МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе № 4

по дисциплине «Программирование на языках высокого уровня»

Тема: «Табличный компонент»

Выполнил: Юрков Д. А.,

группа МВА-122

Проверил: Самойлова Т. А.

Москва 2023

**Цель работы**

Целью работы является ознакомление со средствами вывода, редактирования и сортировки табличных данных.

Формулировка индивидуального задания представлена ниже:

Отобразить данные класса, созданного в лабораторной работе №1, в виде таблицы. Количество полей в классе должно быть не меньше 5, хотя бы одно из них - типа Boolean. Предусмотреть возможность добавления и сортировки данных в таблице. Добавить 2 произвольных отрисовщика.

Текст класса Room представлен ниже:

public class Room {  
 private final String name;  
 private final int area;  
 public Room(String name, double area) {  
 this.name = name;  
 this.area = (int) area;  
 }  
 public String getName() {  
 return name;  
 }  
 public double getArea() {  
 return area;  
 }  
 @Override  
 public String toString() {  
 return "Помещение: " + name + ", площадь: " + area + " м2";  
 }  
}

Текст класса ResidentialRoom представлен ниже:

public class ResidentialRoom extends Room {

private final int numberOfBathrooms;

private final boolean balcony;

private final String heating;

public ResidentialRoom(String name, double area, int numberOfBathrooms, boolean balcony, String heating) {

super(name, area);

this.numberOfBathrooms = numberOfBathrooms;

this.balcony = balcony;

this.heating = heating;

}

public int getNumberOfBathrooms() {

return numberOfBathrooms;

}

public boolean hasBalcony() {

return balcony;

}

public String getHeatingType() {

return heating;

}

@Override

public String toString() {

String balconyInfo = balcony ? "Есть балкон" : "Без балкона";

return super.toString() + ", количество ванных комнат: " + numberOfBathrooms + ", " + balconyInfo + ", тип отопления: " + heating;

}

}

Текст класса RoomTableApp представлен ниже:

import javafx.application.Application;

import javafx.beans.property.SimpleBooleanProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import org.jetbrains.annotations.NotNull;

import java.util.Comparator;

public class RoomTableApp extends Application {

private final TableView<ResidentialRoom> table = new TableView<>();

private final ObservableList<ResidentialRoom> rooms = FXCollections.observableArrayList();

private boolean ascendingSort = true;

private final TextField nameTextField = new TextField();

private final TextField areaTextField = new TextField();

private final TextField bathroomsTextField = new TextField();

private final CheckBox balconyCheckBox = new CheckBox("Has Balcony");

private final ComboBox<String> heatingComboBox = new ComboBox<>(FXCollections.observableArrayList("Центральное", "Электрическое", "Нет"));

public static void main(String[] args) {

Application.launch(args);

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Room Table");

TableColumn<ResidentialRoom, String> nameColumn = new TableColumn<>("Name");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<ResidentialRoom, Double> areaColumn = new TableColumn<>("Area");

areaColumn.setCellValueFactory(new PropertyValueFactory<>("area"));

TableColumn<ResidentialRoom, Integer> numberOfBathroomsColumn = new TableColumn<>("Number of Bathrooms");

numberOfBathroomsColumn.setCellValueFactory(new PropertyValueFactory<>("numberOfBathrooms"));

TableColumn<ResidentialRoom, Boolean> hasBalconyColumn = getResidentialRoomBooleanTableColumn();

TableColumn<ResidentialRoom, String> heatingColumn = getResidentialRoomStringTableColumn();

table.getColumns().addAll(nameColumn, areaColumn, numberOfBathroomsColumn, hasBalconyColumn, heatingColumn);

table.setItems(rooms);

Button addButton = new Button("Add Room");

addButton.setOnAction(e -> addRoom());

Button addFromArrayButton = new Button("Add Rooms from Array");

addFromArrayButton.setOnAction(e -> addAllRoomsFromArray());

Button sortButton = new Button("Sort by Area");

sortButton.setOnAction(e -> sortRoomsByArea());

ToggleGroup sortToggleGroup = new ToggleGroup();

RadioButton ascendingSortButton = new RadioButton("Ascending");

ascendingSortButton.setToggleGroup(sortToggleGroup);

ascendingSortButton.setSelected(true);

RadioButton descendingSortButton = new RadioButton("Descending");

descendingSortButton.setToggleGroup(sortToggleGroup);

HBox sortBox = new HBox(10);

sortBox.getChildren().addAll(sortButton, new Label("Sort Order:"), ascendingSortButton, descendingSortButton);

HBox inputBox = new HBox(10);

inputBox.getChildren().addAll(

new Label("Name:"),

nameTextField,

new Label("Area:"),

areaTextField,

new Label("Bathrooms:"),

bathroomsTextField,

balconyCheckBox,

new Label("Heating:"),

heatingComboBox,

addButton

);

VBox vbox = new VBox(10);

vbox.getChildren().addAll(table, inputBox, addFromArrayButton, sortBox);

Scene scene = new Scene(vbox);

primaryStage.setScene(scene);

primaryStage.show();

}

@NotNull

private static TableColumn<ResidentialRoom, String> getResidentialRoomStringTableColumn() {

TableColumn<ResidentialRoom, String> heatingColumn = new TableColumn<>("Heating");

heatingColumn.setCellValueFactory(cellData -> {

ResidentialRoom room = cellData.getValue();

String heating = room.getHeatingType();

return new SimpleStringProperty(heating);

});

heatingColumn.setCellFactory(column -> new TableCell<>() {

@Override

protected void updateItem(String item, boolean empty) {

super.updateItem(item, empty);

if (item == null || empty) {

setText(null);

setStyle("");

} else {

setText(item);

if ("Центральное".equals(item)) {

setStyle("-fx-background-color: yellow;");

} else if ("Электрическое".equals(item)) {

setStyle("-fx-background-color: lightblue;");

} else {

setStyle("-fx-background-color: lightcoral;");

}

}

}

});

return heatingColumn;

}

private static TableColumn<ResidentialRoom, Boolean> getResidentialRoomBooleanTableColumn() {

TableColumn<ResidentialRoom, Boolean> hasBalconyColumn = new TableColumn<>("Has Balcony");

hasBalconyColumn.setCellValueFactory(cellData -> {

ResidentialRoom room = cellData.getValue();

boolean hasBalcony = room.hasBalcony();

return new SimpleBooleanProperty(hasBalcony);

});

hasBalconyColumn.setCellFactory(column -> new TableCell<>() {

@Override

protected void updateItem(Boolean item, boolean empty) {

super.updateItem(item, empty);

if (item == null || empty) {

setText(null);

setStyle("");

} else {

ResidentialRoom room = getTableView().getItems().get(getIndex());

setText(item ? "Да" : "Нет");

if (!item) {

setStyle("-fx-background-color: lightcoral;");

} else {

setStyle("-fx-background-color: lightgreen;");

}

}

}

});

return hasBalconyColumn;

}

private void addRoom() {

try {

String name = nameTextField.getText();

double area = Double.parseDouble(areaTextField.getText());

int bathrooms = Integer.parseInt(bathroomsTextField.getText());

boolean hasBalcony = balconyCheckBox.isSelected();

String heating = heatingComboBox.getValue();

ResidentialRoom newRoom = new ResidentialRoom(name, area, bathrooms, hasBalcony, heating);

rooms.add(newRoom);

nameTextField.clear();

areaTextField.clear();

bathroomsTextField.clear();

balconyCheckBox.setSelected(false);

heatingComboBox.setValue(null);

} catch (NumberFormatException e) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Input Error");

alert.setHeaderText("Invalid input");

alert.setContentText("Please enter valid numeric values for Area and Bathrooms.");

alert.showAndWait();

}

}

private void addAllRoomsFromArray() {

rooms.addAll(getRoomsFromArray());

}

private ObservableList<ResidentialRoom> getRoomsFromArray() {

ObservableList<ResidentialRoom> roomsArray = FXCollections.observableArrayList();

roomsArray.add(new ResidentialRoom("Спальня", 20.5, 1, true, "Центральное"));

roomsArray.add(new ResidentialRoom("Гостиная", 30.0, 0, false, "Электрическое"));

roomsArray.add(new ResidentialRoom("Детская", 15.0, 2, true, "Нет"));

roomsArray.add(new ResidentialRoom("Кухня", 12.0, 0, true, "Нет"));

roomsArray.add(new ResidentialRoom("Туалет", 9.0, 1, false, "Электрическое"));

return roomsArray;

}

private void sortRoomsByArea() {

ascendingSort = !ascendingSort;

Comparator<ResidentialRoom> comparator = ascendingSort

? Comparator.comparingDouble(ResidentialRoom::getArea)

: (r1, r2) -> Double.compare(r2.getArea(), r1.getArea());

rooms.sort(comparator);

}

}

Результат выполнения программы представлен на рис. 1.

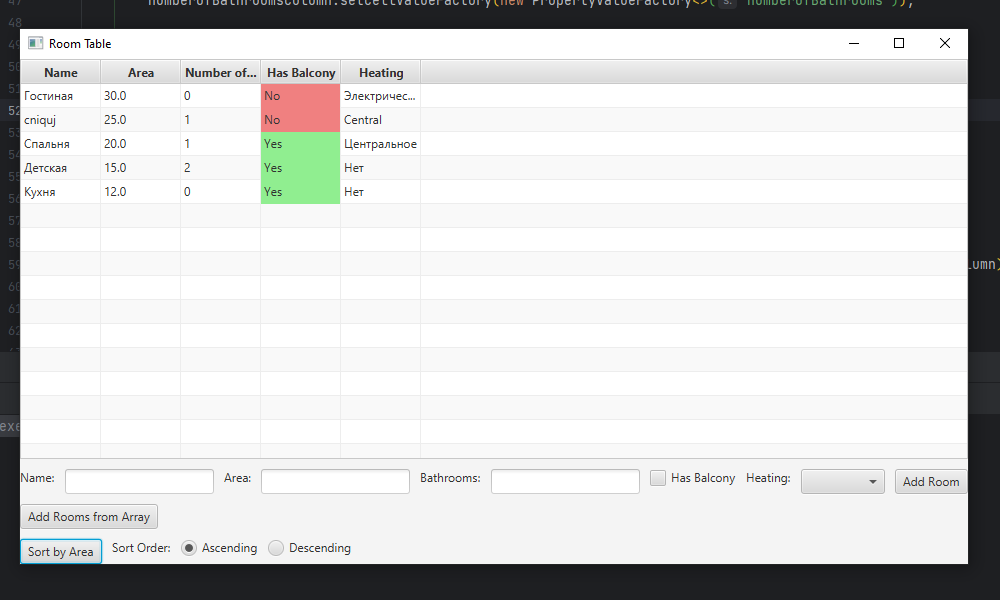


Рис. 1. Результат выполнения.