc69596928)

[5. Инструкция пользователя 28](#_Toc69596929)

[Заключение 31](#_Toc69596930)

[библиографический список 32](#_Toc69596931)

[Приложение А](#_Toc69596932) - [Структура программы 33](#_Toc69596934)

[Приложение Б](#_Toc69596935) - [Интерфейс программы 34](#_Toc69596937)

[Приложение В](#_Toc69596938) - [Диаграммы последовательностей реализующие основные функции программы 35](#_Toc69596940)

[Приложение Г](#_Toc69596941) - [Код программы, включая файлы на «fxml» 36](#_Toc69596943)

# **Введение**

Данный документ преследует цель показать процесс реализации навыков разработки приложений на платформе JavaFX, на изучаемом языке программирования Java и изучение применения для типовых задач. Научиться этому является основной целью курсовой работы и необходимо для выполнения всех последующих лабораторных работ.

**Основная часть**

1. **Техническое задание**

«Журнал преподавателя». Пользователь должен иметь возможность ввести группы студентов, в группы — студентов (фамилии, имя). Для групп можно назначать занятия (задания) с весовым коэффициентом. Для студентов можно задавать процент выполнения каждого задания. Программа должна рассчитывать рейтинг студентов используя весовые коэффициенты.

**2. Средства разработки**

IntelliJ IDEA - интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ.

Scene Builder — это инструмент, с помощью которого мы можем конструировать наши окна в виде графического интерфейса и после их сохранять, и эта программа на основании результата будет конструировать xml файлы, которые мы будем подтягивать в нашем приложении.

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.

Java Architecture for XML Binding (JAXB) — это программная структура, которая позволяет разработчикам отображать классы Java в представления XML. JAXB позволяет делать сохранение объекта в файл и обратно в объекты.

**3. Структура программы**

Для понимания, того как работает программа, необходимо найти все связи и понять их принцип работы. Ниже представлена структура данной программы, рисунок 1. Так же её можно увидеть, в приложении А.

|  |
| --- |
|  |
| Рисунок 1 – Визуальное отображение структуры данной программы |

**4. Реализация программы**

**4.1. Создание интерфейса**

Для взаимодействия с данными, нам нужен интерфейс программы. Для кода, относящегося к графическому интерфейсу, создадим файл «sample.fxml». Этот файл можно редактировать как обычный код или (что более предпочтительно) открыть его в визуальном редакторе «Scene Builder». Что и было сделано, как показано на рисунке 2.

|  |
| --- |
|  |
| Рисунок 2 – Визуальный редактор «Scene Builder» |

Исходя из нашего задания, нам нужен «TableView», а так же выпадающий список и несколько кнопок, для управления. Основная часть нашей панели будет служить, для отображения данных, поэтому в панели «BorderPane», мы расположим «TableView», а сверху кнопки управления. А так же нам нужно отдельное окно для добавления различных данных. Что и было сделано, как показано на рисунке 3 и 4.

|  |
| --- |
|  |
| Рисунок 3 – Создание основного интерфейса в «Scene Builder» |
|  |
| Рисунок 4 – Создание интерфейса в «Scene Builder» для добавления данных |
|  |

После сохранения, можно увидеть изменения в самом файле, каждый элемент будет соответствовать своему расположению на форме. А так же необходимо добавить атрибуты управления такие как «fx:id="nameColumn"» для того что бы его можно было найти в нашем контроллере, «onMouseClicked="#addColumn"» для создания методов обработки нажатия мышки, как показано на рисунке 5.

|  |
| --- |
|  |
| Рисунок 5 – Создание интерфейса в «Scene Builder» |

Более подробно можно посмотреть код, который приводится в виде листинга, в приложения Б.

**4.2. Отображение интерфейса**

Для отображения интерфейса, нам необходимо изначально создать и загрузить файл с расширением «FXML», как показано на рисунке 6.

|  |
| --- |
|  |
| Рисунок 6 – Загрузка файла с расширением «FXML», для визуализации нашего будущего окна |

Далее необходимо подгрузить класс, отвечающий за контроль всех наших действий, а так же отобразить наше окно. Параллельно создаем обработку закрытия окна, она нам пригодится в будущем при сохранении наших данных. Всё это представлено на картинке 7.

|  |
| --- |
|  |
| Рисунок 7 – Загрузка класса с контролем и отображение окна |

Так же мы видим, что у контроллера сразу вызывается функция «init()». Она необходима, для начала инициализации всех наших данных. Так как мы создавали основное окно программы, а так же подключили его к окну, то теперь необходимо, отобразить наши элементы, вот их и будем создавать. Как мы создаём контроллер, представлено на рисунке 8.

|  |
| --- |
|  |
| Рисунок 8 – Создаём класс контроллера |

Так как данных мы вводим много, поэтому изначально и создаём массив. Но не простой, а массив выбранного класса данных, так как наши переменные в таблице будут разного типа. То и создаём класс с этими типами, представлено на рисунке 9.

|  |
| --- |
|  |
| Рисунок 9 – Создаём класс данных о ученике |

А так же добавляем ему конструктор, для того что бы можно было легко его инициализировать, как представлено на рисунке 10.

|  |
| --- |
|  |
| Рисунок 10 – Создаём конструктор класса данных о ученике |

При инициализации основного окна в контроллере, вводим начальную группу, на данный момент, что бы она не была пустой, запишем туда значение «101», после загружаем данные, но это мы реализуем позже. И запускаем инициализацию, основных полей данных на панели. Всё это представлено на рисунке 11.

|  |
| --- |
|  |
| Рисунок 11 – Инициализация контроллера, начало |

Для инициализации основных данных на панели, мы получаем данные из загрузки, после выбранной группы. Для этого мы создаем обработку события, при открытии «ComboBox», представлено на рисунке 12. Но это сохранение и загрузку, мы реализуем позже. А так же загрузку наших данных, об учениках и разрешение редактирование полей.

|  |
| --- |
|  |
| Рисунок 12 – Инициализация контроллера, выбор группы и загрузка данных об учениках |

Так же нам необходимо загрузить самих учеников и их оценки, и разрешить их редактирование, как представлено на рисунке 13.

|  |
| --- |
|  |
| Рисунок 13 – Загрузка данных самих учеников и их оценки, а так же действия при попытки редактировать |

Но при изменении нам необходимо, сохранить наши изменения, и для этого мы создаём действие вызвывающие сохранение данных, если мы нажали на таблицу, как представлено на рисунке 14. Но это мы реализуем чуть позже.

|  |
| --- |
|  |
| Рисунок 14 – Нажатие мышки на таблицу и сохранение данных |

Более подробно можно посмотреть код, который приводится в виде листинга, в приложения Б.

**4.3. Обработка данных**

После прописанных запросов, на изменение, сохранение и загрузку данных, их необходимо обработать. Другими словами создать, и начнём выгрузки данных из основного класса «bill». Для этого создадим к каждому полю загрузку и выгрузку данных, пример для некоторых из них представлен на рисунке 15. Остальные делаются по аналогии.

|  |
| --- |
|  |
| Рисунок 15 – Загрузка и выгрузка данных, для полей об учеников |

Теперь необходимо сделать так, что бы при вводе данных они были в виде числа. Для этого создаём класс «FloatCell», который и наследует возможности таблицы. А так же передадим ему данные, с которыми будем работать. Всё это представлено на рисунке 16.

|  |
| --- |
|  |
| Рисунок 16 – Создание FloatCell |

Далее запускается запуск, на изменение данных. Здесь важно передать данные, которые у нас были введены до этого. Что бы в случаи малых исправлений или случайному открытию редактора, мы их не удалили. Так как здесь мы редактируем только оценки, то нам нужно понять, какое полевыбрал пользователь, для этого мы и устанавливаем «Switch». Всё это представлено на рисунке 17.

|  |
| --- |
|  |
| Рисунок 17 – Переопределение функции, по началу редактирования |

Кстати, здесь же мы используем класс «FloatField», для ограничения ввода наших значений, но это мы реализуем позже.

Когда мы закончили редактирование, необходимо вставить весь наш ведённый текст в выбранное поле. Для этого мы переопределяем функцию, как представлено на рисунке 18.

При этом для сохранения, нам необходимо загрузить полное поле данных. Согласен что это было излишне и можно было сделать более оптимизировано, но было решено взять за основу готовый конструктор и заменять строку данных целиком.

|  |
| --- |
|  |
| Рисунок 18 – Переопределение функции, в конце редактирования, сохранение наших данных |

Но так как наше поле изменилось, то нам теперь необходимо, обновить средний балл и рейтинг ученика. Для этого мы запускаем сортировку, и в зависимости от средних баллов, делать перерасчёт для всех значений. Всё это представлено на рисунке 19.

|  |
| --- |
|  |
| Рисунок 19 – Вычисление рейтинга |

Но так как надо показать свою возможности, то сортировку данных надо сделать самому, а не через стандартную функцию сортировки коллекций. Поэтому мы создаём её, как представлено на рисунке 20.

|  |
| --- |
|  |
| Рисунок 20 – Сортировка коллекции |

А так же нам для полного счастья, необходимо обработка обновления поля, её мы тоже переопределяем и проверяем на пустые значения, как представлено на рисунке 21.

|  |
| --- |
|  |
| Рисунок 21 – Обновление поля |

Как уже говорилось ранее, мы используем наше поле для ввода данных, то его необходимо создать.

Так как вместо поля «TextField» мы берем поле «FloatField», то его необходимо создать, наследую я от «TextField». Полная его реализация представлено на рисунке 22.

|  |
| --- |
|  |
| Рисунок 22 – Создание FloatField |

Более подробно можно посмотреть код, который приводится в виде листинга, в приложения Б.

**4.4. Сохранение и загрузка данных**

Далее для того что-бы наши данные не исчезали при выключении программы или переключении между группами. Нам необходимо сохранять любые изменения. Для этого мы пишем следующий код, как представлено на рисунке 23.

|  |
| --- |
|  |
| Рисунок 23 – Создание метода сохранения данных |

При необходимости мы создаём папку, далее мы сохраняем значения для полей студентов. Через цикл перебираем каждую строчку. И в конце, по такому же принципу сохраняем и группы, в виде «XML» файла.

Далее по такому же принципу, загружаем поля обратно при открытии окна или выборе другой группы. Для этого мы пишем следующий код, как представлено на рисунке 24.

|  |
| --- |
|  |
| Рисунок 24 – Загрузка данных |

Так как мы уже ранее поняли, что групп у нас может быть несколько, поэтому мы их вводим в виде отдельного класса, что бы они были сохранены, а так же в будущем и выгружены. Так как нам нужны лишь названия, то нету смысла их делать разными файлами, а нужно сделать в виде массива строчек. Для этого мы пишем следующий код, как представлено на рисунке 25.

|  |
| --- |
|  |
| Рисунок 25 – Класс Group |

Сама функция сохранения преобразует наши данные в следующий вид «XML» файла, как представлено на рисунке 26 и 27.

|  |
| --- |
|  |
| Рисунок 26 – Файл «XML», для данных ученика |

|  |
| --- |
|  |
| Рисунок 27 – Файл «XML», для данных групп |

Более подробно можно посмотреть код, который приводится в виде листинга, в приложения Б.

**4.5. Удаление и добовление данных**

После сохранения и загрузки данных, встаёт вопрос по поводу удаления и добавления новых полей, для учеников и групп. Для этого мы и создавали кнопки на панели. Теперь же их необходимо инициализировать. Начнём с добавления новых полей учеников, для этого мы уже создали файл «FXML». Его вид приводился выше, на рисунке 4, а мы теперь подгружаем его, контроллер для него. Для этого мы пишем следующий код, как представлено на рисунке 28.

|  |
| --- |
|  |
| Рисунок 28 – Открытие окна, для добавления данных нового ученика |

Как вы уже поняли выше, для нового окна нам нужен контроллер. Для этого мы пишем следующий код, как представлено на рисунке 29.

|  |
| --- |
|  |
| Рисунок 29 – Контроллер окна, для добавления данных нового ученика |

И инициализируем кнопку для удаления поля. Для этого мы пишем следующий код, как представлено на рисунке 30.

|  |
| --- |
|  |
| Рисунок 30 – Инициализация кнопки удаления |

Теперь так же инициализируем кнопку с добавлением групп. Здесь тоже понадобиться файл «FXML». Его вид приводился выше, на рисунке 4, а мы теперь подгружаем его, контроллер для него. Для этого мы пишем следующий код, как представлено на рисунке 31.

|  |
| --- |
|  |
| Рисунок 31 – Инициализация кнопки добавления группы |

Как вы уже поняли выше, для нового окна нам нужен контроллер. Для этого мы пишем следующий код, как представлено на рисунке 32.

|  |
| --- |
|  |
| Рисунок 32 – Контроллер окна, для добавления данных новой группы |

И инициализируем кнопку для удаления поля. Для этого мы пишем следующий код, как представлено на рисунке 33.

|  |
| --- |
|  |
| Рисунок 33 – Инициализация кнопки удаления |

Более подробно можно посмотреть код, который приводится в виде листинга, в приложения Б.

**5. Инструкция пользователя**

При включении программы «OMGUPS\_Sem\_6\_KR\_1.jar», у нас откроется следующее окно, как представлено на рисунке 34.

|  |
| --- |
|  |
| Рисунок 34 – Открытие программы |

Далее, можно редактировать текст, нажав два раза на выделенную область, как это показано на рисунке 35.

|  |
| --- |
|  |
| Рисунок 35 – Редактирование полей |

А для того что бы сохранить свои действия, необходимо нажать в любое другое место или на клавишу «Enter», как это показано на рисунке 36.

|  |
| --- |
|  |
| Рисунок 36 – Редактирование имени |

Так же можно выбрать другую группу, для запыления и изменения данных», как это показано на рисунке 37.

|  |
| --- |
|  |
| Рисунок 37 – Выбор группы |

На кнопке добавить вылезет дополнительное окно, и при нажатии на добавление данных, значение исчезнет, как представлено на рисунке 38 и 39.

|  |
| --- |
|  |
| Рисунок 37 – Добавление группы |
|  |
| Рисунок 37 – Добавление ученика |

**Заключение**

При выполнении данной работы, был показан процесс реализации навыков разработки приложений на платформе JavaFX, на изучаемом языке программирования Java. Научиться этому являлось основной целью курсовой работы и необходимо для выполнения всех последующих лабораторных работ. Так же было изучена технология «JAXB».

**библиографический список**

* 1. IntelliJ Idea the Java IDE [/https://www.jetbrains.com/idea/](https://www.jetbrains.com/idea/)
  2. Scene Builder — Gluon [/http://gluonhq.com/open-source/scene-](http://gluonhq.com/open-source/scene-builder/) [builder/](http://gluonhq.com/open-source/scene-builder/)
  3. Using JavaFX UI Controls [/https://docs.oracle.com/javafx /2/ui\_controls/jfxpub-ui\_controls.htm](https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm)
  4. Документация по платформе JavaFX [/https://docs.oracle .com/javase/8/javafx/api/overview-summary.html](https://docs.oracle.com/javase/8/javafx/api/overview-summary.html)
  5. Учебник по JavaFX 8 — Часть 5: Хранение данных в XML[/http://code.makery.ch/library/javafx-8-tutorial/ru/part5/](http://code.makery.ch/library/javafx-8-tutorial/ru/part5/)
  6. Package javafx.print [/https://docs.oracle.com/javase/8/javafx/api](https://docs.oracle.com/javase/8/javafx/api/javafx/print/package-summary.html)

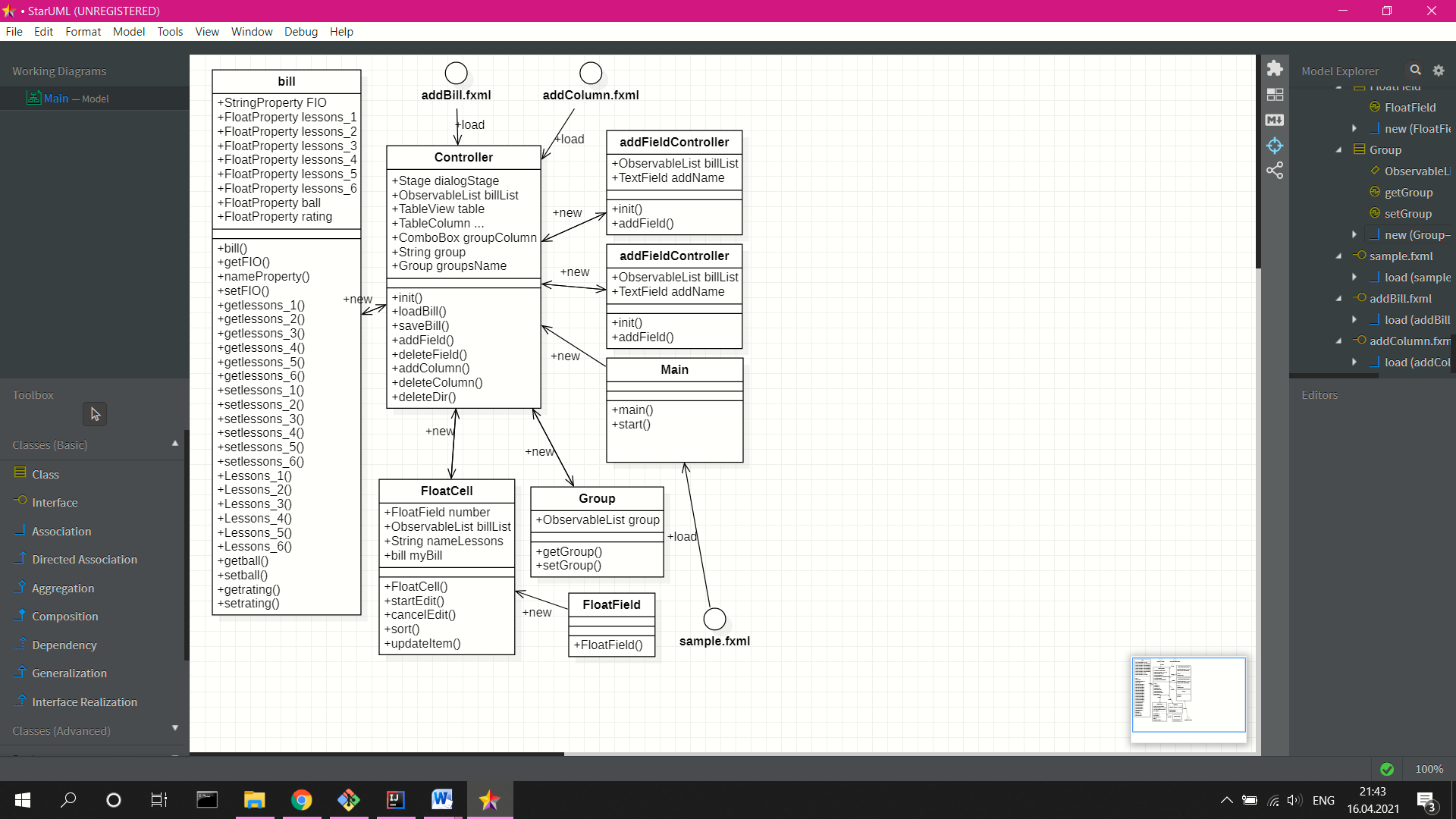
[/javafx/print/package-summary.html](https://docs.oracle.com/javase/8/javafx/api/javafx/print/package-summary.html)

* 1. Учебник по JavaFX 8 — Часть 7: Развертывание [/http://code.makery.ch/library/javafx-8-tutorial/ru/part7/](http://code.makery.ch/library/javafx-8-tutorial/ru/part7/)
  2. СТП ОмГУПС–1.2–2005 // Омск, ОмГУПС 2005

Приложение А

(справочное)

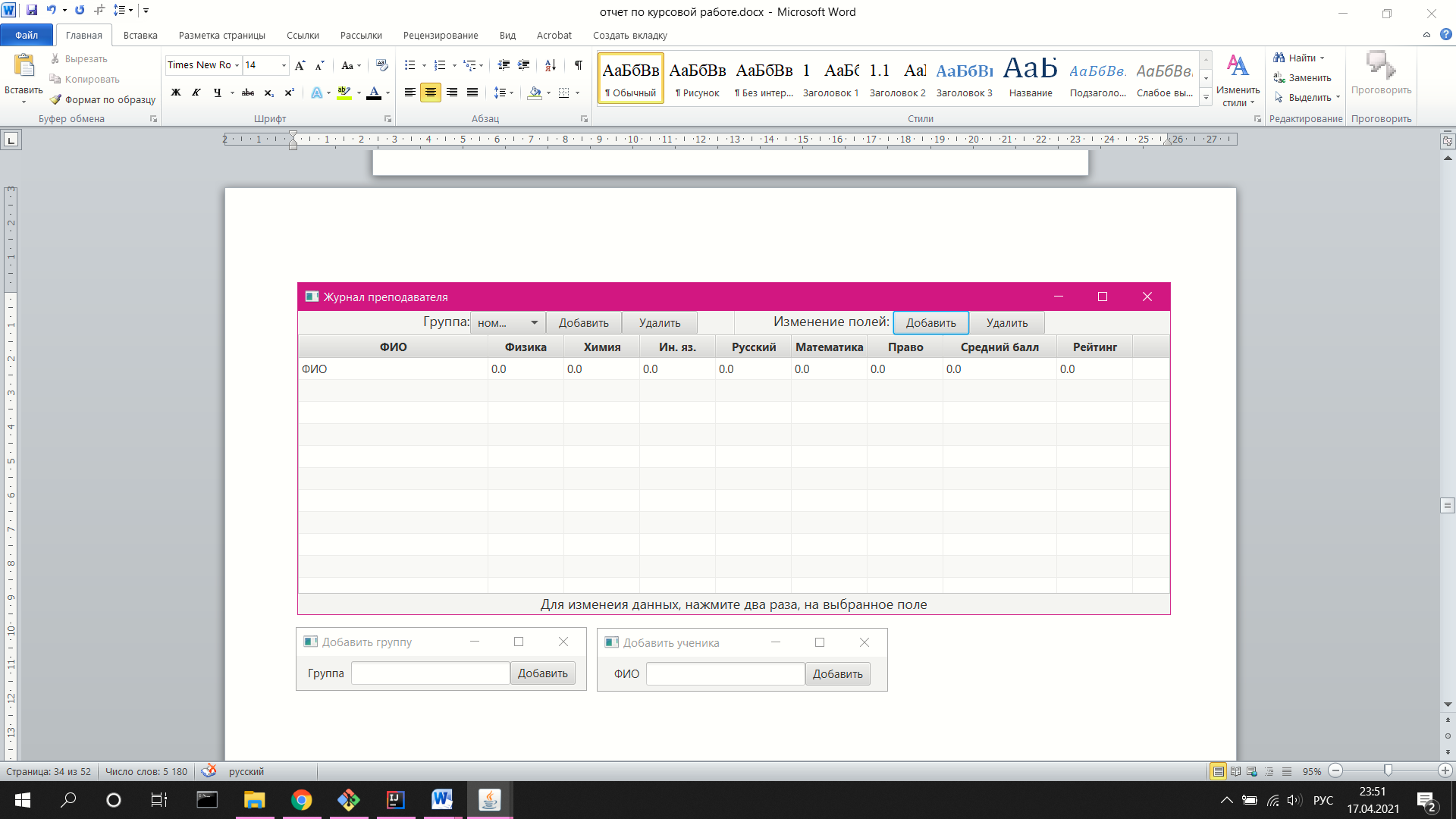
Структура программы



Приложение Б

(справочное)

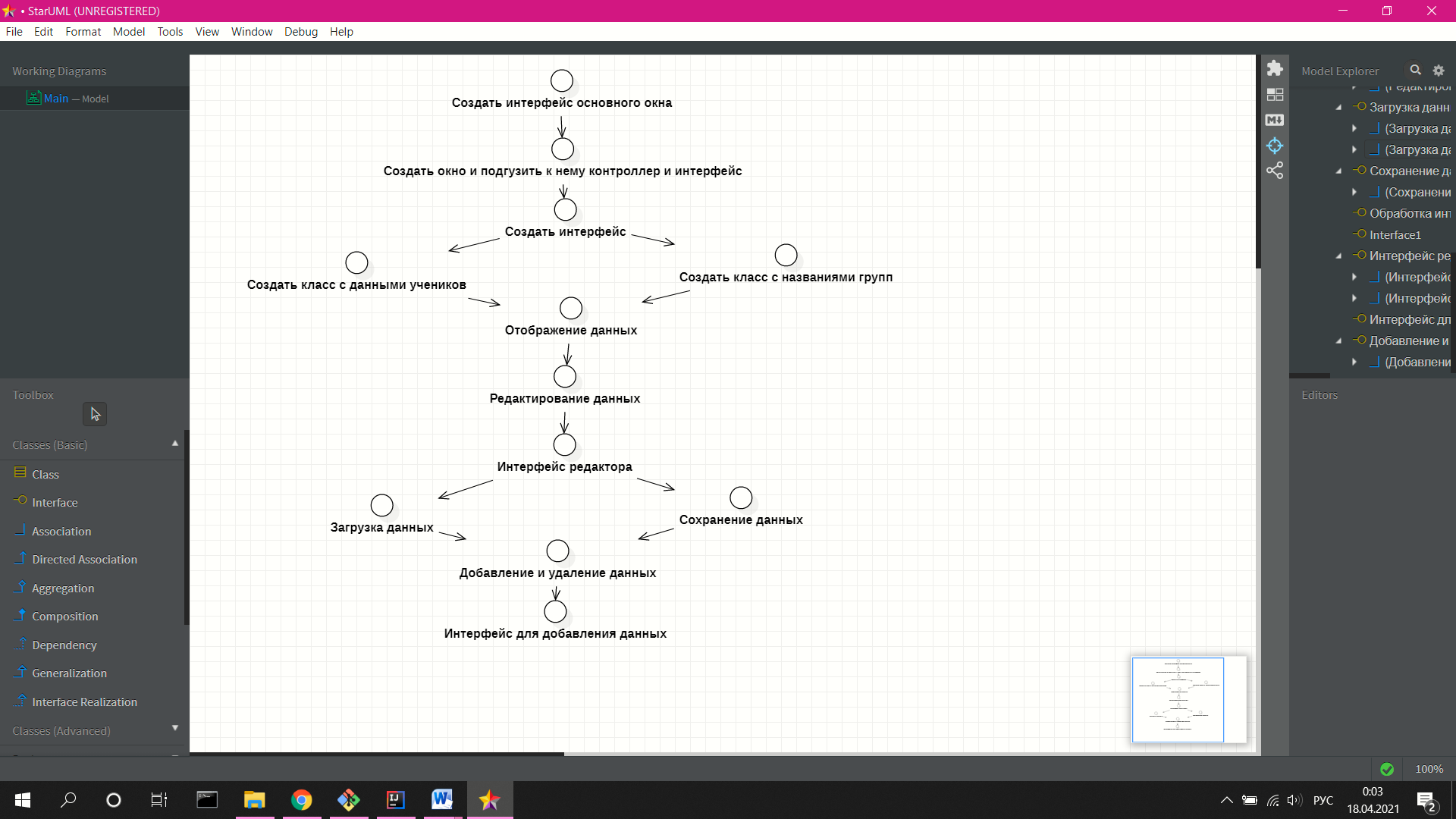
Интерфейс программы



Приложение В

(справочное)

Диаграммы последовательностей реализующие основные функции программы



Приложение Г

(справочное)

Код программы, включая файлы на «fxml»

**sample.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.Separator***?>  
<?***import javafx.scene.control.TableColumn***?>  
<?***import javafx.scene.control.TableView***?>  
<?***import javafx.scene.layout.BorderPane***?>  
<?***import javafx.scene.layout.HBox***?>  
<?***import javafx.scene.text.Font***?>*<**BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="1050.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**center**>  
 <**TableView fx:id="table" prefHeight="373.0" prefWidth="786.0" BorderPane.alignment="CENTER"**>  
 <**columns**>  
 <**TableColumn fx:id="nameColumn" prefWidth="250" text="ФИО"** />  
 <**TableColumn fx:id="lessons\_1" prefWidth="100" text="Физика"** />  
 <**TableColumn fx:id="lessons\_2" prefWidth="100" text="Химия"** />  
 <**TableColumn fx:id="lessons\_3" prefWidth="100" text="Ин. яз."** />  
 <**TableColumn fx:id="lessons\_4" prefWidth="100" text="Русский"** />  
 <**TableColumn fx:id="lessons\_5" prefWidth="100" text="Математика"** />  
 <**TableColumn fx:id="lessons\_6" prefWidth="100" text="Право"** />  
 <**TableColumn fx:id="ball" prefWidth="150" text="Средний балл"** />  
 <**TableColumn fx:id="rating" prefWidth="100" text="Рейтинг"** />  
 </**columns**>  
 </**TableView**>  
 </**center**>  
 <**top**>  
 <**HBox alignment="TOP\_CENTER" prefHeight="26.0" prefWidth="600.0" BorderPane.alignment="CENTER"**>  
 <**children**>  
 <**Label text="Группа:"**>  
 <**font**>  
 <**Font size="18.0"** />  
 </**font**>  
 </**Label**>  
 <**ComboBox fx:id="groupColumn" prefHeight="33.0" prefWidth="100.0" promptText="101"** />  
 <**Button onMouseClicked="#addColumn" mnemonicParsing="false" prefHeight="32.0" prefWidth="100.0" text="Добавить"** />  
 <**Button onMouseClicked="#deleteColumn" mnemonicParsing="false" prefHeight="31.0" prefWidth="100.0" text="Удалить"** />  
 <**Separator orientation="VERTICAL" prefHeight="27.0" prefWidth="100.0"** />  
 <**Label text="Изменение полей: "**>  
 <**font**>  
 <**Font size="18.0"** />  
 </**font**>  
 </**Label**>  
 <**Button onMouseClicked="#addField" mnemonicParsing="false" prefHeight="34.0" prefWidth="100.0" text="Добавить"** />  
 <**Button onMouseClicked="#deleteField" mnemonicParsing="false" prefHeight="30.0" prefWidth="100.0" text="Удалить"** />  
 </**children**>  
 </**HBox**>  
 </**top**>  
 <**bottom**>  
 <**Label text="Для изменеия данных, нажмите два раза, на выбранное поле" BorderPane.alignment="CENTER"**>  
 <**font**>  
 <**Font size="18.0"** />  
 </**font**>  
 </**Label**>  
 </**bottom**>  
</**BorderPane**>

**Main**

**package** sample;  
  
**import** javafx.application.Application;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.stage.Stage;  
  
**public class** Main **extends** Application {  
  
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception{  
 FXMLLoader loader = **new** FXMLLoader(); *// создаем загрузку FXML* loader.setLocation(getClass().getResource(**"sample.fxml"**)); *// добавляем нашу FXML* Parent root = loader.load(); *// создаем панель и загружаем наши данные* Controller controller = loader.getController(); *// создаем контроллер* controller.init(); *// запускаем в контроллере метод init* primaryStage.setTitle(**"Журнал преподавателя"**); *// заголовок формы* primaryStage.setScene(**new** Scene(root, 1150, 400)); *// размеры формы и наша панель, в форму(окно)* primaryStage.show(); *// показать форму* primaryStage.setOnCloseRequest(event -> { *// сохранение, при закрытии формы* **try** {  
 controller.saveBill();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

**Controller**

**package** sample;  
  
**import** javafx.beans.property.FloatProperty;  
**import** javafx.beans.property.SimpleFloatProperty;  
**import** javafx.beans.property.SimpleObjectProperty;  
**import** javafx.beans.value.ObservableValue;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.EventHandler;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.ComboBoxTableCell;  
**import** javafx.scene.control.cell.TextFieldTableCell;  
**import** javafx.scene.input.MouseEvent;  
**import** javafx.scene.layout.Pane;  
**import** javafx.stage.Modality;  
**import** javafx.stage.Stage;  
**import** javafx.stage.WindowEvent;  
**import** javafx.util.Callback;  
  
**import** javax.xml.bind.JAXBContext;  
**import** javax.xml.bind.Marshaller;  
**import** javax.xml.bind.JAXBException;  
**import** javax.xml.bind.Unmarshaller;  
**import** java.awt.\*;  
**import** java.io.File;  
**import** java.io.FileInputStream;  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.nio.file.Files;  
**import** java.nio.file.Paths;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** Controller {  
  
 *// -------------------------------------------------------------------- переменные ---------------------------------------------* ObservableList<bill> **billList** = FXCollections.*observableArrayList*();  
  
 @FXML TableView<bill> **table**;  
 @FXML TableColumn<bill, String> **nameColumn**;  
 @FXML TableColumn<bill, Number> **lessons\_1**;  
 @FXML TableColumn<bill, Number> **lessons\_2**;  
 @FXML TableColumn<bill, Number> **lessons\_3**;  
 @FXML TableColumn<bill, Number> **lessons\_4**;  
 @FXML TableColumn<bill, Number> **lessons\_5**;  
 @FXML TableColumn<bill, Number> **lessons\_6**;  
 @FXML TableColumn<bill, Number> **ball**;  
 @FXML TableColumn<bill, Number> **rating**;  
 @FXML ComboBox **groupColumn**;  
  
 String **group**;  
 Group **groupsName** = **new** Group();  
  
 *// ----------------------------------------- инициализация ----------------------* **public void** init() {  
 File file = **new** File(**"group"**); *// файл с таким именем* **if** (!file.exists()) {file.mkdir();} *// если нету папки, то создать!* **groupColumn**.setValue(**"101"**); *// начальное значение поля (по умолчанию)* **group** = **"group/"**+**groupColumn**.getValue(); *// путь для группы* **this**.loadBill(); *//загрузка данных* **this**.init(**billList**); *// инициализация данных (отображение)* }  
  
  
 *// ------------------------------------------------------- загрузка данных ---------------------------------------------------------------* **public void** loadBill() {  
 **try** {  
 File file = **new** File(**"nameGroup.xml"**); *// наш файл* JAXBContext context = JAXBContext.*newInstance*(Group.**class**); *// переводим наши дынные, в вид JAXBContext* Unmarshaller um = context.createUnmarshaller(); *// переводим из xml в наш вид* **groupsName** = (Group) um.unmarshal(file); *// выбранный файл* **groupColumn**.setItems(**groupsName**.getGroup());  
  
 **group** = **"group/"**+**groupColumn**.getValue(); *// путь для группы* file = **new** File(**group**);  
 **if** (!file.exists()) {file.mkdir();}  
  
 context = JAXBContext.*newInstance*(bill.**class**); *// переводим наши дынные, в вид JAXBContext* um = context.createUnmarshaller();  
  
 **billList**.clear();  
 *// Обёртываем наши данные об адресатах.* **int** i=1;  
 **while** (file.exists()){  
 file = **new** File(**group**+**"/"**+i+**".xml"**);  
 **if**(file.exists()){  
 *//InputStream inStream = new FileInputStream( file );* bill wrapper = (bill) um.unmarshal(file);  
 **billList**.addAll(wrapper);*// Чтение XML из файла и демаршализация.* }  
 i++;  
 }  
 } **catch** (Exception e) { *// окно с ошибкой* Alert alert = **new** Alert(Alert.AlertType.***ERROR***);  
 alert.setTitle(**"Error"**);  
 alert.setHeaderText(**"Could not load data"**);  
 alert.showAndWait();  
 }  
 }  
  
 *// ----------------------------------------------------------------- инициализация данных (отображение)---------------------------------------* **public void** init (ObservableList<bill> billList){  
 **groupColumn**.setValue(**"101"**); *// устанавливаем выбранный элемент по умолчанию  
  
 // получаем выбранный элемент* **groupColumn**.setOnAction(event -> {  
 **try** {  
 **this**.saveBill(); *// сохранение* **this**.loadBill(); *// загрузка* } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 });  
  
 **table**.setItems(billList) ;  
 **table**.setEditable(**true**);  
  
 *// -------------------------------------------------------------------- имя  
 // добавление строки в столбик (имя)* **nameColumn**.setCellValueFactory(cellData ->  
 cellData.getValue().nameProperty());  
  
 *// запрос на изменение данных* **nameColumn**.setCellFactory(TextFieldTableCell.<bill> *forTableColumn*());  
  
 *// -------------------------------------------------------------------- занятия  
 // добавление строк* **lessons\_1**.setCellValueFactory(cellData -> cellData.getValue().Lessons\_1());*// добавление строки в столбик (предмет 1)* **lessons\_2**.setCellValueFactory(cellData -> cellData.getValue().Lessons\_2());*// добавление строки в столбик (предмет 2)* **lessons\_3**.setCellValueFactory(cellData -> cellData.getValue().Lessons\_3());*// добавление строки в столбик (предмет 3)* **lessons\_4**.setCellValueFactory(cellData -> cellData.getValue().Lessons\_4());*// добавление строки в столбик (предмет 4)* **lessons\_5**.setCellValueFactory(cellData -> cellData.getValue().Lessons\_5());*// добавление строки в столбик (предмет 5)* **lessons\_6**.setCellValueFactory(cellData -> cellData.getValue().Lessons\_6());*// добавление строки в столбик (предмет 6)* **ball**.setCellValueFactory(cellData -> cellData.getValue().getBall());*// добавление строки в столбик (предмет 6)* **rating**.setCellValueFactory(cellData -> cellData.getValue().getRating());*// добавление строки в столбик (предмет 6)  
  
 // запрос на изменение данных* **lessons\_1**.setCellFactory(cellData -> **new** FloatCell (billList, cellData.getId()));  
 **lessons\_2**.setCellFactory(cellData -> **new** FloatCell (billList, cellData.getId()));  
 **lessons\_3**.setCellFactory(cellData -> **new** FloatCell (billList, cellData.getId()));  
 **lessons\_4**.setCellFactory(cellData -> **new** FloatCell (billList, cellData.getId()));  
 **lessons\_5**.setCellFactory(cellData -> **new** FloatCell (billList, cellData.getId()));  
 **lessons\_6**.setCellFactory(cellData -> **new** FloatCell (billList, cellData.getId()));  
  
 *//-------------------------------------------------------------------- подсчёт и сохранение* **table**.setOnMouseClicked(event -> {  
 **try** {  
 saveBill(); *// вызывает метод сохранения* } **catch** (Exception e) {  
 e.printStackTrace(); *// обработка ошибок* }  
 });  
 }  
  
 *//--------------------------------------Сохраняет текущую информацию об адресатах в указанном файле.--------------------------------------------* **public void** saveBill() **throws** Exception{  
 *// настройка пути к файлу* File file = **new** File(**group**);  
 **if** (!file.exists()) {file.mkdir();} *// если нету папки, то создать!  
  
 //обработка сохранения* **try** {  
 JAXBContext context = JAXBContext.*newInstance*(bill.**class**); *// какой тип данных будем сохранять* Marshaller marshaller = context.createMarshaller(); *// подготовка к расписанию данных* marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, **true**); *// для того, что бы не было в одну строчку  
  
 // Обёртываем наши данные об адресатах.* **int** i=1;  
 **for** (bill b:**billList**){ *// пока есть данные типа bill* file = **new** File(**group**+**"/"**+i+**".xml"**); *// указываем путь к файлу* file.createNewFile(); *// при необходимости создаём новый* marshaller.marshal(b, file); *// Маршаллируем и сохраняем XML в файл.* i++;  
 }  
  
 file = **new** File(**"nameGroup.xml"**);  
 context = JAXBContext.*newInstance*(Group.**class**); *// какой тип данных будем сохранять* marshaller = context.createMarshaller(); *// подготовка к расписанию данных* marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, **true**); *// для того, что бы не было в одну строчку* marshaller.marshal(**groupsName**, file); *// Маршаллируем и сохраняем XML в файл.* } **catch** (Exception e) { *// окно с ошибкой* Alert alert = **new** Alert(Alert.AlertType.***ERROR***);  
 alert.setTitle(**"Error"**);  
 alert.setHeaderText(**"Could not save data"**);  
 alert.showAndWait();  
 }  
 }  
  
 *//---------------------------------------------------------------------- добавляем новую строку --------------------------------------------* **static** Stage *dialogStage*;  
 **public void** addField(MouseEvent event) **throws** Exception {  
 FXMLLoader loader = **new** FXMLLoader(); *// создаем загрузку FXML* loader.setLocation(getClass().getResource(**"addBill.fxml"**)); *// добавляем нашу FXML* Pane page = loader.load(); *// создание панели  
 dialogStage* = **new** Stage();  
 *dialogStage*.initModality( Modality.***WINDOW\_MODAL***); *// блокировка предыдущей формы, при открытии новой* Scene scene = **new** Scene(page) ; *// создание сцены  
 dialogStage*.setScene(scene);  
  
 addFieldController addF = loader.getController(); *// контроллер* addF.init(**billList**); *// отправка данных этой формы, на новую  
 dialogStage*.setTitle(**"Добавить ученика"**);  
 *dialogStage*.showAndWait();  
 **this**.saveBill(); *// сохраняем* }  
  
 *//---------------------------------------------------------------------- Удаляем строку --------------------------------------------* **public void** deleteField(MouseEvent event) **throws** Exception {  
 **int** editingIndex = **table**.getSelectionModel().getFocusedIndex(); *// узнаём индекс* **billList**.remove(editingIndex); *// удаляем* File file = **new** File(**group**+**"/"**+editingIndex+**".xml"**); *// находим файл* **if** (file.exists()) {  
 Files.*delete*(Paths.*get*(**group**+**"/"**+(**billList**.size()+1)+**".xml"**)); *// удаляем* }  
  
 **this**.saveBill(); *// сохраняем* }  
  
 *//---------------------------------------------------------------------- добавляем новую группу --------------------------------------------* **public void** addColumn(MouseEvent event) **throws** Exception {  
 FXMLLoader loader = **new** FXMLLoader(); *// создаем загрузку FXML* loader.setLocation(getClass().getResource(**"addColumn.fxml"**)); *// добавляем нашу FXML* Pane page = loader.load(); *// создание панели  
 dialogStage* = **new** Stage();  
 *dialogStage*.initModality( Modality.***WINDOW\_MODAL***); *// блокировка предыдущей формы, при открытии новой* Scene scene = **new** Scene(page) ; *// создание сцены  
 dialogStage*.setScene(scene);  
  
 addColumnController addF = loader.getController(); *// контроллер* addF.init(**groupsName**); *// отправка данных этой формы, на новую  
 dialogStage*.setTitle(**"Добавить группу"**);  
 *dialogStage*.showAndWait();  
 }  
  
 *//---------------------------------------------------------------------- Удаляем группу --------------------------------------------* **public void** deleteColumn(MouseEvent event) **throws** Exception {  
 String del = **""**+**groupColumn**.getValue(); *// узнаём имя выбранной группы* ObservableList<String> list = **groupsName**.getGroup(); *// выгружаем группы* **for** (**int** i=0;i<list.size();i++) {  
 **if** (list.get(i).equals(del)) {list.remove(i);} *// находим нашу, по циклу и удаляем* }  
 **groupsName**.setGroup(list); *// загружаем группы, обратно* File file = **new** File(**"group/"**+del); *// удаляем файлы с данной группой* **this**.deleteDir(file); *// вызываем метод удаления* **groupColumn**.setValue(**null**);  
 }  
  
 **void** deleteDir(File file) { *// метод удаления* File[] contents = file.listFiles(); *// узнаём содержимое* **if** (contents != **null**) { *// если оно не равно нулю* **for** (File f : contents) { *// то проходимся по ним* **if** (! Files.*isSymbolicLink*(f.toPath())) {  
 deleteDir(f); *// и запускаем рекурсию, на удалнеие* }  
 }  
 }  
 file.delete(); *// удаление файла или папки* }  
}

**bill**

**package** sample;  
  
**import** javafx.beans.property.FloatProperty;  
**import** javafx.beans.property.SimpleFloatProperty;  
**import** javafx.beans.property.SimpleStringProperty;  
**import** javafx.beans.property.StringProperty;  
  
**import** javax.xml.bind.annotation.\*;  
  
*// Используется для сохранения списка адресатов в XML.*@XmlRootElement(name = **"bills"**)  
@XmlAccessorType(XmlAccessType.***PROPERTY***)  
@XmlType(name = **"bills"**, propOrder = {  
 **"FIO"**,  
 **"lessons\_1"**,  
 **"lessons\_2"**,  
 **"lessons\_3"**,  
 **"lessons\_4"**,  
 **"lessons\_5"**,  
 **"lessons\_6"**,  
 **"ball"**,  
 **"rating"**})  
**public class** bill {  
 *//--------------------------------------------------------------------------------- переменные ------------------------------------------------------------* StringProperty **FIO** = **new** SimpleStringProperty(**""**); *// имя* FloatProperty **lessons\_1** = **new** SimpleFloatProperty( 0 ) ; *// предмет 1* FloatProperty **lessons\_2** = **new** SimpleFloatProperty( 0 ) ; *// предмет 2* FloatProperty **lessons\_3** = **new** SimpleFloatProperty( 0 ) ; *// предмет 3* FloatProperty **lessons\_4** = **new** SimpleFloatProperty( 0 ) ; *// предмет 4* FloatProperty **lessons\_5** = **new** SimpleFloatProperty( 0 ) ; *// предмет 5* FloatProperty **lessons\_6** = **new** SimpleFloatProperty( 0 ) ; *// предмет 6* FloatProperty **ball** = **new** SimpleFloatProperty( 0 ) ; *//* FloatProperty **rating** = **new** SimpleFloatProperty( 0 ) ; *// рейтинг* **public** bill (){  
 }  
 *//-------------------------------------------------------------------------------------конструтор -----------------------------------------------* **public** bill (  
 String name,  
 Float lessons\_1,  
 Float lessons\_2,  
 Float lessons\_3,  
 Float lessons\_4,  
 Float lessons\_5,  
 Float lessons\_6  
 )  
 {  
 **this**.**FIO** = **new** SimpleStringProperty(name); *// добавить имя в класс* **this**.**lessons\_1** = **new** SimpleFloatProperty(lessons\_1); *// добавить предмет в класс* **this**.**lessons\_2** = **new** SimpleFloatProperty(lessons\_2); *// добавить предмет в класс* **this**.**lessons\_3** = **new** SimpleFloatProperty(lessons\_3); *// добавить предмет в класс* **this**.**lessons\_4** = **new** SimpleFloatProperty(lessons\_4); *// добавить предмет в класс* **this**.**lessons\_5** = **new** SimpleFloatProperty(lessons\_5); *// добавить предмет в класс* **this**.**lessons\_6** = **new** SimpleFloatProperty(lessons\_6); *// добавить предмет в класс* **this**.**ball** = **new** SimpleFloatProperty(  
 (lessons\_1+lessons\_2+lessons\_3+lessons\_4+lessons\_5+lessons\_6)/6  
 );  
 **this**.**rating** = **new** SimpleFloatProperty(0); *// добавить предмет в класс* }  
  
 **public** bill(String text) {  
 **this**.**FIO** = **new** SimpleStringProperty(text); *// добавить имя в класс* }  
  
 *// --------------------------------------------------------------------------------- обработка событый -------------------------------------------------------  
  
 //---------------------------------------------------------- ИМЯ  
 //@XmlElement(name = "name")* **public** String getFIO () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return FIO**.get();  
 }*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public** StringProperty nameProperty () { *// вернуть значение-строку с именем* **return FIO**;  
 }  
  
 **public void** setFIO (String name) { *// записать имя (ещё не использовали, но вдруг пригодиться)* **this**.**FIO**.set(name);  
 }  
 *//----------------------------------------------------------- предмет 1* @XmlElement(name = **"lessons\_1"**)  
 **public** Float getlessons\_1() {**return lessons\_1**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setlessons\_1(Float lessons\_1) {**this**.**lessons\_1**.set(lessons\_1);}  
  
 **public** FloatProperty Lessons\_1 () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return lessons\_1**;  
 }  
 *//----------------------------------------------------------- предмет 2* @XmlElement(name = **"lessons\_2"**)  
 **public** Float getlessons\_2() {**return lessons\_2**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setlessons\_2(Float lessons\_2) {**this**.**lessons\_2**.set(lessons\_2);}  
  
 **public** FloatProperty Lessons\_2 () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return lessons\_2**;  
 }  
 *//----------------------------------------------------------- предмет 3* @XmlElement(name = **"lessons\_3"**)  
 **public** Float getlessons\_3() {**return lessons\_3**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setlessons\_3(Float lessons\_3) {**this**.**lessons\_3**.set(lessons\_3);}  
  
 **public** FloatProperty Lessons\_3 () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return lessons\_3**;  
 }  
 *//----------------------------------------------------------- предмет 4* @XmlElement(name = **"lessons\_4"**)  
 **public** Float getlessons\_4() {**return lessons\_4**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setlessons\_4(Float lessons\_4) {**this**.**lessons\_4**.set(lessons\_4);}  
  
 **public** FloatProperty Lessons\_4 () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return lessons\_4**;  
 }  
 *//----------------------------------------------------------- предмет 5* @XmlElement(name = **"lessons\_5"**)  
 **public** Float getlessons\_5() {**return lessons\_5**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setlessons\_5(Float lessons\_5) {**this**.**lessons\_5**.set(lessons\_5);}  
  
 **public** FloatProperty Lessons\_5 () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return lessons\_5**;  
 }  
 *//----------------------------------------------------------- предмет 6* @XmlElement(name = **"lessons\_6"**)  
 **public** Float getlessons\_6() {**return lessons\_6**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setlessons\_6(Float lessons\_6) {**this**.**lessons\_6**.set(lessons\_6);}  
  
 **public** FloatProperty Lessons\_6 () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return lessons\_6**;  
 }  
 *//----------------------------------------------------------- рейтинг* @XmlElement(name = **"ball"**)  
 **public** Float getball() {**return ball**.get();} *// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setball(Float ball) {**this**.**ball**.set(ball);}  
  
 **public** FloatProperty getBall () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return ball**;  
 }  
 @XmlElement(name = **"rating"**)  
 **public** Float getrating() {**return rating**.get();}*// необходимо для сериализации в file.xml(marshaller/Unmarshaller)* **public void** setrating(Float rating) {**this**.**rating**.set(rating);}  
  
 **public** FloatProperty getRating () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return rating**;  
 }  
 **public void** setRating (**float** rating) { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **this**.**rating**=**new** SimpleFloatProperty(rating);  
 }  
 **public void** setBall(**float** ball) { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **this**.**ball**=**new** SimpleFloatProperty(ball);  
 }  
}

**FloatCell**

**package** sample;  
  
**import** javafx.collections.ObservableList;  
**import** javafx.scene.control.TableCell;  
**import** sample.FloatField;  
  
**import** java.util.ArrayList;  
**import** java.util.Collections;  
**import** java.util.List;  
  
**public class** FloatCell **extends** TableCell<bill,Number> { *// наследуем таблицу переменная / число* FloatField **number**; *// наше поле для числа* ObservableList<bill> **billList**; *// список переменных* String **nameLessons**=**""**;  
 bill **myBill**; *// наше поле для переменной  
  
 // конструктор* **public** FloatCell(ObservableList<bill> expenseList, String nameLessons) {  
 **this**.**billList** = expenseList; *// присваиваем список* **this**.**nameLessons** = nameLessons;  
 }  
  
 @Override  
 **public void** startEdit() { *// начало редактирования* **if**(!isEmpty()){  
 **super**.startEdit();  
 **number**=**new** FloatField(); *// новое поле FloatField* **myBill**=getTableView().getSelectionModel().getSelectedItem(); *// выбранное поле* **switch** (**nameLessons**){  
 **case** (**"lessons\_1"**): **number**.setText(**myBill**.Lessons\_1().getValue().toString()); *// вставляем наш текст* **break**;  
 **case** (**"lessons\_2"**): **number**.setText(**myBill**.Lessons\_2().getValue().toString()); *// вставляем наш текст* **break**;  
 **case** (**"lessons\_3"**): **number**.setText(**myBill**.Lessons\_3().getValue().toString()); *// вставляем наш текст* **break**;  
 **case** (**"lessons\_4"**): **number**.setText(**myBill**.Lessons\_4().getValue().toString()); *// вставляем наш текст* **break**;  
 **case** (**"lessons\_5"**): **number**.setText(**myBill**.Lessons\_5().getValue().toString()); *// вставляем наш текст* **break**;  
 **case** (**"lessons\_6"**): **number**.setText(**myBill**.Lessons\_6().getValue().toString()); *// вставляем наш текст* **break**;  
 }  
 setGraphic(**number**); *//отображаем число* }  
 }  
  
 @Override  
 **public void** cancelEdit() { *// отмена редактирования* **try** {  
 **super**.cancelEdit();  
 **int** index = **billList**.indexOf(**myBill**); *// узнаем индекс* **float** newCost = Float.*parseFloat*(**number**.getText()); *// переводим число в Float* bill newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), **myBill**.**lessons\_2**.getValue(), **myBill**.**lessons\_3**.getValue(), **myBill**.**lessons\_4**.getValue(), **myBill**.**lessons\_5**.getValue(), **myBill**.**lessons\_6**.getValue()); *// подгружаем таблицу* **switch** (**nameLessons**) {  
 **case** (**"lessons\_1"**):  
 newExpense = **new** bill(**myBill**.getFIO(), newCost, **myBill**.**lessons\_2**.getValue(), **myBill**.**lessons\_3**.getValue(), **myBill**.**lessons\_4**.getValue(), **myBill**.**lessons\_5**.getValue(), **myBill**.**lessons\_6**.getValue());  
 **break**; *// подгружаем таблицу* **case** (**"lessons\_2"**):  
 newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), newCost, **myBill**.**lessons\_3**.getValue(), **myBill**.**lessons\_4**.getValue(), **myBill**.**lessons\_5**.getValue(), **myBill**.**lessons\_6**.getValue());  
 **break**; *// подгружаем таблицу* **case** (**"lessons\_3"**):  
 newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), **myBill**.**lessons\_2**.getValue(), newCost, **myBill**.**lessons\_4**.getValue(), **myBill**.**lessons\_5**.getValue(), **myBill**.**lessons\_6**.getValue());  
 **break**; *// подгружаем таблицу* **case** (**"lessons\_4"**):  
 newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), **myBill**.**lessons\_2**.getValue(), **myBill**.**lessons\_3**.getValue(), newCost, **myBill**.**lessons\_5**.getValue(), **myBill**.**lessons\_6**.getValue());  
 **break**; *// подгружаем таблицу* **case** (**"lessons\_5"**):  
 newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), **myBill**.**lessons\_2**.getValue(), **myBill**.**lessons\_3**.getValue(), **myBill**.**lessons\_4**.getValue(), newCost, **myBill**.**lessons\_6**.getValue());  
 **break**; *// подгружаем таблицу* **case** (**"lessons\_6"**):  
 newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), **myBill**.**lessons\_2**.getValue(), **myBill**.**lessons\_3**.getValue(), **myBill**.**lessons\_4**.getValue(), **myBill**.**lessons\_5**.getValue(), newCost);  
 **break**; *// подгружаем таблицу* **default**:  
 newExpense = **new** bill(**myBill**.getFIO(), **myBill**.**lessons\_1**.getValue(), **myBill**.**lessons\_2**.getValue(), **myBill**.**lessons\_3**.getValue(), **myBill**.**lessons\_4**.getValue(), **myBill**.**lessons\_5**.getValue(), **myBill**.**lessons\_6**.getValue());  
 **break**; *// подгружаем таблицу* }  
  
 **billList**.set(index, newExpense); *// меняем значение* setGraphic(**null**); *//не отображаем число* }**catch** (Exception e){  
 System.***out***.println(e);  
 }  
  
 *//---------------------------------------------------------------------------------------------- sorted ------------------------------------------* **try** {  
 List<Float> ballList = **new** ArrayList<Float>(); *// список баллов* **for** (bill b: **billList**){  
 ballList.add(b.getBall().getValue());  
 }  
  
 *// сортировка  
 //Collections.sort(ballList, Collections.reverseOrder()); // программная сортировка коллекций* **this**.*sort*(ballList); *// наша сортировка* **for** (**int** j = 0; j < **billList**.size(); j++){ *// установка значений рейтинга* **for**(**int** i = 0; i < ballList.size(); i++){  
 **float** f1=ballList.get(i);  
 **float** f2=**billList**.get(j).getBall().getValue();  
 **if**(f1==f2){ *// если совпали баллы, то записать рейтинг в нашу коллекцию* bill b = **billList**.get(j);  
 b.setRating(i+1);  
 **billList**.set(j, b);  
 }  
 }  
 }  
 }**catch** (Exception e){  
 System.***out***.println(e);  
 }  
 }  
  
 **public static** List<Float> sort(List<Float> array) { *// сортировка* **for** (**int** i = 0; i < array.size(); i++) {  
 **for** (**int** j = 0; j < array.size(); j++) {  
 **if** (array.get(j) < array.get(i)) { *// сравниваем два рядом стоящих значения* **float** aj = array.get(j);  
 **float** ai = array.get(i);  
 array.set(j, ai); *// меняем местами* array.set(i, aj);  
 }  
 }  
 }  
 **return** array; *// возвращаем коллекцию* }  
  
 @Override  
 **public void** updateItem(Number item, **boolean** empty) { *// обновление поля* **super**.updateItem (item, empty);  
 **if**(empty) { *// пустое?* setText(**null**); *// пустое поле* setGraphic(**null**); ;  
 } **else  
 if**(isEditing()) *// редаткируем? в(°ロ°)д* setGraphic(**number**); *//отображаем число* **else** {  
 setText(getItem().toString()) ;  
 setGraphic(**null**) ;  
 }  
 }  
}

**FloatField**

**package** sample;  
  
**import** javafx.scene.control.TextField;  
**import** javafx.scene.control.TextFormatter;  
  
**public class** FloatField **extends** TextField { *// Создание кастомного(своего) текстового поля* **public** FloatField(){ *// конструктор* **super**(); *// вызывает конструктор родителя* **this**.setText(**"0"**);*// запись текста "0"* **this**.setTextFormatter(**new** TextFormatter<String>( *// изменение формата текста* s->{  
 **if** (s.getControlNewText().isEmpty()) *// он пустой?* **return** s;*// да?! тогда верни его обратно!* **try** {  
 Float.*parseFloat*(s.getControlNewText());*// а если всё хорошо, то распарси его во Float* **return** s; *// и верни уже её* } **catch**(NumberFormatException e){ *// обработка исключений* **return null**;  
 }  
 }  
) ) ;  
 }  
}

**addFieldController**

**package** sample;  
  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.fxml.FXML;  
**import** javafx.scene.control.TextField;  
**import** javafx.scene.input.MouseEvent;  
  
**public class** addFieldController {  
  
 ObservableList<bill> **billList**;  
 @FXML TextField **addName**;  
  
 **public void** addField(MouseEvent event) { *// добавление полей учеников* String s=**addName**.getText();  
 s = s.replaceAll(**"[^A-Za-zА-Яа-я0-9]"**, **""**);  
 **if** (!s.equals(**""**)){  
 **billList**.add(**new** bill(s)); *// добавляем в нашу коллекцию, данные из текстового поля* **addName**.setText(**""**); *// очистить поле* }  
 }  
  
 **public void** init(ObservableList<bill> billList) {  
 **this**.**billList** = billList;  
 }  
}

**addBill.fxml**

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.layout.HBox***?>*<**HBox alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="45.0" prefWidth="382.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.addFieldController"**>  
 <**children**>  
 <**Label text="ФИО "** />  
 <**TextField fx:id="addName" prefHeight="26.0" prefWidth="210.0"** />  
 <**Button onMouseClicked="#addField" mnemonicParsing="false" text="Добавить"** />  
 </**children**>  
</**HBox**>

**addColumnController**

**package** sample;  
  
**import** javafx.collections.ObservableList;  
**import** javafx.fxml.FXML;  
**import** javafx.scene.control.TextField;  
**import** javafx.scene.input.MouseEvent;  
  
**import** java.io.File;  
  
**public class** addColumnController {  
 Group **groups**;  
 @FXML TextField **addName**;  
  
 **public void** addField(MouseEvent event) { *// добавление названий групп* String s=**addName**.getText();  
 s = s.replaceAll(**"[^A-Za-zА-Яа-я0-9]"**, **""**);  
 **for** (String g:**groups**.getGroup()){  
 **if**(g.equals(s))s=**""**;  
 }  
 **if** (!s.equals(**""**)) {  
 **groups**.getGroup().add(s); *// добавляем в нашу коллекцию, данные из текстового поля* File file = **new** File(**"group/"**+s); *// файл с таким именем* **if** (!file.exists()) {file.mkdir();} *// если нету папки, то создать!* **addName**.setText(**""**); *// очистить поле* }  
 }  
  
 **public void** init(Group groups) {  
 **this**.**groups** = groups;  
 }  
}

**addColumn.fxml**

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.layout.HBox***?>*<**HBox alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="45.0" prefWidth="382.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.addColumnController"**>  
 <**children**>  
 <**Label text="Группа "** />  
 <**TextField fx:id="addName" prefHeight="26.0" prefWidth="210.0"** />  
 <**Button onMouseClicked="#addField" mnemonicParsing="false" text="Добавить"** />  
 </**children**>  
</**HBox**>

**Group**

**package** sample;  
  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**import** javax.xml.bind.annotation.\*;  
  
*/\*\*  
 \* Вспомогательный класс для обёртывания списка адресатов.  
 \* Используется для сохранения списка адресатов в XML.  
 \*/*@XmlRootElement(name = **"Group"**)  
@XmlAccessorType(XmlAccessType.***PROPERTY***)  
@XmlType(name = **"Group"**, propOrder = {  
 **"group"**})  
**public class** Group {  
  
 **private** ObservableList<String> **group** = FXCollections.*observableArrayList*(); *// коллекция названий наших групп  
  
 //@XmlElement(name = "group")* **public** ObservableList<String> getGroup() { *// вернуть* **return group**;  
 }  
 **public void** setGroup(ObservableList<String> group) { *// получить* **this**.**group** = group;  
 }  
 **public void** setGroup(String ...group) { *// получить в виде строчек* ObservableList<String> myArray = **null**;  
 **for** (String s:group){  
 myArray.add(s);  
 }  
 setGroup(myArray);  
 }  
}

nameGroup.xml

*<?***xml version="1.0" encoding="UTF-8" standalone="yes"***?>*<**Group**>  
 <**group**>101</**group**>  
 <**group**>103</**group**>  
 <**group**>105</**group**>  
 <**group**>107</**group**>  
 <**group**>109</**group**>  
</**Group**>

group/101/1.xml

*<?***xml version="1.0" encoding="UTF-8" standalone="yes"***?>*<**bills**>  
 <**FIO**>Рожков Иван Иванович</**FIO**>  
 <**lessons\_1**>89.0</**lessons\_1**>  
 <**lessons\_2**>22.0</**lessons\_2**>  
 <**lessons\_3**>78.0</**lessons\_3**>  
 <**lessons\_4**>56.0</**lessons\_4**>  
 <**lessons\_5**>66.0</**lessons\_5**>  
 <**lessons\_6**>78.0</**lessons\_6**>  
 <**ball**>64.833336</**ball**>  
 <**rating**>2.0</**rating**>  
</**bills**>