MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

Expresii lambda și procesarea stream-urilor

Documentație

Ciubotaru Andrei-Mihai

Grupa: 30227

Cuprins

1.Obiectivul temei

Obiectivul acestei teme este sa realizam o aplicatie care indeplineste anumite task-uri pe baza unui fisier de activitati.

Aplicatia va dispune de o interfata „User Friendly”, care poate fi utilizata de oricine.

2.Analiza problemei

Pentru realizarea cerintelor trebuie folosite expresii lambda si procesarea stream-urilor. Trebuie implementate cinci task-uri :

- Numararea zilelor diferite din lista de activitati
- Sa se numere de cate ori apare fiecare activitate in lista
- Pentru fiecare zi sa se afiseze activitatile si numarul lor
- Sa se afiseze cu o durata mai mare de 10 ore
- Sa se afiseze activitatile care in 90% din aparitii au o durata mai mica de 5 minute

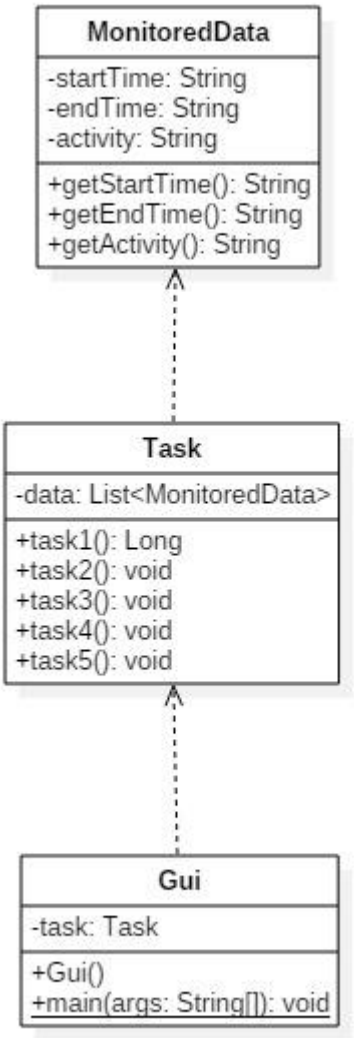
Rezultatele fiecarui task vor fi scrise in fisiere text.

3.Proiectare

Aplicatia are o structura simpla fiind alcatuita din 3 clase.

Diagrama UML presupune modelarea unui sistem prin lucrurile care sunt importante pentru acesta. Aceste lucruri sunt modelate folosind clase.

3.1.Diagrama de clase



4.Implementare

4.1.Clase si pachete

Clasa MonitoredData are ca variabile instanta startTime, endTime si activity.

```
public class MonitoredData {
    private String startTime;
    private String endTime;
    private String activity;

    public MonitoredData(String startTime, String endTime, String activity)
{
        this.startTime = startTime;
        this.endTime = endTime;
        this.activity = activity;
    }

    public String getStartTime() {
        return startTime;
    }

    public String getEndTime() {
        return endTime;
    }

    public String getActivity() {
        return activity;
    }
}
```

In clasa Task se implementeaza metodele pentru rezolvarea cerintelor.

Aceasata are ca variabila instanta o lista de obiecte de tipul MonitoredData.

```
public Task() {
    try {
        Stream<String> stream =
Files.lines(Paths.get("E://Proiecte/TP/tema5/Activities.txt"));
        stream.forEach(s->{
            System.out.println(s);
            String[] elements = s.split("\t\t");
            data.add(new
MonitoredData(elements[0],elements[1],elements[2]));});
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

In constructor citesc datele din fisier si le adaug intr-o lista. Pentru fiecare stream parsez datele(cu ajutorul metodei split) si fac un nou obiect de tipul MonitoredData si il adaug in lista.

```
public long task1() {

    return data.stream().map(s->{String[] elements =
s.getStartTime().split(" ");
    return elements[0];}).distinct().count();
}
```

Pentru a numara zilele diferite din liata de activitati aplic metoda map penru a mapa stream-urile dupa data de inceput. Pentru numara zilele distincte am folosit distinct.

```
public void task2() {
    Map<String,Long> activities;
    activities = data.stream()
            .map(s->s.getActivity())

.collect(Collectors.groupingBy(Function.identity(),Collectors.counting()));

    try (PrintWriter writer = new PrintWriter("count_activities.txt", "UTF-
8")) {
        for (Map.Entry m : activities.entrySet())
            writer.println(m.getKey() + " " + m.getValue());
        writer.close();
        System.out.println("Task_2: OK");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Pentru a numara de cate ori apare fiecare activitate in lista am mapat stream-ul in functie de activitate si rezultatul l-am salvat intr-un map Map<String,Long>. Am folosit groupingBy pentru a mapa in hashmap in functie de activitate, iar valoarea este reprezentata de nr de activitati.

```
public void task3() {
    Map<String,Map<String,Long>> activities;
    activities = data.stream()
            .collect(Collectors.groupingBy(s->{String[] elements =
s.getStartTime().split(" ");
                return
elements[0];},Collectors.groupingBy(MonitoredData::getActivity,Collectors.c
ounting())));

    try (PrintWriter writer = new PrintWriter("day_activities.txt", "UTF-
8")) {
        for (Map.Entry m : activities.entrySet()) {
            writer.println(m.getKey() + ":");
            for(Map.Entry entry :  ((Map<String, Long>)
m.getValue()).entrySet())
                writer.println(entry.getKey() + " " + entry.getValue());
            writer.println("");
```

```
        }
        writer.close();
        System.out.println("Task_3: OK");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Pentru a numara activitatile din fiecare zi am utilizata o structura de date de tipul map<String,map<String,Long>>. Am mapat stringurile dupa data de inceput, iar cheia este alcatuita din alt map format din activitate si numarul de aparitii.

```
public void task4() {

    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Map<String,Long> activities;
    activities = data.stream()
            .collect(Collectors.groupingBy(s->s.getActivity(),
    Collectors.summingLong(s->{try {
        Date date1 = df.parse(s.getEndTime());
        Date date2 = df.parse(s.getStartTime());
        long diffInMillies = date1.getTime() - date2.getTime();
        return diffInMillies/3600000;
    } catch (ParseException e) {
        e.printStackTrace();
        return -1;
    }}
    )));

    List<Map.Entry<String, Long>> activities_10;
    activities_10 = activities.entrySet()
            .stream()
            .filter(e->e.getValue()>10)
            .collect(Collectors.toList());

    try (PrintWriter writer = new PrintWriter("duration_activities.txt",
"UTF-8")) {
        for (Map.Entry m : activities.entrySet())
            writer.println(m.getKey() + " " + m.getValue());
        writer.println("");
        writer.println("larger than 10:");
        for(Map.Entry entry : activities_10)
            writer.println(entry.getKey() + " " + entry.getValue());
        writer.close();
        System.out.println("Task_4: OK");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Acest task filtreaza activitatile care au durata totala mai mare de 10 ore.

Am grupat intr-un map dupa activitate, iar cheia este reprezentata de timpul total calculat astfel: Am transformat stringurile start date si end date in obiecte de

tipul Date si am facut diferenta iar pentru a transforma in ore din milisecunde am impartit rezultatul la 60*1000*60. Apoi in cel de-al doilea map am filtrat activitatile > 10 ore.

```
public void task5() {

    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Map<String,Long> activities_5;
    activities_5 = data.stream()
            .filter(s->{try {
                Date date1 = df.parse(s.getEndTime());
                Date date2 = df.parse(s.getStartTime());
                long diffInMillies = date1.getTime() - date2.getTime();
                return (diffInMillies / 60000) < 5;
            } catch (ParseException e) {
                e.printStackTrace();
                return false;
            }})
            .collect(Collectors.groupingBy(s->s.getActivity(),
                    Collectors.counting()));

    Map<String,Long> activities;
    activities = data.stream()
            .map(s->s.getActivity())

.collect(Collectors.groupingBy(Function.identity(),Collectors.counting()));

    System.out.println(activities);
    System.out.println("<5\n" +activities_5);

    List<String> result;
    result = activities.entrySet().stream()
            .filter(a-> {List<Long>
nr=activities_5.entrySet().stream().filter(s->
s.getKey().equals(a.getKey()))
                    .map(Map.Entry::getValue).collect(Collectors.toList());
                    return nr.size()==1 && nr.get(0)/a.getValue()>=0.9; })
            .map(a->a.getKey()).collect(Collectors.toList());

    System.out.println(result);

    try (PrintWriter writer = new PrintWriter("90%_activities.txt", "UTF-
8")) {
        for (String i : result)
            writer.println(i);
            writer.close();
        System.out.println("Task_5: OK");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

In structura activities_5 am stocat numarul de aparitii al fiecarei activitati < 5 minute. In structura activities am pus rezultatul de la task-ul 2. In lista result am filtrat activitatile < 5 minute care au ponderea mai mare de 90% . In fiecare metoda (mai putin task-ul 1) am scris rezultatele intr-un fisier text.

4.2.GUI

In clasa Gui se realizeaza interfata.

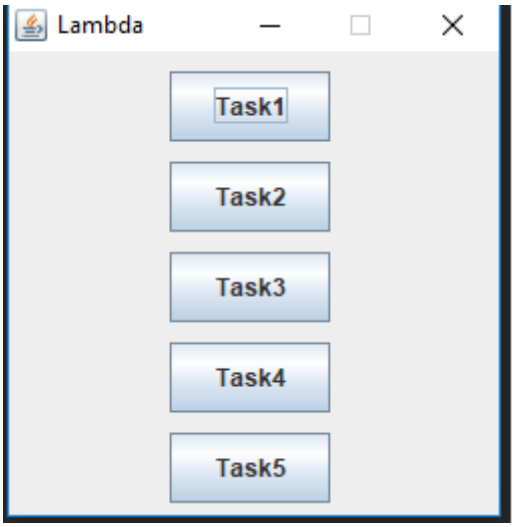
In constructor este creat frame-ul principal cu 5 butoane, iar la apasarea unui buton se apeleaza metoda corespunzatoare task-ului si se va deschide un fisier text cu rezultatul.

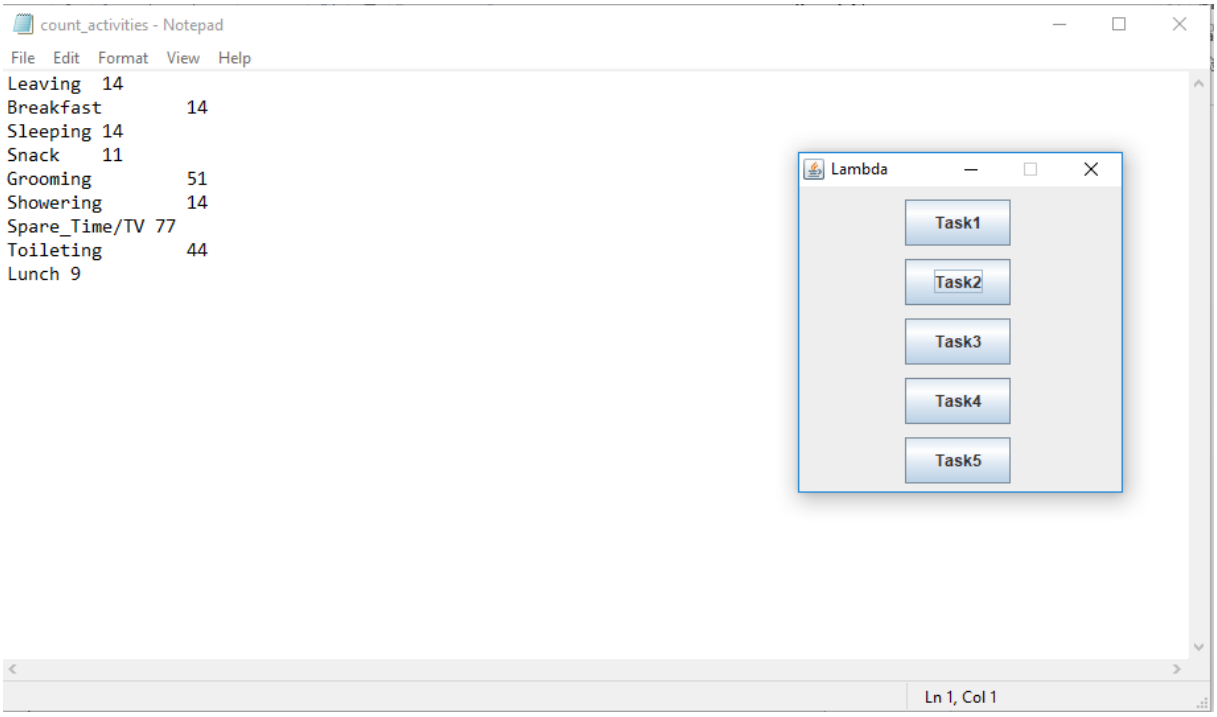
```
        import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;

public class Gui extends JPanel{

    private Task task = new Task();

    public Gui() {
        JFrame frame = new JFrame("Lambda");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(250, 260);

        JPanel panel = new JPanel();
        JButton task1 = new JButton("Task1");
        JButton task2 = new JButton("Task2");
        JButton task3 = new JButton("Task3");
        JButton task4 = new JButton("Task4");
        JButton task5 = new JButton("Task5");
        frame.add(task1);
        task1.setBounds(80,10,80,35);
        frame.add(task2);
        task2.setBounds(80,55,80,35);
        frame.add(task3);
        task3.setBounds(80,100,80,35);
        frame.add(task4);
        task4.setBounds(80,145,80,35);
        frame.add(task5);
        task5.setBounds(80,190,80,35);
        frame.add(panel);
        frame.setResizable(false);
        frame.setVisible(true);

        task1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                long nr = task.task1();
                JOptionPane.showMessageDialog(frame,"Nr. activitati: " +
nr);
            }
        });

        task2.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    task.task2();
                    Desktop.getDesktop().open(new
java.io.File("count_activities.txt"));
```

```
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        });

        task3.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    task.task3();
                    Desktop.getDesktop().open(new
java.io.File("day_activities.txt"));
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        });

        task4.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    task.task4();
                    Desktop.getDesktop().open(new
java.io.File("duration_activities.txt"));
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        });

        task5.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    task.task5();
                    Desktop.getDesktop().open(new
java.io.File("90%_activities.txt"));
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        });

    }

    public static void main(String[] args) {
        Gui gui = new Gui();
    }

}
```

5.Testare

Din panoul principal se alege operatia dorita.





6.Rezultate

Aplicatia gestioneaza fisier care contine o lista de activitati si genereaza urmatoarele rezultate:

Numararea zilelor diferite din lista de activitati.

Sa se numere de cate ori apare fiecare activitate in lista.

Pentru fiecare zi sa se afiseze activitatile si numarul lor.

Sa se afiseze cu o durata mai mare de 10 ore.

Sa se afiseze activitatile care in 90% din aparitii au o durata mai mica de 5 minute.

7.Concluzii

In aceasta tema am invatat sa lucrez cu Clasa Stream si expresii Lambda si conversia in obiect de tip Date.

8.Bibliografie

www.stackoverflow.com

https://docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html

https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html