Programming Project Final Report

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

Abstract:

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.

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.

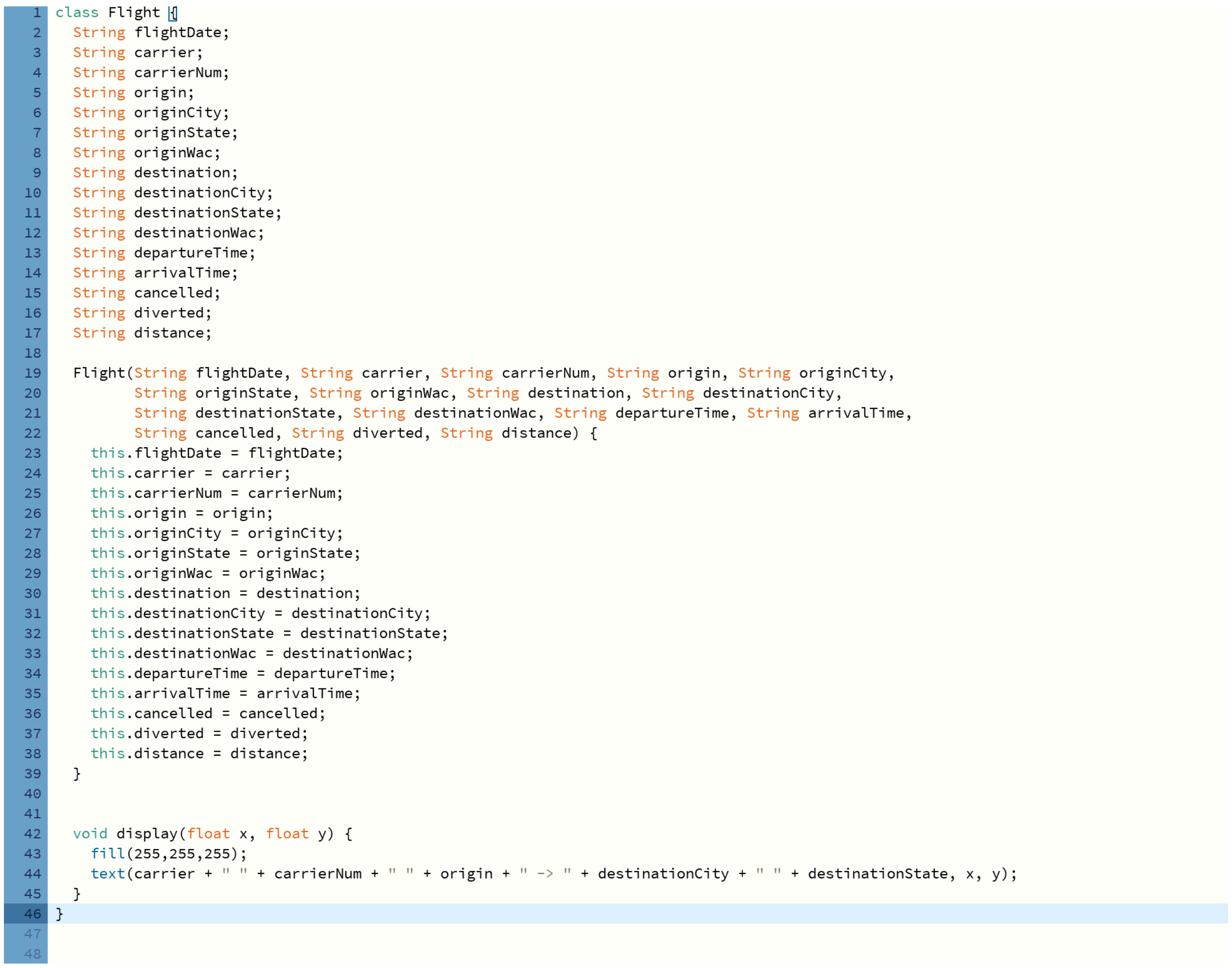
Uzoma Eze-Ejikeme:I was responsible for implementing the button functionalities and working on the Main class. My contributions involved handling user interaction, and ensuring smooth user navigation within the program.

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.

Execution and Results

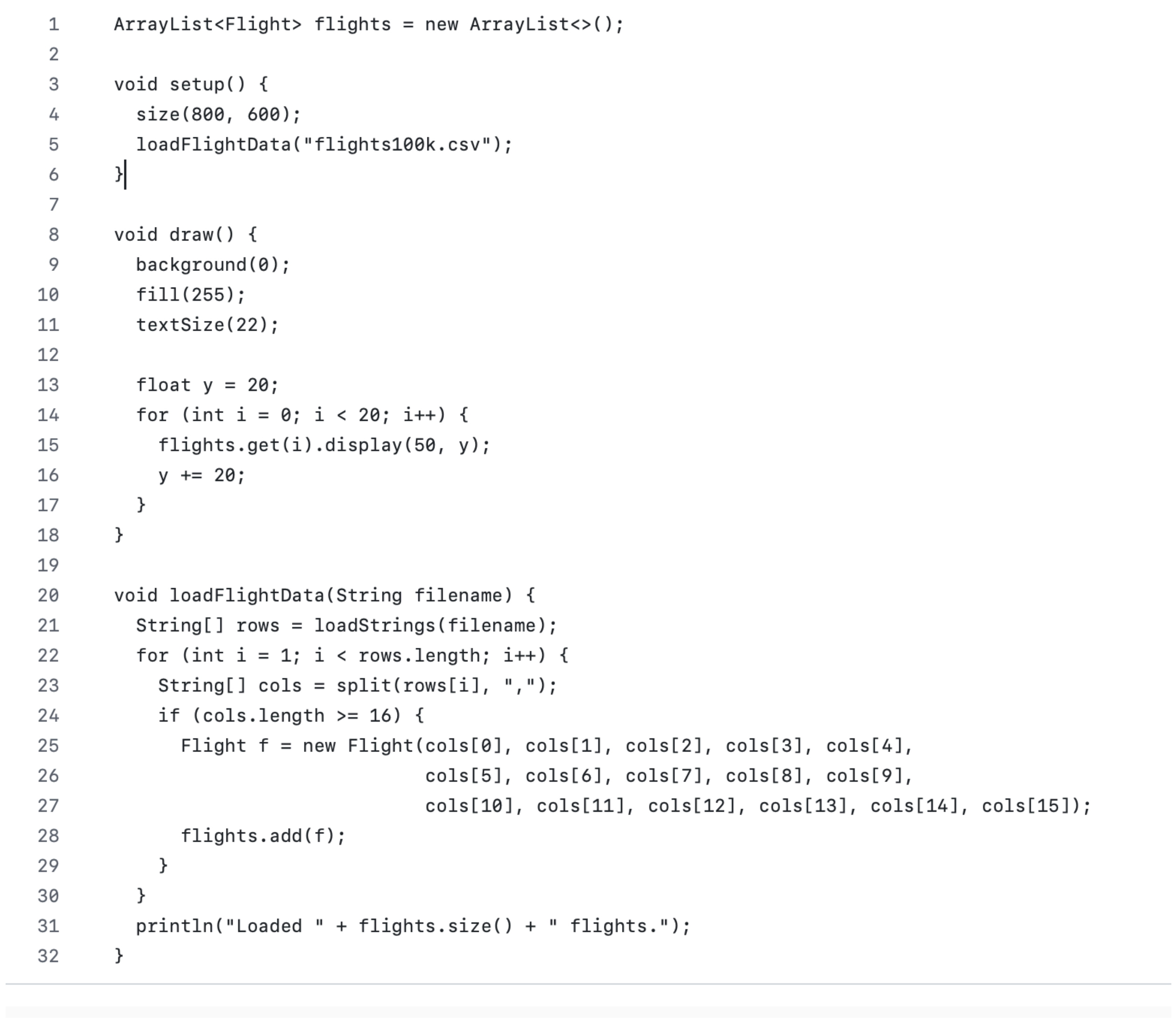
Week 8:

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



```java
class Flight {
  String flightDate;
  String carrier;
  String carrierNum;
  String origin;
  String originCity;
  String originState;
  String originWac;
  String destination;
  String destinationCity;
  String destinationState;
  String destinationWac;
  String departureTime;
  String arrivalTime;
  String cancelled;
  String diverted;
  String distance;

  Flight(String flightDate, String carrier, String carrierNum, String origin, String originCity,
         String originState, String originWac, String destination, String destinationCity,
         String destinationState, String destinationWac, String departureTime, String arrivalTime,
         String cancelled, String diverted, String distance) {
    this.flightDate = flightDate;
    this.carrier = carrier;
    this.carrierNum = carrierNum;
    this.origin = origin;
    this.originCity = originCity;
    this.originState = originState;
    this.originWac = originWac;
    this.destination = destination;
    this.destinationCity = destinationCity;
    this.destinationState = destinationState;
    this.destinationWac = destinationWac;
    this.departureTime = departureTime;
    this.arrivalTime = arrivalTime;
    this.cancelled = cancelled;
    this.diverted = diverted;
    this.distance = distance;
  }


  void display(float x, float y) {
    fill(255,255,255);
    text(carrier + " " + carrierNum + " " + origin + " -> " + destinationCity + " " + destinationState, x, y);
  }
}
```

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.

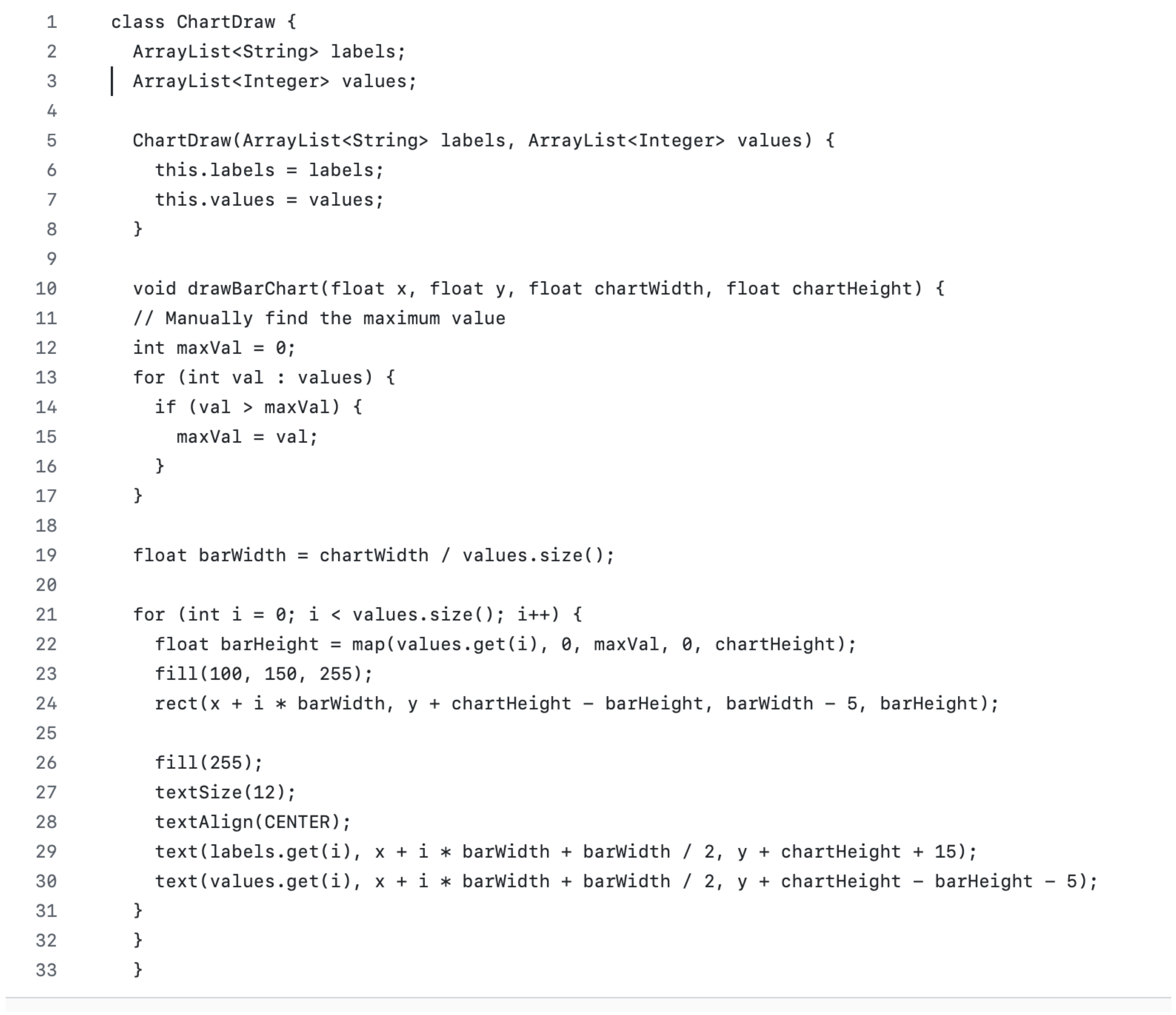


```
ArrayList<Flight> flights = new ArrayList<>();

void setup() {
  size(800, 600);
  loadFlightData("flights100k.csv");
}

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.");
}
```

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

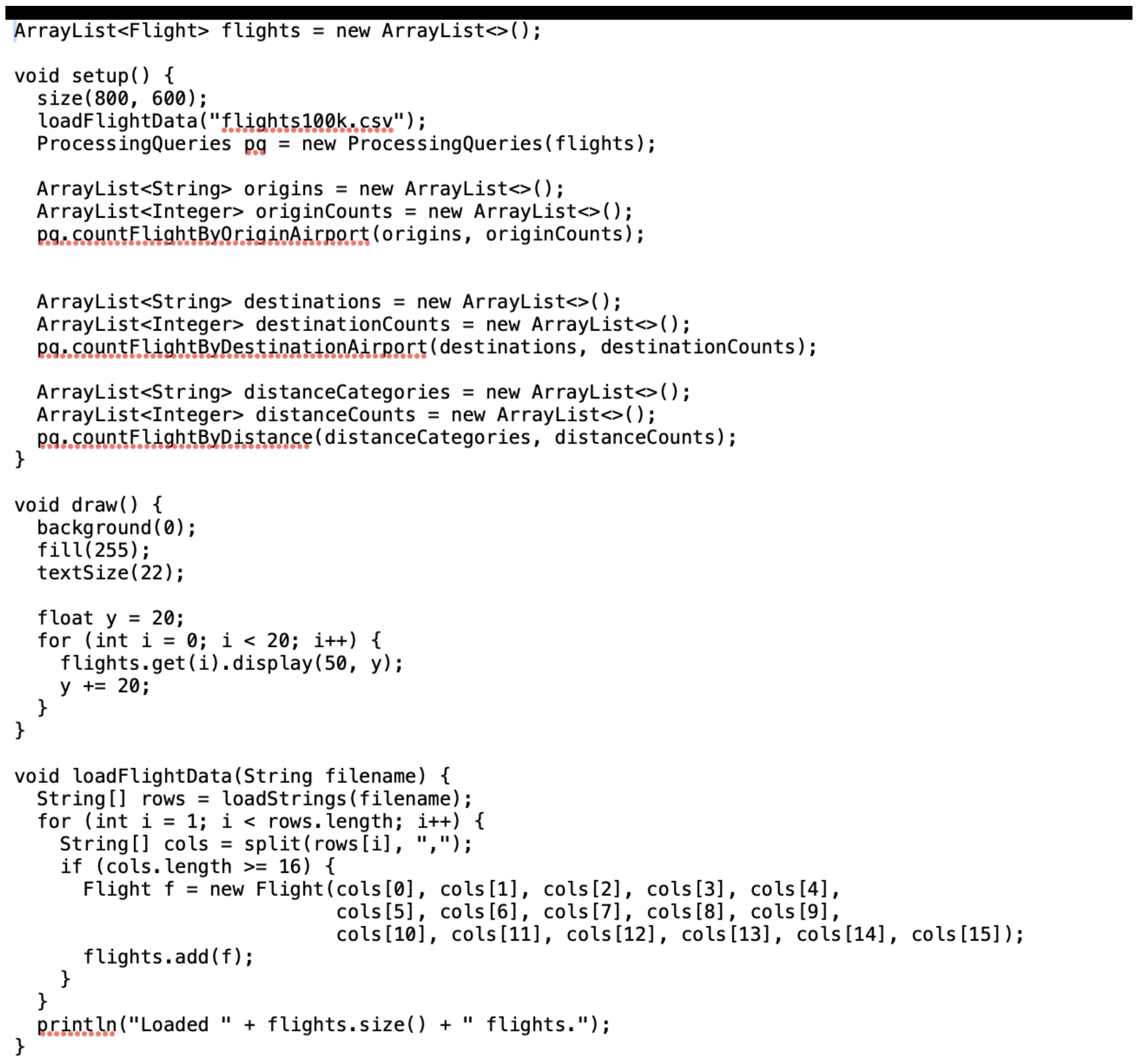
Week 9:

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.



```
class ChartDraw {
  ArrayList<String> labels;
  ArrayList<Integer> values;

  ChartDraw(ArrayList<String> labels, ArrayList<Integer> values) {
    this.labels = labels;
    this.values = values;
  }

  void drawBarChart(float x, float y, float chartWidth, float chartHeight) {
  // Manually find the maximum value
  int maxVal = 0;
  for (int val : values) {
    if (val > maxVal) {
      maxVal = val;
    }
  }

  float barWidth = chartWidth / values.size();

  for (int i = 0; i < values.size(); i++) {
    float barHeight = map(values.get(i), 0, maxVal, 0, chartHeight);
    fill(100, 150, 255);
    rect(x + i * barWidth, y + chartHeight - barHeight, barWidth - 5, barHeight);

    fill(255);
    textSize(12);
    textAlign(CENTER);
    text(labels.get(i), x + i * barWidth + barWidth / 2, y + chartHeight + 15);
    text(values.get(i), x + i * barWidth + barWidth / 2, y + chartHeight - barHeight - 5);
  }
  }
  }
```

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.

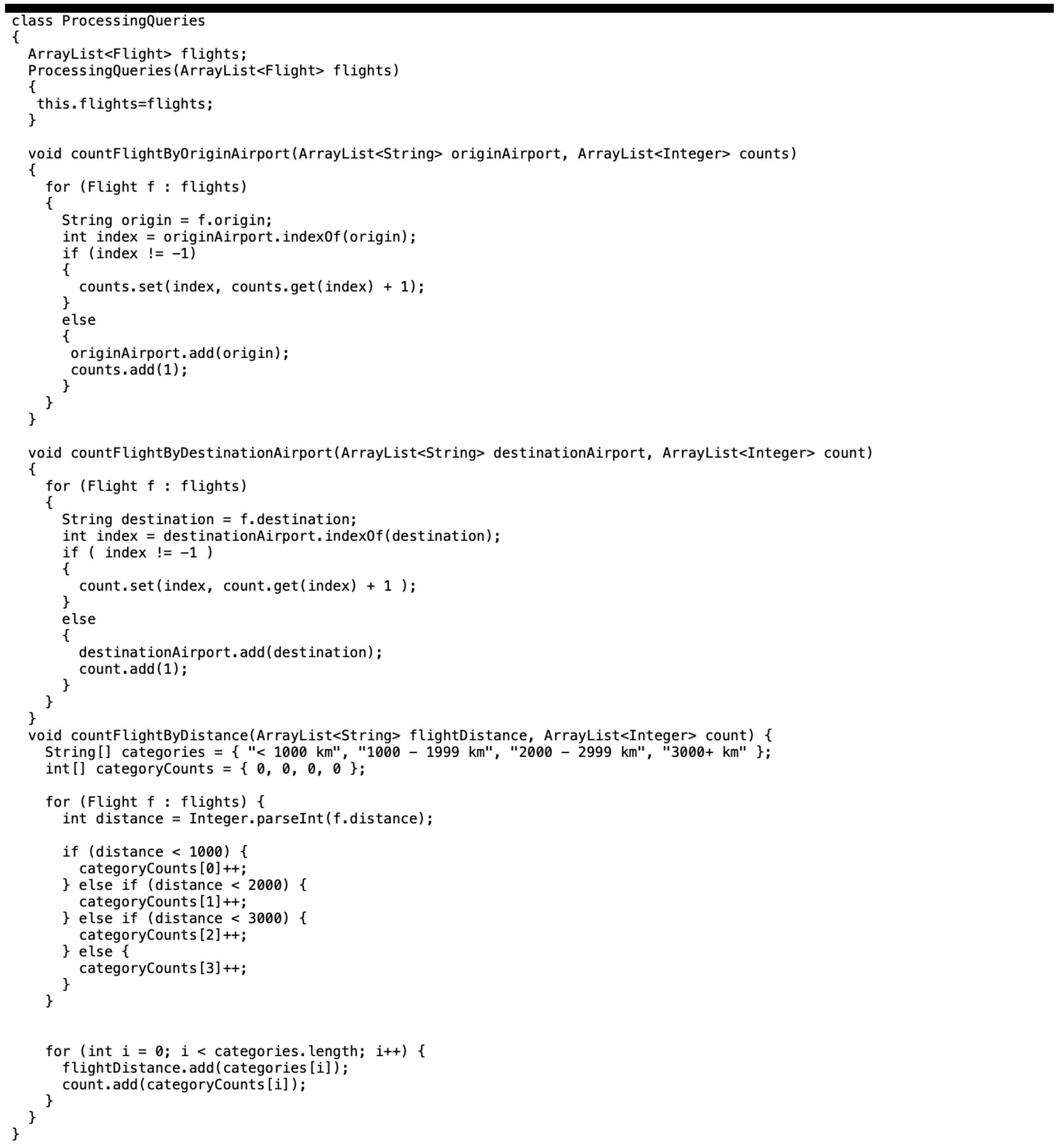


```
ArrayList<Flight> flights = new ArrayList<>();

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.

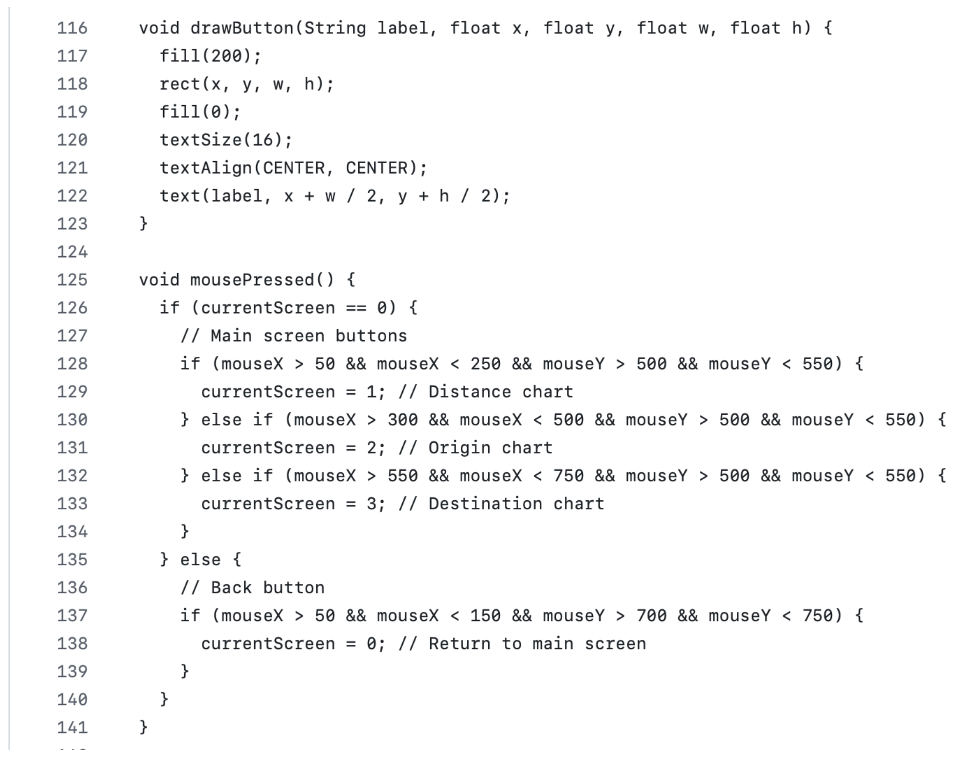
Week 10

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.

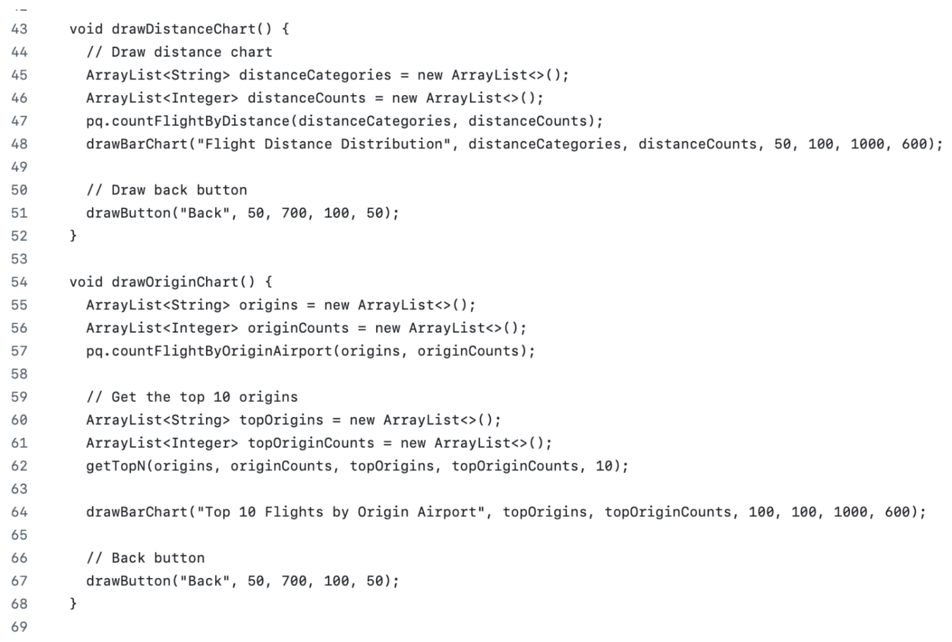


```
class ProcessingQueries
{
  ArrayList<Flight> flights;
  ProcessingQueries(ArrayList<Flight> flights)
  {
   this.flights=flights;
  }

  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.



```
void drawButton(String label, float x, float y, float w, float h) {
  fill(200);
  rect(x, y, w, h);
  fill(0);
  textSize(16);
  textAlign(CENTER, CENTER);
  text(label, x + w / 2, y + h / 2);
}

void mousePressed() {
  if (currentScreen == 0) {
    // Main screen buttons
    if (mouseX > 50 && mouseX < 250 && mouseY > 500 && mouseY < 550) {
      currentScreen = 1; // Distance chart
    } else if (mouseX > 300 && mouseX < 500 && mouseY > 500 && mouseY < 550) {
      currentScreen = 2; // Origin chart
    } else if (mouseX > 550 && mouseX < 750 && mouseY > 500 && mouseY < 550) {
      currentScreen = 3; // Destination chart
    }
  } else {
    // Back button
    if (mouseX > 50 && mouseX < 150 && mouseY > 700 && mouseY < 750) {
      currentScreen = 0; // Return to main screen
    }
  }
}
```

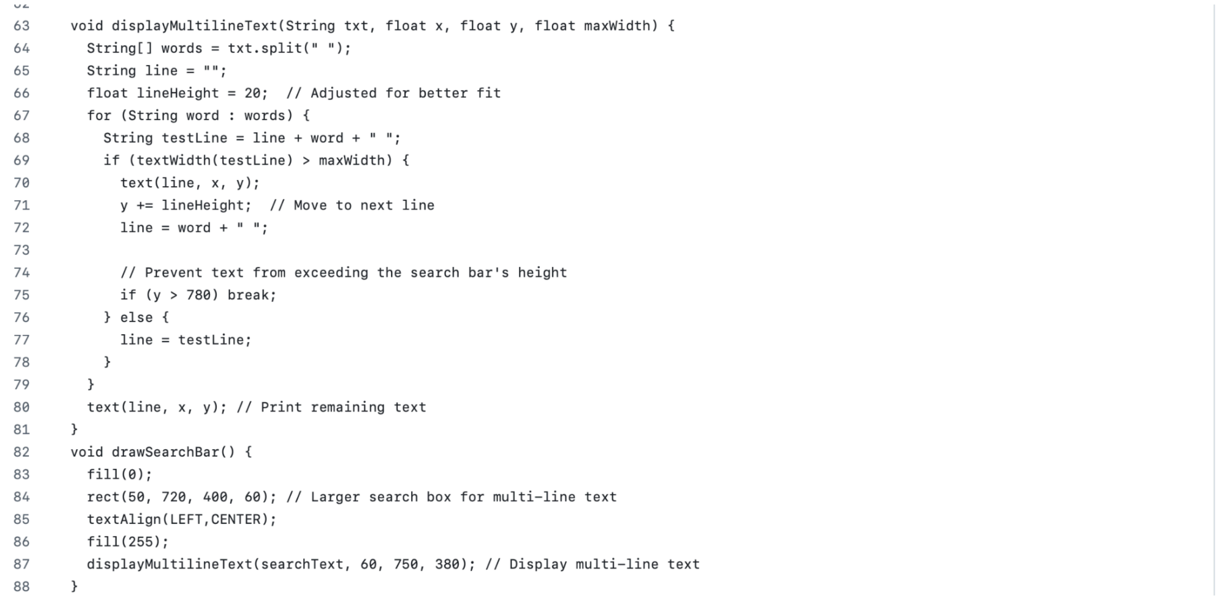


```
void drawDistanceChart() {
  // Draw distance chart
  ArrayList<String> distanceCategories = new ArrayList<>();
  ArrayList<Integer> distanceCounts = new ArrayList<>();
  pq.countFlightByDistance(distanceCategories, distanceCounts);
  drawBarChart("Flight Distance Distribution", distanceCategories, distanceCounts, 50, 100, 1000, 600);

  // Draw back button
  drawButton("Back", 50, 700, 100, 50);
}

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

  // Get the top 10 origins
  ArrayList<String> topOrigins = new ArrayList<>();
  ArrayList<Integer> topOriginCounts = new ArrayList<>();
  getTopN(origins, originCounts, topOrigins, topOriginCounts, 10);

  drawBarChart("Top 10 Flights by Origin Airport", topOrigins, topOriginCounts, 100, 100, 1000, 600);

  // Back button
  drawButton("Back", 50, 700, 100, 50);
}
```

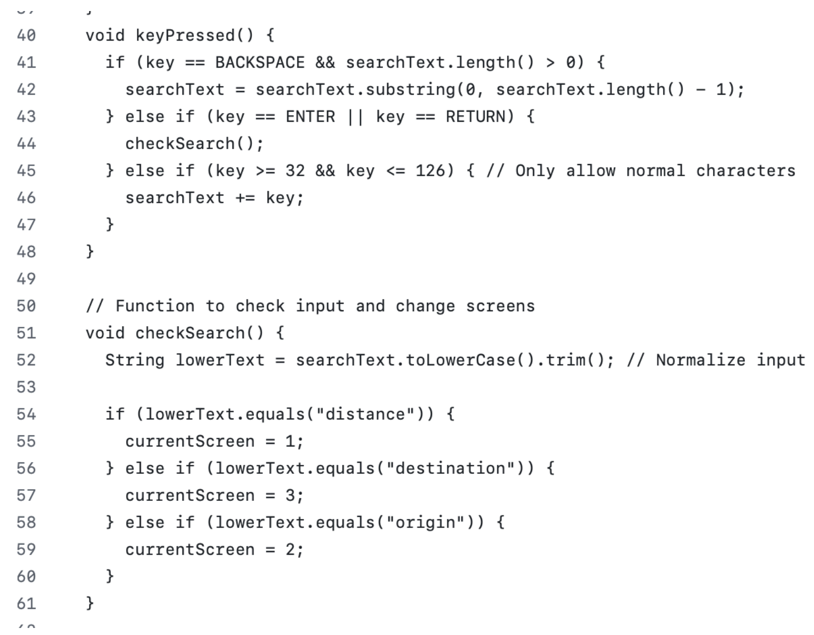
The photo above shows how we were able to call on the chartDraw class from earlier in the main class to actually allow the user to view the charts once the program was run.

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.



```
void displayMultilineText(String txt, float x, float y, float maxWidth) {
  String[] words = txt.split(" ");
  String line = "";
  float lineHeight = 20;  // Adjusted for better fit
  for (String word : words) {
    String testLine = line + word + " ";
    if (textWidth(testLine) > maxWidth) {
      text(line, x, y);
      y += lineHeight;  // Move to next line
      line = word + " ";

      // Prevent text from exceeding the search bar's height
      if (y > 780) break;
    } else {
      line = testLine;
    }
  }
  text(line, x, y); // Print remaining text
}
void drawSearchBar() {
  fill(0);
  rect(50, 720, 400, 60); // Larger search box for multi-line text
  textAlign(LEFT,CENTER);
  fill(255);
  displayMultilineText(searchText, 60, 750, 380); // Display multi-line text
}
```

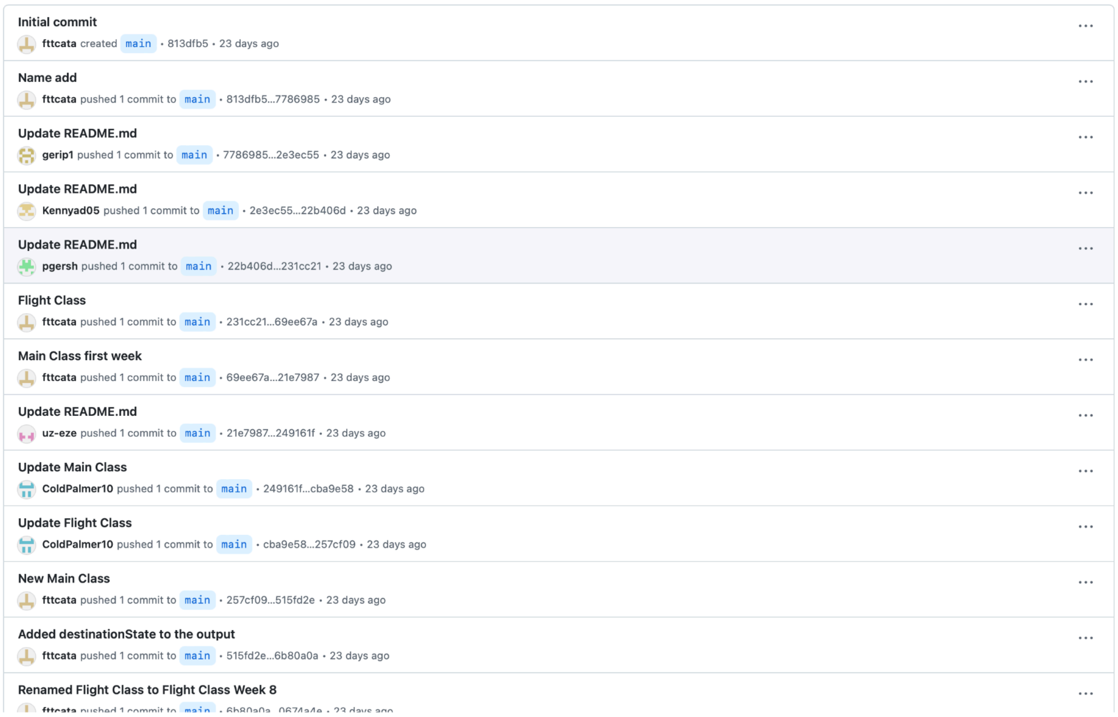


```
void keyPressed() {
  if (key == BACKSPACE && searchText.length() > 0) {
    searchText = searchText.substring(0, searchText.length() - 1);
  } else if (key == ENTER || key == RETURN) {
    checkSearch();
  } else if (key >= 32 && key <= 126) { // Only allow normal characters
    searchText += key;
  }
}

// Function to check input and change screens
void checkSearch() {
  String lowerText = searchText.toLowerCase().trim(); // Normalize input

  if (lowerText.equals("distance")) {
    currentScreen = 1;
  } else if (lowerText.equals("destination")) {
    currentScreen = 3;
  } else if (lowerText.equals("origin")) {
    currentScreen = 2;
  }
}
```

```
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;
  }
}
```

Full git history:



Initial commit
fttcata created main • 813dfb5 • 23 days ago

Name add
fttcata pushed 1 commit to main • 813dfb5...7786985 • 23 days ago

Update README.md
gerip1 pushed 1 commit to main • 7786985...2e3ec55 • 23 days ago

Update README.md
Kennyad05 pushed 1 commit to main • 2e3ec55...22b406d • 23 days ago

Update README.md
pgersh pushed 1 commit to main • 22b406d...231cc21 • 23 days ago

Flight Class
fttcata pushed 1 commit to main • 231cc21...69ee67a • 23 days ago

Main Class first week
fttcata pushed 1 commit to main • 69ee67a...21e7987 • 23 days ago

Update README.md
uz-eze pushed 1 commit to main • 21e7987...249161f • 23 days ago

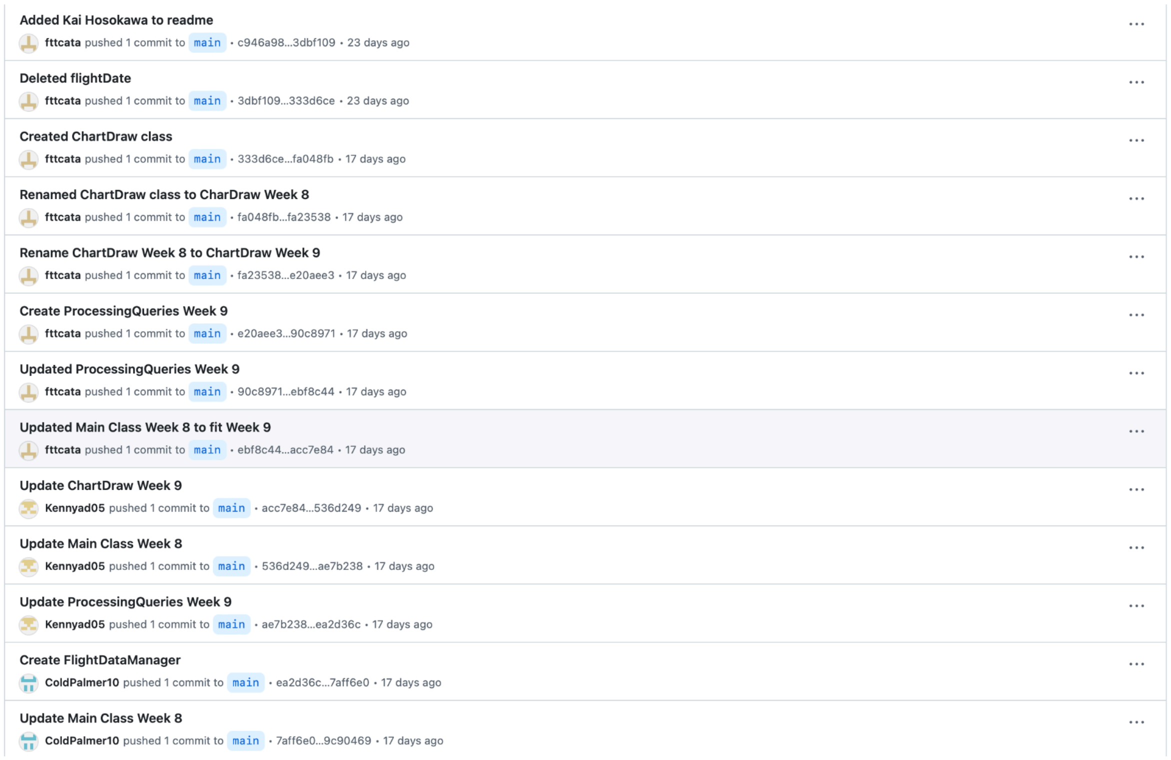
Update Main Class
ColdPalmer10 pushed 1 commit to main • 249161f...cba9e58 • 23 days ago

Update Flight Class
ColdPalmer10 pushed 1 commit to main • cba9e58...257cf09 • 23 days ago

New Main Class
fttcata pushed 1 commit to main • 257cf09...515fd2e • 23 days ago

Added destinationState to the output
fttcata pushed 1 commit to main • 515fd2e...6b80a0a • 23 days ago

Renamed Flight Class to Flight Class Week 8
fttcata pushed 1 commit to main • 6b80a0a...0674a4e • 23 days ago



Added Kai Hosokawa to readme
fttcata pushed 1 commit to main • c946a98...3dbf109 • 23 days ago

Deleted flightDate
fttcata pushed 1 commit to main • 3dbf109...333d6ce • 23 days ago

Created ChartDraw class
fttcata pushed 1 commit to main • 333d6ce...fa048fb • 17 days ago

Renamed ChartDraw class to CharDraw Week 8
fttcata pushed 1 commit to main • fa048fb...fa23538 • 17 days ago

Rename ChartDraw Week 8 to ChartDraw Week 9
fttcata pushed 1 commit to main • fa23538...e20aee3 • 17 days ago

Create ProcessingQueries Week 9
fttcata pushed 1 commit to main • e20aee3...90c8971 • 17 days ago

Updated ProcessingQueries Week 9
fttcata pushed 1 commit to main • 90c8971...ebf8c44 • 17 days ago

Updated Main Class Week 8 to fit Week 9
fttcata pushed 1 commit to main • ebf8c44...acc7e84 • 17 days ago

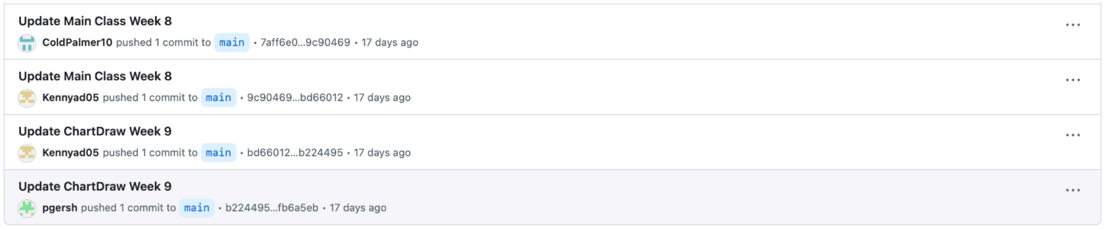
Update ChartDraw Week 9
Kennyad05 pushed 1 commit to main • acc7e84...536d249 • 17 days ago

Update Main Class Week 8
Kennyad05 pushed 1 commit to main • 536d249...ae7b238 • 17 days ago

Update ProcessingQueries Week 9
Kennyad05 pushed 1 commit to main • ae7b238...ea2d36c • 17 days ago

Create FlightDataManager
ColdPalmer10 pushed 1 commit to main • ea2d36c...7aff6e0 • 17 days ago

Update Main Class Week 8
ColdPalmer10 pushed 1 commit to main • 7aff6e0...9c90469 • 17 days ago



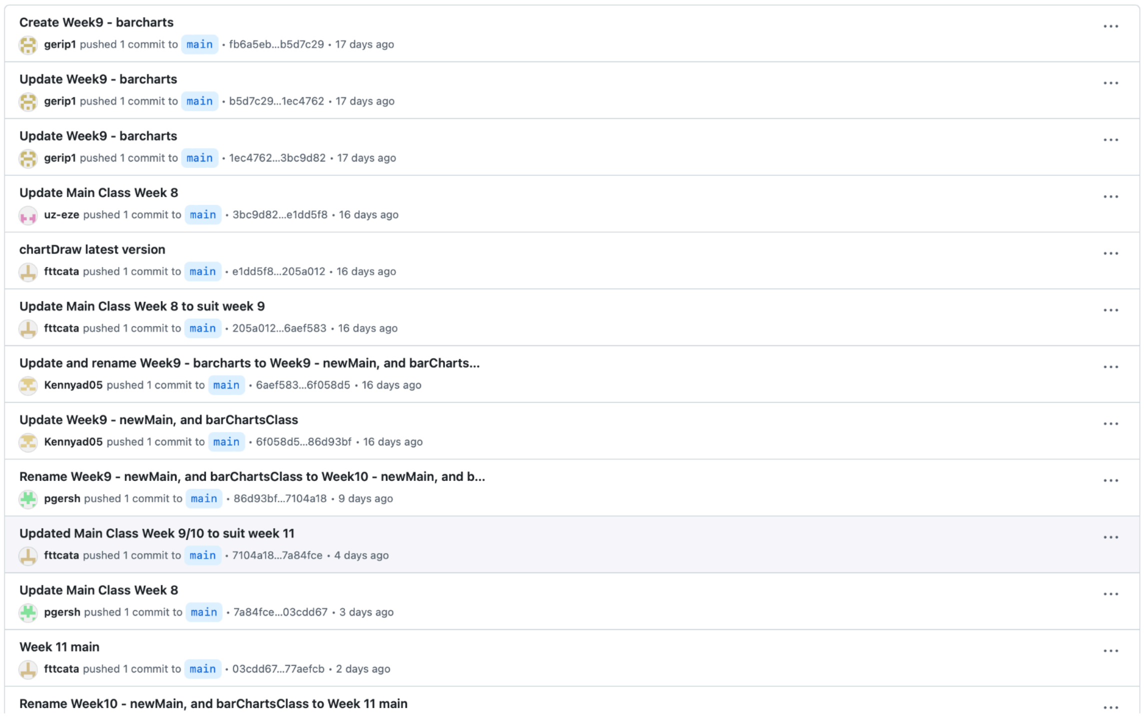
Update Main Class Week 8
ColdPalmer10 pushed 1 commit to main • 7aff6e0...9c90469 • 17 days ago

Update Main Class Week 8
Kennyad05 pushed 1 commit to main • 9c90469...bd66012 • 17 days ago

Update ChartDraw Week 9
Kennyad05 pushed 1 commit to main • bd66012...b224495 • 17 days ago

Update ChartDraw Week 9
pgersh pushed 1 commit to main • b224495...fb6a5eb • 17 days ago

Share feedback about this page



Create Week9 - barcharts
gerip1 pushed 1 commit to main • fb6a5eb...b5d7c29 • 17 days ago

Update Week9 - barcharts
gerip1 pushed 1 commit to main • b5d7c29...1ec4762 • 17 days ago

Update Week9 - barcharts
gerip1 pushed 1 commit to main • 1ec4762...3bc9d82 • 17 days ago

Update Main Class Week 8
uz-eze pushed 1 commit to main • 3bc9d82...e1dd5f8 • 16 days ago

chartDraw latest version
fttcata pushed 1 commit to main • e1dd5f8...205a012 • 16 days ago

Update Main Class Week 8 to suit week 9
fttcata pushed 1 commit to main • 205a012...6aef583 • 16 days ago

Update and rename Week9 - barcharts to Week9 - newMain, and barCharts...
Kennyad05 pushed 1 commit to main • 6aef583...6f058d5 • 16 days ago

Update Week9 - newMain, and barChartsClass
Kennyad05 pushed 1 commit to main • 6f058d5...86d93bf • 16 days ago

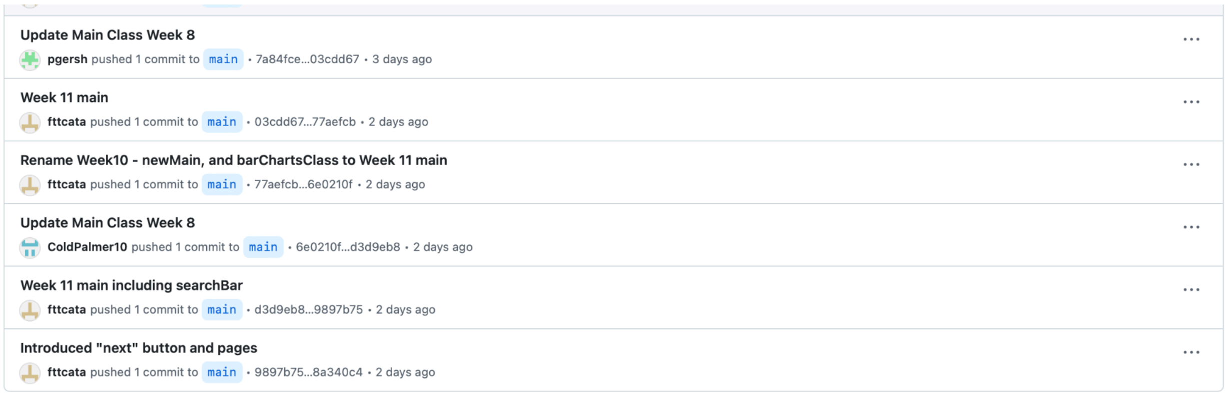
Rename Week9 - newMain, and barChartsClass to Week10 - newMain, and b...
pgersh pushed 1 commit to main • 86d93bf...7104a18 • 9 days ago

Updated Main Class Week 9/10 to suit week 11
fttcata pushed 1 commit to main • 7104a18...7a84fce • 4 days ago

Update Main Class Week 8
pgersh pushed 1 commit to main • 7a84fce...03cdd67 • 3 days ago

Week 11 main
fttcata pushed 1 commit to main • 03cdd67...77aefcb • 2 days ago

Rename Week10 - newMain, and barChartsClass to Week 11 main



Update Main Class Week 8
pgersh pushed 1 commit to main • 7a84fce...03cdd67 • 3 days ago

Week 11 main
fttcata pushed 1 commit to main • 03cdd67...77aefcb • 2 days ago

Rename Week10 - newMain, and barChartsClass to Week 11 main
fttcata pushed 1 commit to main • 77aefcb...6e0210f • 2 days ago

Update Main Class Week 8
ColdPalmer10 pushed 1 commit to main • 6e0210f...d3d9eb8 • 2 days ago

Week 11 main including searchBar
fttcata pushed 1 commit to main • d3d9eb8...9897b75 • 2 days ago

Introduced "next" button and pages
fttcata pushed 1 commit to main • 9897b75...8a340c4 • 2 days ago

Share feedback about this page

Conclusion:

Throughout this project, our group strived to create the most user friendly application possible and we believe we achieved that by implementing plenty of features aimed at the user, as well as delivering easy to digest interpretations of the data.