Assignment 2

Team number: 41
Team members

Name	Student Nr.	Email
Nikolay Filipov	2691580	n.r.filipov@student.vu.nl
Asim Manzaij	2671198	a.k.manzaij@student.vu.nl
Alejandro Pascual	2708337	a.pascualpintado@student.vu.nl
Andres Latorre	2708338	a.latorremagaz@student.vu.nl

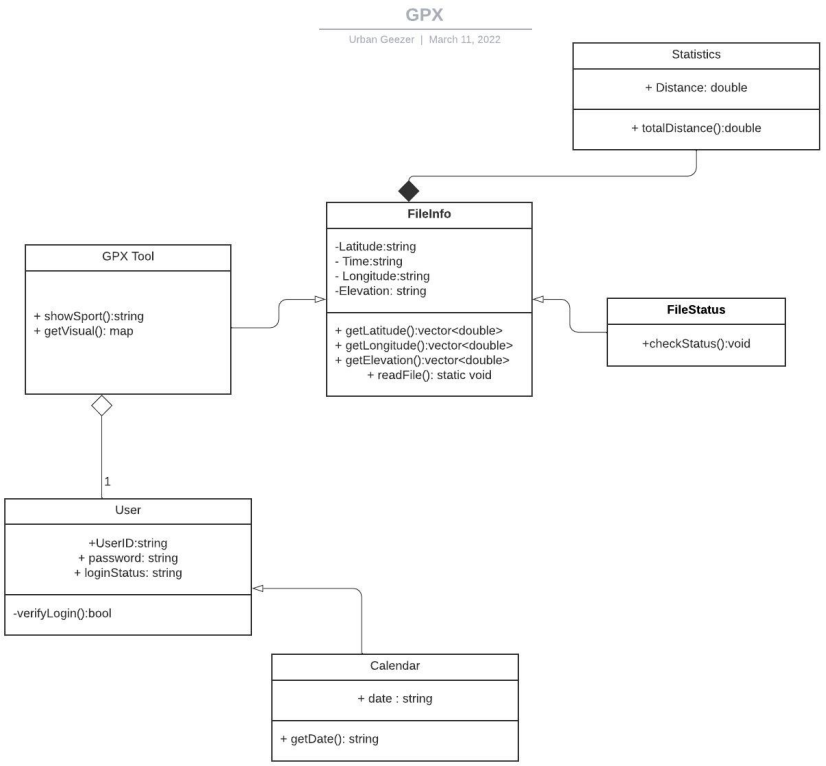
Implemented feature

ID	Short name	Description
F1	Distance	Calculates the total distance between all the points and shows the user.

Used modelling tool: LucidChart

Class diagram

Author(s): Nikolay Filipov and Asim Manzaij



The main class in our program is the FileInfo which contains the latitude, longitude, time, and elevation as attributes. These three are given as a string and are made public so other classes can use them. The operations of getLatitude, getLongitude, and getElevation() are given as vectors and are made private as they cannot be accessed by other classes. Also, the operation called readFile() is given as a static void and it reads and parses the file.

The next class we have is the GPX Tool which is connected to the GPX File with inheritance as it inherits the information first provided in the GPX File. The GPX Tool then has 2 operations, one of which is getVisual() which shows a map. The other operation is showSport() which is a string and outputs the sport. Both of these operations are made public so they can be accessed by other classes like the user class.

From the GPX Tool there is an aggregation relationship with the next class which is User. Aggregation means that one class is loosely associated with the other. The attributes of the user class are the userID, the user password, and the login status. All of these attributes are given as strings. The User class also has one operation which is to verify the login status of the user. This is given as a boolean function and inputs true if the user is logged in and false if the user is not logged in. In this example, the user can still login without the GPX Tool necessarily being there. Per GPX Tool , we only have one user who's information will be shown.

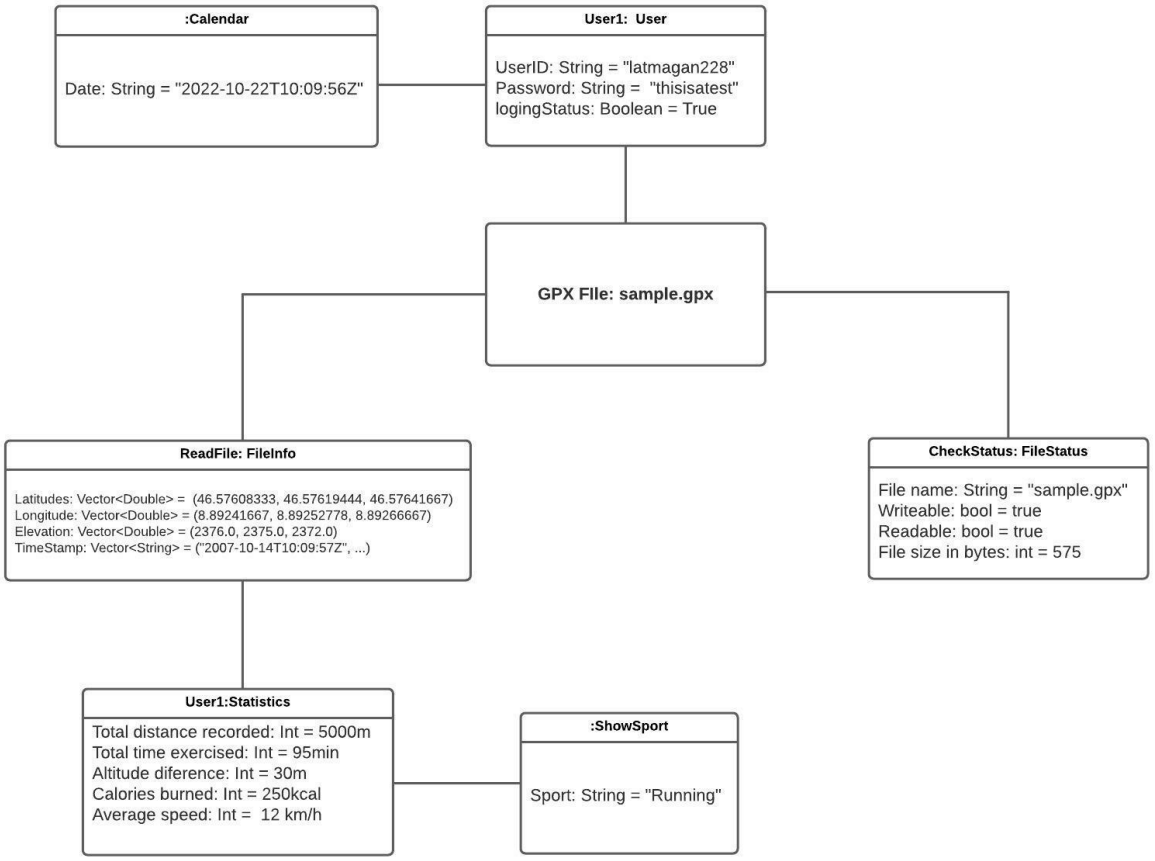
The next class we have is the calendar class which inherits the information from the user and shows a date to the user. It is connected to the user class using inheritance. It also has one attribute of its own which in this case is the dates the user logged into the app. This attribute is given as a string. The calendar class also has one operation which is getDate() using a string.

The next class is the statistics class. The class calculates the total distance of the track. It is connected to the GPX file class using composition meaning that if the GPX class is destroyed, the statistics class will also be destroyed. Within the statistics class, we have an attribute of distance characterised as a double and an operation of totalDistance() also as a double.

The final class is the FileStatus class which has one method. The method is to determine the status of the file and to check whether it is readable. This class is connected to the GPX file class using inheritance.

Object diagram

Author(s): Alex Pascual and Andres Latorre

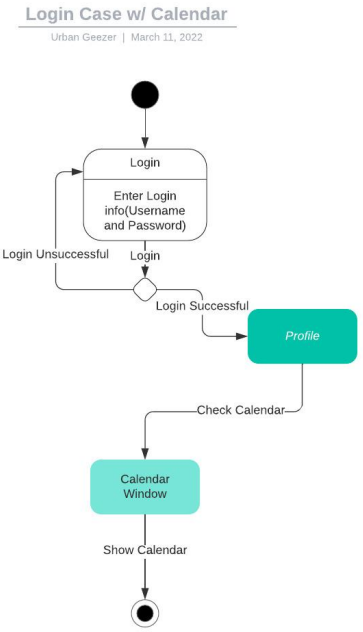


Our object diagram shows a snapshot of how our program would run when tested with a sample gpx file. In this example we can see how the different classes are related. For example, we observe the clear relation between the class Statistics() and the class Showsport() where in correlation to the data obtained we can determine the sport performed by the user. Fileinfo() and FileStatus() obtain their results directly from the sample.gpx file.

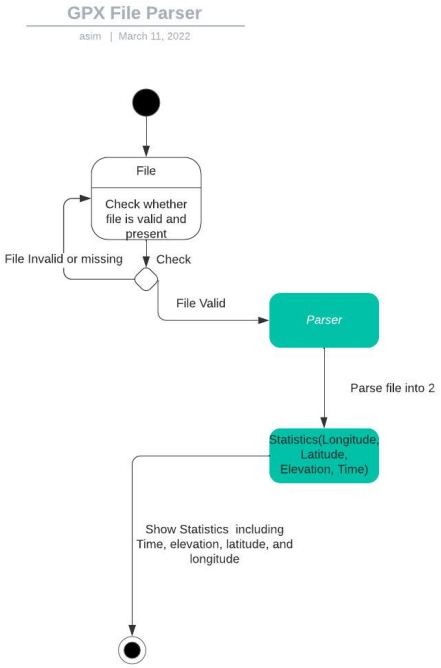
Maximum number of pages for this section: 1

State machine diagrams

Author(s): Nikolay Filipov and Asim Manzaij



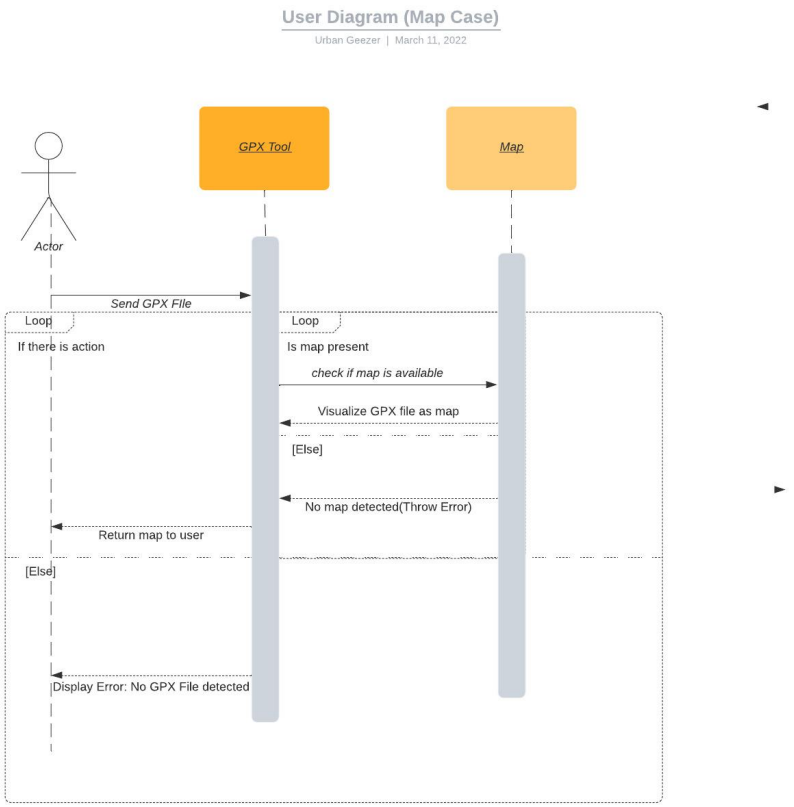
The User activity in order to see the calendar starts with the process of the user login. It is part of the user class. If the user is unable to login, it takes them back to the login screen. If the login is successful it sends the user to the next state which is the user profile. From the user profile, the user can check their calendar and it sends them to the Calendar Window state. The system then shows the calendar and this system of events is then finished.



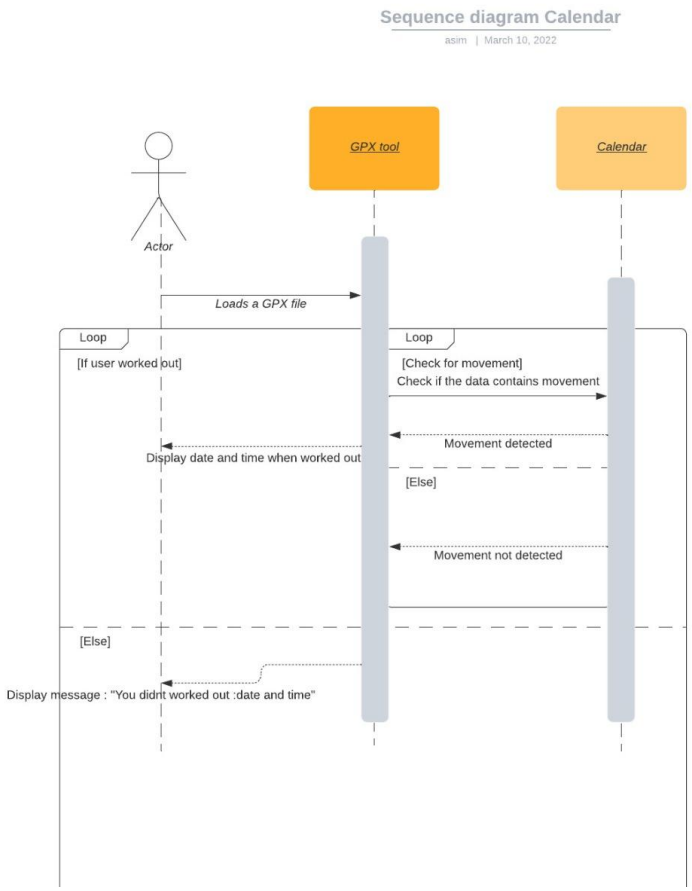
This state machine diagram begins by checking if a file is valid or not. It is part of the GPX parser class. If the file is invalid, the system should return to the first step and request a new, valid file. If the file is valid, the system should go to the Parser state. There, the parser should parse the file to obtain the most important information(the latitude, longitude and elevation) going into one state. That state will then output the statistics in the terminal to the user.

Sequence diagrams

Author(s): Nikolay Filipov and Asim Manzaij



- The first step in a sequence diagram is the actor which in our case is the user that uses our GPX tool. We then have two objects which here are the GPX Tool and the map. The user loads a GPX file which the tool will check if it is a valid GPX file. With this the diagram enters into a loop with the condition of whether the file is valid or not. In this case valid means, if there is enough data to represent a map or if the GPX file is not corrupted. If the GPX file is valid then the GPX tool communicates with the map object and requests a visual of the file. If that is successful the map sends the GPX Tool the visual, which can then be accessed by the user. However, if there is no map the map will communicate with the tool that no map is present. The tool will then throw an error and skip out of the loop to display an error message to the user. If any step of this process is unsuccessful then the GPX Tool will display an error message to the user.



- The first step in a sequence diagram is the actor which in our case is the user that uses our GPX tool. We then have two objects which here are the GPX Tool and the Calendar. The user loads a GPX file which the tool will check if it is a valid GPX file. With this the diagram enters into a loop with the condition of whether the file is valid or not. In this case valid means, if the user worked out that day. If the GPX file is valid then the GPX tool communicates with the calendar object and checks whether or not movement was registered on that day. If that is successful the calendar sends the GPX Tool the date and time of the workout, which can then be accessed by the user. However, if there is no data on that day, the Calendar will communicate with the GPX Tool that no movement was registered. The tool will then throw an error and skip out of the loop to display an error message to the user. If any step of this process is unsuccessful then the GPX Tool will display an error message to the user.

Implementation

Author(s): Andres Latorre, Alejandro Pascual

The strategy we followed in order to move from the UML models to the implementation was at first to create simple diagrams with the basic information we wanted to include in our implementation. Afterwards, we started working on the implementation of these main aspects. When working on this, we realised many other features we needed to include and that would be useful to achieve a better result. Finally, we updated the changes into the UML models so that they would match our current implementation.

In order to extract the needed information from the gpx file we parsed the file. Firstly, we read the gpx file as a string and then we split the string in two as we wanted the information contained between “<trkseg>” and “</trkseg>“. Then, we used regex to split the string into a string array to obtain an array which contained each waypoint starting with "<trkpt". Finally, we extracted each value of latitude, longitude and elevation and stored the values in an ArrayList. Another important part of our implementation was to manage to print the total distance of the track performed. We managed this by implementing a mathematical formula that calculates the distance between two waypoints using latitude and longitude. After we added each distance to calculate the total distance.

Main java class: src/main/java/softwaredesign/Main.java
GithubUrl:
https://github.com/latmagan228/Software-Design/blob/74e4484d35fc7187ba2edd0744a21ecc4eeee0d1/src/main/java/softwaredesign/Main.java

Jar File: out/artifacts/software_design_vu_2020_jar/software-design-vu-2020.jar
GithubUrl:
https://github.com/latmagan228/Software-Design/blob/74e4484d35fc7187ba2edd0744a21ecc4eeee0d1/out/artifacts/software_design_vu_2020_jar/software-design-vu-2020.jar

YouTube Video Url:
https://youtu.be/xQEPpjxfFU4

Time logs

<Copy-paste here a screenshot of your time logs - a template for the table is available on Canvas>

Team number	41		
Member	**Activity**	**Week number**	**Hours**
Nikolay Filipov	Merging Github and Installation	1	1
Andres Latorre	Merging Github and Installation	1	1
Asim Manzaij	Merging Github and Installation	1	1
Alex Pascual	Merging Github and Installation	1	1
Nikolay Filipov	Class Diagram	2	2
Andres Latorre	Implementation	2	2
Asim Manzaij	Class Diagram	2	2
Alex Pascual	Implementation	2	2
Nikolay Filipov	Sequence and State Diagram	3	4
Andres Latorre	Implementation	3	5
Asim Manzaij	Sequence and State Diagram	3	4
Alex Pascual	Implementation	3	5
		Total	**30**