Звіт

Лабораторна работа 12. Регулярні вирази. Обробка тексту

Мета роботи: Ознайомлення з принципами використання регулярних виразів для обробки тексту.

ВИМОГИ

1. Використовуючи програми рішень попередніх задач, продемонструвати ефективне (оптимальне) використання регулярних виразів при вирішенні прикладної задачі.
2. Передбачити можливість незначної зміни умов пошуку.
3. Продемонструвати розроблену функціональність в діалоговому та автоматичному режимах.

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

2. ОПИС ПРОГРАМИ

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

2.2. 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. Також є клас Data, який виконує роль покажчика на елемент і клас MyCollection, який містить покажчик на головний елемент та методи обробки масиву елементів.

2.3. Важливі фрагменти програми:

Main12.java

```
public class Main12
{
	public static void main(String[] args) throws ParseException
	{

		for (var str : args)
		{
			if (str.equals("-auto"))
			{
				try
				{
					Helper.Auto();
				} catch (IOException e)
```

```
                        {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                        return;
                }
        }
        try
        {
                Helper.Menu();
        } catch (IOException e)
        {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
    }
}
```

Helper.java

```
public class Helper
{
    static <T extends Route> void sort(MyCollection<T> collection, ESort choose)
    {
        boolean przEnd = true;
        while (przEnd)
        {
            przEnd = false;
            for (int i = 0; i < collection.getSize() - 1; i++)
            {
                switch (choose)
                {
                case TOTALNUMBEROFSEATS:
                    if
(collection.get(i).getTotal_number_of_seats().compareTo(collection.get(i +
1).getTotal_number_of_seats()) > 0)
                    {
                        collection.swap(i, i + 1);
                        przEnd = true;
                    }
                    break;
                case DAYOFTHEWEEK:
                    if
(collection.get(i).getDays_of_the_week().compareTo(collection.get(i +
1).getDays_of_the_week()) > 0)
                    {
                        collection.swap(i, i + 1);
                        przEnd = true;
                    }
                    break;
                case FLIGHTNUMBER:
                    if
(collection.get(i).getFlight_number().compareTo(collection.get(i +
1).getFlight_number()) > 0)
                    {
                        collection.swap(i, i + 1);
                        przEnd = true;
                    }
                    break;
                default:
                    break;
                }
            }
```

```
		}
	}

	static void TransitRoutes(MyCollection<Route> collection)
	{
		String patternStation = "(Харьков)";
		String patternStatus = "(Промежуточная)";
		String patternDepartureTime = "((1[6-9])|(2[0-3])):[0-5][0-9]";
		String patternArrivalTime = "((1[6-9])|(2[0-3])):[0-5][0-9]";
		Pattern rStation = Pattern.compile(patternStation);
		Pattern rStatus = Pattern.compile(patternStatus);
		Pattern rDepartureTime = Pattern.compile(patternDepartureTime);
		Pattern rArrivalTime = Pattern.compile(patternArrivalTime);

		for(var value : collection)
		{
			if((rStation.matcher(value.getStation_name())).find() &&
(rStatus.matcher(value.getStatus_station())).find() &&
(rDepartureTime.matcher(value.getDeparture_time())).find() &&
(rArrivalTime.matcher(value.getArrival_time())).find())
			{
				System.out.println("Маршрут " + value.getNameRoute() + "
есть транзитным.");
			}
			if(!(rStation.matcher(value.getStation_name())).find() &&
!(rStatus.matcher(value.getStatus_station())).find() &&
!(rDepartureTime.matcher(value.getDeparture_time())).find() &&
!(rArrivalTime.matcher(value.getArrival_time())).find())
			{
				System.out.println("Маршрут " + value.getNameRoute() + "
есть не транзитным.");
			}
			if((rStation.matcher(value.getStation_name())).find() &&
(rStatus.matcher(value.getStatus_station())).find() &&
!(rDepartureTime.matcher(value.getDeparture_time())).find() &&
!(rArrivalTime.matcher(value.getArrival_time())).find())
			{
				System.out.println("Маршрут " + value.getNameRoute() + "
есть не транзитным.");
			}
			if(!(rStation.matcher(value.getStation_name())).find() &&
!(rStatus.matcher(value.getStatus_station())).find() &&
(rDepartureTime.matcher(value.getDeparture_time())).find() &&
(rArrivalTime.matcher(value.getArrival_time())).find())
			{
				System.out.println("Маршрут " + value.getNameRoute() + "
есть не транзитным.");
			}
			if((rStation.matcher(value.getStation_name())).find() &&
!(rStatus.matcher(value.getStatus_station())).find() &&
(rDepartureTime.matcher(value.getDeparture_time())).find() &&
(rArrivalTime.matcher(value.getArrival_time())).find())
			{
				System.out.println("Маршрут " + value.getNameRoute() + "
есть не транзитным.");
			}
			if((rStation.matcher(value.getStation_name())).find() &&
!(rStatus.matcher(value.getStatus_station())).find() &&
!(rDepartureTime.matcher(value.getDeparture_time())).find() &&
!(rArrivalTime.matcher(value.getArrival_time())).find())
			{
```

```
                        System.out.println("Маршрут " + value.getNameRoute() + "
есть не транзитным.");
                    }
            }
      }

      enum ESort
      {
            TOTALNUMBEROFSEATS, DAYOFTHEWEEK, FLIGHTNUMBER
      }

      static void saveToFile(String filename, String str) throws IOException
      {

            FileWriter file = new FileWriter(filename);
            file.write(str);
            file.close();
      }

      static String readFromFile(String filename) throws IOException
      {
            FileReader file = new FileReader(filename);
            String str = new String();

            int c = 0;
            while ((c = file.read()) != -1)
            {
                  str += new String(new char[] { (char) c });
            }
            file.close();
            return str;
      }

      static MyCollection<Route> parsingRoute(String str)
      {
            MyCollection<Route> array = new MyCollection<Route>();
            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();
            while (str.indexOf("NameRoute: ") >= 0 && str.length() > 0)
            {
                  name = str.substring(str.indexOf("NameRoute: ") + 11,
str.indexOf("Station Name: ") - 1);
                  station = str.substring(str.indexOf("Station Name: ") + 14,
str.indexOf("Departure time: ") - 1);
                  departure = str.substring(str.indexOf("Departure time: ") + 16,
str.indexOf("Arrival time: ") - 1);
                  arrival = str.substring(str.indexOf("Arrival time: ") + 14,
str.indexOf("Number of free seats: ") - 1);
                  number = str.substring(str.indexOf("Number of free seats: ") +
22, str.indexOf("Status station: ") - 1);
                  status = str.substring(str.indexOf("Status station: ") + 16,
str.indexOf("Total number of seats: ") - 1);
                  total_number = str.substring(str.indexOf("Total number of seats:
") + 23, str.indexOf("Days of the week: ") - 1);
```

```
                    days = str.substring(str.indexOf("Days of the week: ") + 18,
str.indexOf("Flight number: ") - 1);
                    flight = new String();
                    for (int i = str.indexOf("Flight number: ") + 15; str.charAt(i)
!= '\n' && i < str.length(); i++)
                    {
                          flight += str.charAt(i);
                    }

                    try
                    {
                          array.add(new Route(name, station, departure, arrival,
number, status, total_number, days, flight));

                    } catch (ParseException e)
                    {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }
                    str = str.substring(str.indexOf(flight) + flight.length() + 1);
              }
              return array;
        }

        static void Auto() throws IOException, ParseException
        {
              MyCollection<Route> collection = new MyCollection<Route>();
              collection.add(new Route("Лозовая-
Харьков","Лозовая","15:20","15:24","55","Начальная","200","11.05.2021","1"));
              collection.add(new Route("Минск-
Запорожье","Харьков","00:19","00:41","68","Промежуточная","150","21.03.2021","2"));
              System.out.println(collection);
              sort(collection, ESort.TOTALNUMBEROFSEATS);
              System.out.println("После сортировки: Общее количество мест");
              System.out.println(collection);
              sort(collection, ESort.DAYOFTHEWEEK);
              System.out.println("После сортировки: День недели");
              System.out.println(collection);
              sort(collection, ESort.FLIGHTNUMBER);
              System.out.println("После сортировки: Номер рейса");
              System.out.println(collection);
        }

        static void Menu() throws IOException
        {
              MyCollection<Route> collection = new MyCollection<Route>();
              Scanner scan = new Scanner(System.in);
              boolean prz = true;
              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();
              while (prz)
              {
                    System.out.println(
```

```
                                    "\n1.Добавить элемент\n2.Удалить
элемент\n3.Сортировать\n4.Вывод всех элементов.\n5.Записать в файл\n6.Считать с
файла\n7.Найти транзитные маршруты\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("Номер элемента (начало с 0): ");
                              collection.delete(scan.nextInt());
                              break;
                        case 3:
                              System.out.println(
                                          "Как сортировать?1.По общему количеству
мест.\n2.По дню недели.\n3.По номеру рейса.\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 4:
```

```
                        System.out.println(collection);
                        break;
                case 5:
                        scan.nextLine();
                        System.out.println("Введите имя файла: ");
                        saveToFile(scan.nextLine(), collection.toString());
                        break;
                case 6:
                        scan.nextLine();
                        System.out.println("Введите имя файла: ");
                        collection = parsingRoute(readFromFile(scan.nextLine()));
                        break;
                case 7:
                        Helper.TransitRoutes(collection);
                        break;
                case 0:
                        prz = false;
                        break;
                default:
                        break;
                }
            }
            scan.close();
        }

}
```

Data.java

```
public class Data<T>
{

        public T obj;

        public Data<T> next;

        public Data<T> prev;

        Data()
        {

        }
        Data(T obj, Data<T> prev, Data<T> next)
        {
                this.obj = obj;
                this.next = next;
                this.prev = prev;
        }
}
```

MyCollection.java

```
public class MyCollection<T> implements Iterable<T>, Serializable
{
        static final long serialVersionUID = 1L;
        private int size;
        private Data<T> start;
        private Data<T> last;

        public void saveSer(String fileName) throws IOException
        {
                FileOutputStream outputStream = new FileOutputStream(fileName);
                ObjectOutputStream objectOutputStream = new
ObjectOutputStream(outputStream);
```

```
		for (T value : this)
			objectOutputStream.writeObject(value);
		objectOutputStream.close();
	}

	@SuppressWarnings("unchecked")
	public void downloadSer(String fileName) throws IOException,
ClassNotFoundException
	{
		FileInputStream inStream = new FileInputStream(fileName);
		ObjectInputStream objectInStream = new ObjectInputStream(inStream);
		try
		{
			while (true)
			{
				add((T) objectInStream.readObject());
			}
		} catch (EOFException e)
		{
			objectInStream.close();
		}

	}

	public void swap(int itr1, int itr2)
	{
		if (itr1 >= size && itr2 >= size && itr1 == itr2)
			return;

		Data<T> temp1 = start.next;
		Data<T> temp2 = start.next;
		for (int i = 0; i < itr1; i++)
		{
			temp1 = temp1.next;
		}

		for (int i = 0; i < itr2; i++)
		{
			temp2 = temp2.next;
		}
		T temp = temp1.obj;
		temp1.obj = temp2.obj;
		temp2.obj = temp;

	}

	public boolean find(T obj)
	{
		for (T value : this)
		{
			if (value.equals(obj))
				return true;
		}
		return false;
	}

	public String toString()
	{
		String str = new String();
		for (T value : this)
		{
			str += value + "\n";
```

```
            }
            return str;
      }

      public void clear()
      {
            start.next = last;
            last.prev = start;
            size = 0;
      }

      public void saveXml(String fileName) throws FileNotFoundException
      {
            XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new
FileOutputStream(fileName)));

            for (T value : this)
                  encoder.writeObject(value);
            encoder.close();
            System.out.println("Сериализация прошла успешно\n");
      }

      @SuppressWarnings("unchecked")
      public void downloadXml(String fileName) throws FileNotFoundException
      {
            XMLDecoder d = new XMLDecoder(new BufferedInputStream(new
FileInputStream(fileName)));
            try
            {
                  while (true)
                  {
                        add((T) d.readObject());
                  }
            } catch (ArrayIndexOutOfBoundsException e)
            {
                  d.close();
                  System.out.println("Десериализация прошла успешно\n");
            } catch (Exception e)
            {
                  throw e;
            }

      }

      public int getSize()
      {
            return size;
      }

      MyCollection()
      {
            size = 0;
            start = new Data<T>(null, null, null);
            last = new Data<T>(null, start, null);
            start.next = last;
      }

      void add(T obj)
      {
            Data<T> temp = last.prev;
            temp.next = new Data<T>();
            last.prev = temp.next;
```

```
        temp.next.obj = obj;

        temp.next.next = last;
        temp.next.prev = temp;
        size++;
}

void delete(int itr)
{
        if (itr >= size)
                return;

        Data<T> temp = start.next;
        Data<T> temp2 = null;

        for (int i = 0; i < itr; i++)
        {
                temp = temp.next;
        }

        temp2 = temp.prev;
        temp2.next = temp.next;
        temp.next.prev = temp2;
        size--;
}

T get(int itr)
{
        if (itr >= size && itr < 0)
                return null;

        Data<T> temp = start.next;

        for (int i = 0; i < itr; i++)
        {
                temp = temp.next;
        }

        return temp.obj;
}

public T[] toArray(T[] arr)
{
        for (int i = 0; i < size; i++)
        {
                arr[i] = get(i);
        }
        return arr;
}

@Override
public Iterator<T> iterator()
{
        return new Iterator<T>()
        {
                int itr = 0;

                @Override
                public boolean hasNext()
                {
```

```
                        return itr < size;
                }

                @Override
                public T next()
                {
                        return get(itr++);
                }

                @Override
                public void remove()
                {
                        delete(itr - 1);
                }
        };
    }
}
```

Route.java

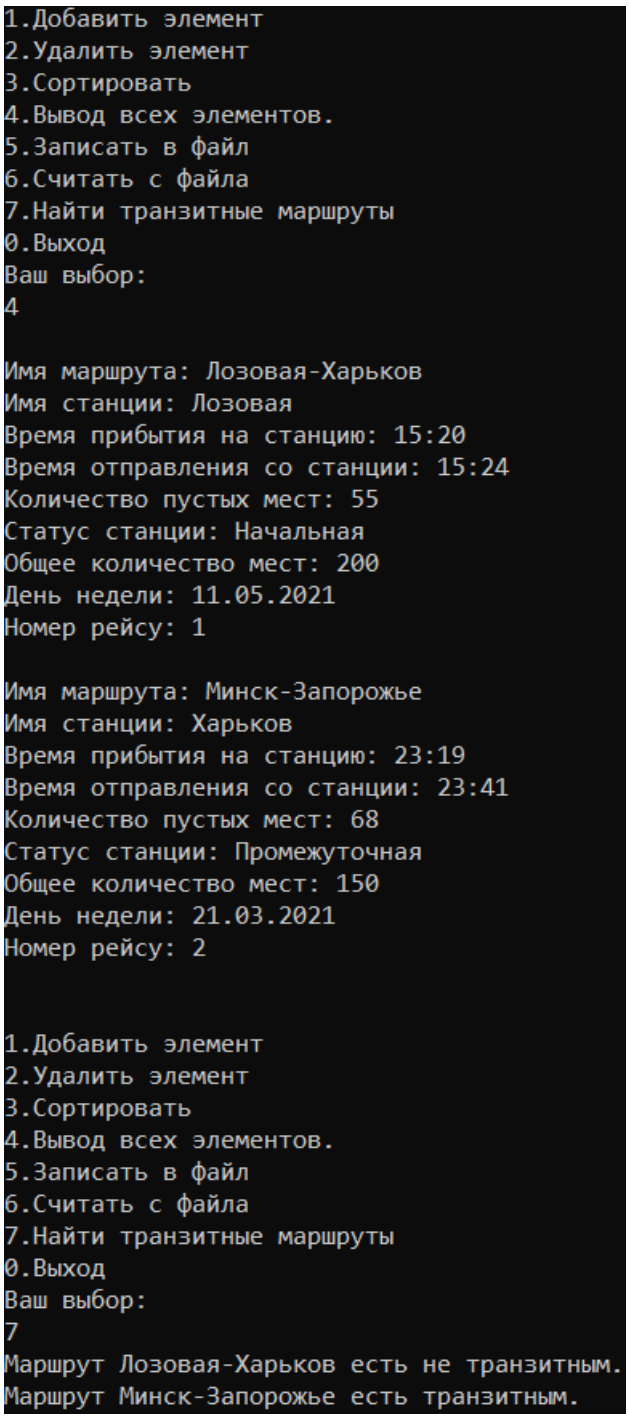
```
public class Route implements Serializable
{
    private static final long serialVersionUID = 1L;
    private String name_route;
    private String station_name;
    private String departure_time;
    private String arrival_time;
    private String number_of_free_seats;
    private String status_station;
    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]{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]{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 регулярними виразами у тексті.

Програма протестована, виконується без помилок.