```
CLASS HistogramApp
    FUNCTION __init__(traffic_data, date)
        SET self.traffic_data = traffic_data
        SET self.date = date
        CREATE main Tkinter window (self.root)
        SET self.canvas = None

    FUNCTION setup_windows()
        SET window title to "Histogram"
        SET window dimensions to 1500x900
        CREATE a canvas (self.canvas) with dimensions 1500x900 and white background
        PACK the canvas into the window

    FUNCTION draw_histogram()
        SET hourly_data = process_hourly_data()
        SET max_count = maximum count from hourly_data
        SET gap_between_groups = 55
        SET bar_width = 20
        SET x_offset = 100
        SET y_offset = 600
        SET group_gap = gap_between_groups - 2 * bar_width

        DEFINE color_codes for junctions

        DRAW X-axis and Y-axis on the canvas
        ADD labels for X-axis and Y-axis

        SET current_x = x_offset + group_gap

        FOR each hour, counts in hourly_data
            FOR each junction, count in counts
                SET bar_height = (count / max_count) * 350
                SET bar_color = color_codes[junction]
```

```
            DRAW the bar on the canvas
            ADD count label above the bar

            INCREMENT current_x by bar_width

        ADD hour label below the group of bars
        INCREMENT current_x by group_gap

    FUNCTION process_hourly_data()
        INITIALIZE hourly_counts for each hour and junction
        FOR each record in self.traffic_data
            EXTRACT hour from record
            GET junction name from record
            IF hour is in hourly_counts
                INCREMENT count for the junction and hour
        RETURN hourly_counts

    FUNCTION add_legend()
        DRAW legend for each junction on the canvas
        ADD title for the histogram

    FUNCTION run()
        CALL setup_windows()
        CALL draw_histogram()
        CALL add_legend()
        START Tkinter main loop

CLASS MultiCSVProcessor
    FUNCTION __init__()
        SET self.traffic_data = None

    FUNCTION load_csv_files(file_pathway)
```

```
    TRY
        OPEN file at file_pathway
        READ CSV file into self.traffic_data
        PRINT success message
    EXCEPT FileNotFoundError
        PRINT error message
        SET self.traffic_data = []
    EXCEPT Exception as e
        PRINT error message
        SET self.traffic_data = []

FUNCTION clear_previous_data()
    SET self.traffic_data = []

FUNCTION handle_user_interaction()
    WHILE True
        PROMPT user for input (Y/N)
        IF user_input is 'y'
            PROMPT for file pathway
            CALL clear_previous_data()
            CALL load_csv_files(file_pathway)
            PROMPT for date
            CREATE HistogramApp instance with traffic_data and date
            CALL app.run()
        ELSE IF user_input is 'n'
            PRINT exit message
            RETURN user_input
        ELSE
            PRINT invalid input message

FUNCTION process_files()
    WHILE True
        CALL Validate_date_input() and store result in file_pathway
```

```
        IF file_pathway is invalid
            PRINT invalid file path message
            CONTINUE

        CALL load_csv_files(file_pathway)

        IF traffic_data is empty
            PRINT no valid data message
            CONTINUE

        SET outcomes = process_csv_data(file_pathway)
        IF outcomes is empty
            PRINT no data found message
            CONTINUE

        CALL display_outcomes(outcomes)
        CALL save_results_to_file(outcomes)

        TRY
            SET date from file_pathway
            CREATE HistogramApp instance with traffic_data and date
            CALL app.run()
        EXCEPT Exception as e
            PRINT error displaying histogram message

        SET repeat_program = Validate_continue_input()
        IF repeat_program is not valid
            PRINT exit message
            BREAK

MAIN
    CREATE MultiCSVProcessor instance (processor)
    CALL processor.process_files()
```