Technical University of Cluj-Napoca

Programming Techniques

Laboratory-Assignment 5

Teacher: prof. Ioan Salomie

Teacher Assistant: Viorica Chifu

Student:Stioiu Denis-Adrian

Group:30424

1. Objective

Consider designing, implementing and testing an application for analysing the behaviour of a person recorded by a set of sensors installed in its house. The historical log of the person's activity is stored as tuples (start_time, end_time, activity_label), where start_time and end_time represent the date and time when each activity has started and ended while the activity label represents the type of activity performed by the person: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner, Snack, Spare_Time/TV, Grooming. The data is spread over several days as many entries in the log Activities.txt.

Write a program that uses functional programming in Java with lambda expressions and stream processing to perform the tasks listed in the table below. The results of each task must be written in a separate .txt file (each .txt file must be named according to the following template task_number.txt, for example Task_1.txt).

2. Problem analysis, scenarios, use cases

A. General Overview

The application processes through stream processes the tuples representing the activities and the time they took place collected by sensors. The streams are processed using lambdas, the biggest feature of Java8. The tuples are read from the file Activities.txt. The data is thus easier to be parsed than using classical methods.

Lambda expression is a new and important feature of Java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.

Java lambda expression is treated as a function, so compiler does not create .class file.

B. Input and Output

Using streams the application parses a txt file and extract the necessary information. The output are 6 txt files representing the tasks we are asked to solve the tasks are as follows:

Task Number	Objective
Task_1	Define a class MonitoredData with 3 fields: start time, end time and activity as string. Read the data from the file Activity.txt using streams and split each line in 3 parts: start_time, end_time and activity_label, and create a list of objects of type MonitoredData
Task_2	Count the distinct days that appear in the monitoring data
Task_3	Count how many times each activity has appeared over the entire monitoring period. • Return a structure of type Map representing the mapping of each distinct activity to the number of occurrences in the log; therefore the key of the Map will represent a String object corresponding to the activity name, and the value will represent an Integer object corresponding to the number of times the activity has appeared over the monitoring period.
Task_4	Count for how many times each activity has appeared for each day over the monitoring period. • Return a structure of type Map> that contains the activity count for each day of the log; therefore the key of the Map will represent an Integer object corresponding to the number of the monitored day, and the value will represent a Map (in this map the key which is a String object corresponds to the name of the activity, and the value which is an Integer object corresponds to the number of times that activity has appeared within the day)
Task_5	For each activity compute the entire duration over the monitoring period. • Return a structure of type Map in which the key of the Map will represent a String object corresponding to the activity name, and the value will represent a LocalTime object corresponding to the entire

	duration of the activity over the monitoring period.
Task_6	Filter the activities that have more than 90% of the monitoring records with duration less than 5 minutes, collect the results in a List containing only the distinct activity names and return the list.

Stream represents a sequence of objects from a source, which supports aggregate operations. Following are the characteristics of a Stream −

- Sequence of elements − A stream provides a set of elements of specific type in a sequential manner. A stream gets/ computes elements on demand. It never stores the elements.
- Source − Stream takes Collections, Arrays, or I/O resources as input source.
- Aggregate operations − Stream supports aggregate operations like filter, map, limit, reduce, find, match, and so on.
- Pipelining − Most of the stream operations return stream itself so that their result can be pipelined. These operations are called intermediate operations and their function is to take input, process them, and return output to the target. Collect() method is a terminal operation which is normally present at the end of the pipelining operation to mark the end of the stream.
- Automatic iterations − Stream operations do the iterations internally over the source elements provided, in contrast to Collections where explicit iteration is required.

3. Data Structures

ArrayList: for storing the monitored data read from the stream

Hash Maps: for storing and indexing the data, and solving the tasks

4. Class design

a. Main Class: Used to implement all the tasks
b. MonitoredData: Class used to store a tuple, having the attributes activityLabel, a string to store the name of the activity, and startTime and endTime used to store the time recorder by the sensors, and are of type LocalDateTime. The class has a constructor, getters and setters.
c. Task1: Creates a list of objects of type MonitoredData
d. Task2: Is used to count the distinct days that appear in the monitoring data
e. Task3: Counts the frequency of activities in the monitoring period
f. Task4: Counts for how many times each activity has appeared for each day over the monitoring period.

g. Task5: Computes the duration of each activity over the monitoring period
h. Task6: Filter the activities that have more than 90% of the monitoring records with duration less than 5 minutes.

5. Conclusion

I find this project as an excellent opportunity to learn how to work with lambdas and streams in java, and to perfect my OOP skills. This application could be developed by adding a graphical user interface, and creating a network for users to compare their results and activities, and pieces of advice from specialists about how the way of life can be improved.

6. Bibliography

https://archive.ics.uci.edu/ml/datasets/Activities+of+Daily+Living+(ADLs)+Recognition+Using+Binary+Sensors

https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html

https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/

Resources provided at lectures and laboratories by the teachers