****

**Java Project**

**Car Meter**

**Under supervision of Eng.: Eman Hesham**

**Abdullah Hanafy**

**Ahmed Sherif**

**Asmaa Ebrahim**

**Asmaa Saied**

**Mostafa Ahmed**

**Overview**

The project is about a desktop application that allows the user to open a map and start tracking his location on this map.

It also shows the user the speed he is moving with by changing the value of the speed in a speedometer that appears in the GUI.

The user also has the ability to save his trip after finishing it by pressing the button “End trip” the application asks the user if he wants to save the trip he has just finished or not.

The user can find the saved trips by pressing the button “View Saved Trips” that shows the user a list of hyperlinks with the names of all trips he saved.

When the user press on one of the hyperlinks it shows the user a map that has two marks one of the start point and the other of the end one.

There also a feature that when the user exceeds a certain speed the application turns on an alarm to get the user attention which might prevent an accident or car crash.

The user also has the ability to delete saved trips by pressing the button “Delete all trips”.

If the GPS module has not been turn on yet and the user pressed “Start” button a message is sent to the user to tell him that the GPS has not started yet and he has to for seconds.

**The GUI**

* **Speedometer:**
  + We Used “Medusa gauges and clocks” library to get the shape of the speedometer and to set its value in a thread while the program running.
* **Map**
  + عبدالله هيكتب هنا
* **Initial scene**
  + The initial scene contains the map, the speedometer, “Start” button and “View Trips” button.
* **“View trips” scene**
  + Contains a list of hyperlinks that does not exceed 5 each hyperlink is a name of a saved trip and by pressing on one of them it shows a map with two marks one of the starting point and the other one is the end point of the trip.
* **“Save trip” scene**
  + Contains a text field that the user should enter the name of the trip in, “Save” button the user presses on it if he wants to save the trip and “Cancel” button if he does not want to save the trip.
* **Error messages dialogs**
  + عبدالله هيكتب هنا

**Serial Communication:**

In this part, the whole focus is on how to get the GPS data so we can show and use it in our application, and that will be through establishing a serial communication between the Mobile phone/GPS module and our application.

1. GPS using Mobile phone:

* Install Share GPS output on your mobile phone.
* Establishing a serial communication could be through USB or Bluetooth, we tried USB but we found that Bluetooth is more easy and accurate.
* Using Bluetooth, we can make the mobile and the laptop paring, and that will create serial ports for both. SerialCommunication.java will sense the mobile serial port and the serial thread will be started.
* Now, we can receive data from GPS.

1. GPS Module:

The data we are interested in are: the speed, the latitude and the longitude. The GPS data sentence are in NMEA sentence structure.

There are three sentence we got from the GPS:

*The First one:*

**“$GPGGA,181908.00,3404.7041778,N,07044.3966270,W,4,13,1.00,495.144,M,29.200,M,0.10,0000\*40”**

1. *GPGGA* --> represent GPS and it might be GL which denotes GLONASS (Global Navigation Satellite System)
2. *181908.00* --> Time stamp for UTC (Coordinated Universal Time) 18 hour 19 minute and 08 second , for Egypt +2 which means 20h 19m 08s
3. *3404.7041778,N* --> is the latitude in DMM.MMMMM format in degrees, minutes and decimal minutes (34 degree 04.7041778 N)
4. *07044.3966270,W* --> is the longitude (70 deg 44.3966270 W)
5. *4* --> position of quality and vary from 0 to 8: 0= invalid 1=GPS fix( SPS) 2=DGPS fix 3=PPs fix 4= Real Time Kinematic (Centimeter precicion) 5=float Real Time Kinematic (decimeter precicion) 6=estimated (dead reckoning 2.3 feature) 7=Manual input mode 8=Simulation mode
6. *13* --> denotes number of satellites used in the coordinate
7. *1.00* --> denotes the HDOP (horizontal dilution of position)
8. *495.144,M* --> altitud, meters, above sea level
9. *29.200,M* --> Height of geoid separation (geoid means ocean leve) subtract this from the altitude of the antenna to arrive at the height above Ellipsoid
10. *0.10* --> time in seconds since last DGPS update
11. *0000* --> DGPS station ID number
12. *40* --> checksum data.

*The Second one:*

**“\**$GPGLL,4916.45,N,12311.12,W,225444,A,1D”***

1. *GLL* --> Geographic position, Latitude and Longitude
2. *4916.45,N* --> Latitude 49 deg. 16.45 min. North
3. *12311.12,W* --> Longitude 123 deg. 11.12 min. West
4. *225444* --> Fix taken at 22:54:44 UTC
5. *A* --> Data Active or V (void)
6. *\*1D* --> checksum data

*The Third one:*

**“$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W\*6A”**

1. *RMC* --> Recommended Minimum Sentence C
2. *123519* --> Fix taken at 12:35:19 UTC (convert according to time zone, for Egypt UTC+2)
3. *A* --> Navigation receiver warning A = OK (means that you are getting a signal and things are working), V = warning (means you do not have a signal)
4. *4807.038,N* --> Latitude 48 deg 07.038' minutes Northern Hemishpere
5. *01131.000,E* --> Longitude 11 deg 31.000' minutes in Eastern Hemisphere
6. *022.4* --> Speed over the ground in knots
7. *084.4* --> Track angle in degrees True
8. *230394* --> Date - 23rd of March 1994 (Day , Month , Year)
9. *003.1,W* --> Magnetic Variation 3.1 deg West
10. *\*6A* --> The checksum data & always begins with \*

**The Audio Package**

This is used to implement the audio alarm if the speed exceeded the recommended speed above the safety limit

First using the file class: The FileSystem object representing the platform's local file system.

Second using media class: Media class represents a media resource. It is instantiated from the string form of a source URI. Information about the media such as duration, metadata, tracks, and video resolution may be obtained from a Media instance.

Then using MediaPlayer class The MediaPlayer class provides the controls for playing media. It is used in combination with the {@link Media} and {@link MediaView} classes to display and control media playback.

And for testing purpose A simple GUI was used to test the playing and pausing and stopping the audio from playing, so a play pause Button was used with the name “play audio”.

Then implementing a Flag to be used to check the status of the audio.

Finally using the default constructor for initialization of each object reference

After this simple test verified its functionality a merge is implemented to CarMeter.java