|  |  |
| --- | --- |
| Федеральное агентство железнодорожного транспорта  Омский государственный университет путей сообщения  Кафедра «Автоматика и системы управления»  РЕДАКТИРОВАНИЕ ДАННЫХ В TABLEVIEW  Лабораторная работа № 5  по теме «Программирование» | |
|  | Студент гр. 78-ИС                           Швед Т.Ю.  «    »                  2021 г.  Руководитель –                           Альтман Е. А.  «    »                  2021 г. |
| Омск 2021 | |

# Цель работы

Изучить методы редактирования данных с помощью  диалоговых окон и изменением ячеек компонента TableView. На изучаемом языке программирования Java и изучение применения Java FX для типовых задач. Для этого нужно выполнить следующие написать программу, в которой повторим пример приложения, приведенный в теоретической части, а это:

повторить пример приложения, приведенный в теоретической части;

переделать приложение для работы с вариантом класса «Expense», разработанным в предыдущей работе;

реализовать класс для ячеек в столбце с категорий затрат так, чтобы пользователь мог выбрать категорию из выпадающего списка.

## Замечания по текущему документу

Данный документ преследует цель показать редактирования данных с помощью диалоговых окон и изменением ячеек компонента TableView на изучаемом языке программирования Java и изучение применения Java FX для типовых задач. Научиться этому является основной целью лабораторной работы и необходимо для выполнения всех последующих лабораторных работ.

## Написание программы, в которой повторим пример приложения, приведенный в теоретической части.

Одним из способов изменения данных в таблице заключается в том, чтобы создать отдельное окно в котором редактируются записи для отдельной строки таблицы. Создадим разметку для окна, приведенную на рисунке 1.

|  |
| --- |
|  |
| Рисунок  – Файл «fxml» |

Теперь для этого окна создадим контроллер, как показано на рисунке 2 и 3.

|  |
| --- |
|  |
| Рисунок  – Контроллер для редактирования, переменные |

|  |
| --- |
|  |
| Рисунок  – Контроллер для редактирования, методы |

Метод «init» должен вызываться перед запуском окна. Он запоминает ссылку на коллекцию и индекс элемента и инициализирует значения полей. Также он устанавливает метод «exit» как обработчик события, возникающего при выходе из окна. Метод «exit» запоминает введенные пользователем данные.

В главное окно программы добавим кнопку «Изменить». Обработчик нажатия на эту кнопку должен запустить диалоговое окно для изменения текущей записи, как показано на рисунке 4.

|  |
| --- |
|  |
| Рисунок  – Кнопка взывающая окно редактирования |

Создадим класс для работы с ячейкой, который бы позволял редактировать в ячейке вещественные числа, как показано на рисунке 5.

|  |
| --- |
|  |
| Рисунок  – Как «Tovar» в самом файле «fxml» |

Для корректной работы класса нам нужно переопределить три метода: «startEdit», который вызывается при начале редактирования ячейки; «cancelEdit», который вызывается при завершении редактирования и «updateItem», который вызывается при необходимости обновить вид ячейки, как показано на рисунке 6, 7 и 8.

|  |
| --- |
|  |
| Рисунок  – Метод «startEdit» |

|  |
| --- |
|  |
| Рисунок  – Метод «cancelEdit» |

|  |
| --- |
|  |
| Рисунок  – Метод «updateItem» |

После создания класса, ответственного за редактирование ячейки, его нужно привязать для вызова в нужном столбце, как показано на рисунке 9.

|  |
| --- |
|  |
| Рисунок  – Привязать для вызова в нужном столбце |

А так же обработку для выпадающего списка, как показано на рисунке 10.

|  |
| --- |
|  |
| Рисунок  – Обработка для выпадающего списка |

И вот что мы увидим, при запуске программы и её редактировании, как показано на рисунке 7 и 8.

|  |
| --- |
|  |
| Рисунок 7 – Проверка работы, основное окно |

|  |
| --- |
|  |
| Рисунок 8 – Проверка работы, окно для редактирования |

## Изменение программы

Текст программы набирается в окне файла с расширением «.java» и «.fxml». Исходный код приводится в виде листинга, в приложения Б.

2. Выводы по лабораторной работе

Из выше указанного, можно точно сказать, что был достигнут искомый результат. Показан основной процесс редактирования данных с помощью диалоговых окон и изменением ячеек компонента TableView на изучаемом языке программирования Java, с применением Java FX .

Приложение А

(справочное)

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(**"TableView"**); *// заголовок формы* primaryStage.setScene(**new** Scene(root, 310, 330)); *// размеры формы и наша панель, в форму(окно)* primaryStage.show(); *// показать форму* }  
   
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

sample.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.geometry.Insets***?>  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.effect.Blend***?>  
<?***import javafx.scene.layout.BorderPane***?>  
<?***import javafx.scene.layout.HBox***?>  
<?***import javafx.scene.layout.VBox***?>  
<?***import javafx.scene.text.Font***?>  
<?***import javafx.scene.control.TableView***?>  
  
<?***import javafx.scene.control.TableColumn***?>*<**BorderPane fx:id="root" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="200.0" prefWidth="300.0" xmlns="http://javafx.com/javafx/8.0.172" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**center**>  
 <**TableView fx:id="table" BorderPane.alignment ="CENTER"**>  
 <**columns** >  
 <**TableColumn fx:id ="nameColumn" text="Название"**/>  
 <**TableColumn fx:id ="costColumn" text="Цена"**/>  
 <**TableColumn fx:id ="categoryColumn" text="Категория расходов"**/>  
 </**columns** >  
 </**TableView** >  
 </**center**>  
 <**bottom**>  
 <**Label fx:id ="sum" BorderPane.alignment ="CENTER"**></**Label**>  
 </**bottom**>  
</**BorderPane**>

Controller

**package** sample;  
  
**import** javafx.beans.value.ObservableValue;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.ActionEvent;  
**import** javafx.event.EventHandler;  
**import** javafx.fxml.FXML;  
**import** javafx.scene.control.\*;  
  
**public class** Controller {  
  
 *// -------------------------------------------------------------------- переменные ---------------------------------------------* ObservableList<Expense> **expenseList** = FXCollections.*observableArrayList*();  
  
 @FXML TableView<Expense> **table**;  
 @FXML TableColumn<Expense, String> **nameColumn**;  
 @FXML TableColumn<Expense, Number> **costColumn**;  
 @FXML TableColumn<Expense, String> **categoryColumn**;  
 @FXML Label **sum**;  
  
 *// ----------------------------------------------------------------------- инициализация ---------------------------------------* **public void** init (ObservableList<Expense> expenseList){  
 **table**.setItems(expenseList) ;  
 *// добавление строки в столбик (имя)* **nameColumn**.setCellValueFactory(cellData ->  
 cellData.getValue().nameProperty()) ;  
 *// добавление строки в столбик (цена)* **costColumn**.setCellValueFactory(cellData ->  
 cellData.getValue().costProperty()) ;  
 *// подсчёт затрат* **float** sumCost=0;  
 **for** (Expense e: expenseList){ *// перебор затрат* sumCost += e.**cost**.getValue();  
 }  
 **sum**.setText(**"Итого: "**+sumCost+**" руб."**); *// вывод затрат  
 // добавление строки в столбик (цена)* **categoryColumn**.setCellValueFactory(cellData ->  
 cellData.getValue().categoryOfExpensesProperty()) ;  
 }  
  
 *// ----------------------------------------- пустая инициализация (если пользователь ничего не передал)* **public void** init() {  
 **this**.testInit();  
 **this**.init(**expenseList**);  
 }  
  
 *// ----------------------------------------- тестовая инициализация* **public void** testInit() {  
 **expenseList**.add(**new** Expense ( **"Хлеб"**, **new** Float(30), **"Еда"**));  
 **expenseList**.add ( **new** Expense ( **"Вода"**, **new** Float(35 ), **"Еда"**));  
 **expenseList**.add ( **new** Expense ( **"Проезд"**, **new** Float(30 ), **"Личное"**));  
 **expenseList**.add ( **new** Expense ( **"Верёвка"**, **new** Float(150 ), **"Хозяйство"**));  
 **expenseList**.add ( **new** Expense ( **"Книга"**, **new** Float(340 ), **"Личное"**));  
 **expenseList**.add ( **new** Expense ( **"Мыло"**, **new** Float(120 ), **"Хозяйство"**));  
 **expenseList**.add ( **new** Expense ( **"Масло"**, **new** Float(35 ), **"Еда"**));  
 **expenseList**.add ( **new** Expense ( **"Сыр"**, **new** Float(220 ), **"Еда"**));  
 **expenseList**.add ( **new** Expense ( **"Табурет"**, **new** Float(1500 ), **"Хозяйство"**));  
 *//ObservableValue<Number> x; // данная строчка не имеет смысла, но может пригодится если переделать на возвращаемую функцию. Но у нас void =D* }  
}

Observable

**package** sample;  
  
**import** javafx.beans.InvalidationListener;  
  
**public interface** Observable {  
 **void** addListener(InvalidationListener listener);  
 **void** removeListener(InvalidationListener listener);  
}

Expense

**package** sample;  
  
**import** javafx.beans.property.FloatProperty;  
**import** javafx.beans.property.SimpleFloatProperty;  
**import** javafx.beans.property.SimpleStringProperty;  
**import** javafx.beans.property.StringProperty;  
**import** javafx.beans.value.ObservableValue;  
  
**public class** Expense {  
 *//--------------------------------------------------------------------------------- переменные ------------------------------------------------------------* StringProperty **name** = **new** SimpleStringProperty(**""**); *// имя* FloatProperty **cost** = **new** SimpleFloatProperty( 0 ) ; *// цена* StringProperty **categoryOfExpenses** = **new** SimpleStringProperty(**""**); *// категория расходов  
  
 // --------------------------------------------------------------------------------- обработка событый -------------------------------------------------------  
  
 //----------------------------- ИМЯ* **public** String getName () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return name**.get();  
 }  
 **public** StringProperty nameProperty () { *// вернуть значение-строку с именем* **return name**;  
 }  
 **public void** setName (String name) { *// записать имя (ещё не использовали, но вдруг пригодиться)* **this**.**name**.set(name);  
 }  
 *//------------------------------ ЦЕНА* **public** FloatProperty costProperty () { *// вернуть значение-дробное с ценой* **return cost**;  
 }  
 *//----------------------------- КАТЕГОРИЯ* **public** String getСategoryOfExpenses () { *// вернуть категорию (ещё не использовали, но вдруг пригодиться)* **return categoryOfExpenses**.get();  
 }  
 **public** StringProperty categoryOfExpensesProperty () { *// вернуть значение-строку с категорией* **return categoryOfExpenses**;  
 }  
 **public void** setСategoryOfExpenses (String name) { *// записать категорию (ещё не использовали, но вдруг пригодиться)* **this**.**categoryOfExpenses**.set(name);  
 }  
  
  
  
 *//-------------------------------------------------------------------------------------конструтор -----------------------------------------------* **public** Expense ( String name , Float cost, String categoryOfExpenses) {  
 **this**.**name** = **new** SimpleStringProperty(name); *// добавить имя в класс* **this**.**cost** = **new** SimpleFloatProperty(cost); *// добавить цену в класс* **this**.**categoryOfExpenses** = **new** SimpleStringProperty(categoryOfExpenses); *// добавить цену в класс* }  
  
}

Приложение Б

(справочное)

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(**"TableCell"**); *// заголовок формы* primaryStage.setScene(**new** Scene(root, 310, 350)); *// размеры формы и наша панель, в форму(окно)* primaryStage.show(); *// показать форму* }  
  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

sample.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.geometry.Insets***?>  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.effect.Blend***?>  
<?***import javafx.scene.layout.BorderPane***?>  
<?***import javafx.scene.layout.HBox***?>  
<?***import javafx.scene.layout.VBox***?>  
<?***import javafx.scene.text.Font***?>  
<?***import javafx.scene.control.TableView***?>  
  
<?***import javafx.scene.control.TableColumn***?>*<**BorderPane fx:id="root" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="200.0" prefWidth="300.0" xmlns="http://javafx.com/javafx/8.0.172" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**center**>  
 <**TableView fx:id="table" BorderPane.alignment ="CENTER"**>  
 <**columns** >  
 <**TableColumn fx:id ="nameColumn" text="Название"**/>  
 <**TableColumn fx:id ="costColumn" text="Цена"**/>  
 <**TableColumn fx:id ="categoryColumn" text="Категория расходов"**/>  
 </**columns** >  
 </**TableView** >  
 </**center**>  
 <**bottom**>  
 <**HBox alignment ="CENTER"**>  
 <**Button id="1" fx:id="ok" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onMouseClicked="#edit" text="Изменить" textAlignment="CENTER" textOverrun="CENTER\_WORD\_ELLIPSIS" BorderPane.alignment="BOTTOM\_CENTER"**>  
 <**font**>  
 <**Font name="System Bold Italic" size="18.0"** />  
 </**font**>  
 </**Button**>  
 <**Label alignment ="CENTER" fx:id ="sum" BorderPane.alignment ="CENTER"** />  
 </**HBox**>  
 </**bottom**>  
</**BorderPane**>

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.ActionEvent;  
**import** javafx.event.EventHandler;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.ComboBoxTableCell;  
**import** javafx.scene.input.MouseEvent;  
**import** javafx.scene.layout.Pane;  
**import** javafx.stage.Modality;  
**import** javafx.stage.Stage;  
**import** javafx.util.Callback;  
  
**import** javafx.application.Application;  
**import** javafx.beans.property.SimpleBooleanProperty;  
**import** javafx.beans.property.SimpleObjectProperty;  
**import** javafx.beans.value.ChangeListener;  
**import** javafx.beans.value.ObservableValue;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.geometry.Insets;  
**import** javafx.geometry.Pos;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.TableCell;  
**import** javafx.scene.control.TableColumn;  
**import** javafx.scene.control.TableColumn.CellDataFeatures;  
**import** javafx.scene.control.TableColumn.CellEditEvent;  
**import** javafx.scene.control.TablePosition;  
**import** javafx.scene.control.TableView;  
**import** javafx.scene.control.cell.CheckBoxTableCell;  
**import** javafx.scene.control.cell.ComboBoxTableCell;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** javafx.scene.control.cell.TextFieldTableCell;  
**import** javafx.scene.layout.StackPane;  
**import** javafx.stage.Stage;  
**import** javafx.util.Callback;  
  
**import** java.io.IOException;  
  
**public class** Controller {  
  
 **static** Stage *dialogStage*;  
  
 *// -------------------------------------------------------------------- переменные ---------------------------------------------* ObservableList<Expense> **expenseList** = FXCollections.*observableArrayList*();  
  
 @FXML TableView<Expense> **table**;  
 @FXML TableColumn<Expense, String> **nameColumn**;  
 @FXML TableColumn<Expense, Number> **costColumn**;  
 @FXML TableColumn<Expense, CategoryOfExpenses> **categoryColumn**;  
 @FXML Label **sum**;  
  
 *// ----------------------------------------------------------------------- инициализация ---------------------------------------* **public void** init (ObservableList<Expense> expenseList){  
 **table**.setItems(expenseList) ;  
 **table**.setEditable(**true**);  
  
 *// добавление строки в столбик (имя)* **nameColumn**.setCellValueFactory(cellData ->  
 cellData.getValue().nameProperty());  
  
 *// добавление строки в столбик (цена)* **costColumn**.setCellValueFactory(cellData ->  
 cellData.getValue().costProperty());  
  
 **costColumn**.setCellFactory(cellData ->  
 **new** FloatCell (expenseList)  
 );  
  
 *// подсчёт затрат* **table**.setOnMouseClicked(event -> itogoUpdate());  
  
 *// ================================================================== Category Of Expenses (COMBO BOX)* ObservableList<CategoryOfExpenses> categoryList = FXCollections.*observableArrayList*( *// лист с категориями (enum)* CategoryOfExpenses.*values*());  
  
 *// добавление строки в столбик (Категория)* **categoryColumn**.setCellValueFactory(**new** Callback<TableColumn.CellDataFeatures<Expense, CategoryOfExpenses>, ObservableValue<CategoryOfExpenses>>() {  
  
 @Override  
 **public** ObservableValue<CategoryOfExpenses> call(TableColumn.CellDataFeatures<Expense, CategoryOfExpenses> param) {  
 Expense name = param.getValue(); *// передаём параметры поля  
 // E,X,M* String categoryCode = name.getСategoryOfExpenses(); *// вытаскиваем код категории (строчка)* CategoryOfExpenses category = CategoryOfExpenses.*getByCode*(categoryCode); *// возвращает категорию по коду* **return new** SimpleObjectProperty<CategoryOfExpenses>(category);  
 }  
 });  
  
 *// подгрузка ComboBox, для изменения данных* **categoryColumn**.setCellFactory(ComboBoxTableCell.*forTableColumn*(categoryList));  
  
 *//запись выбранного поля ComboBox* **categoryColumn**.setOnEditCommit((TableColumn.CellEditEvent<Expense, CategoryOfExpenses> event) -> {  
 TablePosition<Expense, CategoryOfExpenses> pos = event.getTablePosition(); *// где меняем* CategoryOfExpenses newcategory = event.getNewValue(); *// значение новой, выбранной категории* **int** row = pos.getRow(); *// где меняем* Expense category = event.getTableView().getItems().get(row);  
  
 category.setСategoryOfExpenses(newcategory.getCode()); *// записываем новую категорию* });  
 }  
  
 *//--------------------------------------обновление результата! \(＾∀＾)/--------------------------------------------* **private void** itogoUpdate(){  
 FloatProperty sumCost= **new** SimpleFloatProperty(0f);  
 **for** (Expense e: **expenseList**){ *// перебор затрат* sumCost.setValue(sumCost.getValue()+e.**cost**.getValue());  
 }  
 **sum**.setText(**" Итого: "**+sumCost.getValue()+**" руб. "**); *// вывод затрат* }  
  
 *// ----------------------------------------- пустая инициализация (если пользователь ничего не передал)----------------------* **public void** init() {  
 **this**.testInit();  
 **this**.init(**expenseList**);  
 }  
  
 *// ----------------------------------------- тестовая инициализация---------------------------------------------------------------* **public void** testInit() {  
 **expenseList**.add(**new** Expense ( **"Хлеб"**, **new** Float(30), CategoryOfExpenses.***EAT***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Вода"**, **new** Float(35 ), CategoryOfExpenses.***EAT***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Проезд"**, **new** Float(30 ), CategoryOfExpenses.***MAN***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Верёвка"**, **new** Float(150 ), CategoryOfExpenses.***XOZ***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Книга"**, **new** Float(340 ), CategoryOfExpenses.***MAN***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Мыло"**, **new** Float(120 ), CategoryOfExpenses.***XOZ***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Масло"**, **new** Float(35 ), CategoryOfExpenses.***EAT***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Сыр"**, **new** Float(220 ), CategoryOfExpenses.***EAT***.getCode()));  
 **expenseList**.add ( **new** Expense ( **"Табурет"**, **new** Float(1500 ), CategoryOfExpenses.***XOZ***.getCode()));  
 *//ObservableValue<Number> x; // данная строчка не имеет смысла, но может пригодится если переделать на возвращаемую функцию. Но у нас void =D* }  
  
 *// ------------------------------------------------- Изменения через кнопку ---------------------------------------------------------------* **public void** edit(MouseEvent mouseEvent) **throws** Exception {  
 FXMLLoader loader = **new** FXMLLoader(); *// создание формы* loader.setLocation (getClass().getResource(**"editWin.fxml"**)); *// её разлиновка в xml* 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);  
  
 EditWinController editWinController = loader.getController(); *// контроллер* **int** editingIndex = **table**.getSelectionModel().getFocusedIndex();  
 editWinController.init(**expenseList**, editingIndex); *// отправка данных этой формы, на новую  
 dialogStage*.showAndWait();  
 }  
}

Observable

**package** sample;  
  
**import** javafx.beans.InvalidationListener;  
  
**public interface** Observable {  
 **void** addListener(InvalidationListener listener);  
 **void** removeListener(InvalidationListener listener);  
}

Expense

**package** sample;  
  
**import** javafx.beans.property.FloatProperty;  
**import** javafx.beans.property.SimpleFloatProperty;  
**import** javafx.beans.property.SimpleStringProperty;  
**import** javafx.beans.property.StringProperty;  
**import** javafx.beans.value.ObservableValue;  
  
**public class** Expense {  
 *//--------------------------------------------------------------------------------- переменные ------------------------------------------------------------* StringProperty **name** = **new** SimpleStringProperty(**""**); *// имя* FloatProperty **cost** = **new** SimpleFloatProperty( 0 ) ; *// цена* StringProperty **categoryOfExpenses** = **new** SimpleStringProperty(**""**); *// категория расходов  
  
 // --------------------------------------------------------------------------------- обработка событый -------------------------------------------------------  
  
 //----------------------------- ИМЯ* **public** String getName () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return name**.get();  
 }  
 **public** StringProperty nameProperty () { *// вернуть значение-строку с именем* **return name**;  
 }  
 **public void** setName (String name) { *// записать имя (ещё не использовали, но вдруг пригодиться)* **this**.**name**.set(name);  
 }  
 *//------------------------------ ЦЕНА* **public** FloatProperty costProperty () { *// вернуть значение-дробное с ценой* **return cost**;  
 }  
 **public** FloatProperty getCost () { *// вернуть имя (ещё не использовали, но вдруг пригодиться)* **return cost**;  
 }  
 *//----------------------------- КАТЕГОРИЯ* **public** String getСategoryOfExpenses () { *// вернуть категорию (ещё не использовали, но вдруг пригодиться)* **return categoryOfExpenses**.get();  
 }  
 **public** StringProperty categoryOfExpensesProperty () { *// вернуть значение-строку с категорией* **return categoryOfExpenses**;  
 }  
 **public void** setСategoryOfExpenses (String name) { *// записать категорию (ещё не использовали, но вдруг пригодиться)* **this**.**categoryOfExpenses**.set(name);  
 }  
  
  
  
 *//-------------------------------------------------------------------------------------конструтор -----------------------------------------------* **public** Expense ( String name , Float cost, String categoryOfExpenses) {  
 **this**.**name** = **new** SimpleStringProperty(name); *// добавить имя в класс* **this**.**cost** = **new** SimpleFloatProperty(cost); *// добавить цену в класс* **this**.**categoryOfExpenses** = **new** SimpleStringProperty(categoryOfExpenses); *// добавить category в класс* }  
}

FloatCell

**package** sample;  
  
**import** javafx.collections.ObservableList;  
**import** javafx.scene.control.TableCell;  
  
**public class** FloatCell **extends** TableCell<Expense,Number> { *// наследуем таблицу переменная / число* FloatField **number**; *// наше поле для числа* ObservableList<Expense> **expenseList**; *// список переменных* Expense **expense**; *// наше поле для переменной  
  
 // конструктор* **public** FloatCell(ObservableList<Expense> expenseList )  
 {  
 **this**.**expenseList** = expenseList; *// присваиваем список* }  
  
 @Override  
 **public void** startEdit() { *// начало редактирования* **if**(!isEmpty()){  
 **super**.startEdit();  
 **number**=**new** FloatField(); *// новое поле FloatField* **expense**=getTableView().getSelectionModel().getSelectedItem(); *// выбранное поле* **number**.setText(**expense**.getCost().getValue().toString()); *// вставляем наш текст* setGraphic(**number**); *//отображаем число* }  
 }  
  
 @Override  
 **public void** cancelEdit() { *// отмена редактирования* **super**.cancelEdit();  
 **int** index = **expenseList**.indexOf(**expense**); *// узнаем индекс* **float** newCost = Float.*parseFloat*(**number**.getText()); *// переводим число в Float* Expense newExpense = **new** Expense(**expense**.getName(), newCost, **expense**.**categoryOfExpenses**.getValue()); *// подгружаем таблицу* **expenseList**.set(index, newExpense); *// меняем значение* setGraphic(**null**); *//не отображаем число* }  
  
 @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?* 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?! (」°ロ°)」 тогда верни его обратно! Немедленно!!!* **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**;  
 }  
 }  
) ) ;  
 }  
}

EditWinController.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import java.lang.\****?>  
<?***import java.util.\****?>  
<?***import javafx.scene.\****?>  
<?***import javafx.scene.control.\****?>  
<?***import javafx.scene.layout.\****?>  
  
<?***import javafx.scene.text.Font***?>  
<?***import sample.FloatField***?>*<**BorderPane fx:id="root" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="200.0" prefWidth="300.0" xmlns="http://javafx.com/javafx/8.0.172" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.EditWinController"**>  
 <**center**>  
 <**VBox**>  
 <**HBox**>  
 <**Label BorderPane.alignment ="CENTER" text="Название "**/>  
 <**TextField fx:id="name"** />  
 </**HBox**>  
 <**HBox**>  
 <**Label BorderPane.alignment ="CENTER" text="Цена "**/>  
 <**FloatField fx:id="cost"** />  
 </**HBox**>  
 <**HBox**>  
 <**Label BorderPane.alignment ="CENTER" text="Категория расходов "**/>  
 <**ComboBox fx:id="category"**/>  
 </**HBox**>  
 </**VBox**>  
 </**center**>  
</**BorderPane**>

EditWinController

**package** sample;  
  
**import** javafx.beans.property.StringProperty;  
**import** javafx.beans.value.ObservableValue;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.ActionEvent;  
**import** javafx.event.EventHandler;  
**import** javafx.fxml.FXML;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.input.MouseEvent;  
  
**public class** EditWinController {  
 ObservableList<Expense> **expenseList**;  
 **int editingIndex** ;  
 @FXML TextField **name**;  
 @FXML FloatField **cost**;  
 @FXML ComboBox **category**;  
 @FXML Button **ok**;  
 @FXML Button **no**;  
  
 **public void** init (ObservableList<Expense > expenseList, **int** editingIndex ) {  
 **this**.**expenseList** = expenseList;  
 **this**.**editingIndex** = editingIndex;  
 Expense expense = expenseList.get(editingIndex); *// подгрузка старых значений и выборка изменяемых* **name**.setText(expense.getName());  
 **cost**.setText((expense.getCost().getValue()).toString());  
  
 ObservableList<CategoryOfExpenses> categoryList = FXCollections.*observableArrayList*( *// лист с категориями (enum)* CategoryOfExpenses.*values*());  
 **category**.setItems(categoryList); *// записываем наш список внутрь ComboBox* String categoryCode = expense.getСategoryOfExpenses(); *// вытаскиваем код категории (строчка)* CategoryOfExpenses categoryOfExpenses = CategoryOfExpenses.*getByCode*(categoryCode); *// возвращает категорию по коду* **category**.setValue(categoryOfExpenses); *// выбранное значение ComboBox  
  
 // если сделалили действие* **name**.getParent().setOnMouseExited(e -> exit());  
 **cost**.getParent().setOnMouseExited(e -> exit());  
 **category**.getParent().setOnMouseExited(e -> exit());  
 }  
  
 **private void** exit() {  
 CategoryOfExpenses categoryOfExpenses = CategoryOfExpenses.*getByName*(**category**.getValue().toString()); *// возвращает категорию по названию* Expense expense = **new** Expense ( *// запись новых значений* **name**.getText(),  
 Float.*parseFloat*(**cost**.getText()),  
 categoryOfExpenses.getCode());  
 **expenseList**.set(**editingIndex**, expense) ;  
 }  
}

CategoryOfExpenses

**package** sample;  
  
**import** javafx.scene.control.ComboBox;  
**import** javafx.scene.control.TableCell;  
  
**public enum** CategoryOfExpenses {  
 ***EAT***(**"E"**, **"Еда"**), ***XOZ***(**"X"**, **"Хозяйство"**), ***MAN***(**"M"**, **"Личное"**),;  
  
 **private** String **code**;  
 **private** String **text**;  
  
 **private** CategoryOfExpenses(String code, String text) { *// конструктор* **this**.**code** = code;  
 **this**.**text** = text;  
 }  
  
 **public** String getCode() {  
 **return code**;  
 } *// вернуть код* **public** String getText() {  
 **return text**;  
 } *// вернуть текст* **public static** CategoryOfExpenses getByCode(String categoryCode) { *// возвращает категорию по коду* **for** (CategoryOfExpenses g : CategoryOfExpenses.*values*()) {  
 **if** (g.**code**.equals(categoryCode)) {  
 **return** g;  
 }  
 }  
 **return null**;  
 }  
  
 **public static** CategoryOfExpenses getByName(String nameCode) { *// возвращает категорию по названию* **for** (CategoryOfExpenses g : CategoryOfExpenses.*values*()) {  
 **if** (g.**text**.equals(nameCode)) {  
 **return** g;  
 }  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return this**.**text**;  
 } *// вернуть текст*}