NAME: IHEMEBIRI CHINASA MARTINS

STUDENT NO: 2981843

MODULE: MSCC-MD

ASSIGNMENT 3 DOCUMENTATION

BRACKET 8:

This documentation bracket contains description of every method used in my program

Please note that because some methods are repeated in multiple activities with similar functionalities they would only be listed once

MainActivity.java:

- onCreate() : is run when the app is restarted
- setContentView(): Selects the xml file that is sent as an input and displays it on the main activity page of the app
- setOnClickListener(): Add a listener to the buttons on the main activity to reset the game and to trigger the settings activity
- onClick(): Overridden method to handle to handle a button click
- findViewById(): Retrieves the id of a layout when called
- getSystemService(): to retrieve a accessibility manager for giving the user feedback for UI events through the registered event listeners.
- addLoscationListener():Method that will add a location listener to the location manager
- getInstance(): Retrieves the current date and time and stores in a calendar variable
- distanceTo: calculates the distance and speed between two locations
- getTimeinMillis: return the time in milliseconds
- add(): adds a new element to an arraylist
- checkSelfPermissions()L Checks if permissions are granted to the application
- onLoacationChanged(): Listens if user has changed location
- calcSpeedPerKm(): calculated the speed for every km using the distance and time elapsed

- onProvideDisabled(): Checks if GPS has been turned off
- getLastLocation(): Returns the last known gps location recorded
- setText(): Sets the text of a text view
- requestLocationUpdates(): returns location information based on the specified time period in the method

AnalysisListActivity.java:

- onCreate() : is run when the app is restarted
- setContentView(): Selects the xml file that is sent as an input and displays it on the main activity page of the app
- getFilesDir(): Retrieves the internal file directory
- toString(): Returns the value of an object as a string
- exists(): Returns true or false if files exist in a directory
- lists(): Lists out the filenames within a directory
- setOnLongClickListener(): When a run is long pressed its name is stored and will be used for run analysis
- getItemAtPositon(): Returns the item that was long pressed in the list view
- maketext(): Used by toast class to generate text
- readLine(): Buffered Reader makes use of this method to read files line by line
- size(): Returns the length of an array list
- get(): returns the element in an array list
- putStringArrayListExtra(): Used to send array lists across activities
- startActivity(): Triggers a new activity

AnalysisViewActivity.java:

- onCreate() : is run when the app is restarted

- setContentView(): Selects the xml file that is sent as an input and displays it on the main activity page of the app
- getIntent(): Retrieves the intent that started this activity and the values that was sent to it
- updateLineChart(): Method in AnalysisView.java that is called to update the custom view for analysing runs

CustomPoints.java:

- getKm(): Returns the Km of each custom point entry
- getSpeed(): return the speed of each custom point entry

CustomView.java:

- init(): initialises values within the custom view such as array lists, colors etc

- setColor(): Sets the color of a paint object using hex values

- invalidate(): Called to redraw the game whenever changes are made instead of calling the onDraw() method as calling onDraw() could cause a looping error

- getResources(): Helps in retrieving values from resources such string, layout resources, in the custom view it is used to get the dimensions and width of the screen where it is being displayed

- getDisplayMetrics(): A structure describing general information about a display, such as its size, density, and font scaling.

- Min(): Finds the minimum value between two numbers, in the custom view it used to get the minimum value between the width and height, which will then be used to make the view square shaped

- onDraw(): Overridden method that will be called to draw the contents of the custom view

- translate(): Moves the origin (initiallay at (0,0)) of drawing on the canvas to a new location

- drawRect(): Responsible for drawing the squares of the boards after a translate operation is done

- onMeasure(): gets the width and height and forces the custom view to be square

- drawLine(): Draws a line from a start point to an end point based on x y coordinates

- drawText(): Used to construct texts such as x and y axis labels on the custom view

- save(): Saves the current location of the canvas's initial drawing point

- restore(): Restores the canvas to initial position of last save()

- Math.round(): Used to convert Float values to integer values by rounding up the decimals

- updateLineChart(): Accepts two array Lists representing x and y axis and uses its values to plot a line chart

- updateLineChartLive(): Accepts two arguments representing each kilometre covered and time elapsed , this values are used to plot the line chart live

AnalysisView.java:

- init(): initialises values within the custom view such as array lists, colors etc

- setColor(): Sets the color of a paint object using hex values

- invalidate(): Called to redraw the game whenever changes are made instead of calling the onDraw() method as calling onDraw() could cause a looping error

- getResources(): Helps in retrieving values from resources such string, layout resources, in the custom view it is used to get the dimensions and width of the screen where it is being displayed

- getDisplayMetrics(): A structure describing general information about a display, such as its size, density, and font scaling.

- Min(): Finds the minimum value between two numbers, in the custom view it used to get the minimum value between the width and height, which will then be used to make the view square shaped

- onDraw(): Overridden method that will be called to draw the contents of the custom view

- translate(): Moves the origin (initiallay at (0,0)) of drawing on the canvas to a new location

- drawRect(): Responsible for drawing the squares of the boards after a translate operation is done

- onMeasure(): gets the width and height and forces the custom view to be square

- drawLine(): Draws a line from a start point to an end point based on x y coordinates

- drawText(): Used to construct texts such as x and y axis labels on the custom view

- save(): Saves the current location of the canvas's initial drawing point

- restore(): Restores the canvas to initial position of last save()

- Math.round(): Used to convert Float values to integer values by rounding up the decimals

- updateLineChart(): Accepts two array Lists representing two runs and uses their values to plot two line charts for comparison. The line chart uses a fixed y axis of 400Km/h and x axis value of 10kilo metres

BRACKET 9:

This documentation bracket contains description of every data structure used in the program

- Several array lists are used in the program to hold values for x and y coordinates across activities
- The array lists hold primitive data such as Integer, doubles and some custom objects such as Custom Points which represent distance and speed
- In the Custom View two array lists represent the x and y axis, each distance in km and speed and speed in km/h. The x axis will be divided according to the size of the array list holding each km
- The axis will be divided according to the maximum speed recorded in the array list. A line chart of distance against speed will be plotted with these values
- In the analysis custom view. Two array lists representing two runs are used. The array lists are of type string holding the distance and speed information as strings separated by : e.g Km 1: 456 km/h. The strings are then split and used to construct custom point objects of distance and speed
  The custom points are then stored in another array list that holds custom point objects. The custom points array lists are then used to construct the line charts for both runs

BRACKET 10:

This documentation bracket discusses design decisions made to improve the UI as well as the design and mathematics of the custom view used here

DESIGN DECISIONS:

- For simplicity a button two buttons are placed at the top of the screen that are used to start and stop tracking. These buttons are placed clearly at the top as they will be the most frequently used buttons on the app
- A less frequently used button for analysis is placed at the bottom of the custom view
- The y axis can represent speeds up to 400 km/h or more. This would make for a very clogged up y axis if all the values were printed to screen, Instead the corresponding y axis labels are placed at each node representing a kilometre covered in the graph
- A text view at the bottom of main activity will enable a user see the results of a run after they stop tracking
- In the analysis view. Two runs are represented on a single graph. For convenience both runs are coloured differently. Also because the two runs represent different distances and speed, a maximum value for distance covered and max speed is used to compare both runs. The two runs are compared over a distance of 10 km and a max speed of 400 km/h on the y axis

- When the analyse button is clicked on, it triggers a new activity displaying a list of all saved run files. A list view is used for this purpose to make selecting and navigating across the lists of files easy.

CustomView.java:

CUSTOM VIEW DESIGN AND CALCULATIONS:

- The custom view used here is a square
- The square is a rectangle that is forced to be a square by always ensuring the same minimum width is used as the width and height
- In the onMeasure method when the width and height of the display screen is gotten the minimum of the two is gotten to force a square shape custom view, this value is called size
- The x axis is separated by ticks. The ticks are constructed from square primitive shapes and translated to positions of the x axis on the screen
- The x axis is placed at a position size/8 from the bottom of the screen while the y axis is placed at a position (7* size)/8 from the left of the custom view. All texts, lines and shapes within the graph are drawn relative to these two points on the custom view
- Size variable represents the length of sides of the custom view
- The x axis of the custom view is divided according to the number of kilometres on a run, while the y axis is divided according to the maximum speed recorded for each run

AnalysisView.java:

CUSTOM VIEW DESIGN AND CALCULATIONS:

- The custom view used here is a square
- The square is a rectangle that is forced to be a square by always ensuring the same minimum width is used as the width and height
- Because two separate runs are being drawn, a fixed distance and max speed is used to construct the line graph. For this application the two runs are compared over a distance of 10 km and max speed of 400km/h. The x and y axis are dived according to these ranges for comparing the two line charts