# Звіт

**Лабораторна робота 15. Колекції в Java**

# Мета роботи:

* Ознайомлення з бібліотекою колекцій Java SE.
* Використання колекцій для розміщення об'єктів розроблених класів.

# ВИМОГИ

1. Розробити консольну програму для реалізації завдання обробки даних згідно прикладної області.
2. Для розміщення та обробки даних використовувати контейнери (колекції) і алгоритми з Java Collections Framework.
3. Забезпечити обробку колекції об'єктів: додавання, видалення, пошук, сортування згідно розділу Прикладні задачі л.р. №10.
4. Передбачити можливість довготривалого зберігання даних: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.
5. Продемонструвати розроблену функціональність в діалоговому та автоматичному режимах за результатом обробки параметрів командного рядка.

**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. Клас MyThread, який виконує роль потока.

# Важливі фрагменти програми:

**Main15.java**

**public** **class** Main15 {

**public** **static** **void** main(String[] args) {

**for** (**var** str : args) {

**if** (str.equals("-auto")) {

Helper.*Auto*();

**return**;

}

}

Helper.*Menu*();

}

}

**Helper.java**

**public** **class** Helper {

**static** **void** Auto() {

ArrayList<Route> collection = **new** ArrayList<Route>();

collection.add(**new** Route("Лозовая-Харьков", "Лозовая", "15:20", "15:24", "55", "Начальная", "150",

**new** GregorianCalendar(2021, 11, 7), "1"));

collection.add(**new** Route("Харьков-Изюм", "Маяк", "14:52", "14:53", "61", "Промежуточная", "200",

**new** GregorianCalendar(2021, 11, 8), "2"));

collection.add(**new** Route("Красноград-Харьков", "Харьков", "19:34",

"Времени отправления нету, так как это конечная станция.", "78", "Конечная", "170",

**new** GregorianCalendar(2021, 1, 20), "3"));

collection.add(**new** Route("Львов-Мариуполь", "Льзов", "01:07", "01:12", "55", "Начальная", "230",

**new** GregorianCalendar(2021, 2, 15), "4"));

collection.add(**new** Route("Ужгород-Киев", "Мукачево", "17:59", "18:05", "69", "Промежуточная", "200",

**new** GregorianCalendar(2021, 2, 25), "5"));

**for** (**var** value : collection) {

System.***out***.println(value);

}

System.***out***.println("\n\nПосле сортировки\n\n");

*sort*(collection, ESort.***TOTALNUMBEROFSEATS***);

**for** (**var** value : collection) {

System.***out***.println(value);

}

collection.remove(0);

collection.remove(0);

collection.remove(0);

collection.remove(0);

collection.remove(0);

System.***out***.println("\n\nПосле десериализации\n\n");

**try** {

*downloadSer*("s", collection);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**for** (**var** value : collection) {

System.***out***.println(value);

}

}

**static** **void** Menu() {

ArrayList<Route> collection = **new** ArrayList<Route>();

Scanner scan = **new** Scanner(System.***in***);

String name = **new** String();

String station = **new** String();

String departure = **new** String();

String arrival = **new** String();

String number = **new** String();

String status = **new** String();

String total\_number = **new** String();

String days = **new** String();

String flight = **new** String();

**boolean** prz = **true**;

**int** rez = 0;

**while** (prz) {

System.***out***.println(

"\n1.Добавить элемент\n2.Удалить элемент\n3.Вывод всех элементов.\n4.Сортировать\n5.Поиск\n6.Сериализция\n7.Десериализация\n0.Выход\nВаш выбор:");

**switch** (scan.nextInt()) {

**case** 1:

scan.nextLine();

System.***out***.println("Имя маршрута: ");

name = scan.nextLine();

System.***out***.println("Имя станции: ");

station = scan.nextLine();

System.***out***.println("Время прибытия на станцию: ");

departure = scan.nextLine();

System.***out***.println("Время отправления со станции: ");

arrival = scan.nextLine();

System.***out***.println("Количество пустых мест: ");

number = scan.nextLine();

System.***out***.println("Статус станции: ");

status = scan.nextLine();

System.***out***.println("Общее количество мест: ");

total\_number = scan.nextLine();

System.***out***.println("День недели в формате День.Месяц.Год: ");

days = scan.nextLine();

System.***out***.println("Номер рейсу: ");

flight = scan.nextLine();

**try** {

collection.add(

**new** Route(name, station, departure, arrival, number, status, total\_number, days, flight));

} **catch** (ParseException e) {

System.***out***.println("Не удалось коректно считать");

**continue**;

}

**break**;

**case** 2:

System.***out***.println("Введите id для удаления: ");

collection.remove(scan.nextInt());

**break**;

**case** 3:

System.***out***.println();

**for** (**var** value : collection)

System.***out***.println(value);

**break**;

**case** 4:

System.***out***.println(

"Как сортировать?\n1.По количеству пустых мест.\n2.По дню недели.\n3.По номеру маршрута.\nВаш выбор: ");

**switch** (scan.nextInt()) {

**case** 1:

*sort*(collection, ESort.***TOTALNUMBEROFSEATS***);

**break**;

**case** 2:

*sort*(collection, ESort.***DAYOFTHEWEEK***);

**break**;

**case** 3:

*sort*(collection, ESort.***FLIGHTNUMBER***);

**break**;

**default**:

**break**;

}

**break**;

**case** 5:

System.***out***.println(

"По какому полю произвести поиск\n1.По имени маршрута\n2.По статусу станции\n3.По дню недели\nВаш выбор: ");

**switch** (scan.nextInt()) {

**case** 1:

System.***out***.println("Введите имя маршрута: ");

scan.nextLine();

rez = *find*(collection, EFind.***NAME***, scan.nextLine());

**break**;

**case** 2:

System.***out***.println("Введите статус станции: ");

scan.nextLine();

rez = *find*(collection, EFind.***STATUSSTATION***, scan.nextLine());

**break**;

**case** 3:

System.***out***.println("Введите день недели(d.m.y): ");

scan.nextLine();

rez = *find*(collection, EFind.***DAYOFTHEWEEK***, scan.nextLine());

**break**;

**default**:

**break**;

}

**if** (rez == -1)

System.***out***.println("Элемент не найден");

**else**

System.***out***.println("Id элемента: " + rez);

**break**;

**case** 6:

System.***out***.println("Как сериализовать?\n1.Стандартный способ\n2.XML\nВаш выбор: ");

**try** {

**switch** (scan.nextInt()) {

**case** 1:

System.***out***.println("Введите имя файла: ");

scan.nextLine();

*saveSer*(scan.nextLine(), collection);

**break**;

**case** 2:

scan.nextLine();

System.***out***.println("Введите имя файла: ");

*saveXml*(scan.nextLine(), collection);

**break**;

**default**:

**break**;

}

} **catch** (IOException e) {

System.***out***.println("Не удалось найти файл");

**continue**;

}

**break**;

**case** 7:

System.***out***.println("Как Десериализовать?\n1.Стандартный способ\n2.XML\nВаш выбор: ");

**try** {

**switch** (scan.nextInt()) {

**case** 1:

scan.nextLine();

System.***out***.println("Введите имя файла: ");

*downloadSer*(scan.nextLine(), collection);

**break**;

**case** 2:

scan.nextLine();

System.***out***.println("Введите имя файла: ");

*downloadXml*(scan.nextLine(), collection);

**break**;

**default**:

**break**;

}

} **catch** (IOException e) {

System.***out***.println("Не удалось найти файл");

**continue**;

} **catch** (ClassNotFoundException e) {

System.***out***.println("Не удалось преобразовать тип");

**continue**;

}

**break**;

**case** 0:

prz = **false**;

**break**;

**default**:

**continue**;

}

}

scan.close();

}

**static** <T **extends** Route> **void** sort(ArrayList<T> collection, ESort choose) {

**boolean** przEnd = **true**;

T temp = **null**;

**while** (przEnd) {

przEnd = **false**;

**for** (**int** i = 0; i < collection.size() - 1; i++) {

**switch** (choose) {

**case** ***TOTALNUMBEROFSEATS***:

**if** (collection.get(i).getTotal\_number\_of\_seats()

.compareTo(collection.get(i + 1).getTotal\_number\_of\_seats()) > 0) {

temp = collection.get(i);

collection.remove(i);

collection.add(i + 1, temp);

przEnd = **true**;

}

**break**;

**case** ***DAYOFTHEWEEK***:

**if** (collection.get(i).getDays\_of\_the\_week()

.compareTo(collection.get(i + 1).getDays\_of\_the\_week()) > 0) {

temp = collection.get(i);

collection.remove(i);

collection.add(i + 1, temp);

przEnd = **true**;

}

**break**;

**case** ***FLIGHTNUMBER***:

**if** (collection.get(i).getFlight\_number().compareTo(collection.get(i + 1).getFlight\_number()) > 0) {

temp = collection.get(i);

collection.remove(i);

collection.add(i + 1, temp);

przEnd = **true**;

}

**break**;

**default**:

**break**;

}

}

}

}

**static** <T **extends** Route> **int** find(ArrayList<T> collection, EFind choose, String str) {

Route temp = **null**;

**try** {

**switch** (choose) {

**case** ***NAME***:

temp = **new** Route(str, "Темп", "11:11", "11:11", "11", "Темп", "111", "22.22.2001", "1");

**break**;

**case** ***STATUSSTATION***:

temp = **new** Route("Темп-Темп", "Темп", "11:11", "11:11", "11", str, "111", "22.22.2001", "1");

**break**;

**case** ***DAYOFTHEWEEK***:

temp = **new** Route("Темп-Темп", "Темп", "11:11", "11:11", "11", "Темп", "111", str, "1");

**break**;

**default**:

**break**;

}

} **catch** (ParseException | IllegalArgumentException e) {

System.***out***.println("Неверно переданы данные");

**return** -1;

}

**for** (**int** i = 0; i < collection.size(); i++) {

**switch** (choose) {

**case** ***NAME***:

**if** (collection.get(i).getNameRoute().equals(temp.getNameRoute()))

**return** i;

**break**;

**case** ***STATUSSTATION***:

**if** (collection.get(i).getStatus\_station().equals(temp.getStatus\_station()))

**return** i;

**break**;

**case** ***DAYOFTHEWEEK***:

**if** (collection.get(i).getDays\_of\_the\_week().equals(temp.getDays\_of\_the\_week()))

**return** i;

**break**;

**default**:

**break**;

}

}

**return** -1;

}

**static** **public** **void** saveXml(String fileName, ArrayList<Route> collection) **throws** FileNotFoundException {

XMLEncoder encoder = **new** XMLEncoder(**new** BufferedOutputStream(**new** FileOutputStream(fileName)));

**for** (**var** value : collection)

encoder.writeObject(value);

encoder.close();

System.***out***.println("Сереализация прошла успешно.");

}

**static** **public** **void** downloadXml(String fileName, ArrayList<Route> collection) **throws** FileNotFoundException {

XMLDecoder d = **new** XMLDecoder(**new** BufferedInputStream(**new** FileInputStream(fileName)));

**try** {

**while** (**true**) {

collection.add((Route) d.readObject());

}

} **catch** (ArrayIndexOutOfBoundsException e) {

d.close();

System.***out***.println("Десереализация прошла успешно\n");

} **catch** (Exception e) {

**throw** e;

}

}

**static** **public** **void** saveSer(String fileName, ArrayList<Route> collection) **throws** IOException {

FileOutputStream outputStream = **new** FileOutputStream(fileName);

ObjectOutputStream objectOutputStream = **new** ObjectOutputStream(outputStream);

**for** (**var** value : collection)

objectOutputStream.writeObject(value);

objectOutputStream.close();

}

**static** **public** **void** downloadSer(String fileName, ArrayList<Route> collection)

**throws** IOException, ClassNotFoundException {

FileInputStream inStream = **new** FileInputStream(fileName);

ObjectInputStream objectInStream = **new** ObjectInputStream(inStream);

**try** {

**while** (**true**) {

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

}

} **catch** (EOFException e) {

objectInStream.close();

}

}

}

**enum** EFind {

***NAME***, ***DAYOFTHEWEEK***, ***STATUSSTATION***

}

**enum** ESort {

***TOTALNUMBEROFSEATS***, ***DAYOFTHEWEEK***, ***FLIGHTNUMBER***

}

**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;

**public** **void** setNameRoute(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 getNameRoute()

{

**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** Route()

{

**super**();

}

@Override

**public** String toString()

{

SimpleDateFormat sdf1 = **new** SimpleDateFormat("dd.MM.yyyy", Locale.***ENGLISH***);

**return** **new** String("\nИмя маршрута: " + **this**.getNameRoute()+"\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, String total\_number,Calendar days,String flight)

{

**this**.setNameRoute(name);

setTotal\_number\_of\_seats(total\_number);

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

setFlight\_number(flight);

}

Route(String name, String total\_number,String days,String flight) **throws** ParseException

{

**this**.setNameRoute(name);

setTotal\_number\_of\_seats(total\_number);

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

setFlight\_number(flight);

}

Route(String name, String name1, String time,String time1, String number, String status, String total\_number, Calendar days, String flight)

{

**this**.setNameRoute(name);

**this**.setStation\_name(name1);

**this**.setArrival\_time(time);

**this**.setDeparture\_time(time1);

**this**.setNumber\_of\_free\_seats(number);

**this**.setStatus\_station(status);

setTotal\_number\_of\_seats(total\_number);

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

setFlight\_number(flight);

}

Route(String name, String name1, String time,String time1, String number, String status, String total\_number, String days,String flight) **throws** ParseException

{

**this**.setNameRoute(name);

**this**.setStation\_name(name1);

**this**.setArrival\_time(time);

**this**.setDeparture\_time(time1);

**this**.setNumber\_of\_free\_seats(number);

**this**.setStatus\_station(status);

setTotal\_number\_of\_seats(total\_number);

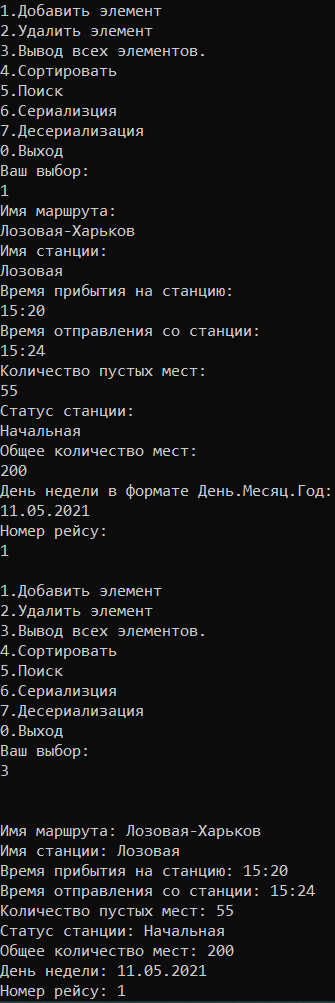
**this**.setDays\_of\_the\_week(days);

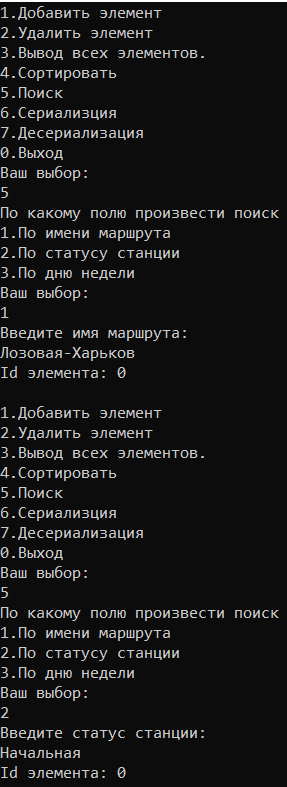
setFlight\_number(flight);

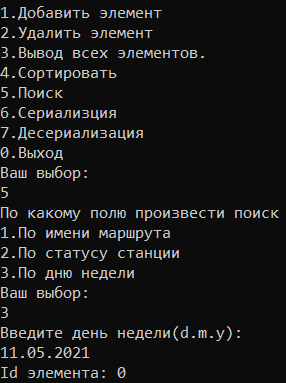
}

}

# Результат роботи програми







**Висновки**

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

Програма протестована, виконується без помилок.