# Project B

Kevin Kingsbury
Foundations of Software Design
Professor Lomelino

WSMV - Weather Visualization

# Table of Contents

# Concept Statement & Definition

The purpose of this visualization is to display historical weather data in a visually appealing way. The visualization will use a range of colors mapped to average daily temperatures in Nashville, TN. This will give the viewer an interesting view of temperature fluctuations throughout the year. Data can be viewed for any year where information is available in a CSV file that will drive the data in the visualization. Mouse hover will be used to update an info panel and keyboard arrows will be used to filter the data by year.

## *Who?*

WSMV is the local NBC affiliate in Nashville, TN. They were the first television station in Nashville and have been on the air since 1950. Alumni staff of the station include Pat Sajak, Robin Roberts, and John Tesh. The station has an excellent weather program that has been a Nashville favorite for many years. 4Warn is the current branding for the station's weather reporting and is headed by Chief Meteorologist Lisa Spencer.

## *What?*

WSMV would like a heatmap visualization of historical weather data to display on their website. The visualization should be able to display historical average temperatures in the Nashville area for any year where there is temperature data. Initial test data will be provided in a CSV file. A colorful way of displaying the data should be used to help show at-a-glance temperature differences throughout the year. Any other information may also be displayed as available.

## Concept Statement & Definition

### *Where?*

The visualization is intended to be displayed on the WSMV website. A new section of the site will be added to show historical weather-related data. In the future the visualization may possibly be used on mobile apps or even on the air during a weather segment.

### *When?*

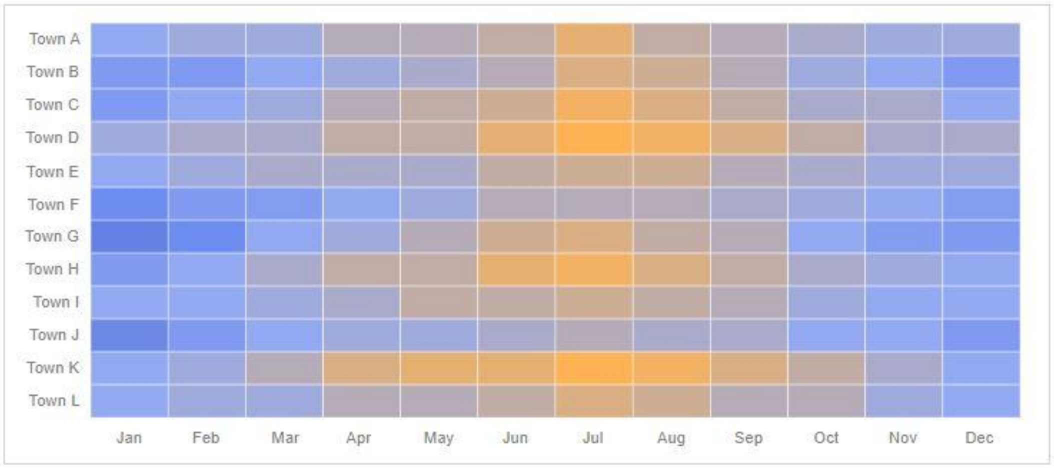
The visualization is requested as soon as possible. Expected delivery date is within the next two weeks. Additional functionality such as pulling data from other sources will be requested at a later time.

### *Why?*

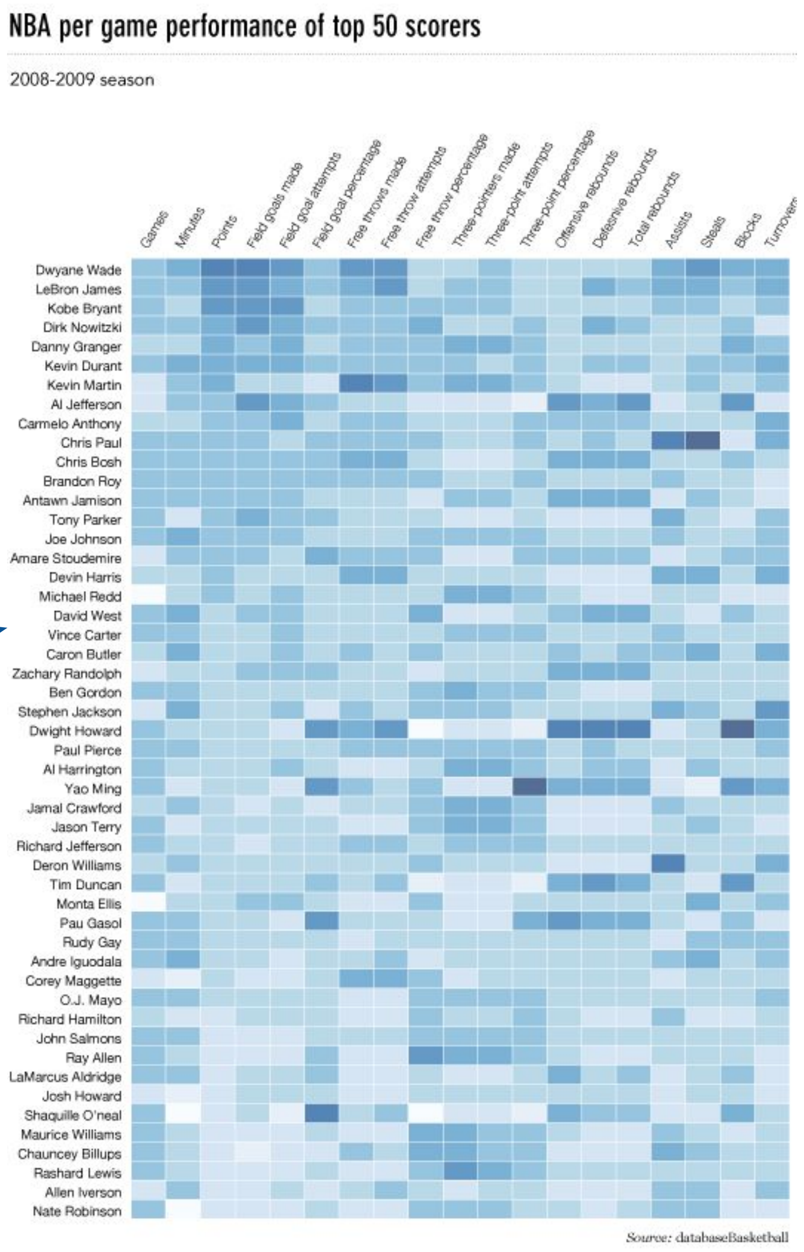
The visualization is meant to give the person viewing it a sense of the fluctuations in average temperature for the Nashville area throughout a given year. Having the visualization show the temperatures using color offers an interesting view of weather in the area. It will also show that WSMV is interested in providing its viewers a wider variety of features on its website. Data visualizations that can be interacted with by using a mouse and keyboard will encourage users see WSMV as a relevant source of weather information.

# Reference Imagery

## Heatmap (Matrix)

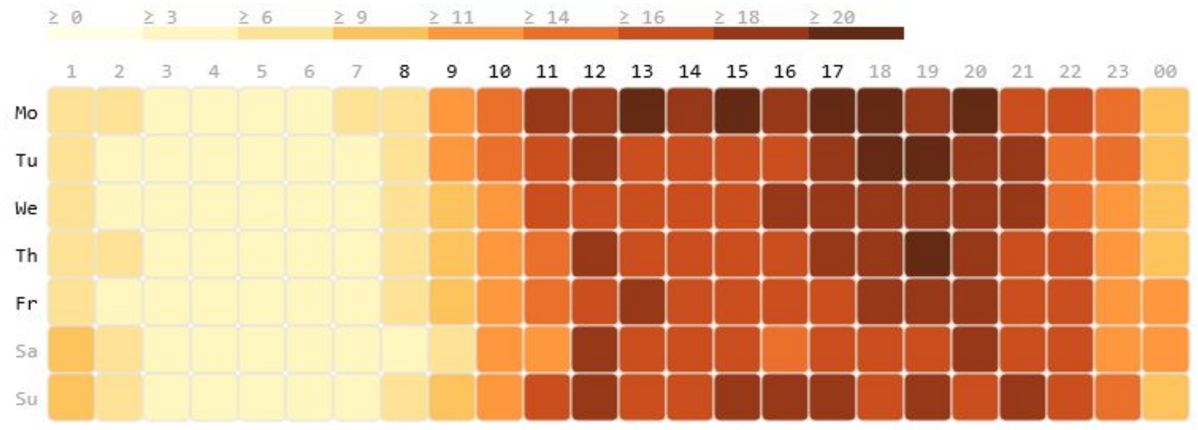


These colors are visually appealing and the layout is intuitive.



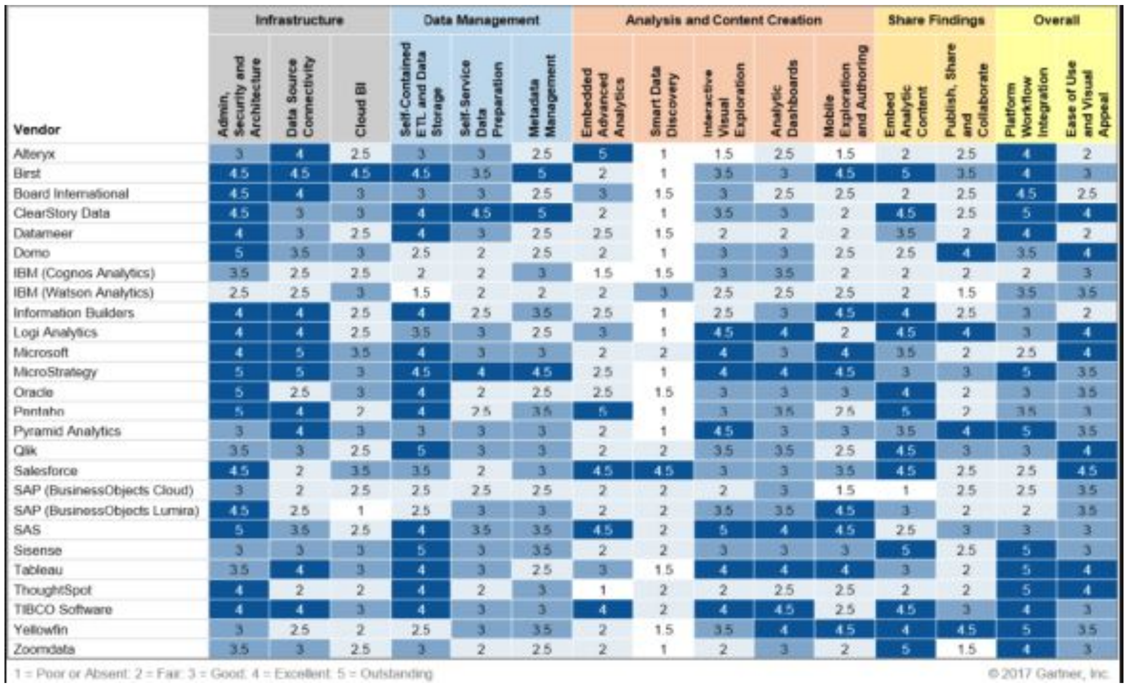
The colors here look good, but it is difficult visually to decide which column and row you are looking at. The data is too specific to use a heatmap maybe.

# Reference Imagery



Notice the gradient legend at the top of this heatmap display. We at least need to give the viewer the start and end points for reference.

This chart looks bad because it is too busy. The actual data values being on different background and in different colors is visually confusing.

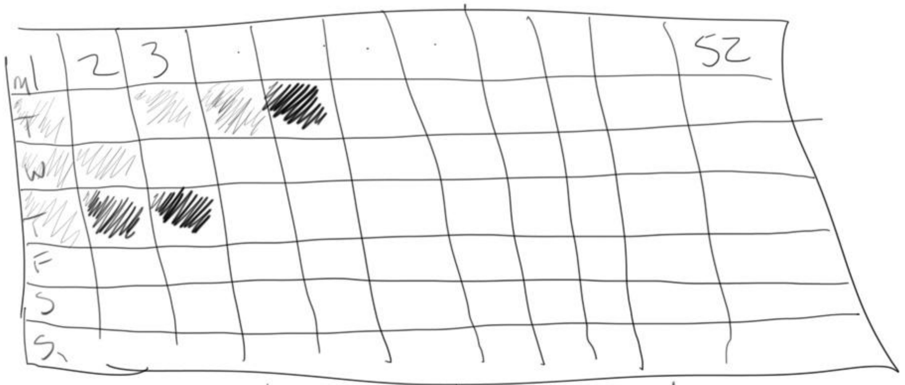


| | Infrastructure | | | Data Management | | | Analysis and Content Creation | | | | | Share Findings | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vendor | Admin, Security and Architecture | Data Source Connectivity | Cloud BI | Self-Contained ETL and Data Storage | Self-Service Data Preparation | Metadata Management | Embedded Advanced Analytics | Smart Data Discovery | Interactive Visual Exploration | Analytic Dashboards | Mobile Exploration and Authoring | Embed Analytic Content | Publish, Share and Collaborate | Platform Workflow Integration | Ease of Use and Visual Appeal |
| Alteryx | 3 | 4 | 2.5 | 3 | 3 | 2.5 | 5 | 1 | 1.5 | 2.5 | 1.5 | 2 | 2.5 | 4 | 2 |
| Birst | 4.5 | 4.5 | 4.5 | 4.5 | 3.5 | 5 | 2 | 1 | 3.5 | 3 | 4.5 | 5 | 3.5 | 4 | 3 |
| Board International | 4.5 | 4 | 3 | 3 | 3 | 2.5 | 3 | 1.5 | 3 | 2.5 | 2.5 | 2 | 2.5 | 4.5 | 2.5 |
| ClearStory Data | 4.5 | 3 | 3 | 4 | 4.5 | 5 | 2 | 1 | 3.5 | 3 | 2 | 4.5 | 2.5 | 5 | 4 |
| Datameer | 4 | 3 | 2.5 | 4 | 3 | 2.5 | 2.5 | 1.5 | 2 | 2 | 2 | 3.5 | 2 | 4 | 2 |
| Domo | 5 | 3.5 | 3 | 2.5 | 2 | 2.5 | 2 | 1 | 3 | 3 | 2.5 | 2.5 | 4 | 3.5 | 4 |
| IBM (Cognos Analytics) | 3.5 | 2.5 | 2.5 | 2 | 2 | 3 | 1.5 | 1.5 | 3 | 3.5 | 2 | 2 | 2 | 2 | 3 |
| IBM (Watson Analytics) | 2.5 | 2.5 | 3 | 1.5 | 2 | 2 | 2 | 3 | 2.5 | 2.5 | 2.5 | 2 | 1.5 | 3.5 | 3.5 |
| Information Builders | 4 | 4 | 2.5 | 4 | 2.5 | 3.5 | 2.5 | 1 | 2.5 | 3 | 4.5 | 4 | 2.5 | 3 | 2 |
| Logi Analytics | 4 | 4 | 2.5 | 3.5 | 3 | 2.5 | 3 | 1 | 4.5 | 4 | 2 | 4.5 | 4 | 3 | 4 |
| Microsoft | 4 | 5 | 3.5 | 4 | 3 | 3 | 2 | 2 | 4 | 3 | 4 | 3.5 | 2 | 2.5 | 4 |
| MicroStrategy | 5 | 5 | 3 | 4.5 | 4 | 4.5 | 2.5 | 1 | 4 | 4 | 4.5 | 3 | 3 | 5 | 3.5 |
| Oracle | 5 | 2.5 | 3 | 4 | 2 | 2.5 | 2.5 | 1.5 | 3 | 3 | 3 | 4 | 2 | 3 | 3.5 |
| Pentaho | 5 | 4 | 2 | 4 | 2.5 | 3.5 | 5 | 1 | 3 | 3.5 | 2.5 | 5 | 2 | 3.5 | 3 |
| Pyramid Analytics | 3 | 4 | 3 | 3 | 3 | 3 | 2 | 1 | 4.5 | 3 | 3 | 3.5 | 4 | 5 | 3.5 |
| Qlik | 3.5 | 3 | 2.5 | 5 | 3 | 3 | 2 | 2 | 3.5 | 3.5 | 2.5 | 4.5 | 3 | 3 | 4 |
| Salesforce | 4.5 | 2 | 3.5 | 3.5 | 2 | 3 | 4.5 | 4.5 | 3 | 3 | 3.5 | 4.5 | 2.5 | 2.5 | 4.5 |
| SAP (BusinessObjects Cloud) | 3 | 2 | 2.5 | 2.5 | 2.5 | 2.5 | 2 | 2 | 2 | 3 | 1.5 | 1 | 2.5 | 2.5 | 3.5 |
| SAP (BusinessObjects Lumira) | 4.5 | 2.5 | 1 | 2.5 | 3 | 3 | 2 | 2 | 3.5 | 3.5 | 4.5 | 3 | 2 | 2 | 3.5 |
| SAS | 5 | 3.5 | 2.5 | 4 | 3.5 | 3.5 | 4.5 | 2 | 5 | 4 | 4.5 | 2.5 | 3 | 3 | 3 |
| Sisense | 3 | 3 | 3 | 5 | 3 | 3.5 | 2 | 2 | 3 | 3 | 3 | 5 | 2.5 | 5 | 3 |
| Tableau | 3.5 | 4 | 3 | 4 | 3 | 2.5 | 3 | 1.5 | 4 | 4 | 4 | 3 | 2 | 5 | 4 |
| ThoughtSpot | 4 | 2 | 2 | 4 | 2 | 3 | 1 | 2 | 2 | 2.5 | 2.5 | 2 | 2 | 5 | 4 |
| TIBCO Software | 4 | 4 | 3 | 4 | 3 | 3 | 4 | 2 | 4 | 4.5 | 2.5 | 4.5 | 3 | 4 | 3 |
| Yellowfin | 3 | 2.5 | 2 | 2.5 | 3 | 3.5 | 2 | 1.5 | 3.5 | 4 | 4.5 | 4 | 4.5 | 5 | 3.5 |
| Zoomdata | 3.5 | 3 | 2.5 | 3 | 2 | 2.5 | 2 | 1 | 2 | 3 | 2 | 5 | 1.5 | 4 | 3 |

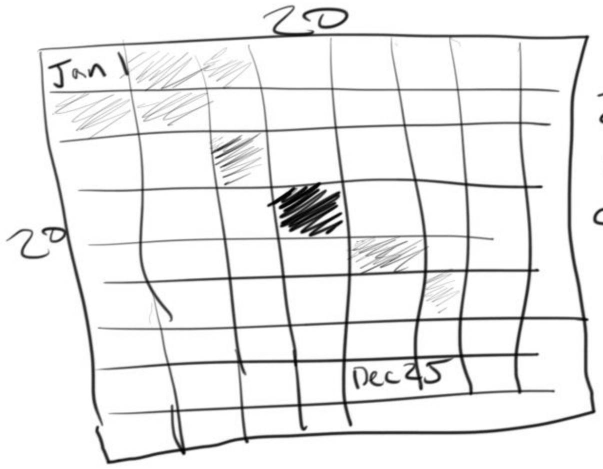
1 = Poor or Absent; 2 = Fair; 3 = Good; 4 = Excellent; 5 = Outstanding

© 2017 Gartner, Inc.

## Sketches & Ideation

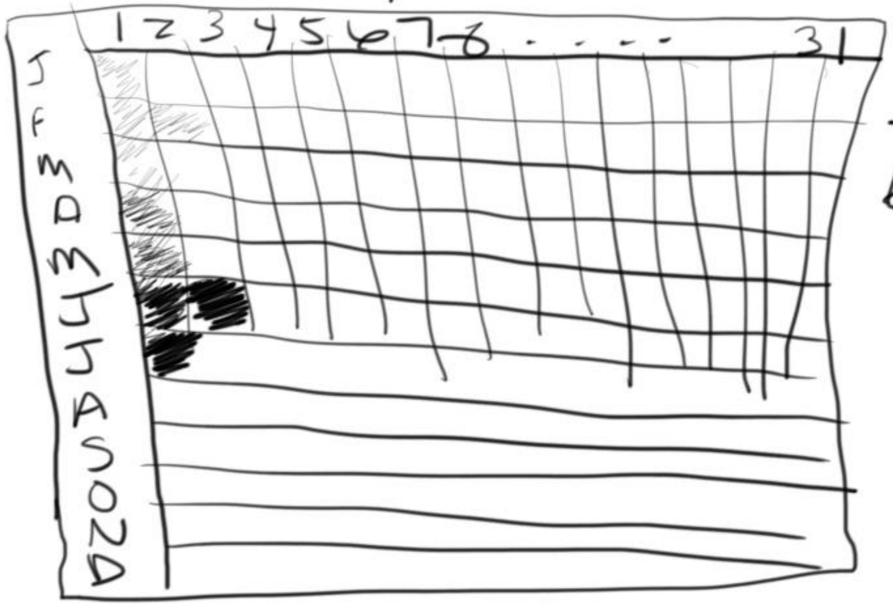


52 week x 7 day layout
Maybe too wide



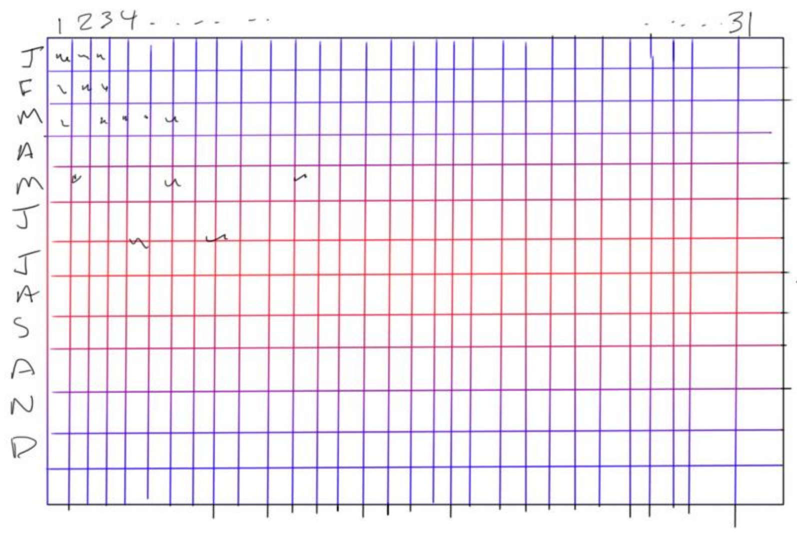
20x20 layout
Not enough days to fill, but looks good as a square

31x12 layout



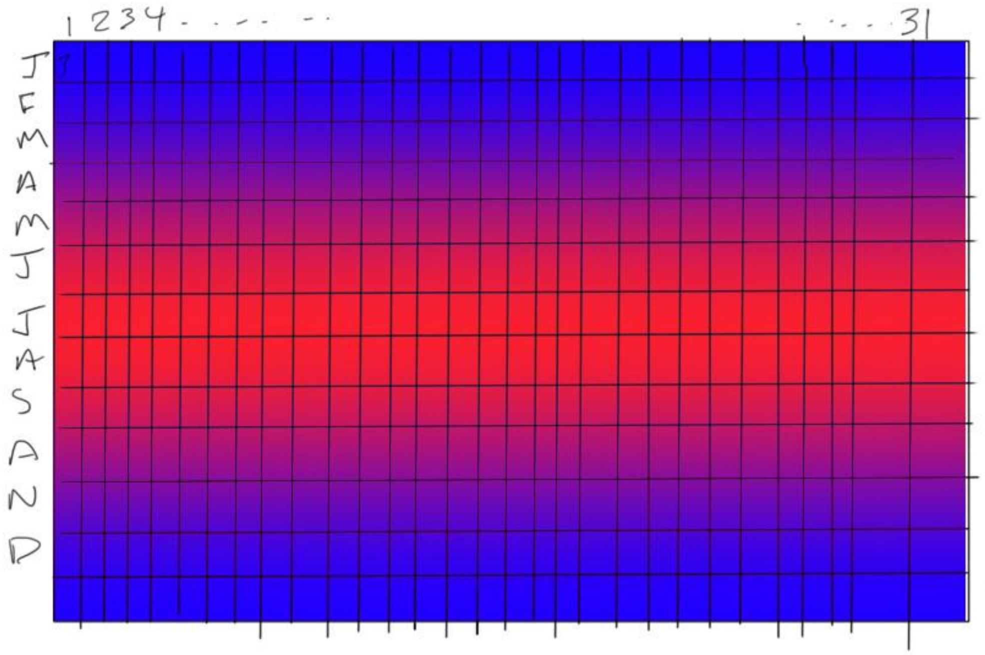
Data points would be tall rectangles

## Sketches & Ideation



Grid lines are colors instead of data squares.

Temp data would be on the grid

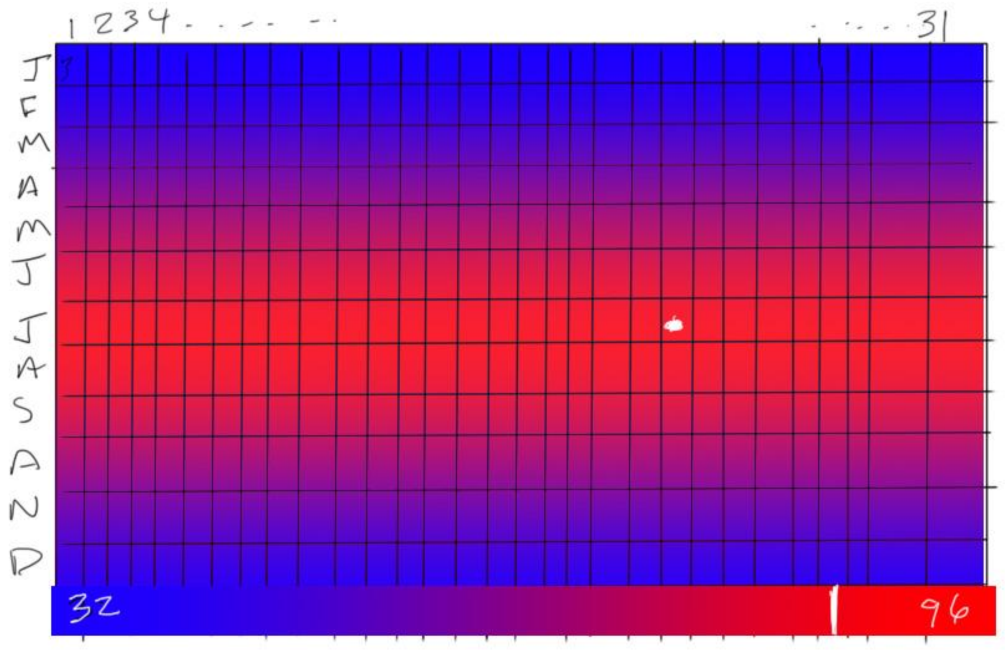


Gradient fill on data squares

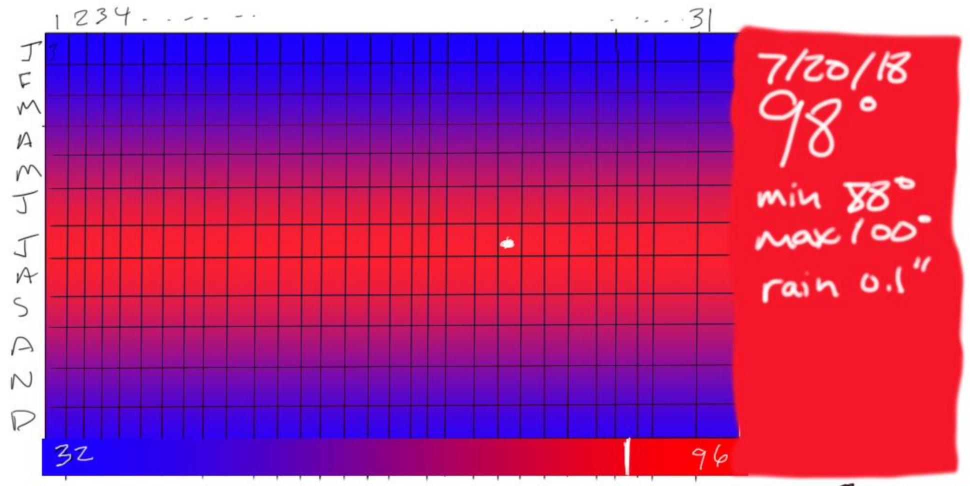
Would be more of a stained glass look with only 1 year of data.

May look good.

## Sketches & Ideation

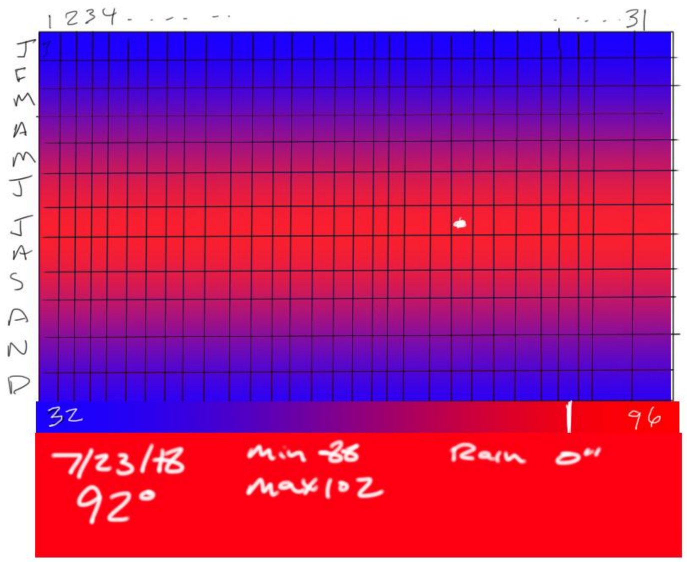


gradient legend across bottom with indicator on mouse hovering



info panel that changes color on mouse hover

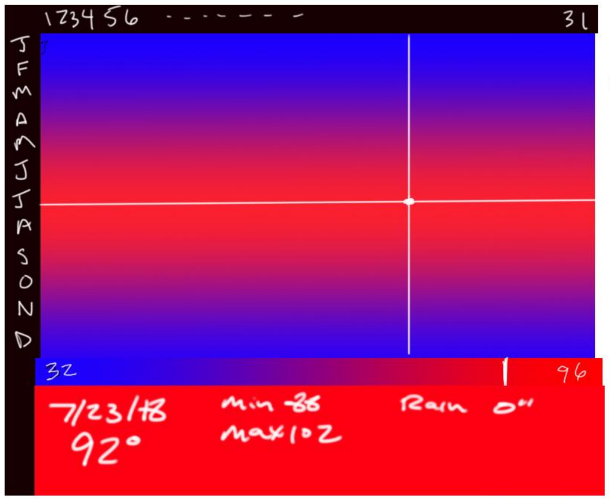
## Sketches & Ideation



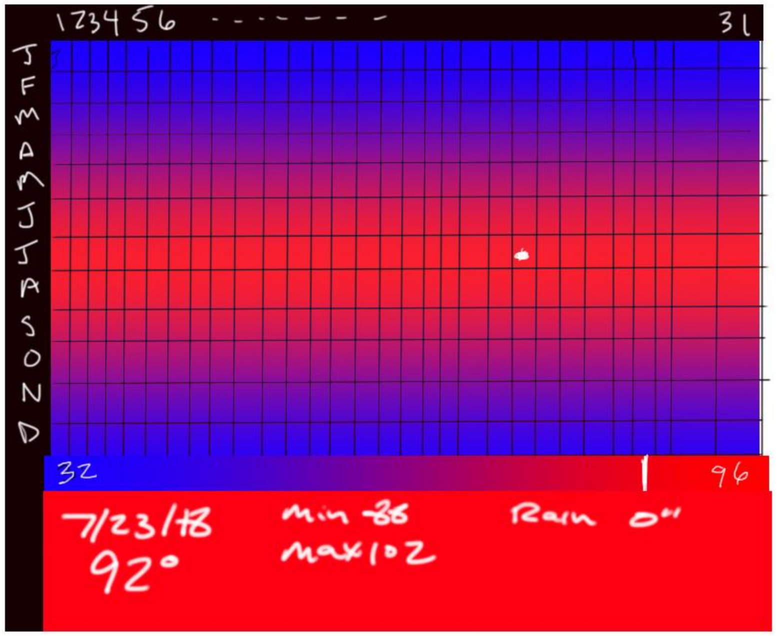
Info panel on bottom

color change on mouse hover

Info update on mouse hover



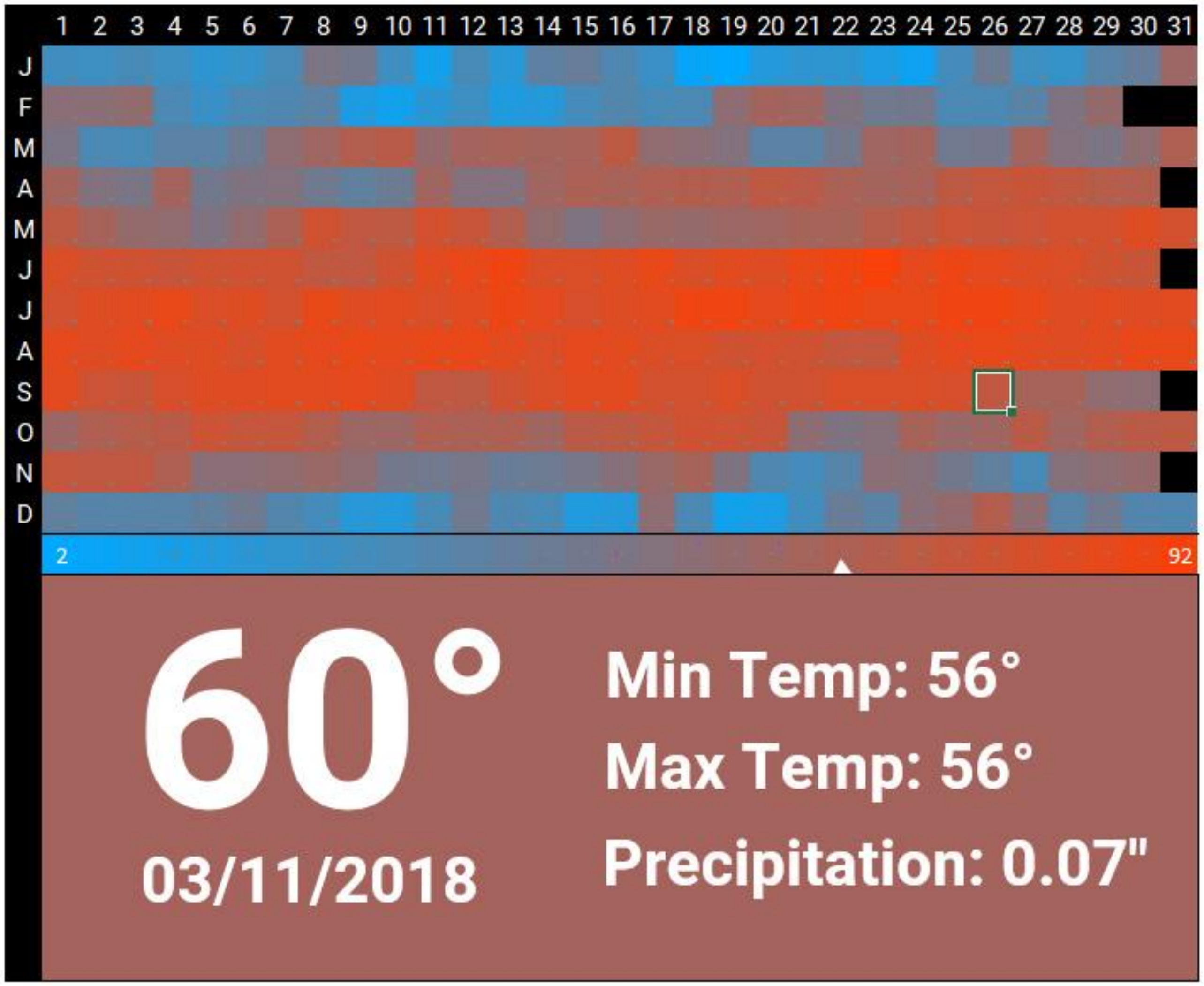
Crosshair design that follows mouse



Black axes with white font gives good contrast

Aspect ratio looks good

# Previsualization Rendering



Previsualization created in Excel and Photoshop

- Low temp color: #00a8ff
- High temp color: #f8410a
- Bottom info panel background will change colors based on mouse hovering over a date
- Indicator on legend will update as mouse is moved
- Left and right keyboard arrows will select the year to load into the visualization

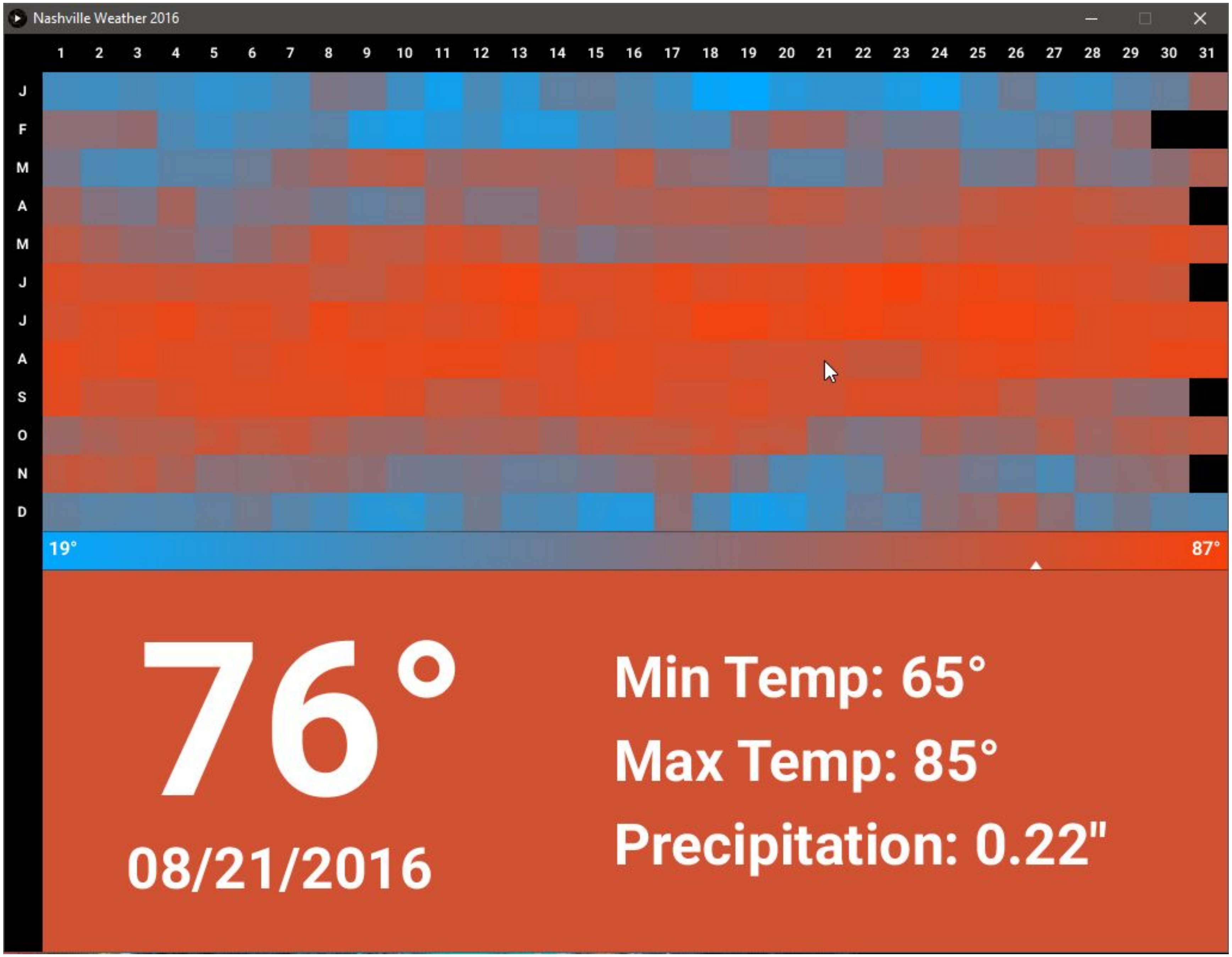
## Pseudocode

```
• Initialize global variables
• Set sketch size and smooth
• Set background to black
• Calculate legend and details panel heights
• Create month letter abbreviations
• Create font
• Import data
    ○ Load table
    ○ Calculate min and max year in data
• Initially call display cards function with minimum
  year found in data
• Display cards function (takes year for parameter)
    ○ Get rows for year from table data
    ○ Set title bar for sketch to year
    ○ Create new WeatherRecord object for each
      day and store in array
    ○ Determine min and max temp for year
    ○ Clear existing card array
    ○ Black out card display area
    ○ Create new Card object for each
      WeatherRecord in array
    ○ Display the card
• Create axes
    ○ Write numbers 1-31 in loop across top row
    ○ Offset each iteration by square size up to the
      width
    ○ Write month letters abbreviations in loop
      down left side of sketch
    ○ Offset each iteration by square size up to top
      of legend area
• Create legend
    ○ Start at offset for left axis and go to width of
      sketch
    ○ Use lerp color to draw colored lines across min
      max temp range for year
    ○ Draw text for min temp for year on left side of
      legend
    ○ Draw text for max temp for year on right side
      of legend
• When drawing
    ○ Check each card to determine if mouse is
      hovering over it
    ○ Update info panel
    ○ Create legend again to remove legend
      indicator
    ○ Draw triangle indicator on top of legend for
      current day temp
```

## Pseudocode

- When updating info panel (takes currently hovered card as parameter)
    - Get the color for the card
    - Draw a rectangle across the info panel in correct color
    - Draw text info from WeatherRecord on the card
- When hitting left and right arrows on keyboard
    - Call display card function with current year minus 1 if left arrow was pressed and current year is not less than or equal to min year from data
    - Call display card function with current year plus 1 if right arrow was pressed and current year is not greater than or equal to max year from data

## Final Output



76°

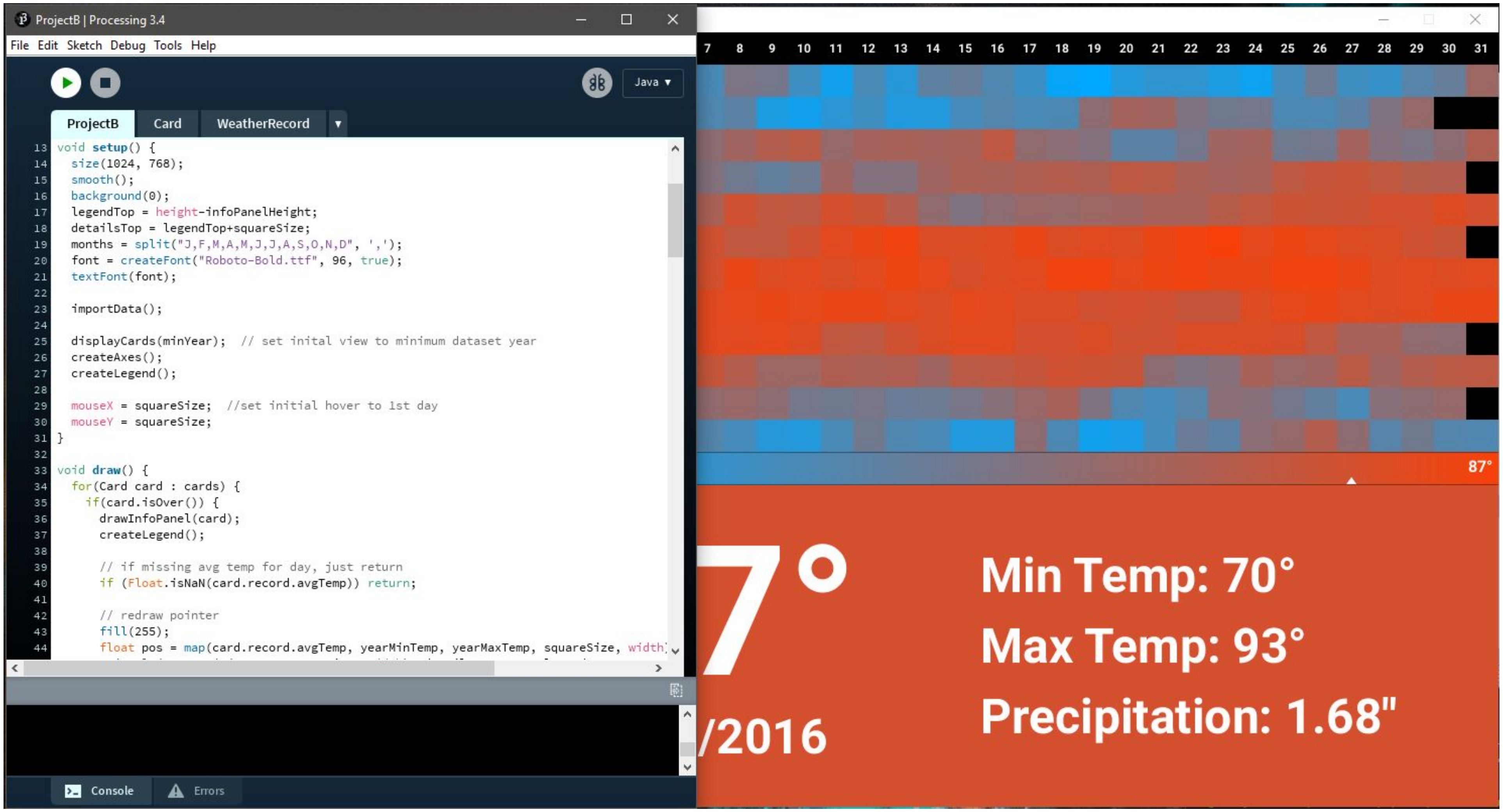
08/21/2016

Min Temp: 65°

Max Temp: 85°

Precipitation: 0.22"

# Final Output



```
void setup() {
  size(1024, 768);
  smooth();
  background(0);
  legendTop = height-infoPanelHeight;
  detailsTop = legendTop+squareSize;
  months = split("J,F,M,A,M,J,J,A,S,O,N,D", ',');
  font = createFont("Roboto-Bold.ttf", 96, true);
  textFont(font);

  importData();

  displayCards(minYear);  // set inital view to minimum dataset year
  createAxes();
  createLegend();

  mouseX = squareSize;  //set initial hover to 1st day
  mouseY = squareSize;
}

void draw() {
  for(Card card : cards) {
    if(card.isOver()) {
      drawInfoPanel(card);
      createLegend();

      // if missing avg temp for day, just return
      if (Float.isNaN(card.record.avgTemp)) return;

      // redraw pointer
      fill(255);
      float pos = map(card.record.avgTemp, yearMinTemp, yearMaxTemp, squareSize, width
```

7°

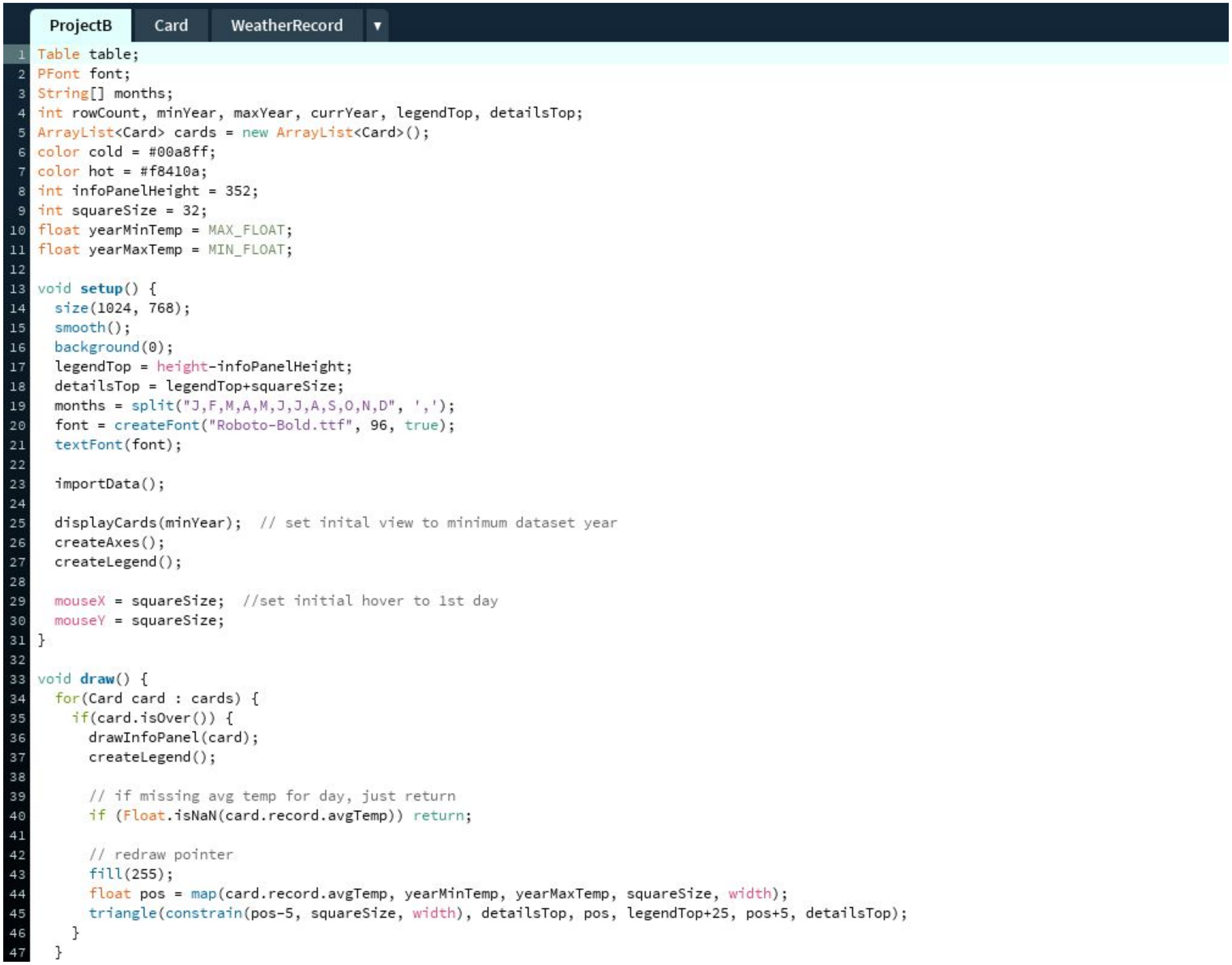
/2016

Min Temp: 70°

Max Temp: 93°

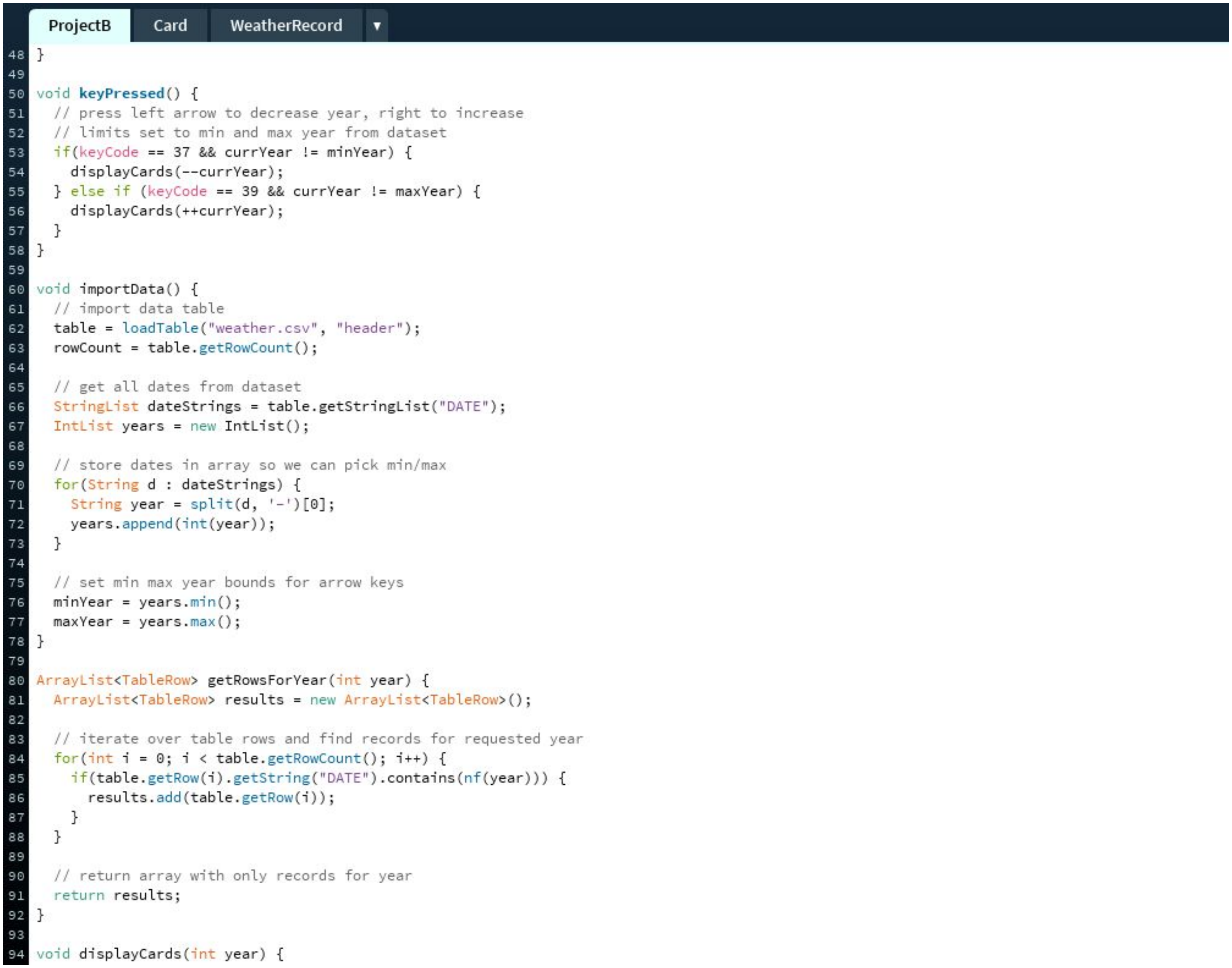
Precipitation: 1.68"

# Final Output



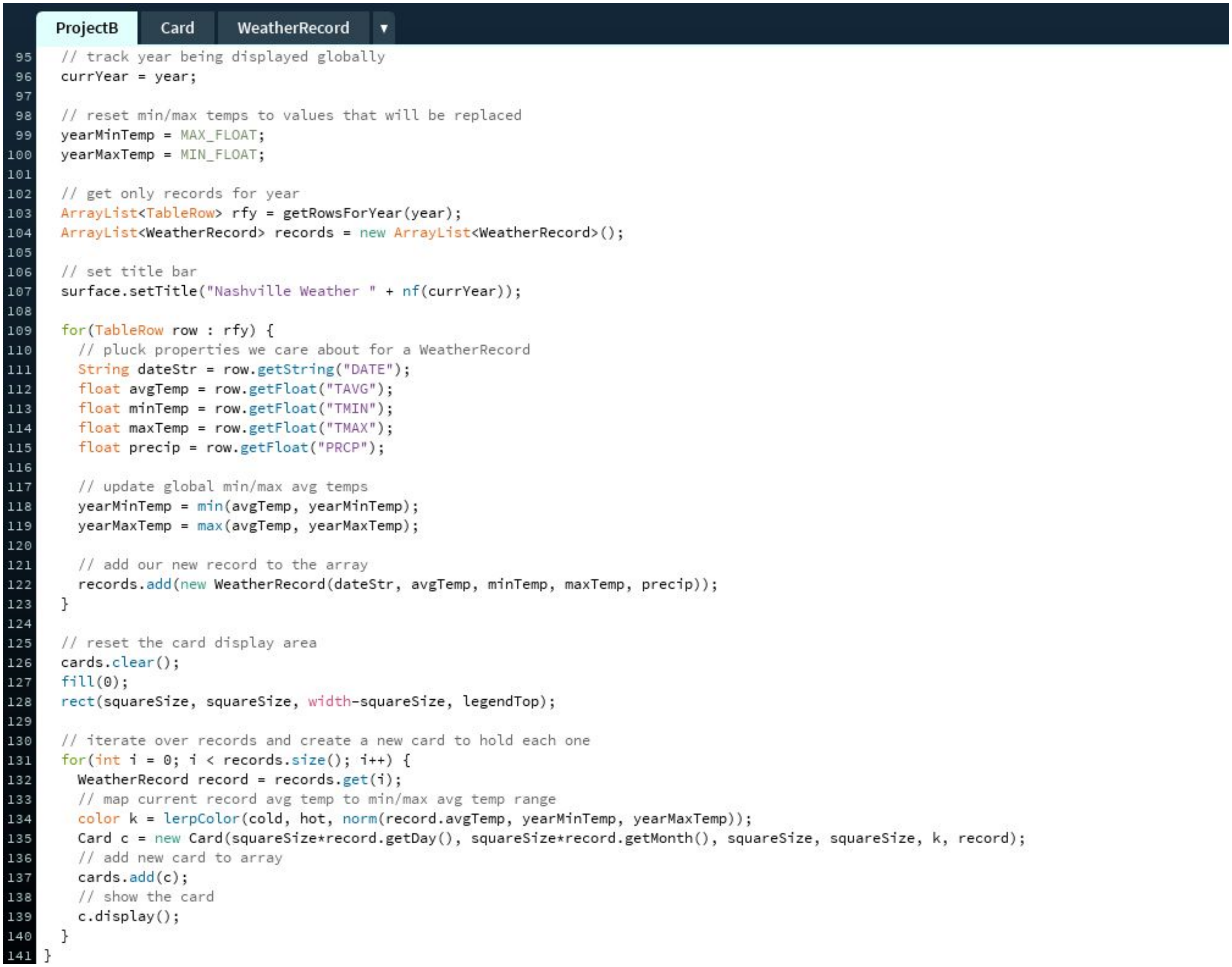
```
Table table;
PFont font;
String[] months;
int rowCount, minYear, maxYear, currYear, legendTop, detailsTop;
ArrayList<Card> cards = new ArrayList<Card>();
color cold = #00a8ff;
color hot = #f8410a;
int infoPanelHeight = 352;
int squareSize = 32;
float yearMinTemp = MAX_FLOAT;
float yearMaxTemp = MIN_FLOAT;

void setup() {
  size(1024, 768);
  smooth();
  background(0);
  legendTop = height-infoPanelHeight;
  detailsTop = legendTop+squareSize;
  months = split("J,F,M,A,M,J,J,A,S,O,N,D", ',');
  font = createFont("Roboto-Bold.ttf", 96, true);
  textFont(font);

  importData();

  displayCards(minYear);  // set inital view to minimum dataset year
  createAxes();
  createLegend();

  mouseX = squareSize;  //set initial hover to 1st day
  mouseY = squareSize;
}

void draw() {
  for(Card card : cards) {
    if(card.isOver()) {
      drawInfoPanel(card);
      createLegend();

      // if missing avg temp for day, just return
      if (Float.isNaN(card.record.avgTemp)) return;

      // redraw pointer
      fill(255);
      float pos = map(card.record.avgTemp, yearMinTemp, yearMaxTemp, squareSize, width);
      triangle(constrain(pos-5, squareSize, width), detailsTop, pos, legendTop+25, pos+5, detailsTop);
    }
  }
```

# Final Output



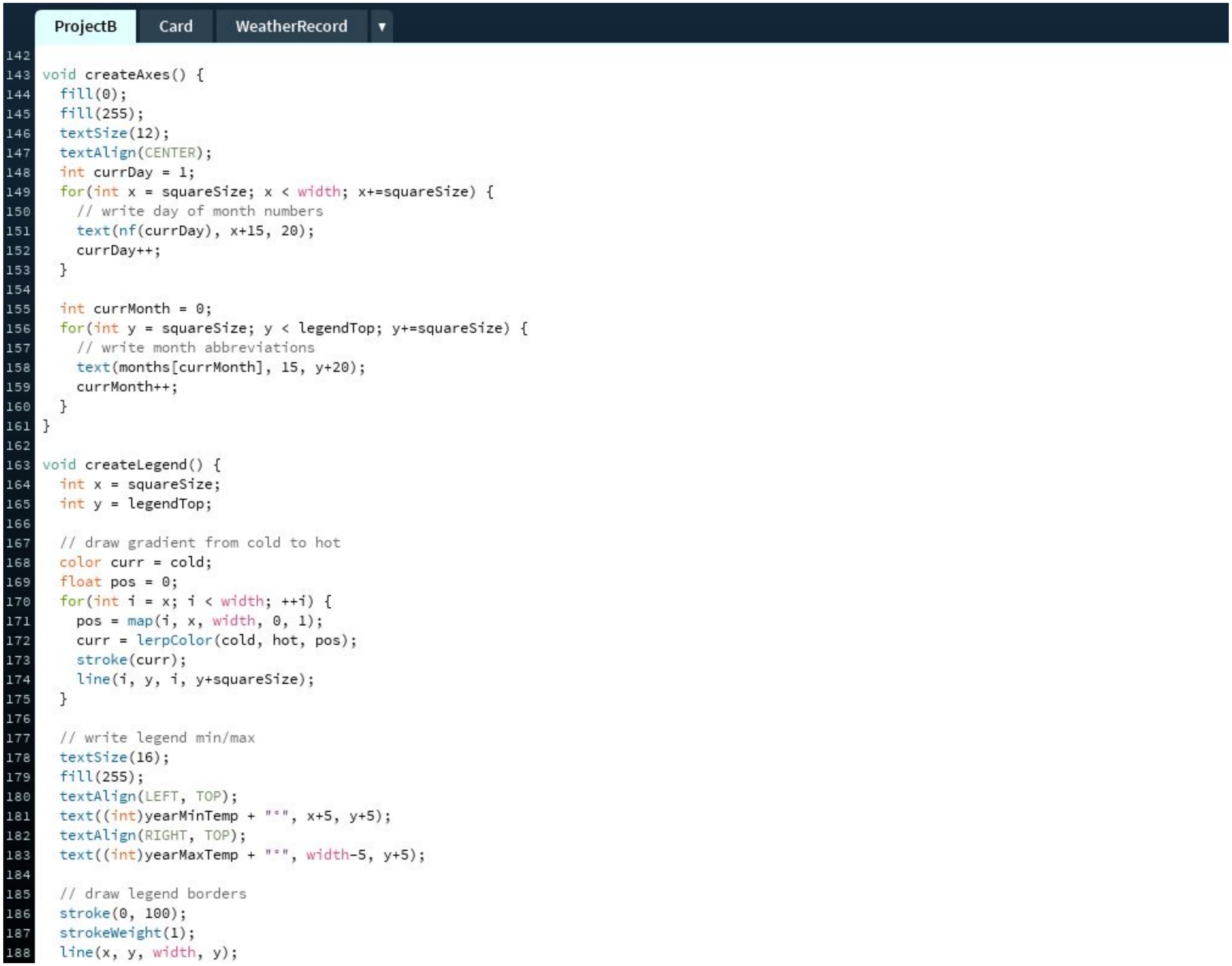
```
}

void keyPressed() {
  // press left arrow to decrease year, right to increase
  // limits set to min and max year from dataset
  if(keyCode == 37 && currYear != minYear) {
    displayCards(--currYear);
  } else if (keyCode == 39 && currYear != maxYear) {
    displayCards(++currYear);
  }
}

void importData() {
  // import data table
  table = loadTable("weather.csv", "header");
  rowCount = table.getRowCount();

  // get all dates from dataset
  StringList dateStrings = table.getStringList("DATE");
  IntList years = new IntList();

  // store dates in array so we can pick min/max
  for(String d : dateStrings) {
    String year = split(d, '-')[0];
    years.append(int(year));
  }

  // set min max year bounds for arrow keys
  minYear = years.min();
  maxYear = years.max();
}

ArrayList<TableRow> getRowsForYear(int year) {
  ArrayList<TableRow> results = new ArrayList<TableRow>();

  // iterate over table rows and find records for requested year
  for(int i = 0; i < table.getRowCount(); i++) {
    if(table.getRow(i).getString("DATE").contains(nf(year))) {
      results.add(table.getRow(i));
    }
  }

  // return array with only records for year
  return results;
}

void displayCards(int year) {
```

Final Output



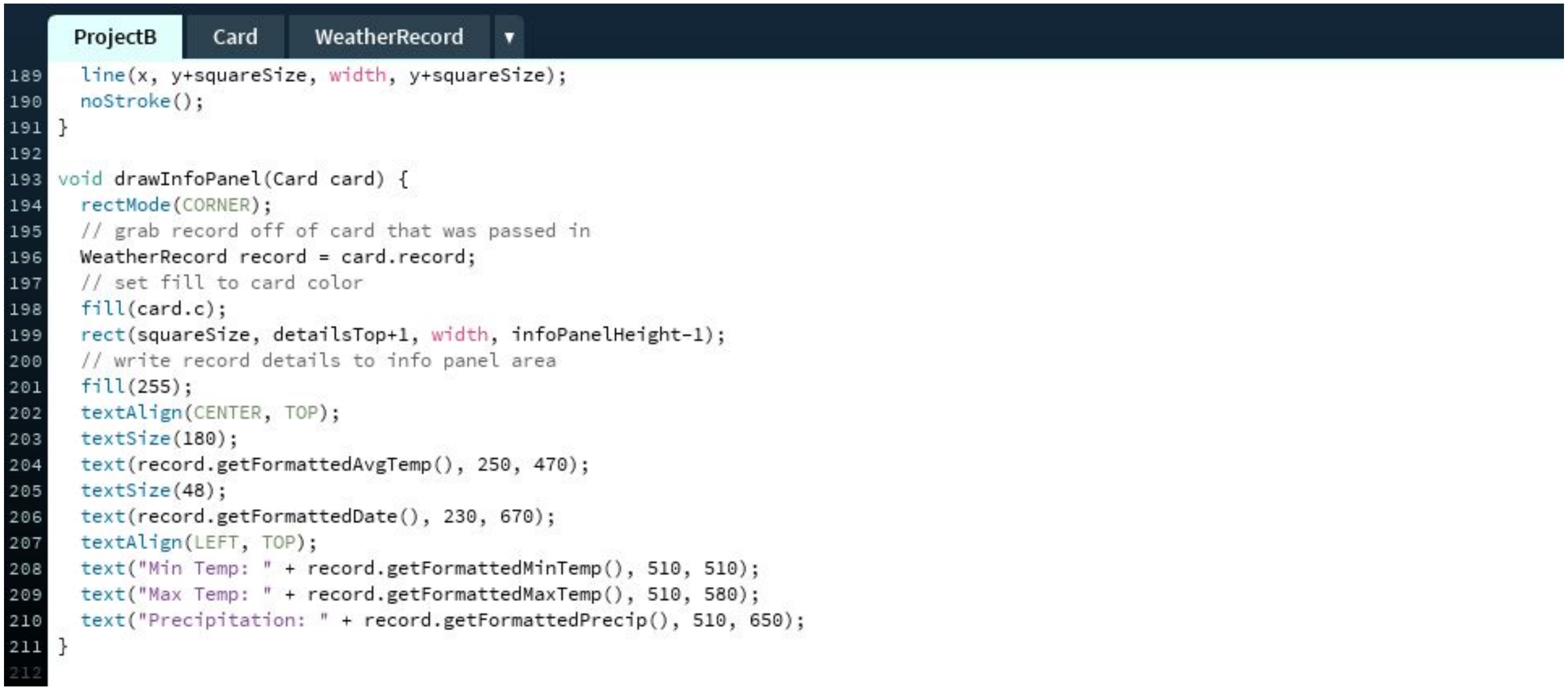
```
  // track year being displayed globally
  currYear = year;

  // reset min/max temps to values that will be replaced
  yearMinTemp = MAX_FLOAT;
  yearMaxTemp = MIN_FLOAT;

  // get only records for year
  ArrayList<TableRow> rfy = getRowsForYear(year);
  ArrayList<WeatherRecord> records = new ArrayList<WeatherRecord>();

  // set title bar
  surface.setTitle("Nashville Weather " + nf(currYear));

  for(TableRow row : rfy) {
    // pluck properties we care about for a WeatherRecord
    String dateStr = row.getString("DATE");
    float avgTemp = row.getFloat("TAVG");
    float minTemp = row.getFloat("TMIN");
    float maxTemp = row.getFloat("TMAX");
    float precip = row.getFloat("PRCP");

    // update global min/max avg temps
    yearMinTemp = min(avgTemp, yearMinTemp);
    yearMaxTemp = max(avgTemp, yearMaxTemp);

    // add our new record to the array
    records.add(new WeatherRecord(dateStr, avgTemp, minTemp, maxTemp, precip));
  }

  // reset the card display area
  cards.clear();
  fill(0);
  rect(squareSize, squareSize, width-squareSize, legendTop);

  // iterate over records and create a new card to hold each one
  for(int i = 0; i < records.size(); i++) {
    WeatherRecord record = records.get(i);
    // map current record avg temp to min/max avg temp range
    color k = lerpColor(cold, hot, norm(record.avgTemp, yearMinTemp, yearMaxTemp));
    Card c = new Card(squareSize*record.getDay(), squareSize*record.getMonth(), squareSize, squareSize, k, record);
    // add new card to array
    cards.add(c);
    // show the card
    c.display();
  }
}
```

## Final Output



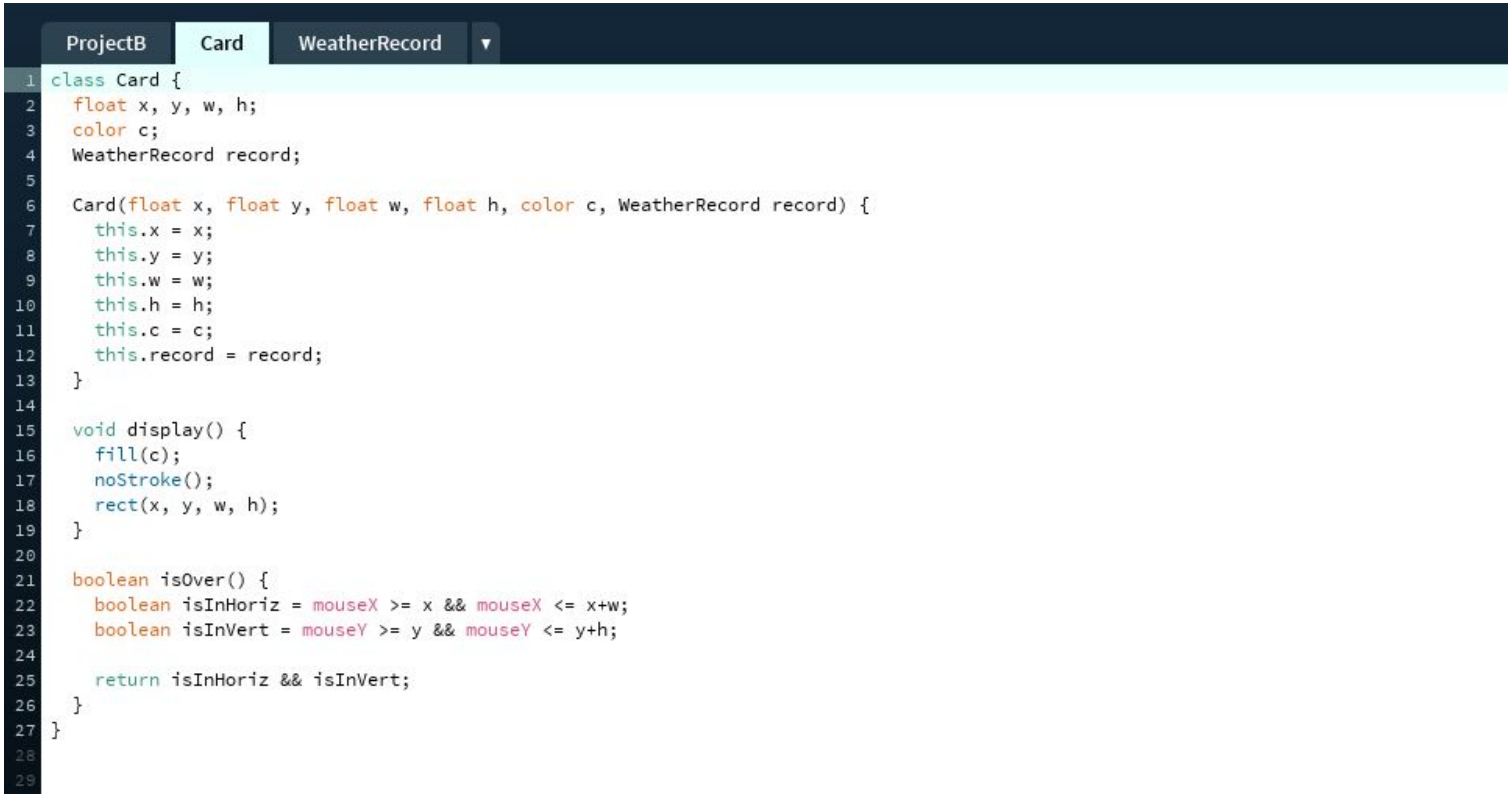
```

void createAxes() {
  fill(0);
  fill(255);
  textSize(12);
  textAlign(CENTER);
  int currDay = 1;
  for(int x = squareSize; x < width; x+=squareSize) {
    // write day of month numbers
    text(nf(currDay), x+15, 20);
    currDay++;
  }

  int currMonth = 0;
  for(int y = squareSize; y < legendTop; y+=squareSize) {
    // write month abbreviations
    text(months[currMonth], 15, y+20);
    currMonth++;
  }
}

void createLegend() {
  int x = squareSize;
  int y = legendTop;

  // draw gradient from cold to hot
  color curr = cold;
  float pos = 0;
  for(int i = x; i < width; ++i) {
    pos = map(i, x, width, 0, 1);
    curr = lerpColor(cold, hot, pos);
    stroke(curr);
    line(i, y, i, y+squareSize);
  }

  // write legend min/max
  textSize(16);
  fill(255);
  textAlign(LEFT, TOP);
  text((int)yearMinTemp + "°", x+5, y+5);
  textAlign(RIGHT, TOP);
  text((int)yearMaxTemp + "°", width-5, y+5);

  // draw legend borders
  stroke(0, 100);
  strokeWeight(1);
  line(x, y, width, y);
```

## Final Output



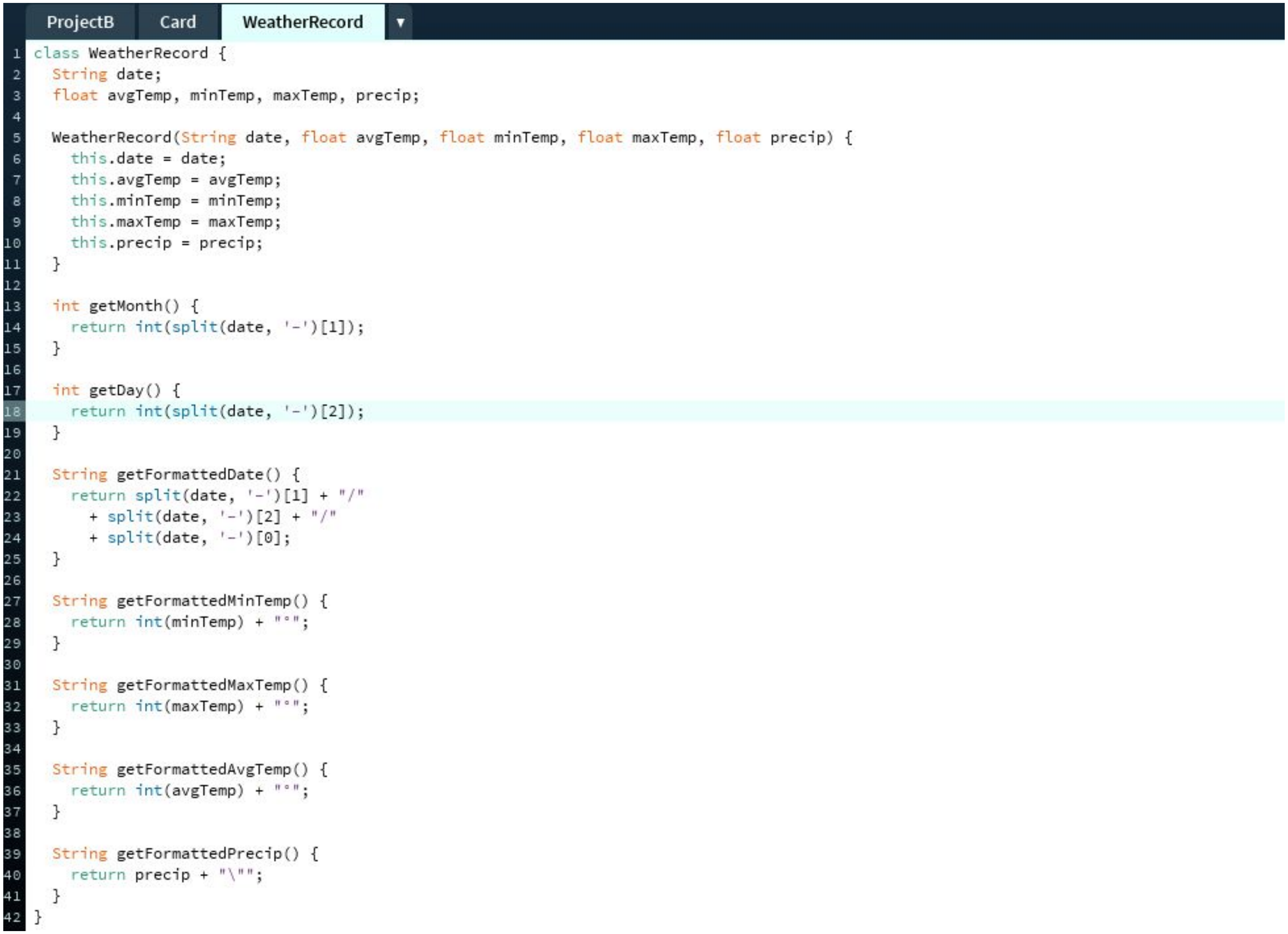
```
  line(x, y+squareSize, width, y+squareSize);
  noStroke();
}

void drawInfoPanel(Card card) {
  rectMode(CORNER);
  // grab record off of card that was passed in
  WeatherRecord record = card.record;
  // set fill to card color
  fill(card.c);
  rect(squareSize, detailsTop+1, width, infoPanelHeight-1);
  // write record details to info panel area
  fill(255);
  textAlign(CENTER, TOP);
  textSize(180);
  text(record.getFormattedAvgTemp(), 250, 470);
  textSize(48);
  text(record.getFormattedDate(), 230, 670);
  textAlign(LEFT, TOP);
  text("Min Temp: " + record.getFormattedMinTemp(), 510, 510);
  text("Max Temp: " + record.getFormattedMaxTemp(), 510, 580);
  text("Precipitation: " + record.getFormattedPrecip(), 510, 650);
}
```

## Final Output



```
class Card {
  float x, y, w, h;
  color c;
  WeatherRecord record;

  Card(float x, float y, float w, float h, color c, WeatherRecord record) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.c = c;
    this.record = record;
  }

  void display() {
    fill(c);
    noStroke();
    rect(x, y, w, h);
  }

  boolean isOver() {
    boolean isInHoriz = mouseX >= x && mouseX <= x+w;
    boolean isInVert = mouseY >= y && mouseY <= y+h;

    return isInHoriz && isInVert;
  }
}
```

# Final Output



```
class WeatherRecord {
  String date;
  float avgTemp, minTemp, maxTemp, precip;

  WeatherRecord(String date, float avgTemp, float minTemp, float maxTemp, float precip) {
    this.date = date;
    this.avgTemp = avgTemp;
    this.minTemp = minTemp;
    this.maxTemp = maxTemp;
    this.precip = precip;
  }

  int getMonth() {
    return int(split(date, '-')[1]);
  }

  int getDay() {
    return int(split(date, '-')[2]);
  }

  String getFormattedDate() {
    return split(date, '-')[1] + "/"
      + split(date, '-')[2] + "/"
      + split(date, '-')[0];
  }

  String getFormattedMinTemp() {
    return int(minTemp) + "°";
  }

  String getFormattedMaxTemp() {
    return int(maxTemp) + "°";
  }

  String getFormattedAvgTemp() {
    return int(avgTemp) + "°";
  }

  String getFormattedPrecip() {
    return precip + "\"";
  }
}
```