**Processing sensor data of daily living activities**

**Avram Andrei-Alexandru**

**30424**

1. **Assignment’s objective**

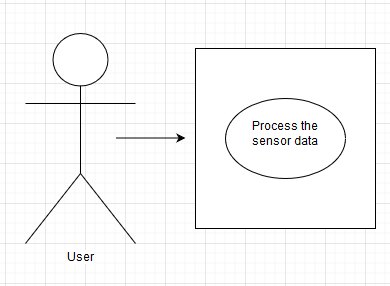
The main objective of this assignment is to design and implement an application that is capable of analyzing the behavior of a person recorded by a set of sensors installed in his house. The entry log will be read from a text file and the format is presented below. As for secondary objectives, certain data processing tasks must be implemented and they are explained later in this documentation. Here is a fragment from the input file.



The first set of date and time is the starting time of the activity, the second set is the end time of the activity and the last string is the activity name. The output for each task will be written in a separate text file having the task number as the filename.

1. **Problem analysis, modelling, scenarios, use-cases**

The possible use-case is presented in the following drawing:



1. Success scenario:

* The input data format is correct and can be processed by the application.

1. Alternative scenario:

* The input data is not correct and the processing cannot be completed.

This application can have many real-world use-cases. For example, you can use this application to analyze the output of many sensors provided that the log of these sensors respects the input format.

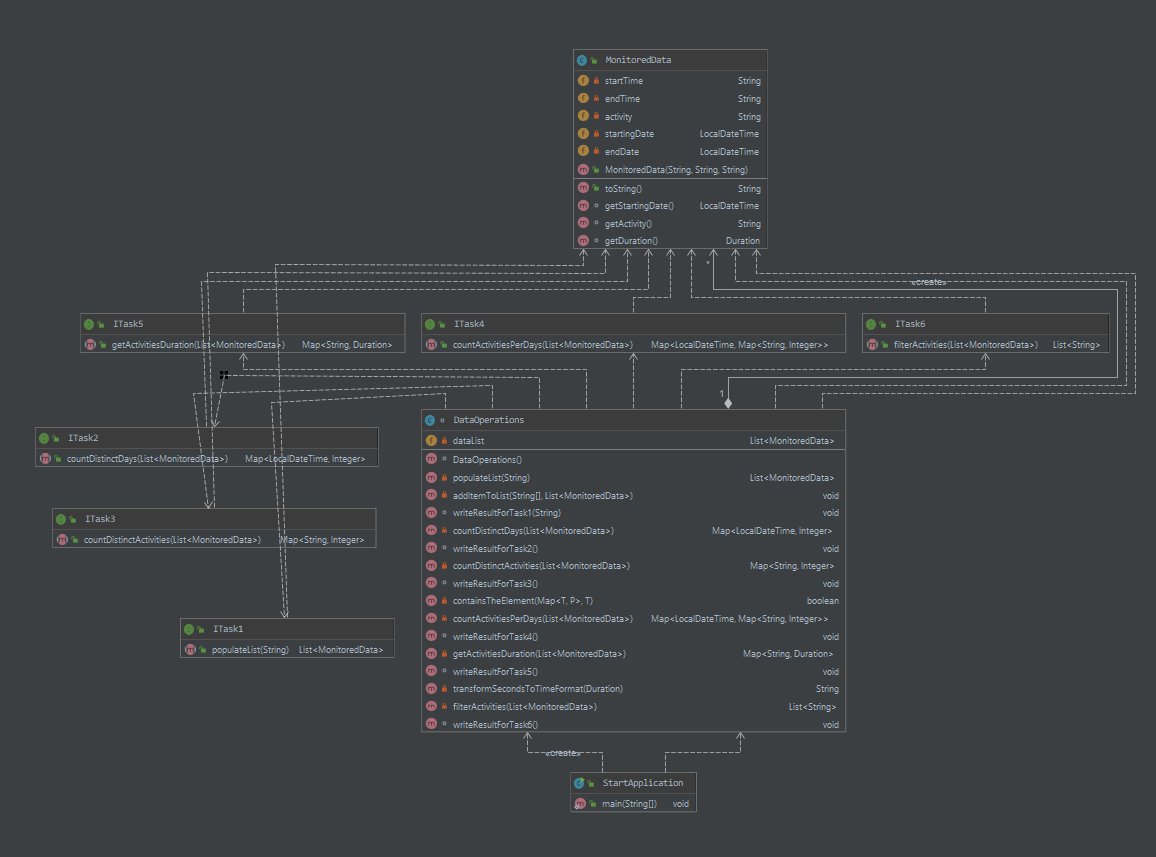
1. **Design (design decisions, UML diagrams, data structures, class design, interface, relations, packages, algorithms)**

For the design of this application only one package was required.

Data structures used in this application:

* Maps and Lists to hold the data processing results.

The UML class diagram for the whole project is presented below:



1. **Implementation**
2. ITask1 interface

* This interface contains the method that is used for the task 1. This method populates the list of MonitoredData objects. This method returns the list of objects and it takes as the parameter a string representing the name of the file containing the sensor data log.

1. ITask2 interface

* This interface contains the method that is used for the task 2. This method counts the distinct days that appear in the monitoring data. This method returns a map <LocalDateTime, Integer> containing the distinct day in the log and the number of occurrences of that day. It takes as the parameter the list of objects containing the data log from the sensor.

1. ITask3 interface

* This interface contains the method that is used for the task 3. This method is used to count the distinct activities that appear in the data log. This method returns a map <String, Integer> containing the distinct activities and number occurrences of that activity. It takes as the parameter the list of objects containing the data log from the sensor.

1. ITask4 interface

* This interface contains the method that is used for the task 4. This method is used to count how many times each activity has appeared for each day over the monitoring period. This method returns a map <LocalDateTime, Map<String, Integer>> containing the distinct day of the monitoring period and a map containing the activity name and the number of times this activity has appeared on that particular day. It takes as the parameter the list of objects containing the data log from the sensor.

1. ITask5 interface

* This interface contains the method that is used for the task 5. This method is used to compute the entire duration over the monitoring data for every activity. This method returns a map <String, Duration> containing the name of the activity and the duration of the activity. It takes as the parameter the list of objects containing the data log from the sensor.

1. ITask6 interface

* This interface contains the method that is used for the task 6. This method is used to filter the activities that have more 90% of the monitoring records with duration less than 5 minutes. This method returns a list of strings containing the activities that respect this requirement. It takes as the parameter the list of objects containing the data log from the sensor.

1. StartApplication class

* This class contains the main method used to run the program. The main method has an object of type DataOperations used to perform the data processing. The method calls all the six methods to write the results for the six tasks that are required to compute in this assignment.

1. MonitoredData class

* This class is used to hold the records that are found inside the input text file. It has 5 fields representing the starting time of the activity, the end time of the activity, a string representing the name of the file, a LocalDateTime representing the starting date of the activity, a LocalDateTime representing the end date of the activity. This class overrides the method toString and has only getters.

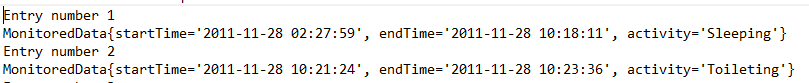
1. DataOperations class

* This class is used to implement all the methods that are required to complete the tasks. Next, I will explain all the methods that are inside this class. This class has only one field that is a list of objects that represent all the log data.
* populateList() method – It is used to read the data from the input file using streams and to populate the list of objects. This method uses lambda expressions to implement the method that is inside the ITask1 interface. This method reads each line from the file, splits the line based on the two tab characters that separate start, end time and the activity name and for each line another method (addItemToList()) is called.
* addItemToList() method – It is used to create and insert an object of type MonitoredData inside the list of objects. It takes as the parameters an array of strings representing the starting/ending time and activity name and the dataList where to add the object.
* writeResultForTask1() method – It is used to write the results of the task 1 in a text file having the name ‘task\_1.txt’. It iterates through all the objects that are in the data list and for every object it writes the result of the toString method call.
* countDistinctDays() method – It is used to count the distinct days that appear in the monitoring data. This method uses lambda expressions to implement the method that is inside the ITask2 interface. This method returns a map containing the distinct day represented as a LocalDateTime and an integer representing the number of occurrences of that day. The algorithm for this operation is quite simple. It iterates through all the objects inside the data list and if the day is not found in the map it adds the day with the number of occurrences of 1. If the day is found it increments the number of occurrences.
* writeResultForTask2() method – It is used to write the results of the task 2 in a text file having the name ‘task\_2.txt’. It goes through all the objects inside the map of distinct days and writes the day and the number of occurrences in the text file.
* countDistinctActivities() method - It is used to count the distinct activities that appear in the monitoring data. This method uses lambda expressions to implement the method that is inside the ITask3 interface. This method returns a map containing the activity name as a string and an integer representing the number of times this activity was found inside the monitoring data. The algorithm for this operation is: it iterates through all the objects inside the data list and if the activity is not found in the map it adds the activity with the number of occurrences of 1. If the activity is found it increments the number of occurrences.
* writeResultForTask3() method - It is used to write the results of the task 3 in a text file having the name ‘task\_3.txt’. It goes through all the objects inside the map of distinct activities and writes the activity and the number of occurrences in the text file.
* countActivitiesPerDays() method – It is used to count how many times each activity has appeared for each day over the monitoring period. This method uses lambda expressions to implement the method that is inside the ITask4 interface.
* writeResultForTask4() method – It is used to write the results of the task 4 in a text file having the name ‘task\_4.txt’.
* getActivitiesDuration() method – It is used to compute the entire duration over the monitoring period. This method uses lambda expressions to implement the method that is inside the ITask5 interface.
* writeResultForTask5() method – It is used to write the results of the task 5 in a text file having the name ‘task\_5.txt’.
* filterActivities() method – It is used to filter out the activities where the 90% of their records are with a duration of 5 minutes or less. This method uses lambda expressions to implement the method that is inside the ITask6 interface.
* writeResultForTask6() method – It is used to write the results of the task 6 in a text file having the name ‘task\_6txt’.

1. **Results**

After successfully processing the data the following results will appear in the output text files. These photos do not represent the whole output, only a small part of it.

Task 1:



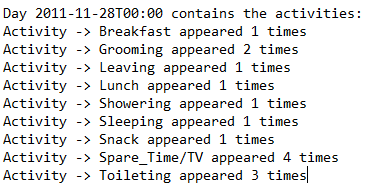
Task 2:



Task 3:



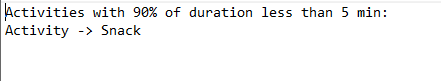
Task 4:



Task 5:



Task 6:



1. **Conclusions**

After completing this assignment, I have learned how to use java streams and lambda expressions. In a future development of this application one can extend the functionality to accept different types of input.

1. **Bibliography**

[**https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/**](https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/)

[**https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html**](https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html)