Звіт

Лабораторна робота 16. Розробка графічного інтерфейсу користувача

Мета роботи:

Придбання навичок використання засобів клієнтських технологій (Client Technologies) платформи Java SE.

ВИМОГИ

Розробити графічний інтерфейс користувача для програми рішення попередньої лабораторної роботи з використанням засобів JavaFX.

1.1. Розробник: Капелька Ярослав Іванович, КІТ-119а, варіант №9.

2. ОПИС ПРОГРАМИ

2.1. Засоби ООП: клас, метод класу, поле класу.

Ієрархія та структура класів: один публічний клас Main, публічний клас Route, у якого є поля: назва маршруту, загальна кількість місць, дні тижня; номер рейсу, назва станції, час прибуття , час відправлення, кількість вільних місць, статус станції, гетери, сетери, конструктор класу та метод виведення даних класу.

2.2. Важливі фрагменти програми:

Main.java

```
public class Main extends Application {

    private TableView<Route> table = new TableView<Route>();
    private final ObservableList<Route> data = FXCollections.observableArrayList();
    final HBox hb = new HBox();

    public static void main(String[] args) {
        launch(args);
    }

    @SuppressWarnings({ "unchecked", "rawtypes" })
       @Override
    public void start(Stage stage) {
        Scene scene = new Scene(new Group());
        stage.setTitle("Капелька Ярослав Лабораторная работа №16");
        stage.setWidth(1440);
        stage.setHeight(540);

        final Label label = new Label("Билетная касса");
        label.setFont(new Font("Jackport College NCV", 20));

        table.setEditable(true);

        TableColumn nameCol = new TableColumn("Название маршрута");
        nameCol.setMinWidth(150);
        nameCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("name_route"));
```

```java
        TableColumn stationCol = new TableColumn("Название станции");
        stationCol.setMinWidth(150);
        stationCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("station_name"));

        TableColumn departureCol = new TableColumn("Время отправления с станции");
        departureCol.setMinWidth(200);
        departureCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("departure_time"));

        TableColumn arrivalCol = new TableColumn("Время прибытия на станцию");
        arrivalCol.setMinWidth(200);
        arrivalCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("arrival_time"));

        TableColumn numberfreeCol = new TableColumn("Количество пустых мест");
        numberfreeCol.setMinWidth(200);
        numberfreeCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("number_of_free_seats"));

        TableColumn statusCol = new TableColumn("Статус станции");
        statusCol.setMinWidth(100);
        statusCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("status_station"));

        TableColumn totalCol = new TableColumn("Общее количество мест");
        totalCol.setMinWidth(200);
        totalCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("total_number_of_seats"));

        TableColumn daysCol = new TableColumn("День недели");
        daysCol.setMinWidth(100);
        daysCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("days"));

        TableColumn flightCol = new TableColumn("Номер рейса");
        flightCol.setMinWidth(100);
        flightCol.setCellValueFactory(
                new PropertyValueFactory<Route, String>("flight_number"));


        table.setItems(data);
        table.getColumns().addAll(nameCol, stationCol, departureCol, arrivalCol,
numberfreeCol, statusCol, totalCol, daysCol, flightCol);

        final TextField addName = new TextField();
        addName.setPromptText("Название маршрута");
        addName.setMaxWidth(nameCol.getPrefWidth());

        final TextField addStation = new TextField();
        addStation.setPromptText("Название станции");
        addStation.setMaxWidth(stationCol.getPrefWidth());

        final TextField addDeparture = new TextField();
        addDeparture.setPromptText("Время отправления со станции");
        addDeparture.setMaxWidth(departureCol.getPrefWidth());

        final TextField addArrival = new TextField();
        addArrival.setPromptText("Время прибытия на станцию");
        addArrival.setMaxWidth(arrivalCol.getPrefWidth());

        final TextField addNumber = new TextField();
        addNumber.setPromptText("Количество пустых мест");
        addNumber.setMaxWidth(numberfreeCol.getPrefWidth());
        final TextField addStatus = new TextField();
        addStatus.setPromptText("Статус станции");
        addStatus.setMaxWidth(statusCol.getPrefWidth());

        final TextField addTotal = new TextField();
        addTotal.setPromptText("Общее количество мест");
        addTotal.setMaxWidth(totalCol.getPrefWidth());

        final TextField addDays = new TextField();
        addDays.setPromptText("День недели");
        addDays.setMaxWidth(daysCol.getPrefWidth());

        final TextField addFlight = new TextField();
        addFlight.setPromptText("Номер рейса");
        addFlight.setMaxWidth(flightCol.getPrefWidth());

        final Button addButton = new Button("Добавить");
        addButton.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent e) {
                try {
                                data.add(new Route(
                                             addName.getText(),
                                             addStation.getText(),
                                             addDeparture.getText(),
                                             addArrival.getText(),
                                             addNumber.getText(),
                                             addStatus.getText(),
                                             addTotal.getText(),
                                             addDays.getText(),
                                             addFlight.getText()));
                        } catch (ParseException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                addName.clear();
                addStation.clear();
                addDeparture.clear();
                addArrival.clear();
                addNumber.clear();
                addStatus.clear();
                addTotal.clear();
                addDays.clear();
                addFlight.clear();
            }
        });

        final TextField serializable = new TextField();
        serializable.setPromptText("Сереализация");
        serializable.setMaxWidth(addFlight.getPrefWidth());

        final Button serbtn = new Button("Сохранить");
        serbtn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent e) {
             FileOutputStream outputStream;
                        try {
                                outputStream = new
FileOutputStream(serializable.getText());
                                serializable.clear();
                    ObjectOutputStream objectOutputStream = new
ObjectOutputStream(outputStream);
                    for (var value : data)
                        objectOutputStream.writeObject(value);
                    objectOutputStream.close();
            } catch (FileNotFoundException e1) {
             // TODO Auto-generated catch block
             e1.printStackTrace();
            } catch (IOException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                    }
            }
        });
        final TextField deserialize = new TextField();
        deserialize.setPromptText("Десериализация");
        deserialize.setMaxWidth(addFlight.getPrefWidth());

        final Button desbtn = new Button("Скачать");
        desbtn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent e) {



             FileInputStream inStream = null;
                        try {
                              inStream = new FileInputStream(deserialize.getText());
                        } catch (FileNotFoundException e1) {
                              // TODO Auto-generated catch block
                              e1.printStackTrace();
                        }
                        deserialize.clear();
                  ObjectInputStream objectInStream = null;
                        try {
                              objectInStream = new ObjectInputStream(inStream);
                        } catch (IOException e1) {
                              // TODO Auto-generated catch block
                              e1.printStackTrace();
                        }
                  while (true) {
                        try {

                                    data.add((Route)objectInStream.readObject());

                              }
                        catch (EOFException e1) {
                                    try {
                                          objectInStream.close();
                                          return;
                                    } catch (IOException e2) {
                                          // TODO Auto-generated catch block
                                          e2.printStackTrace();
                                    }
                              }     catch (ClassNotFoundException | IOException e1)
{

                              }

                  }

            }
        });

        hb.getChildren().addAll(addName, addStation, addDeparture, addArrival,
addNumber, addStatus, addTotal, addDays, addFlight, addButton, serializable, serbtn,
deserialize, desbtn);
        hb.setSpacing(3);

        final VBox vbox = new VBox();
        vbox.setSpacing(5);
        vbox.setPadding(new Insets(10, 0, 0, 10));
        vbox.getChildren().addAll(label, table, hb);

        ((Group) scene.getRoot()).getChildren().addAll(vbox);

        stage.setScene(scene);
        stage.show();
    }
}
```

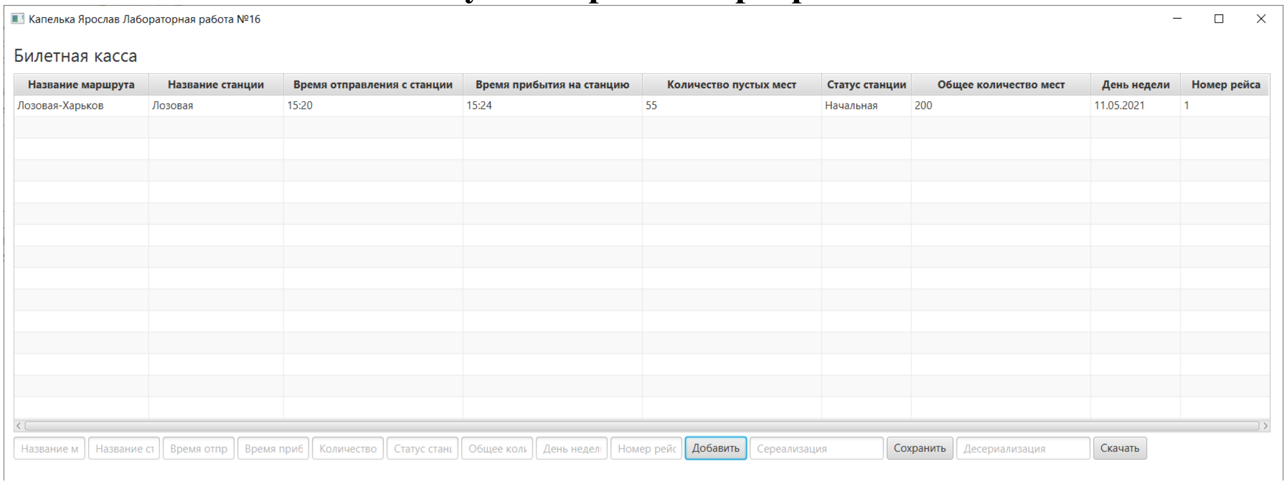
Route.java

```
public class Route implements Serializable
{
	private static final long serialVersionUID = 1L;
	private String name_route;
	private String station_name;
	private String departure_time;
	private String arrival_time;
	private String number_of_free_seats;
	private String status_station;
	private String total_number_of_seats;
	private Calendar days_of_the_week;
	private String flight_number;
	@SuppressWarnings("unused")
	private String daysoftheweek;

	public void setName_route(String name_route)
	{
		String pattern = "^\\b[А-Я][а-я]{1,}[-]\\b[А-Я][а-я]{1,}$";
		Pattern r = Pattern.compile(pattern);
		Matcher m = r.matcher(name_route);
		if(!m.find())
			throw new IllegalArgumentException();;
		this.name_route = name_route;
	}
	public String getName_route()
	{
		return name_route;
	}
	public String getStation_name()
	{
		return station_name;
	}
	public void setStation_name(String station_name)
	{
		String pattern = "^\\b[А-Я][а-я]{1,}$";
		Pattern r = Pattern.compile(pattern);
		Matcher m = r.matcher(station_name);
		if(!m.find())
			throw new IllegalArgumentException();;
		this.station_name = station_name;
	}
	public String getDeparture_time()
	{
		return departure_time;
	}
	public void setDeparture_time(String departure_time)
	{
		String pattern = "^(([0,1][0-9])|(2[0-3])):[0-5][0-9]$";
		Pattern r = Pattern.compile(pattern);
		Matcher m = r.matcher(departure_time);
		if(!m.find())
			throw new IllegalArgumentException();;
		this.departure_time = departure_time;
	}
	public String getArrival_time()
```

```
{
        return arrival_time;
}
public void setArrival_time(String arrival_time)
{
        String pattern = "^(([0,1][0-9])|(2[0-3])):[0-5][0-9]$";
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(arrival_time);
        if(!m.find())
                throw new IllegalArgumentException();;
        this.arrival_time = arrival_time;
}
public String getNumber_of_free_seats()
{
        return number_of_free_seats;
}
public void setNumber_of_free_seats(String number_of_free_seats)
{
        String pattern = "^[0-9]{1,2}$";
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(number_of_free_seats);
        if(!m.find())
                throw new IllegalArgumentException();;
        this.number_of_free_seats = number_of_free_seats;
}
public String getStatus_station()
{
        return status_station;
}
public void setStatus_station(String status_station)
{
        String pattern = "^\\b[А-Я][а-я]{1,}$";
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(status_station);
        if(!m.find())
                throw new IllegalArgumentException();;
        this.status_station = status_station;
}
public void setTotal_number_of_seats(String total_number_of_seats)
{
        String pattern = "^[0-9]{3}$";
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(total_number_of_seats);
        if(!m.find())
                throw new IllegalArgumentException();;
        this.total_number_of_seats = total_number_of_seats;
}
public String getTotal_number_of_seats()
{
        return total_number_of_seats;
}
public void setDays_of_the_week(String days_of_the_week) throws ParseException
{
        String pattern = "^[0-9]{1,2}[.][0-9]{1,2}[.][0-2][0-9]{3}$";
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(days_of_the_week);
        if(!m.find())
                throw new IllegalArgumentException();;
        SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy", Locale.ENGLISH);
        Calendar cal1 = new GregorianCalendar();
        cal1.setTime(sdf.parse(days_of_the_week));
        this.days_of_the_week = cal1;

}
public void setDays_of_the_week(Calendar days_of_the_week)
{
```

```
            this.days_of_the_week = days_of_the_week;
        }
        public Calendar getDays_of_the_week()
        {
            return days_of_the_week;
        }
        public String getFlight_number()
        {
            return flight_number;
        }
        public void setFlight_number(String flight_number)
        {
            String pattern = "^[0-9]{1}$";
            Pattern r = Pattern.compile(pattern);
            Matcher m = r.matcher(flight_number);
            if(!m.find())
                throw new IllegalArgumentException();;
            this.flight_number = flight_number;
        }
        public void setDays(String daysoftheweek) throws ParseException{
            setDays_of_the_week(daysoftheweek);
        }
        public String getDays() {
            SimpleDateFormat formatForDateNow = new SimpleDateFormat("dd.MM.yyyy");
            String str = formatForDateNow.format(days_of_the_week.getTime());

            return str;
        }
        public Route()
        {
            super();
        }

        @Override
        public String toString()
        {
            SimpleDateFormat sdf1 = new SimpleDateFormat("dd.MM.yyyy",
Locale.ENGLISH);
            return new String("\nИмя маршрута: " + this.getName_route()+"\nИмя
станции: "+ this.getStation_name() + "\nВремя прибытия на станцию: " +
this.getArrival_time()+ "\nВремя отправления со станции: " +
this.getDeparture_time()+"\nКоличество пустых мест: "+
this.getNumber_of_free_seats()+"\nСтатус станции: "+ this.getStatus_station()+"\nОбщее
количество мест: "+ this.getTotal_number_of_seats()+"\nДень недели: "+
sdf1.format(this.getDays_of_the_week().getTime())+"\nНомер рейсу: "+
this.getFlight_number());
        }
        Route(String name_route, String total_number_of_seats, Calendar days, String
flight_number) {
            this.setName_route(name_route);
            this.setTotal_number_of_seats(total_number_of_seats);
            this.setDays_of_the_week(days);
            this.setFlight_number(flight_number);
        }

        Route(String name_route, String total_number_of_seats, String days, String
flight_number)
                throws ParseException {
            this.setName_route(name_route);
            this.setTotal_number_of_seats(total_number_of_seats);
            this.setDays_of_the_week(days);
            this.setFlight_number(flight_number);
        }

        Route(String name_route, String station_name, String departure_time, String
arrival_time,
                    String number_of_free_seats, String status_station, String
total_number_of_seats, Calendar days,
                    String flight_number) {
            this.setName_route(name_route);
            this.setStation_name(station_name);
            this.setDeparture_time(departure_time);
            this.setArrival_time(arrival_time);
            this.setNumber_of_free_seats(number_of_free_seats);
            this.setStatus_station(status_station);
            this.setTotal_number_of_seats(total_number_of_seats);
            this.setDays_of_the_week(days);
            this.setFlight_number(flight_number);
      }

      Route(String name_route, String station_name, String departure_time, String
arrival_time,
                    String number_of_free_seats, String status_station, String
total_number_of_seats, String days,
                    String flight_number) throws ParseException {
            this.setName_route(name_route);
            this.setStation_name(station_name);
            this.setDeparture_time(departure_time);
            this.setArrival_time(arrival_time);
            this.setNumber_of_free_seats(number_of_free_seats);
            this.setStatus_station(status_station);
            this.setTotal_number_of_seats(total_number_of_seats);
            this.setDays_of_the_week(days);
            this.setFlight_number(flight_number);
      }

}
```

Результат роботи програми



Название маршрута	Название станции	Время отправления с станции	Время прибытия на станцию	Количество пустых мест	Статус станции	Общее количество мест	День недели	Номер рейса
Лозовая-Харьков	Лозовая	15:20	15:24	55	Начальная	200	11.05.2021	1

Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з JavaFX.

Програма протестована, виконується без помилок.