

Activities monitoring

UTCN CTI ENGLISH

Muresan Daniel, 30422

Contents

1. Goal of the application
2. Problem analysis
3. Projecting
4. Implementation
5. Results
6. Conclusions
7. Bibliography

1. Goal of the application

The main goal of the application is to analyze the behavior of a person recorded by a set of sensors over a longer period of time. Sensors are a vital component of the contemporary society and they are used everywhere, in every device and by every person who owns any piece of technology. Sensors are very accessible these days and very useful to monitor and automatize everything that surrounds us.

Having a house with lots of sensors may be a good idea from security point of view. But having sensors that focus on the activity of a person can be useful to analyze the behavior, the health and the way a person manages his time. With data from those sensors, a program may create some statistics about the person behavior and maybe help the person to improve his health status or to manage better the time.

This application is meant to analyze such a set of data provided by a set of sensors and to compute some statistics based on the activities of a person inside his house over few days. The application will use streams to filter the data from the sensors and obtain the required statistics.

2. Problem analysis

The historical log of the person's activity is stored as tuples (start time, end time and activity label) where start time and end time represent the date and time 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 from the log is spread over several days as many entries in a text file named Activities.txt.

There were no assumptions to make for this application, only the fact that all the entries from the text file Activities.txt containing the log of events should

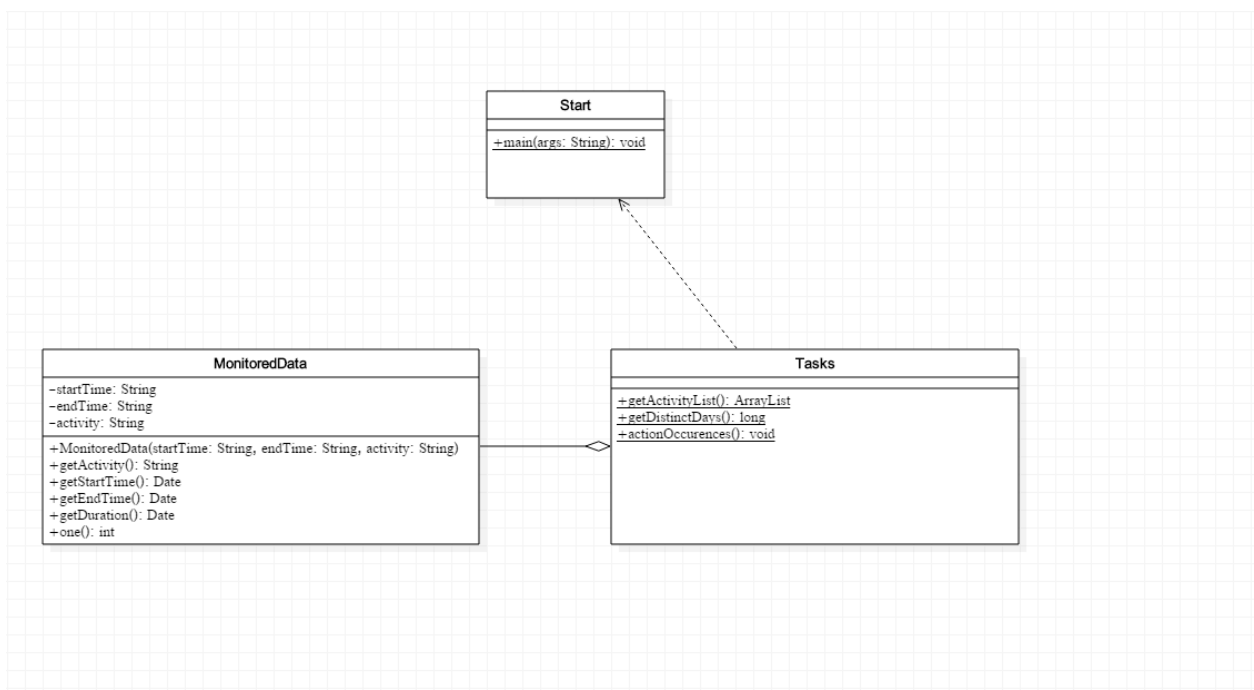
have the same format but since the data is provided by a set of sensors this is a straight forward requirement.

There were no use cases for this application because when the application is launched it will perform the required tasks, for analyzing the activity log, and display the results in console or in a text file, depending on the task. So the user won't have too many options regarding the functionality of the application, just launch the application and receive the results.

3. Projecting

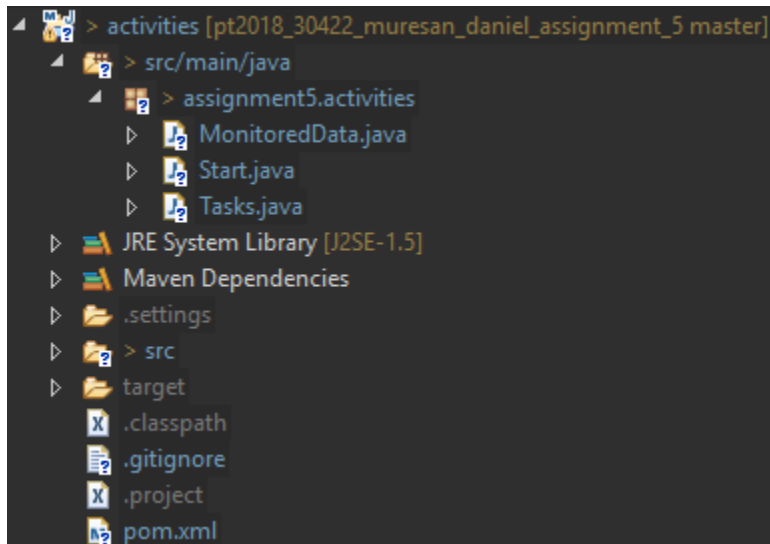
The classes I used for developing this application are the following ones : MonitoredData to model the entries from the log of activities, Tasks with the methods that perform the required tasks and Start with the main method where the application is launched and the tasks are performed.

All the classes and the relationships between them are displayed in the next class diagram.



For this application I used just one package assignment5.activities which contains the classes mentioned above.

The structure of packages is presented in the next image.



4. Implementation

MonitoredData class

Is the class that models the activities from the log with data. This class has few attributes: startTime which is a string where the start time of an activity from the log is saved, endTime is also a string where the end time of an activity from the log is saved and activity which is also a string with the label of the activity from the log. The class has a constructor with three parameters all strings, corresponding to the attributes of the class. There is also a method which returns the start time from the attribute of the class as a Date type and another method which returns the end time in the same manner. Another method is used to compute and return the duration of an activity as a Date type too. One last method just returns the integer 1 and is used in a stream for a specific task.

Tasks class

Is the class where the requirements are fulfilled with some static methods. The first static method is `getActivityList()` which creates and returns an Array List with objects of type `MonitoredData` representing all the entries in the log file. In order to create those objects the method uses a stream to get the file line by line, split the lines into three strings corresponding to the start time, end time and activity label, create and add new objects with the strings in the Array List which will be returned as a result.

Another static method called `getDistinctDays()` counts and returns a long which represents the number of distinct days from the log. This method also uses streams. Firstly it creates an Array List with all the start times from the log file by using a stream and then on that list creates another stream which is used to count the distinct days.

The last static method is called `actionOccurrences()` and its purpose is to create a map with each activity from the log as key and the number of occurrences as values and then print the contents of the map into a text file. Firstly this method uses `getActivityList()` method to obtain the Array List with all the activities from the log and then uses a stream on that Array List to map the activities and the number of occurrences respectively. After that a print writer is used to overwrite the contents of a text file with the contents of the previously created map.

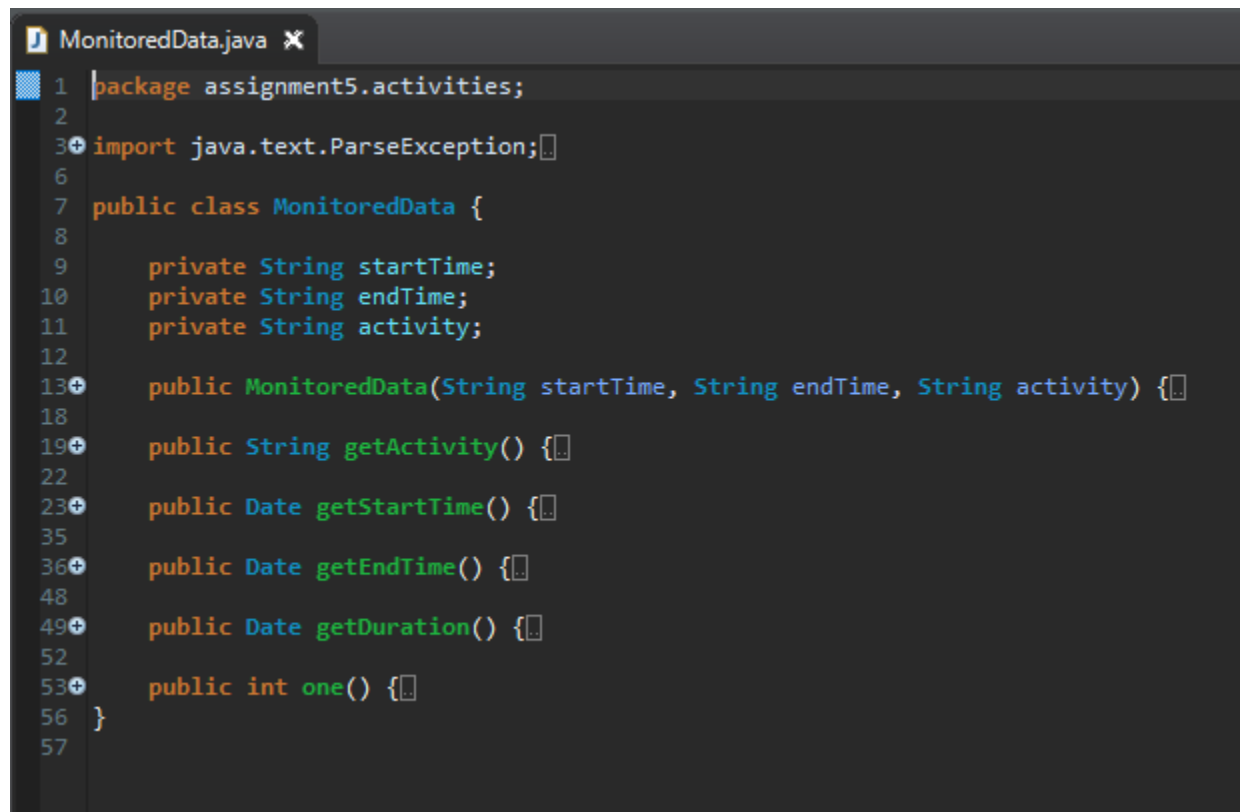
Start class

Is the class which contains the `main()` method used to launch the application. There all the static methods from the `Tasks` class are called in order to obtain the results required by the problem specification.

5. Results

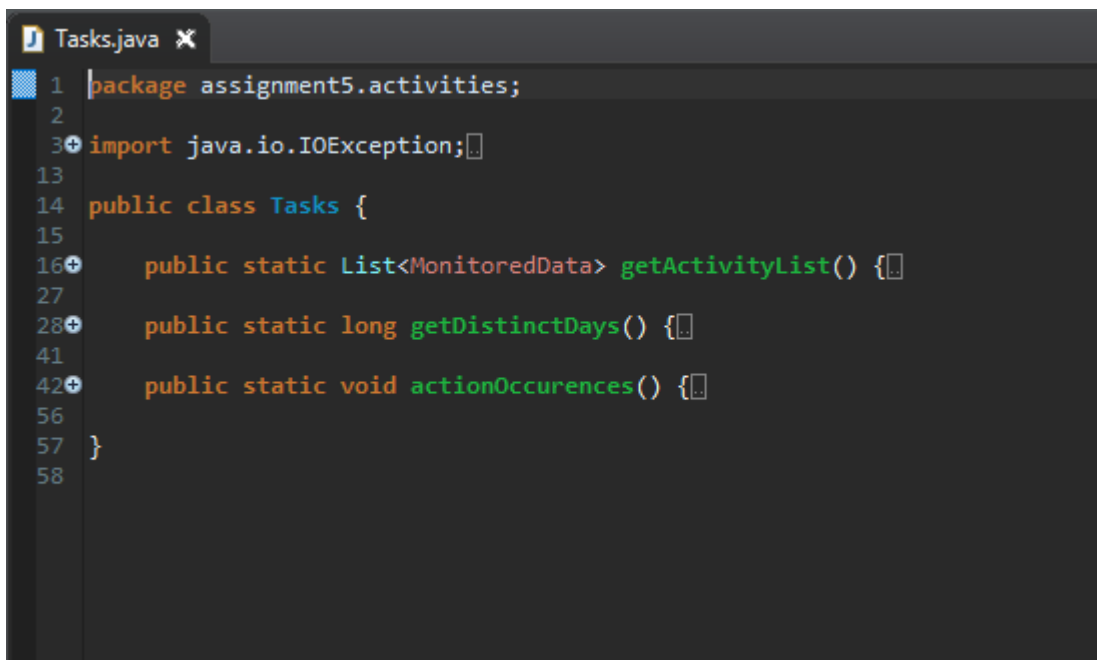
In this section I will present few screenshots with the final application containing code and results of the requirements.

In the first image is the structure of the class MonitoredData.



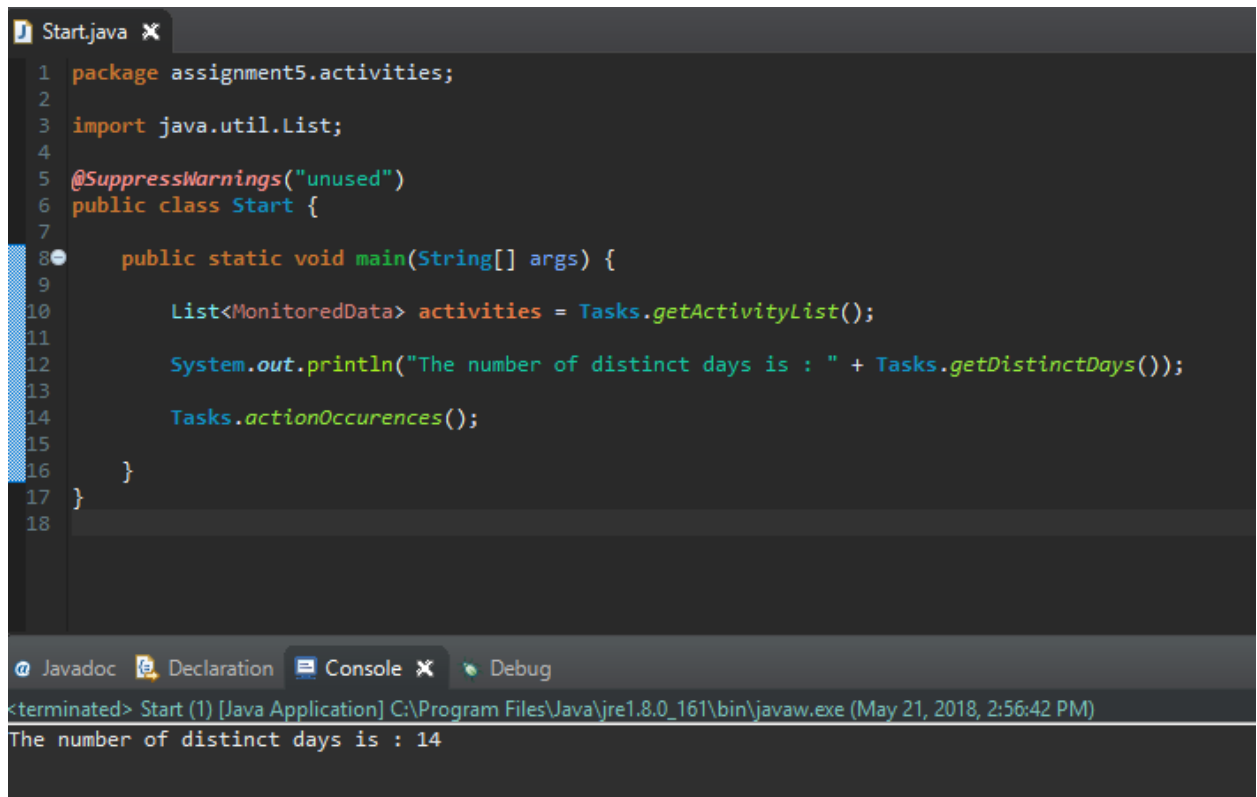
```
MonitoredData.java X
1 package assignment5.activities;
2
3 import java.text.ParseException;
4
5
6
7 public class MonitoredData {
8
9     private String startTime;
10    private String endTime;
11    private String activity;
12
13    public MonitoredData(String startTime, String endTime, String activity) {}
14
15
16    public String getActivity() {}
17
18
19    public Date getStartTime() {}
20
21
22    public Date getEndTime() {}
23
24
25    public Date getDuration() {}
26
27
28    public int one() {}
29 }
30
31
```

The next image is the structure of the Tasks class.



```
Tasks.java X
1 package assignment5.activities;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12
13 public class Tasks {
14
15
16    public static List<MonitoredData> getActivityList() {}
17
18
19    public static long getDistinctDays() {}
20
21
22    public static void actionOccurences() {}
23
24
25 }
26
27
```

The next image is the main() method from class Start and the result of the getDistinctDays() method.



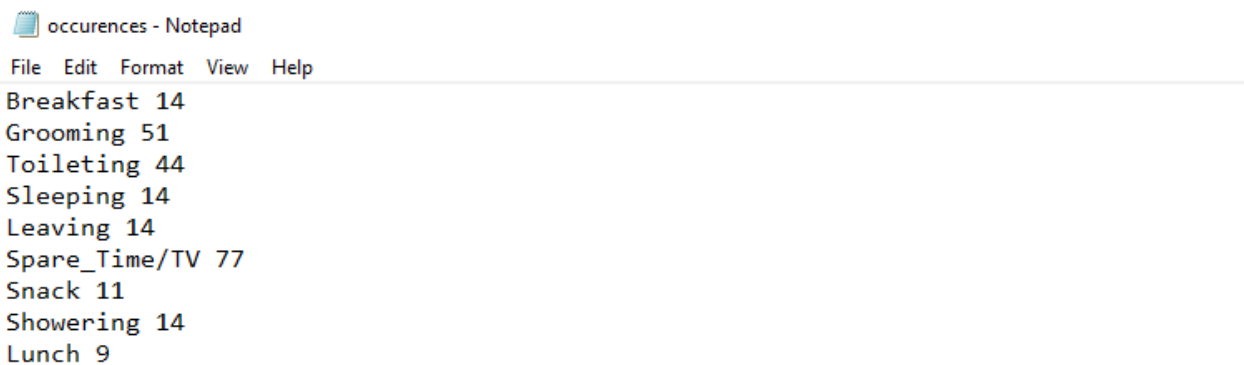
The screenshot shows an IDE window titled 'Start.java'. The code is as follows:

```
1 package assignment5.activities;
2
3 import java.util.List;
4
5 @SuppressWarnings("unused")
6 public class Start {
7
8     public static void main(String[] args) {
9
10         List<MonitoredData> activities = Tasks.getActivityList();
11
12         System.out.println("The number of distinct days is : " + Tasks.getDistinctDays());
13
14         Tasks.actionOccurences();
15
16     }
17 }
18
```

Below the code editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> Start (1) [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (May 21, 2018, 2:56:42 PM)
The number of distinct days is : 14
```

The last image is the result saved in a text file from the method actionOccurences().



The screenshot shows a Notepad window titled 'occurences - Notepad'. The text content is as follows:

```
File Edit Format View Help
Breakfast 14
Grooming 51
Toileting 44
Sleeping 14
Leaving 14
Spare_Time/TV 77
Snack 11
Showering 14
Lunch 9
```


6. **Conclusions**

By developing this application I learnt to use Java8 features like streams and lambda expressions. I observed that this type of code is more readable and maybe SQL like, the operations form streams being similar in a way with the queries form databases. I did not manage to fulfill all the tasks required in the problem specification which involved more advanced stream operations.

7. **Bibliography**

<http://www.mkyong.com/tutorials/java-date-time-tutorials/>

<https://www.youtube.com/watch?v=0bHCxjkkU0s&t=405s>

<https://medium.freecodecamp.org/learn-these-4-things-and-working-with-lambda-expressions-b0ab36e0fffc>