**Pseudocode: Task D and E (OOP)**

(Task A, B and C)

// --- Global Variables and Helper Functions (not part of pseudocode for main classes) --- //

Helper function `process\_csv\_data(file\_path)`:

// Loads and processes data from a CSV file. Returns data as a list of dictionaries.

// Helper function `validate\_date\_input()`:

// Prompts user for date input, validates it and returns integers for dd, mm, yyyy

// Helper function `display\_outcomes(data\_frame)`:

// Performs analysis on the data and returns a dictionary with results.

// Helper function `save\_results\_to\_file(results, file\_name)`:

// Saves given analysis results to a file.

// --- Class: HistogramApp --- **CLASS**

**HistogramApp**

// Constructor:

**METHOD \_\_init\_\_(traffic\_data, date)**

// Initialize variables

SET self.traffic\_data = traffic\_data

SET self.date = date

CREATE a Tkinter root window and assign to self.root

SET title of root window to "Traffic Data Histogram - {date}"

SET self.canvas = None

SET self.bar\_width = 15

SET self.padding\_x = 30 SET self.padding\_y = 80

SET self.x\_label\_offset = 25 SET self.y\_label\_offset = 20

SET self.colors = ['lightgreen', 'lightpink']

SET self.histogram\_data = empty dictionary

SET self.max\_count = 0

CALL self.prepare\_histogram\_data()

// Sets up the Tkinter window and canvas

**METHOD setup\_window()**

SET window size to 1200x600

CREATE a canvas widget with white background and assign to self.canvas ADD canvas to the window, allowing it to expand in size

// Prepare Histogram Data

**METHOD prepare\_histogram\_data()**

SET junctions = ["Elm Avenue/Rabbit Road", "Hanley Highway/Westway"]

CREATE empty dictionary hourly\_counts for junctions with default values 0 for each hour (0-23)

// Loop through each traffic data row

FOR EACH row in self.traffic\_data

GET junction name from row

GET time of day from row, default to "00:00"

// If junction is valid and time is valid and formatted as hh:mm

IF junction exists in hourly\_counts AND time\_of\_day contains ":"

GET hour by splitting time\_of\_day and converting the hour part to integer

INCREMENT hourly\_counts for the junction and hour

SET self.histogram\_data to hourly\_counts

SET self.max\_count to the largest count in hourly counts

// Draw Histogram

**METHOD draw\_histogram()**

// Exit if no data

IF self.traffic\_data is empty, PRINT error message and RETURN

GET junctions as a list of the keys from self.histogram\_data

SET canvas\_width = 1200

SET canvas\_height = 600

SET bar\_chart\_height = canvas\_height - (self.padding\_y \* 2)

CONFIGURE canvas width and height

SET start\_x = self.padding\_x

SET axis\_bottom = canvas\_height - self.padding\_y

//Calculate end\_x for the x-axis

SET end\_x = calculate position of x for the end of the last bar

//Draw X-Axis

DRAW line from start\_x, axis\_bottom to end\_x, axis\_bottom

//Loop through all 24 hours

FOR each hour from 0 to 23

SET x1 to starting x-position of the bar for the current hour

//Loop through each junction in junctions

FOR each junction in junctions with index i

GET count from histogram data for the current junction and hour

SET x2 = x1 + self.bar\_width

SET y1 = axis\_bottom

SET y2 = y1 - (count / self.max\_count) \* bar\_chart\_height

//Draw the bar for each junction

DRAW rectangle from x1, y1 to x2, y2, filled with color at index i, outline black

//Display the count above the bar

CREATE text at the center top of the bar with the count from histogram data for the current junction and hour

SET x1 = x1 + self.bar\_width

//Draw the hours label

CREATE text below the x-axis indicating the current hour

// Draw Main Title

SET title\_y to self.padding\_y/2

CREATE text for title, centered horizontally at the position title\_y

//Draw the Hours label

SET hours\_y = axis\_bottom + self.padding\_y + 20

CREATE text for "Hours 00:00 to 24:00" label, centered horizontally, and with font Arial

**METHOD add\_legend()**

// Get list of junction names

SET junctions to list of keys in self.histogram\_data

SET legend\_start\_x = 50

SET legend\_start\_y = self.padding\_y / 2

// Loop through each junction in junctions

FOR each junction with index i in junctions

DRAW rectangle at given location with color at index i

CREATE text for junction name next to the rectangle

INCREMENT legend\_start\_y by 20

// Run the Tkinter main loop

**METHOD run()**

CALL self.setup\_window()

CALL self.draw\_histogram()

CALL self.add\_legend()

CALL self.root.mainloop()

// --- Class: MultiCSVProcessor --- **CLASS**

**MultiCSVProcessor**  // Constructor:

**METHOD \_\_init\_\_()**

SET self.collected\_results to empty list

SET self.current\_data to None

// Load and Process CSV File

METHOD load\_csv\_file(file\_path)

CALL process\_csv\_data with given file\_path and assign the result to self.current\_data RETURN self.current\_data

// Clear Previous Data

**METHOD clear\_previous\_data()**

SET self.current\_data to None

// Handles user input and analysis

**METHOD handle\_user\_interaction()**

WHILE True:

// Get validated date input

GET dd, mm, yyyy by calling validate\_date\_input()

// Create CSV filename

SET csv\_filename = create traffic\_data CSV file name using the given validated date

TRY:

// Load and process the data

SET data\_frame to result of CALLING load\_csv\_file(csv\_filename)

IF data\_frame exists

// Analysis and save results

SET results to result of calling display\_outcomes(data\_frame)

ADD (results, csv\_filename) to self.collected\_results

//Display the results

FOR each key-value pair in results

PRINT key and value

//Create and display a Histogram

CREATE HistogramApp with data\_frame and date and CALL run method to display

ELSE:

PRINT message "No data found"

CATCH FileNotFoundError:

PRINT error message for file not found

CATCH any other Exception:

PRINT error message

// Check if the user wants to continue

GET user\_input = input "Do you want to analyze another file?"

IF user\_input != 'yes':

// Save results and break

PRINT message "Saving results to results.txt"

CALL save\_results\_to\_file(self.collected\_results, "results.txt")

PRINT message "Saved successfully. Exiting"

BREAK

ELSE:

CALL clear\_previous\_data()

// Main processing loop for files

**METHOD process\_files()**

CALL handle\_user\_interaction()

// --- Main Program --- CALL MultiCSVProcessor ()

CALL processor.process\_files()