HistogramApp Class

```
CLASS HistogramApp
    METHOD __init__(traffic_data, date)
        SET traffic_data, date
        INITIALIZE root, canvas, csv_processor, scale TO None

    METHOD setup_window()
        DESTROY root IF exists
        CREATE Tkinter canvas
        CONFIGURE canvas size and background

    METHOD draw_histogram()
        PARSE rabbit, hayley FROM traffic_data
        CALCULATE scale BASED ON traffic_data[2]

        INITIALIZE x FOR Rabbit Road bars
        FOR EACH j IN rabbit DO
            DRAW bar AND label ON canvas
            INCREMENT x

        RESET x FOR Hayley bars
        FOR EACH j IN hayley DO
            DRAW bar AND label ON canvas
            INCREMENT x

        DRAW X-axis AND labels
```

```
  METHOD add_legend()
    ADD title, scale indicator, and color legend to canvas
    CREATE buttons: "Load new dataset" (CALL main) and "Close" (CALL exit)
    ADD buttons to canvas

  METHOD run()
    CALL setup_window()
    CALL draw_histogram()
    CALL add_legend()
    START Tkinter main loop
```

MultiCSVProcessor Class

```
CLASS MultiCSVProcessor
  METHOD __init__()
    INITIALIZE current_data AS empty list
    INITIALIZE date, file_name AS None
    INITIALIZE first_run AS "yes"
    INITIALIZE retype AS None
    INITIALIZE new_file AS empty string

  METHOD load_csv_file(file_path)
    TRY
      OPEN file_path
      READ lines
      PARSE lines INTO current_data (skip header)
      SET retype TO "no"
```

```
    CATCH FileNotFoundError
        PRINT error message
        SET retype TO "yes"

METHOD clear_previous_data()
    CLEAR current_data

METHOD handle_user_interaction:
IF self.retype IS "yes" AND self.first_run IS "no" THEN:
    REPEAT:
        PROMPT user WITH "Do you want to load a new file? (Y/N): "
        STORE user input IN self.new_file

        IF self.new_file.lower() NOT IN ["yes", "no", "y", "n"] THEN:
            PRINT "Invalid input, please enter 'yes' or 'no'"
            CONTINUE LOOP

        IF self.new_file.lower() IN ["no", "n"] THEN:
            PRINT "Thank you for using the program"
            EXIT program
        BREAK LOOP
CALL first_part.validate_date_input()
SET self.date TO first_part.date
SET self.file_name TO CONCATENATE "traffic_data", self.date, ".csv"
SET self.first_run TO "no"
```

```
  METHOD process_files()
    INITIALIZE hourly_vehicles_hanley, hourly_vehicles_rabbit_road AS empty dictionaries

    FOR EACH row IN current_data DO
      IF row[0].lower() IS "hanley highway/westway" THEN
        PARSE hour FROM row[2]
        INCREMENT hourly_vehicles_hanley[hour]
      ELSE IF row[0].lower() IS "elm avenue/rabbit road" THEN
        PARSE hour FROM row[2]
        INCREMENT hourly_vehicles_rabbit_road[hour]

    CALL clear_previous_data()

    CALCULATE max_hanley, max_rabbit_road, overall_max
    APPEND hourly_vehicles_rabbit_road, hourly_vehicles_hanley, overall_max TO current_data
    SET retype TO "yes"
```

Main Program

```
MAIN
  REPEAT
    INITIALIZE MultiCSVProcessor AS app
    app.handle_user_interaction()
    app.load_csv_file(app.file_name)

    IF app.retype IS "yes" THEN
```

```
        CONTINUE

    CALL process_csv_data(app.date) FROM Part_ABC
    CALL display_outcomes() FROM Part_ABC
    CALL save_results_to_file(first_part.output, "results.txt") FROM Part_ABC

    app.process_files()
    INITIALIZE HistogramApp(app.current_data, app.date) AS histogram
    histogram.run()
    BREAK
END MAIN
```