Polynomial processing system

Name: Zavaczki Péter-Tibor
Group: 30423
Coordinating professor: Cristina Bianca Pop

1. Objective of the project

The presented project was made for understanding lambda expressions and streams.

2. Method uses

MonitoredData class

This class has its constructor, setters and getters for its attributes, also an override for toString and a custom method getDuration() that returns the difference between the endTime and startTime.

DataPopulation class

This class has a parameter which stores the data read from a text file, the format of this text file's entries, and the default path where this file is found.

int readLines(String path) throws IOException is a method used for counting the number of lines (entries) in the text file.

public String[] readFile(String path) throws IOException is a method used for reading the actual data from the text file and returns it in the String array.

public MonitoredData textToMD(String dataLine) is a method used for parsing a line of String to an object of type MonitoredData.

public int populateDataList(String path) is a method used for populating the list which is the attributes in this class with the data found in the given text file.

public void printList() is a method used to print the data found in the list attribute of this class to the standard display.

TestDataPopulation class

This class calculates the requested values, using the DataPopulation class. The tested class has the data on which we work. I used lambda expressions and streams to achieve this.

nrOfDistinctDays = loc.monitoredData.stream().map(c →c.getStartTime().toLocalDate().getDayOfYear()).distinct().count(); is the stream expression which counts the distinct days of the year appearing in the given data.

Map<String, Long> actionOccur = loc.monitoredData.stream().collect(Collectors.groupingBy(c ->c.getActivityLabel(), Collectors.counting())); is the stream expression which maps the actions appearing in the data to their number of occurences.

Map<Object, Map<Object, Long>> actionOccurPerDay = loc.monitoredData.stream().collect(Collectors.groupingBy(c->c.getStartTime().toLocalDate().getDayOfYear(), Collectors.groupingBy(c ->c.getActivityLabel(), Collectors.counting()))); is the stream / lambda expression used to map the actions appearing in the data to their number of occurences to the sepparate days of the year on which they appear.

Map<String, Long> filteredTotalDur = (Map<String, Long>) totalDur.entrySet().stream().filter(map -> map.getValue()>36000).collect(Collectors.toMap(p -> p.getKey(), p -> p.getValue())); is the stream / lambda expression which maps the actions to their total duration, if the total duration is larger than 10 hrs.

List<String> shortActions = actionDurPercent.entrySet().stream().filter(c->c.getValue() >= 0.9).map(x→x.getKey()).collect(Collectors.toList()); is the stream / lambda expression used to get the actions that have over 90% of their data samples of under 5 mins length.