**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

**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. Клас MyThread, який виконує роль потока.

**2.2. 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", "Львов", "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[A-Я][a-я]{1,}[-]\\b[A-Я][a-я]{1,}$";
		Pattern r = Pattern.compile(pattern);
		Matcher m = r.matcher(name_route);
		if(!m.find())
			throw new IllegalArgumentException();;
		this.name_route = name_route;
	}
	public String getNameRoute()
	{
		return name_route;
	}
	public String getStation_name()
	{
		return station_name;
	}
	public void setStation_name(String station_name)
	{
		String pattern = "^\\b[A-Я][a-я]{1,}$";
		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}$";
	Pattern r = Pattern.compile(pattern);
	Matcher m = r.matcher(number_of_free_seats);
	if(!m.find())
		throw new IllegalArgumentException();;
	this.number_of_free_seats = number_of_free_seats;
}
public String getStatus_station()
{
	return status_station;
}
public void setStatus_station(String status_station)
{
	String pattern = "^\\b[A-Я][a-я]{1,}$";
	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

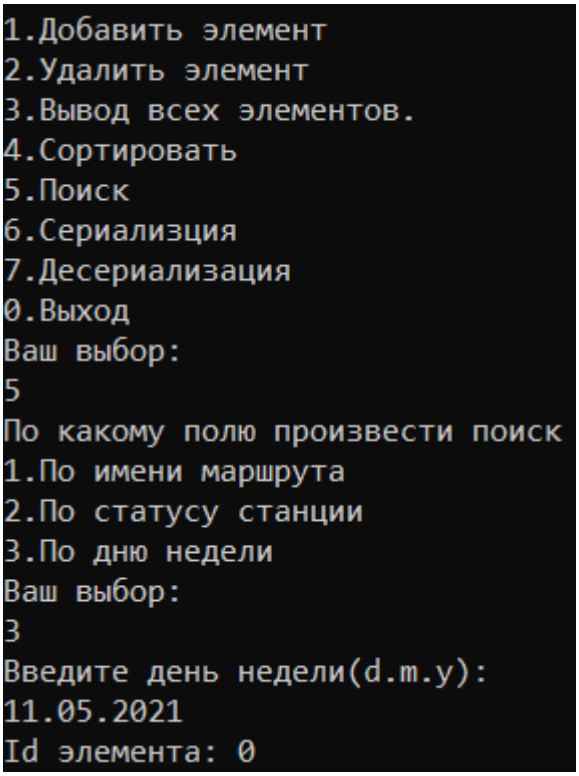
```
1.Добавить элемент
2.Удалить элемент
3.Вывод всех элементов.
4.Сортировать
5.Поиск
6.Сериализция
7.Десериализация
0.Выход
Ваш выбор:
1
Имя маршрута:
Лозовая-Харьков
Имя станции:
Лозовая
Время прибытия на станцию:
15:20
Время отправления со станции:
15:24
Количество пустых мест:
55
Статус станции:
Начальная
Общее количество мест:
200
День недели в формате День.Месяц.Год:
11.05.2021
Номер рейсу:
1

1.Добавить элемент
2.Удалить элемент
3.Вывод всех элементов.
4.Сортировать
5.Поиск
6.Сериализция
7.Десериализация
0.Выход
Ваш выбор:
3


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

```
1.Добавить элемент
2.Удалить элемент
3.Вывод всех элементов.
4.Сортировать
5.Поиск
6.Сериализция
7.Десериализация
0.Выход
Ваш выбор:
5
По какому полю произвести поиск
1.По имени маршрута
2.По статусу станции
3.По дню недели
Ваш выбор:
1
Введите имя маршрута:
Лозовая-Харьков
Id элемента: 0

1.Добавить элемент
2.Удалить элемент
3.Вывод всех элементов.
4.Сортировать
5.Поиск
6.Сериализция
7.Десериализация
0.Выход
Ваш выбор:
5
По какому полю произвести поиск
1.По имени маршрута
2.По статусу станции
3.По дню недели
Ваш выбор:
2
Введите статус станции:
Начальная
Id элемента: 0
```




**Висновки**

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

Програма протестована, виконується без помилок.