# Звіт

# Лабораторна робота 16. [Розробка графічного інтерфейсу користувача](https://oop-khpi.gitlab.io/#tasks)

# Мета роботи:

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

# ВИМОГИ

Розробити графічний інтерфейс користувача для програми рішення [попередньої лабораторної роботи](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task15/) з використанням засобів [JavaFX](https://en.wikipedia.org/wiki/JavaFX).

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

# 2. ОПИС ПРОГРАМИ

* 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.

# 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"));

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);

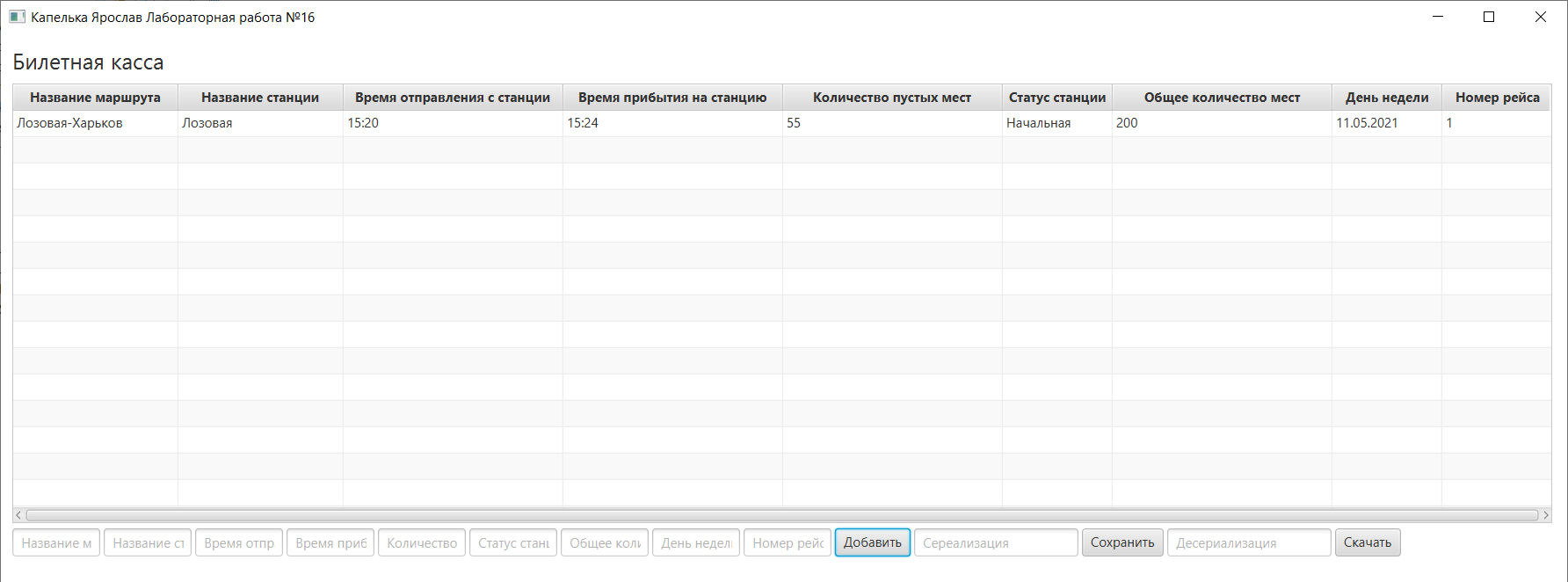
**this**.setDays\_of\_the\_week(days);

**this**.setFlight\_number(flight\_number);

}

}

# Результат роботи програми

****

**Висновки**

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

Програма протестована, виконується без помилок.