# Documentation
# Person Behaviour Analysing

**Student:** Rusu Daniel
**Group:** 30422

**Table of Contents:**

# 1. Objective

**Main objective of the assignment:**
Consider the task of analysing the behaviour of a person recorded by a set of sensors. 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.

**Secondary objectives:**
- Creating a stream out of the given Activities.txt file, in order to read the data.
- Implementing algorithms to process the data coming from the stream, in order to achieve the desired results.
- Display the results nicely, so that the user can read it with little to no effort.

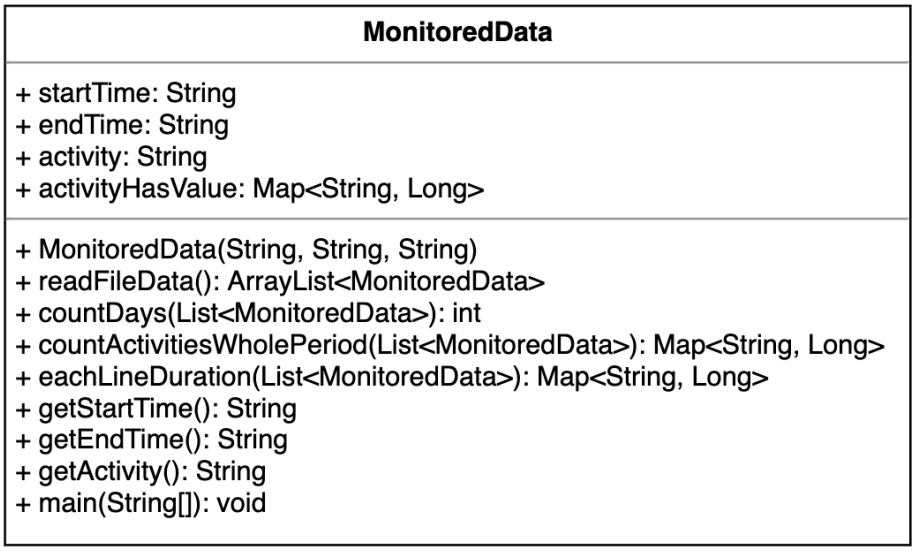
All the secondary objectives presented above will be detailed in the fourth chapter of this documentation.

# 2. Problem analysis, modelling, use-cases.

This application is built to analyse the data received (in a .txt file), regarding the activities performed by a person during a certain period of time. The file should be organised as follows: each line contains information about one activity: start time, end time, and activity label. The total period of time and number of recorded activities are not specified.
The main goal of the application is to manage this "big" input data in order to compute certain stats (e.g. the total time spent showering, during the monitoring period). There are no use cases. The Activities.txt file is created and then filled with data by the sensors, and whenever the application is run, the process is standard.

# 3. Design (decisions, UML diagram, class design, user interface)

The software design approach is OOP style. The application uses a system of interacting objects for the purpose of solving the proposed problem.
The UML class Diagram is presented below:



The class is illustrated as a rectangle divided into three sections: the first one (up) contains the name of the class; the second (middle) contains the class fields and the third (down) contains the implemented class methods.

# 4. Implementation

The application consists of one class, that is presented and detailed below.

**The MonitoredData Class**
This class is used to represent the monitored data object. It contains the fields specific to such an object: start time, end time and activity label, as well as a constructor that receives these fields as attributes.
The class also contains the methods implemented to perform some computations on the input data (as requested in the assignment paper):

- readFileData: this method is used to read the text from the input file Activities.txt. It creates a stream from the file and then uses a “flatMap” and “collect” combination in order to divide the file input into individual lines and then each line into three strings: the start time, end time and activity label. Then it creates an object of type MonitoredData, using these three strings, and places all the objects in one list. This list is the return value.
- countDays: this method is used to count the total number of days, over which the activities were recorded, and also to count how many times has each activity appeared in each day. It does this by calling the countActivitiesWholePeriod method on portions of input corresponding to each day, and then printing the result. The countDays method returns an integer corresponding to the number of recorded days, during the whole monitoring period.
- countActivitiesWholePeriod: this method is used to count how many times has each activity appeared during the period received as parameter, using streams. It is used for the whole monitoring period, as requested by the assignment paper, as well as in the countDays method, for different periods of time.
- eachLineDuration: this method is used to map for each activity (each line) the duration (end time - start time). It is also used to compute the entire duration over the monitoring period for each activity.
- getters for the specified fields: used by other methods in the class, in order to get the requested value (not really needed, since there is only one method… but it is nicer this way).

# 5. Results

By running the application, the result of all the processing can be seen in the console. Given the input file Activities.txt, the program will print the desired output (e.g. the total number of days, and so on…).

# 6. Conclusions

This application, even if not that long and complex, has made me learn some interesting new concepts and techniques: lambda expressions and streams (Java 8). I find these new features very useful, and therefore interesting, due to the way they could make the coding process faster and easier.

**Future developments:**
- Printing the results in a text file
- Completing the last assignment requirement
- Using lambdas and streams more, by changing the current methods.

# 7. Bibliography


- Java 8 streams: https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/
- Lambda expressions: https://www.tutorialspoint.com/java8/java8_lambda_expressions.htm
- Java files and I/O: https://www.tutorialspoint.com/java/java_files_io.htm


Thank you! :)
Rusu Daniel