

# **Programming Project Final Report**

Written by Phillip Gershovich and Uzoma Eze-Ejikeme on behalf of Group 23

## **Introduction**

For the second half of the Programming Project module, we were tasked with making huge data sets of flights in the continental United States more user friendly and easier to dig through. Even though we were handed a multitude of different sized data sets, our group immediately started sorting through the biggest and most challenging dataset to deal with. With 100,000 flights, and their origin, destination, distance, cancellation status, diverted status, carrier and flight number stored, it was crucial to find the most time efficient way to sort through the flights. From starting off with a blank unfiltered dataset to making it user friendly, interactive and visually appealing, this is the process that each group undertook to reach the final goal.

## **Process**

### **Week 8:**

We started off by creating a Flight class. This remained largely unchanged throughout the project.

```

1 class Flight {}
2   String flightDate;
3   String carrier;
4   String carrierNum;
5   String origin;
6   String originCity;
7   String originState;
8   String originWac;
9   String destination;
10  String destinationCity;
11  String destinationState;
12  String destinationWac;
13  String departureTime;
14  String arrivalTime;
15  String cancelled;
16  String diverted;
17  String distance;
18
19  Flight(String flightDate, String carrier, String carrierNum, String origin, String originCity,
20         String originState, String originWac, String destination, String destinationCity,
21         String destinationState, String destinationWac, String departureTime, String arrivalTime,
22         String cancelled, String diverted, String distance) {
23      this.flightDate = flightDate;
24      this.carrier = carrier;
25      this.carrierNum = carrierNum;
26      this.origin = origin;
27      this.originCity = originCity;
28      this.originState = originState;
29      this.originWac = originWac;
30      this.destination = destination;
31      this.destinationCity = destinationCity;
32      this.destinationState = destinationState;
33      this.destinationWac = destinationWac;
34      this.departureTime = departureTime;
35      this.arrivalTime = arrivalTime;
36      this.cancelled = cancelled;
37      this.diverted = diverted;
38      this.distance = distance;
39  }
40
41
42  void display(float x, float y) {
43      fill(255,255,255);
44      text(carrier + " " + carrierNum + " " + origin + " -> " + destinationCity + " " + destinationState, x, y);
45  }
46 }
47
48

```

This flight class allowed us to register in the data from the dataset and assign a value to each variable that we were going to use throughout the code. We then worked on incorporating a main class.

```

1  ArrayList<Flight> flights = new ArrayList<>();
2
3  void setup() {
4      size(800, 600);
5      loadFlightData("flights100k.csv");
6  }
7
8  void draw() {
9      background(0);
10     fill(255);
11     textSize(22);
12
13     float y = 20;
14     for (int i = 0; i < 20; i++) {
15         flights.get(i).display(50, y);
16         y += 20;
17     }
18 }
19
20 void loadFlightData(String filename) {
21     String[] rows = loadStrings(filename);
22     for (int i = 1; i < rows.length; i++) {
23         String[] cols = split(rows[i], ",");
24         if (cols.length >= 16) {
25             Flight f = new Flight(cols[0], cols[1], cols[2], cols[3], cols[4],
26                                   cols[5], cols[6], cols[7], cols[8], cols[9],
27                                   cols[10], cols[11], cols[12], cols[13], cols[14], cols[15]);
28             flights.add(f);
29         }
30     }
31     println("Loaded " + flights.size() + " flights.");
32 }

```

---

We did this by creating an array list of flights which would store all the flight information and then be displayed on the screen.

## Week 9:

For the week after, we needed to come up with a way to display the flight data in some sort of chart. Our group settled on displaying the flight distance in a bar chart format. For this, we needed to implement a class for drawing these charts.

```

1  class ChartDraw {
2      ArrayList<String> labels;
3      | ArrayList<Integer> values;
4
5      ChartDraw(ArrayList<String> labels, ArrayList<Integer> values) {
6          this.labels = labels;
7          this.values = values;
8      }
9
10     void drawBarChart(float x, float y, float chartWidth, float chartHeight) {
11         // Manually find the maximum value
12         int maxVal = 0;
13         for (int val : values) {
14             if (val > maxVal) {
15                 maxVal = val;
16             }
17         }
18
19         float barWidth = chartWidth / values.size();
20
21         for (int i = 0; i < values.size(); i++) {
22             float barHeight = map(values.get(i), 0, maxVal, 0, chartHeight);
23             fill(100, 150, 255);
24             rect(x + i * barWidth, y + chartHeight - barHeight, barWidth - 5, barHeight);
25
26             fill(255);
27             textSize(12);
28             textAlign(CENTER);
29             text(labels.get(i), x + i * barWidth + barWidth / 2, y + chartHeight + 15);
30             text(values.get(i), x + i * barWidth + barWidth / 2, y + chartHeight - barHeight - 5);
31         }
32     }
33 }

```

---

This chart worked by analyzing the flight distances which were stored in a previously created array list and then making rectangles which would represent the number of flights that reached a certain range of flight distance. We also slightly tweaked the main class in order to accommodate for the new chart draw class.

```

ArrayList<Flight> flights = new ArrayList<>();

void setup() {
  size(800, 600);
  loadFlightData("flights100k.csv");
  ProcessingQueries pq = new ProcessingQueries(flights);

  ArrayList<String> origins = new ArrayList<>();
  ArrayList<Integer> originCounts = new ArrayList<>();
  pq.countFlightByOriginAirport(origins, originCounts);

  ArrayList<String> destinations = new ArrayList<>();
  ArrayList<Integer> destinationCounts = new ArrayList<>();
  pq.countFlightByDestinationAirport(destinations, destinationCounts);

  ArrayList<String> distanceCategories = new ArrayList<>();
  ArrayList<Integer> distanceCounts = new ArrayList<>();
  pq.countFlightByDistance(distanceCategories, distanceCounts);
}

void draw() {
  background(0);
  fill(255);
  textSize(22);

  float y = 20;
  for (int i = 0; i < 20; i++) {
    flights.get(i).display(50, y);
    y += 20;
  }
}

void loadFlightData(String filename) {
  String[] rows = loadStrings(filename);
  for (int i = 1; i < rows.length; i++) {
    String[] cols = split(rows[i], ",");
    if (cols.length >= 16) {
      Flight f = new Flight(cols[0], cols[1], cols[2], cols[3], cols[4],
                           cols[5], cols[6], cols[7], cols[8], cols[9],
                           cols[10], cols[11], cols[12], cols[13], cols[14], cols[15]);
      flights.add(f);
    }
  }
  println("Loaded " + flights.size() + " flights.");
}

```

The new array lists allowed for the graphs to be drawn accurately because they counted up the flight variable that we were focusing on for the respective bar charts.

## Week 10

In week 10, our code needed to be able to handle user interaction such as queries about the flights in the dataset. We took this challenge on by creating a ProcessingQueries class.

```
class ProcessingQueries
{
    ArrayList<Flight> flights;
    ProcessingQueries(ArrayList<Flight> flights)
    {
        this.flights=flights;
    }

    void countFlightByOriginAirport(ArrayList<String> originAirport, ArrayList<Integer> counts)
    {
        for (Flight f : flights)
        {
            String origin = f.origin;
            int index = originAirport.indexOf(origin);
            if (index != -1)
            {
                counts.set(index, counts.get(index) + 1);
            }
            else
            {
                originAirport.add(origin);
                counts.add(1);
            }
        }
    }

    void countFlightByDestinationAirport(ArrayList<String> destinationAirport, ArrayList<Integer> count)
    {
        for (Flight f : flights)
        {
            String destination = f.destination;
            int index = destinationAirport.indexOf(destination);
            if ( index != -1 )
            {
                count.set(index, count.get(index) + 1 );
            }
            else
            {
                destinationAirport.add(destination);
                count.add(1);
            }
        }
    }

    void countFlightByDistance(ArrayList<String> flightDistance, ArrayList<Integer> count) {
        String[] categories = { "< 1000 km", "1000 - 1999 km", "2000 - 2999 km", "3000+ km" };
        int[] categoryCounts = { 0, 0, 0, 0 };

        for (Flight f : flights) {
            int distance = Integer.parseInt(f.distance);

            if (distance < 1000) {
                categoryCounts[0]++;
            } else if (distance < 2000) {
                categoryCounts[1]++;
            } else if (distance < 3000) {
                categoryCounts[2]++;
            } else {
                categoryCounts[3]++;
            }
        }

        for (int i = 0; i < categories.length; i++) {
            flightDistance.add(categories[i]);
            count.add(categoryCounts[i]);
        }
    }
}
```

This class created the capability for the code to count and store how many flights met what requirements. For example, the countFlightByDistance method checked which flights were less than 1000 miles, between 1000 and 2000 miles, between 2000 and 3000 miles, and over 3000 miles. This was then stored in an array which would be represented by the bar charts. We then updated the main class for the user to be able to interact with the map by clicking a button to either receive bar graphs showcasing the flight distance, origin airport, or destination airport. We

did this by making a drawButton method. Since it would be unreasonable to show every single origin and destination airport, we just picked the top 10 most commonly appearing airports in the data to showcase in the bar graphs.

```
116 void drawButton(String label, float x, float y, float w, float h) {
117     fill(200);
118     rect(x, y, w, h);
119     fill(0);
120     textSize(16);
121     textAlign(CENTER, CENTER);
122     text(label, x + w / 2, y + h / 2);
123 }
124
125 void mousePressed() {
126     if (currentScreen == 0) {
127         // Main screen buttons
128         if (mouseX > 50 && mouseX < 250 && mouseY > 500 && mouseY < 550) {
129             currentScreen = 1; // Distance chart
130         } else if (mouseX > 300 && mouseX < 500 && mouseY > 500 && mouseY < 550) {
131             currentScreen = 2; // Origin chart
132         } else if (mouseX > 550 && mouseX < 750 && mouseY > 500 && mouseY < 550) {
133             currentScreen = 3; // Destination chart
134         }
135     } else {
136         // Back button
137         if (mouseX > 50 && mouseX < 150 && mouseY > 700 && mouseY < 750) {
138             currentScreen = 0; // Return to main screen
139         }
140     }
141 }
142
143
144 void drawDistanceChart() {
145     // Draw distance chart
146     ArrayList<String> distanceCategories = new ArrayList<>();
147     ArrayList<Integer> distanceCounts = new ArrayList<>();
148     pq.countFlightByDistance(distanceCategories, distanceCounts);
149     drawBarChart("Flight Distance Distribution", distanceCategories, distanceCounts, 50, 100, 1000, 600);
150
151     // Draw back button
152     drawButton("Back", 50, 700, 100, 50);
153 }
154
155 void drawOriginChart() {
156     ArrayList<String> origins = new ArrayList<>();
157     ArrayList<Integer> originCounts = new ArrayList<>();
158     pq.countFlightByOriginAirport(origins, originCounts);
159
160     // Get the top 10 origins
161     ArrayList<String> topOrigins = new ArrayList<>();
162     ArrayList<Integer> topOriginCounts = new ArrayList<>();
163     getTopN(origins, originCounts, topOrigins, topOriginCounts, 10);
164
165     drawBarChart("Top 10 Flights by Origin Airport", topOrigins, topOriginCounts, 100, 100, 1000, 600);
166
167     // Back button
168     drawButton("Back", 50, 700, 100, 50);
169 }
```

The photo above shows how we were able to call on the chartDraw class from earlier in the main class to actually allow the user to view the charts once the program was run.

## Week 11



After altering our code to be able to be user interactive, we were tasked with making the code user interactive in a bigger variety of ways. Since we only implemented buttons in the previous week, we began changing our code to be able to include a search bar in the code, as well as having buttons that could take the user from screen to screen on our code. We did the latter by numbering each screen and then allowing for the buttons to let the user scroll through all the graphs. The search bar also gave the user the chance to enter the words “destination”, “origin”, and “distance”, and the program would accordingly take the user to the appropriate graph. We also made our program more visually appealing by making the background different for each screen, and even changing the way that the data was represented. Since we started off with only bar graphs, we implemented pie charts and line charts as well. All the changes listed above were done exclusively in the pre-existing main class.

```

63 void displayMultilineText(String txt, float x, float y, float maxWidth) {
64     String[] words = txt.split(" ");
65     String line = "";
66     float lineHeight = 20; // Adjusted for better fit
67     for (String word : words) {
68         String testLine = line + word + " ";
69         if (textWidth(testLine) > maxWidth) {
70             text(line, x, y);
71             y += lineHeight; // Move to next line
72             line = word + " ";
73
74             // Prevent text from exceeding the search bar's height
75             if (y > 780) break;
76         } else {
77             line = testLine;
78         }
79     }
80     text(line, x, y); // Print remaining text
81 }
82 void drawSearchBar() {
83     fill(0);
84     rect(50, 720, 400, 60); // Larger search box for multi-line text
85     textAlign(LEFT, CENTER);
86     fill(255);
87     displayMultilineText(searchText, 60, 750, 380); // Display multi-line text
88 }
89
90 void keyPressed() {
91     if (key == BACKSPACE && searchText.length() > 0) {
92         searchText = searchText.substring(0, searchText.length() - 1);
93     } else if (key == ENTER || key == RETURN) {
94         checkSearch();
95     } else if (key >= 32 && key <= 126) { // Only allow normal characters
96         searchText += key;
97     }
98 }
99
100 // Function to check input and change screens
101 void checkSearch() {
102     String lowerText = searchText.toLowerCase().trim(); // Normalize input
103
104     if (lowerText.equals("distance")) {
105         currentScreen = 1;
106     } else if (lowerText.equals("destination")) {
107         currentScreen = 3;
108     } else if (lowerText.equals("origin")) {
109         currentScreen = 2;
110     }
111 }

```

```

void drawLineChart(String title, ArrayList<String> labels, ArrayList<Integer> values, float x, float y, float chartWidth, float chartHeight) {
    fill(0);
    textSize(22);
    textAlign(CENTER);
    text(title, width / 2, 30);

    // Background box behind the chart
    fill(245); // Light gray background
    noStroke();
    rect(x - 40, y - 20, chartWidth + 80, chartHeight + 60);

    // Axis lines
    stroke(0);
    strokeWeight(1);
    line(x, y, x, y + chartHeight); // Y-axis
    line(x, y + chartHeight, x + chartWidth, y + chartHeight); // X-axis

    // Find max value
    int maxVal = 0;
    for (int val : values) {
        if (val > maxVal) {
            maxVal = val;
        }
    }
}

void drawPieChart(String title, ArrayList<String> labels, ArrayList<Integer> values, float centerX, float centerY, float diameter) {
    // Draw the chart title
    fill(0);
    textSize(22);
    textAlign(CENTER);
    text(title, width / 2, 30);

    // Total value
    float total = 0;
    for (int val : values) {
        total += val;
    }

    float startAngle = 0;
    for (int i = 0; i < values.size(); i++) {
        float angle = map(values.get(i), 0, total, 0, TWO_PI);

        // Assign unique color per slice
        fill(100 + i * 30 % 255, 150 + i * 50 % 255, 200 + i * 70 % 255);
        arc(centerX, centerY, diameter, diameter, startAngle, startAngle + angle, PIE);

        // Calculate label position
        float midAngle = startAngle + angle / 2;
        float labelX = centerX + cos(midAngle) * diameter / 2.5;
        float labelY = centerY + sin(midAngle) * diameter / 2.5;

        fill(0);
        textSize(12);
        textAlign(CENTER, CENTER);
        text(labels.get(i) + "\n(" + values.get(i) + ")", labelX, labelY);

        startAngle += angle;
    }
}

```

#### GROUP CONTRIBUTION:

Andrei Catalin Petre: created the structure of the main class and was in charge of deadlines and overseeing the group project, added the search buttons and the backgrounds of the different pages. In charge of troubleshooting and debugging of the program.

Gerald Pirra: worked on the chartDraw class, focusing on handling graphical representation of data. His contributions included designing the graphical output, ensuring data was displayed accurately, and optimizing performance for smooth rendering.














Adam Kenny: collaborated on the chartdraw class alongside Geri. They were responsible for improving the responsiveness of the charts, debugging rendering issues, and refining the layout for better user experience.














Phillip Gershovich: contributed to the Main class which was responsible for structuring the core program workflow, managing the division of data, and ensuring smooth execution of the graphical interface and user input.





Uzoma Eze-Ejikeme: I was responsible for implementing the button functionalities and working on the Main class. My contributions involved handling user interaction, and ensuring smooth user navigation within the program.

Kai Hosokawa: worked on the processing queries and flight classes, tasks included optimizing query execution, representing flight-related attributes like flight date, carrier, origin, destination, departure/arrival time, cancellation status, and distance, and ensuring efficient data handling.













Full git history:







<b>Initial commit</b>	...
 <b>fttcata</b> created <a href="#">main</a> • 813dfb5 • 23 days ago	
<b>Name add</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 813dfb5...7786985 • 23 days ago	
<b>Update README.md</b>	...
 <b>gerip1</b> pushed 1 commit to <a href="#">main</a> • 7786985...2e3ec55 • 23 days ago	
<b>Update README.md</b>	...
 <b>Kennyad05</b> pushed 1 commit to <a href="#">main</a> • 2e3ec55...22b406d • 23 days ago	
<b>Update README.md</b>	...
 <b>pgersh</b> pushed 1 commit to <a href="#">main</a> • 22b406d...231cc21 • 23 days ago	
<b>Flight Class</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 231cc21...69ee67a • 23 days ago	
<b>Main Class first week</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 69ee67a...21e7987 • 23 days ago	
<b>Update README.md</b>	...
 <b>uz-eze</b> pushed 1 commit to <a href="#">main</a> • 21e7987...249161f • 23 days ago	
<b>Update Main Class</b>	...
 <b>ColdPalmer10</b> pushed 1 commit to <a href="#">main</a> • 249161f...cba9e58 • 23 days ago	
<b>Update Flight Class</b>	...
 <b>ColdPalmer10</b> pushed 1 commit to <a href="#">main</a> • cba9e58...257cf09 • 23 days ago	
<b>New Main Class</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 257cf09...515fd2e • 23 days ago	
<b>Added destinationState to the output</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 515fd2e...6b80a0a • 23 days ago	
<b>Renamed Flight Class to Flight Class Week 8</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 6b80a0a...0673a5a • 23 days ago	

<b>Added Kai Hosokawa to readme</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • c946a98...3dbf109 • 23 days ago	
<b>Deleted flightDate</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 3dbf109...333d6ce • 23 days ago	
<b>Created ChartDraw class</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 333d6ce...fa048fb • 17 days ago	
<b>Renamed ChartDraw class to CharDraw Week 8</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • fa048fb...fa23538 • 17 days ago	
<b>Rename ChartDraw Week 8 to ChartDraw Week 9</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • fa23538...e20aee3 • 17 days ago	
<b>Create ProcessingQueries Week 9</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • e20aee3...90c8971 • 17 days ago	
<b>Updated ProcessingQueries Week 9</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • 90c8971...ebf8c44 • 17 days ago	
<b>Updated Main Class Week 8 to fit Week 9</b>	...
 <b>fttcata</b> pushed 1 commit to <a href="#">main</a> • ebf8c44...acc7e84 • 17 days ago	
<b>Update ChartDraw Week 9</b>	...
 <b>Kennyad05</b> pushed 1 commit to <a href="#">main</a> • acc7e84...536d249 • 17 days ago	
<b>Update Main Class Week 8</b>	...
 <b>Kennyad05</b> pushed 1 commit to <a href="#">main</a> • 536d249...ae7b238 • 17 days ago	
<b>Update ProcessingQueries Week 9</b>	...
 <b>Kennyad05</b> pushed 1 commit to <a href="#">main</a> • ae7b238...ea2d36c • 17 days ago	
<b>Create FlightDataManager</b>	...
 <b>ColdPalmer10</b> pushed 1 commit to <a href="#">main</a> • ea2d36c...7aff6e0 • 17 days ago	
<b>Update Main Class Week 8</b>	...
 <b>ColdPalmer10</b> pushed 1 commit to <a href="#">main</a> • 7aff6e0...9c90469 • 17 days ago	

<b>Update Main Class Week 8</b>  ColdPalmer10 pushed 1 commit to <a href="#">main</a> · 7aff6e0...9c90469 · 17 days ago	...
<b>Update Main Class Week 8</b>  Kennya05 pushed 1 commit to <a href="#">main</a> · 9c90469...bd66012 · 17 days ago	...
<b>Update ChartDraw Week 9</b>  Kennya05 pushed 1 commit to <a href="#">main</a> · bd66012...b224495 · 17 days ago	...
<b>Update ChartDraw Week 9</b>  pgersh pushed 1 commit to <a href="#">main</a> · b224495...fb6a5eb · 17 days ago	...

[Share feedback about this page](#)

<b>Create Week9 - barcharts</b>  gerip1 pushed 1 commit to <a href="#">main</a> · fb6a5eb...b5d7c29 · 17 days ago	...
<b>Update Week9 - barcharts</b>  gerip1 pushed 1 commit to <a href="#">main</a> · b5d7c29...1ec4762 · 17 days ago	...
<b>Update Week9 - barcharts</b>  gerip1 pushed 1 commit to <a href="#">main</a> · 1ec4762...3bc9d82 · 17 days ago	...
<b>Update Main Class Week 8</b>  uz-eze pushed 1 commit to <a href="#">main</a> · 3bc9d82...e1dd5f8 · 16 days ago	...
<b>chartDraw latest version</b>  fttcata pushed 1 commit to <a href="#">main</a> · e1dd5f8...205a012 · 16 days ago	...
<b>Update Main Class Week 8 to suit week 9</b>  fttcata pushed 1 commit to <a href="#">main</a> · 205a012...6aef583 · 16 days ago	...
<b>Update and rename Week9 - barcharts to Week9 - newMain, and barCharts...</b>  Kennya05 pushed 1 commit to <a href="#">main</a> · 6aef583...6f058d5 · 16 days ago	...
<b>Update Week9 - newMain, and barChartsClass</b>  Kennya05 pushed 1 commit to <a href="#">main</a> · 6f058d5...86d93bf · 16 days ago	...
<b>Rename Week9 - newMain, and barChartsClass to Week10 - newMain, and b...</b>  pgersh pushed 1 commit to <a href="#">main</a> · 86d93bf...7104a18 · 9 days ago	...
<b>Updated Main Class Week 9/10 to suit week 11</b>  fttcata pushed 1 commit to <a href="#">main</a> · 7104a18...7a84fce · 4 days ago	...
<b>Update Main Class Week 8</b>  pgersh pushed 1 commit to <a href="#">main</a> · 7a84fce...03cdd67 · 3 days ago	...
<b>Week 11 main</b>  fttcata pushed 1 commit to <a href="#">main</a> · 03cdd67...77aefcb · 2 days ago	...
<b>Rename Week10 - newMain, and barChartsClass to Week 11 main</b>	...

<b>Update Main Class Week 8</b>  pgersh pushed 1 commit to <a href="#">main</a> · 7a84fce...03cdd67 · 3 days ago	...
<b>Week 11 main</b>  fttcata pushed 1 commit to <a href="#">main</a> · 03cdd67...77aefcb · 2 days ago	...
<b>Rename Week10 - newMain, and barChartsClass to Week 11 main</b>  fttcata pushed 1 commit to <a href="#">main</a> · 77aefcb...6e0210f · 2 days ago	...
<b>Update Main Class Week 8</b>  ColdPalmer10 pushed 1 commit to <a href="#">main</a> · 6e0210f...d3d9eb8 · 2 days ago	...
<b>Week 11 main including searchBar</b>  fttcata pushed 1 commit to <a href="#">main</a> · d3d9eb8...9897b75 · 2 days ago	...
<b>Introduced "next" button and pages</b>  fttcata pushed 1 commit to <a href="#">main</a> · 9897b75...8a340c4 · 2 days ago	...

[Share feedback about this page](#)