Author: Gary Mac Elhinney

Last Edit: 23/01/2016

==About==========

GPSMaster is a .GPX file analyser. Most modern running/cycling apps generate these files when you do an activity.

The program reads through the file and generates the same data that the app it came from does. This was done as a project aside from college. I wanted to gain an understanding as to how the app I've been using for two years for running works and for this reason I made my own simplified version.

==How to Run==========

In order to run this program you'll need to run a .jar file through command line and pass the file name you wish to analyse as a command line argument (java -jar GPSMaster.jar "File.gpx"). Unfortunately I was unable to send you the .jar due to Google mail protection but the .java files will be sent so you can make the .jar file from them.

==Looking at the code==========

The main file in this project is GPSMaster.java. Below is a brief description as to what the major methods of the program do.

readData()

This method is responsible for looking at each <trkpt> </trkpt> segment and storing the data within these brackets in “GlobalPosition Objects” (line 95) such as elevation, latitude and longitude. These objects will later be vital in calculating running distances and running altitude loss/gain. It also does several corrupted data checks to make sure the data in the file is of the correct format.

totalDistanceAndTimes()

This is the longest method in the program. Moreover there’s a lot going on in this method and you’ll notice that there’s a lot of variables. That’s because I’ve had to keep track of both mile and kilometre distances, times and elevations separately in order to create their independent “Split Objects” (line 224).

There’s a couple of nifty little algorithms in here that do pretty awesome things like getting projected split times(line 302) and readjusting raw split times(line 314) to be more accurate(because GPX files don’t create <trkpt> </trkpt> segments at exact kilometres/miles from your starting point).

As a result I got quiet carried away with this method and tried to make the splits as accurate as possible (I have plans to go even further with this accuracy too!)

writeInfoToFile()

This method using an Iterable object to cycle through the Split objects and create two strings, one for kilometre splits and the other for mile splits. These strings are then written to a .CSV file as output to the program. On a successful run of the program a single string is printed to the console simply stating that the program ran correctly.

==Looking at the output file==========

The output file generated will list all the splits for the .GPX file in both Kilometres and Miles. At the bottom of the file are some basic overall statistics such as total distance, average speed etc.

Also note that the last splits of each split table have a (P) symbol beside the paces and average speeds. This means that these are projected split paces/speeds and are therefore only estimates.