Dan Mitrea
Group 30421
Year of study : 2016-2017

Technical University of Cluj Napoca

# Programming Techniques
# - Assignment 5 -
# Stream processing using lambda expressions

# Contents

## Assignment Objective

A smart house features a set of sensors that may be used to record the behavior of a person living in the house. The historical log of the person's activity is stored as tuples (startTime, endTime, activityLabel), where startTime and endTime 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 attached log file Activities.txt contains a set of activity records over a certain period of time.

Define a class MonitoredData having startTime, endTime and activityLabel as instance variables and read the input file data into the data structure monitoredData of type List<MonitoredData>. Using stream processing techniques and lambda expressions introduced by Java 8, write the following set of short programs for processing the monitoredData.

1. Count the distinct days that appear in the monitoring data.
2. Determine a map of type <String, Integer> that maps to each distinct action type the number of occurrences in the log. Write the resulting map into a text file.
3. Generates a data structure of type Map<Integer, Map<String, Integer>> that contains the activity count for each day of the log (task number 2 applied for each day of the log) and writes the result in a text file.
4. Determine a data structure of the form Map<String, DateTime> that maps for each activity the total duration computed over the monitoring period. Filter the activities with total duration larger than 10 hours. Write the result in a text file.
5. Filter the activities that have 90% of the monitoring samples with duration less than 5 minutes, collect the results in a List<String> containing only the distinct activity names and write the result in a text file.

## Problem Analysis

Nowadays, technology advances at a rate that has never been seen before. Everything has sensors, cameras, or devices that gather data. This data is essential because it ensures our devices work as they should, and that our quality of life is as good as possible. As a result, these big chunks of data need to be processed as fast and efficient as possible, so that they can yield the needed information. So, stream processing techniques have been invented, so that we can treat this continuous flow of data as a stream, without having the need to store it anywhere, only when we truly need it.

Stream processing techniques, as I have said, do not involve storing the data before processing it, but instead, they treat the information as a stream and make changes on it on the go. The information enters the device, it is processed in a predefined way, and then exits the device. This way, these techniques make it extremely efficient to process data from devices that gather information, so that we can later on use only what we need from it. By being efficient, it allows technologies like Internet of Things to become a part of our life.

After a quick analysis of the problem and the input file, it is obvious that ve need to store the time stamps in a date format for a better representation.

## Modelling

Modelling is the process of breaking down a complex problem statement into smaller pieces that can be modeled and can become much easier to be understood. Such process helps for a much clearer design and implementation of the solution for the problem. We stated before that time input should be stored as a date format in order for a better representation of the time stamps and in this way we can have different levels of granularity, from years to even seconds. Such feature allows for more queries to be performed. The activity label is used to differentiate between types of activities performed by the user of the application. At the beginning, we have to process all the data from the input into a data structure, using streams for an improved

efficiency. After having the whole input in the data structure, we can easily pass it to different queries, since they are performed on that data structure. This approach is very convenient since we can read the whole input only at the start of application and then perform our statistics queries on the internal representation of the input.

## Scenarios. Use cases

The user has the option to do a multitude of operations on the log of events available. In the following lines I will present you only one scenario.

1. The user starts the application.
2. He / she chooses to check the number of distinct days that have been logged.
3. The user presses the specific button, and then he will see on the big text field the displayed result.
4. There are no possible error cases, because the program will count just the number of distinct days, in this case.

## Packages:

I have used the standard MVC design pattern (Model - View - Controller). The packages have the role of separating relevant data and to better encapsulate information. Their design is as follows:

1. The classes contained by the “model” package have the role to model a real object and its behaviours into our application.
2. The class contained by the “view” deals with the graphical user interface, or simpler said, manages what the user can see and do with the application.
3. The class contained by the “controller” controls the data flow of the model classes and updates the view whenever necessary.

## Java Streams

The Streams API in Java is a powerful tool of functional programming. A stream is like a sequence of elements on which we can perform various

operations. The collections from Java Collection framework now support methods as stream or parallel stream which create from that collection, a stream. Moreover, the pipeline set of operations can be viewed as a query on a database. If we use parallelStream on a multi-core architecture, we are able to process that stream even faster and obtain better results.

One of the biggest advantages of the stream is that they do not have to store any data, unlike the collections which store their elements in the memory. Streams take the data continuously and process it. Streams are best used in functional programming because they do not modify the source data. By passing a lambda expression to the stream methods, we only specify what operations should be performed on the input data.

## Lambda expressions

Lambda expressions are used to describe an anonymous function. They were introduced in Java 8 and considered one of the major assets of Java 8. A lambda expression is characterized by the following syntax: parameter -> expression body. A Java lambda expression is a function which can be created and can be passed as an object. The lambda expressions pave the way for multi paradigm programming where object oriented programming interfaces with the powers of functional programming. Lambda expressions are extensively used with Functional Interfaces. A Functional Interface is an interface with only one method. Any lambda expression can be assigned to one of the interface called Functional Interface.

## Classes

## MonitoredData class

This is the class chosen to model the individual logs of events. It contains three instance variables : start time, end time and activity label. While the activity label is represented as a string and it names the actual activity the person performs, the other instance variables represent the time of the beggining and ending of the activity. I have chosen to represent this time as a “ LocalDateTime ” because it is the most straightforward one in Java 8. Also, the Date class is deprecated and not used anymore in Java development.

## DataManipulation class

This is the main core of the application. It computes all the neccesary computations based on information read from a file and displays the results in files. So, as you might have guessed, the first method is the “ readInput() ” method, which has the purpose to read the information from the file and transform it into a format that is easy to work with. It has no parameters because the location of the input file is fixed. We tokenize each separate data element in order to obtain the three fields that we have defined in the MonitoredData class. For each line from the file, this method will split them into these three data structures and then repeat. Finally, the method returns a list of MonitoredData objects.

The second method in the class has the purpose of counting the distinct days that appear in the activity log of the person. We take the list received as a parameter as a stream, and then we map based on the day of the month of the starting time. Finally, we add it to a tree set so that we will not have any duplicates. In the end, we compute the size of the tree set and this will be the number of distinct days needed.

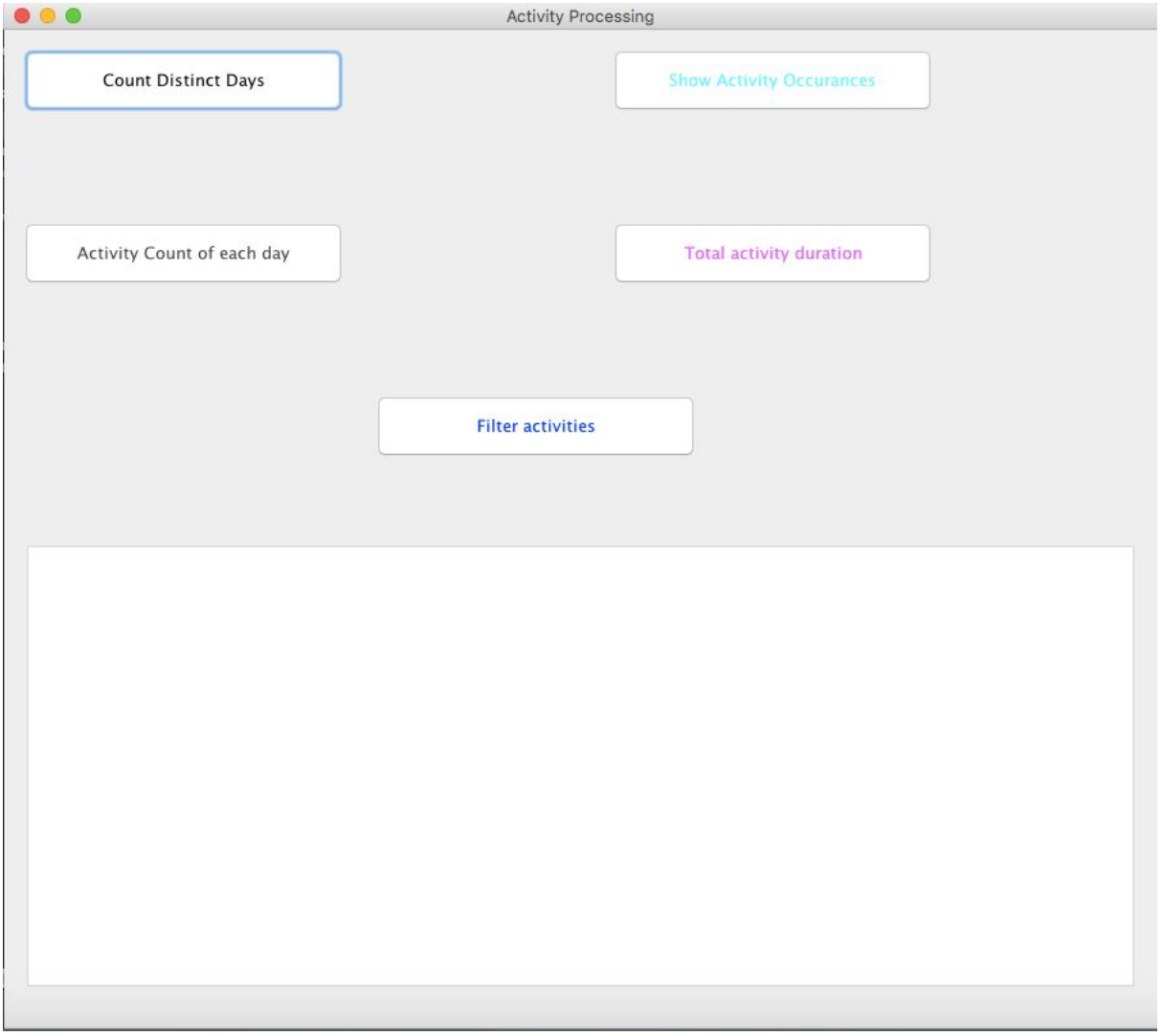
## GUI class:

This is the class that implements the graphical user interface. It has the purpose to make it easy for the user to perform the required operations on the information that is held in the bank. Also, it must help the user to his job as easily as possible, and in a friendly way. It might not have the best design, nor the most fancy one, but it gets the job done. Since the actual functionality of the application is not that complex, the user interface contains only some well defined buttons that will tell the user what can he see after pressing each one of them. The text field is not editable, since it is used just for displaying purposes.

## Main class:

This is the main class, the one that contains the “ public static void main(String [] args) ” method. Since all the operations are done in the graphical user interface class, this main method just creates a new object of type “ GUI “. Everything is done in the constructor of the class “ GUI ”, so, as a result, just creating an object of this kind is enough for the application to run as it should.

## User interface:




As I have previously mentioned, the graphical user interface does not have the best design, but it gets the job done. It is intuitive for the user and tell him enough in order for him / her to use the application with no user manual or something similar.

With nothing special, the interface contains just 5 buttons, implementes as JButtons from the java swing and awt libraries. The frame can be exited pressing the exit button. Each button has an action listener, so that after being pressed, it will not only make all the required computations for that specific method, but also display the results in the text field from below. As a result, the application requires no user manual or special skill in order to be used.

## Implementation and testing

For the implementation, I have extensively read the provided documentations and tutorials on streams and lambda expressions. As they both were concepts new to me, it took some time to get used to them and understand properly how to use them. After procceding with the implementation of the program, I have tested each method separately, before actually designing the graphical user interface. Even though each method required only a few lines of code and one may say it is easy to do such a thing, in fact, it was more difficult to write that line of code than writing maybe tens or hundreds of lines of code that you know.

## Results

The result is, after many hours of coding and debugging, an application that has a great applicability in our society, and also an application that uses some programming techniques that are very helpful and important, that I did not know before. Even though the application has some flaws, I believe that with some minor improvements, it could be used in real life.

## Conclusions

## What did I learn

After finalizing this application, I can sincerely say that I have learned quite a lot. Besides refining my skill in writing java code, which is a skill I will always want to improve, the main thing that I have learned is these new programming techniques. I learned what streams and lambda expressions are and how important are they for modern programming, of data and information that can be used not only nowadays, but also in the future.

I am sure that I can use what I learned after writing this application in many future projects, and that I am one step closer to my goal, which is to become the best programmer I can be.

## Further possibilities for improvement

Although the application does all the required computations on the log of events, some improvements can be made. A great development of this project would be to use real sensors that gather information in real time and try to

process the data using streams techniques. It would be really interesting to see the difference between how fast the Stream API in Java is and other API in other languages, like C++ or Python, languages used in such type of applications.

## Bibliography


1. http://stackoverflow.com
2. https://wordcounter.net
3. https://creately.com