**PROCESSING SENSOR DATA OF DAILY LIVING ACTIVITIES**

1. Objectives
   1. Main Objective

Consider designing, implementing and testing an application for analyzing the behavior 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\_lable represents the type of activity performed by the person: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner, Snack, Spare\_Time/TV, Grooming. The data si spread over several days as many entries in the log input file.

Write a program that uses functional programming in Java with lambda expressions and stream processing to perform the given tasks. The result of each task must be written in a separate .txt file.

* 1. Secondary Objectives
* Implement a class for each of the given tasks.
* Each class should deal with solving a specific task as well as printing the result in the requested fromat.
* Define the class MonitoredData used to store the parsed data from the input file.

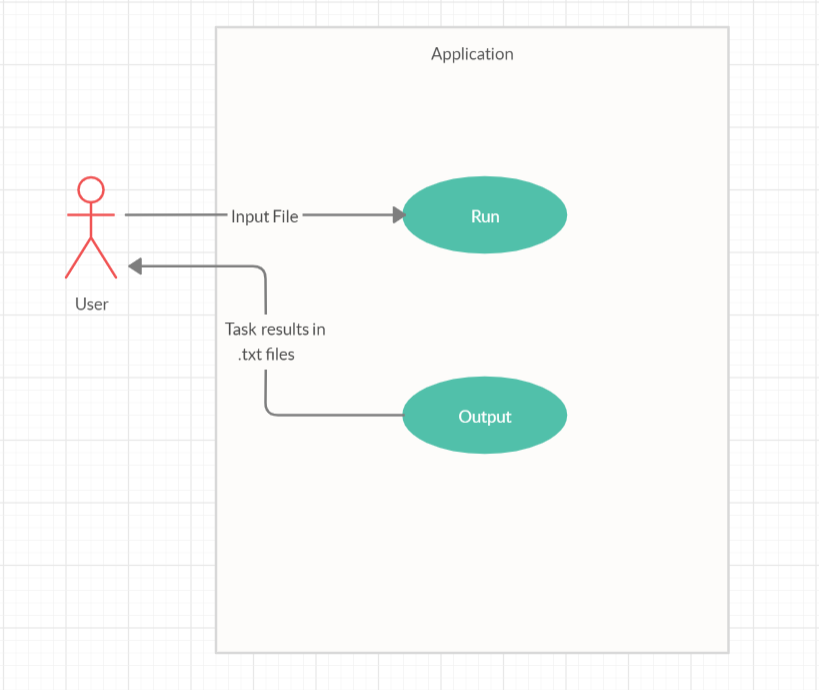
1. Task analysis, Modeling, Scenarios, Uses Cases

The application should receive as argument the path to the file that stores the log of the activities obtained through the sensors. The file should contain any number of rows having the format:

start\_time end\_time activtity\_label

For example:

2011-11-28 02:27:59 2011-11-28 10:18:11 Sleeping



* Use Case: Perform Tasks
* Primary Actor: User
* Main Success Scenario:
  + The user runs the application providing the input file path.
  + After the application is done running a new folder is created named Tasks.
  + In this folder the user can see the results for each individual task stored in its associated Task\_number.txt file
* Alternative Sequences:
  + The user does not provide an input file.
    - The application will exit its running.

1. Design

Given the purpose of the application and its requirements, it is very easy to identify what it should be done. Because the requirements consist in 6 tasks well defined, the application will have a class for each of the tasks having the functionality of performing that and only that specific task.

Task 1 requires the following: “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.”. We first define the class MonitoredDate having the three fields startTime, endTime and activityLabel used for storing the results of performing Task 1. Then we define the class Task1 in which we implement the perform method providing the need functionality as well as the method printFile which will print the result into a .txt file.

Task 2 requires the following: “Count the distinct days that appear in the monitoring data.”. We define the class Task2 in which we implement the perform method providing the need functionality as well as the method printFile which will print the result into a .txt file.

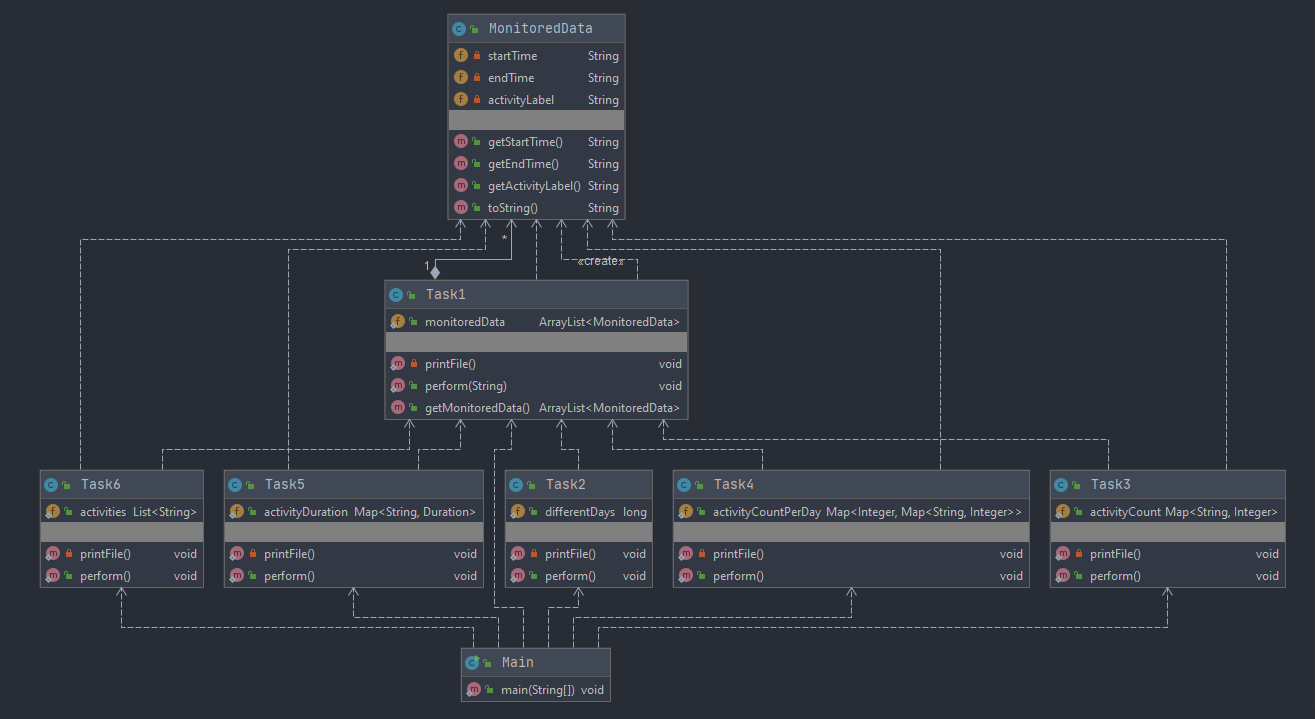
Task 3 requires the following: “Count how many times each activity has appeared over the entire monitoring period.”. We define the class Task3 in which we implement the perform method providing the need functionality as well as the method printFile which will print the result into a .txt file.

Task 4 requires the following: “Count for how many times each activity has appeared for each day over the monitoring period.”. We define the class Task4 in which we implement the perform method providing the need functionality as well as the method printFile which will print the result into a .txt file.

Task 5 requires the following: “For each activity compute the entire duration over the monitoring period.”. We define the class Task5 in which we implement the perform method providing the need functionality as well as the method printFile which will print the result into a .txt file.

Task 6 requires the following: “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.”. We define the class Task6 in which we implement the perform method providing the need functionality as well as the method printFile which will print the result into a .txt file.

For a better understanding of the design, the following schema includes all the classes among with their dependencies.



1. Implementation

* MonitoredData Class

This is the class required by the first task. The purpose of this is class is to provide a convenient way of storing a row from the input file as an object that we can work with. It has three private String fields startTime, endTime and activityLabel for which it also provides getters methods.

* Task1 Class

This class performs task 1. It has a static variable monitoredData of type ArrayList<MonitoredData> in which the result is stored. The method perform is the one that provides the implementation of the task. The result is achieved using stream processing on the input file. The operations on the stream are as follow:

* Map the row by splitting it with the regex ([\\t]+)
* Map the splited row to a new object of type MonitoredData
* Collect the Monitored Data objects into a list
* Task2 Class

This class performs task 2. It has a static variable differentDays of type long in which the result is stored. The method perform is the one that provides the implementation of the task. The result is achieved using stream processing on the list obtained by perfoming Task 1. The operations on the stream are as follow:

* FlatMap the startTime and endTime of all the MonitoredData objects from the list. We obtain a stream containing all the startTime and endTime.
* Map them to their date getting rid of the time stamps.
* Keep only the distinct ones
* Count them to get tha number of days
* Task3 Class

This class performs task 3. It has a static variable activityCount of type Map<String, Integer> in which the result is stored. The method perform is the one that provides the implementation of the task. The result is achieved using stream processing on the list obtained by perfoming Task 1. The operations on the stream are as follow:

* Map the objects to their activityLabel field
* Keep only the distinct ones
* Collect them into a Map where the key is the activityLabel and value is given by processing another stream based on the list obtained by performing Task 1 on which we apply the following operations:
  + Filter by activityLabel
  + Count the number of objects.
* Task4 Class

This class performs task 4. It has a static variable activityCountPerDay of type Map<Integer, Map<String, Integer>> in which the result is stored. The method perform is the one that provides the implementation of the task. The result is achieved using stream processing on the list obtained by perfoming Task 1. FlatMap the startTime and endTime of all the MonitoredData objects from the list. We obtain a stream containing all the startTime and endTime. There are three nested stream having the operations as follows:

* FlatMap the startTime and endTime of all the MonitoredData objects from the list. We obtain a stream containing all the startTime and endTime.
* Map them to their date getting rid of the time stamps.
* Keep only the distinct ones
* Collect them into a Map where the key is the variable index of type AtomicInteger and the value given by another stream:
  + Map objects to their activityLabel
  + Keep only the distinct ones
  + Colect them into a Map where the key is the activityLabel and the value given by another stream:
    - Filter objects to have the activityLabel equal to the ones before from stream 2
    - Filter the objects to have the date from the startingTime equal to the ones from stream 1
    - Count the number of objects
* Task5 Class

This class performs task 5. It has a static variable activityDuration of type Map<String, Duration> in which the result is stored. The method perform is the one that provides the implementation of the task. The result is achieved using stream processing on the list obtained by perfoming Task 1. The operations on the stream are as follow:

* Map the objects to their activityLabel
* Keep only the distinct ones
* Collect them to a Map where the key is the activityLabel and the values are given by another stream:
  + Filter objects to have the activityLabel equal to the ones before from stream 1
  + Map them to their duration computed using the LocalDateTime obtained from the fields startTime and endTime converting them to a from yyyy:mm:ddThh:mm:ss
  + Reduce them to a total duration
* Task6 Class

This class performs task 6. It has a static variable activities of type List<String> in which the result is stored. The method perform is the one that provides the implementation of the task. The result is achieved using stream processing on the list obtained by perfoming Task 1. There are three streams that are being processed. The first two are independent of each other and the third one contains them both

The first stream (getting the number of records of an activity that has a durations less than 5 minutes):

* Filter objects by activityLabel
* Filter objects by duration < 5
* Count the number of objects

The second stream (getting the total number of records of an activity)

* + - Filter objects by activityLabel
    - Count the number of objects

The third stream (getting the required activities labels)

* + - Map objects to their activityLabel
    - Keep only the distinc ones
    - Filter them by a > b\*(90/100) where a = stream 1 and b = stream 2
    - Collect the results to a list

1. Results

The testing of the application consists of checking the results of the tasks. Each task gives as output a text file which contains the data obtained by executing it. The results can be manually verified in most of the cases If the number of logs in the files is not too high.

1. Conclusions

The main objective of this project was to get used working with lambdas expressions as well as understanding the advantages of working with streams provided by java. Lambdas are the base of the functional programming and we get to see how powerful they are when used along with stream processing for example.

In terms of future improvements, the application can support way more tasks that could be performed on a log file in order to obtain all kind of useful data out of it. It could also provide a user interface in order to let the user choose which tasks to be performed.

1. Bibliography

* Ordóñez, F.J.; de Toledo, P.; Sanchis, A. Activity Recognition Using Hybrid Generative/ Discriminative Models on Home Environments Using Binary Sensors. Sensors 2013, 13, 5460-5477.
* <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/>

‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‏‏‎ ‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎ ‏‏‎ ‎