Log

- 10AM: Read the email and went back to sleep
- 11AM : Tried to download the data set
- 11:15: Opened the data in Pycharm Jupyter project
    - Copied the data into a new project
    - Setup a conda environment
    - Installed pandas package
    - Opened the data. Did some analysis
    - Checked most common post codes
    - Checked the number of rows
    - Checked if there was some min/max lat longs I could use
- 11:30: Started looking into ASCII art. I had seen ASCII art before but I had never made one myself
- 11:45: Started reading this blog : https://gamepopper.co.uk/2015/11/20/understanding-ascii-art/
- 12: Thought of doing this in Java.
- 12:15: Had an initial "non-art"

```
....................................................................................
....................................................................................
....................................................................................
.................f..............................................................
....................................................................................
....................................................................................
....................................................................................
................................................................................
....................................................................................
....................................................................................
....................................................................................
................................................................................
....................................................................................
....................................................................................
....................................................................................
................................................................................
....................................................................................
....................................................................................
```

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

```
............................................................................
........................................................................
............................................................................
............................................................................
...................................................................fff......
........................................................................
............................................................................
............................................................................
..............................................f.fff.ff...fff.fff.ff.f.......
........................................................................
............................................................................
............................................................................
...........................................ffffffffffffffffff..fff....ff....
........................................................................
............................................................................
............................................................................
................................f............f....fffffffffff...............
........................................................................
............................................................................
............................................................................
.....................................................fff....................
........................................................................
............................................................................
............................................................................
............................................f...............................
........................................................................
............................................................................
............................................................................
...ff.ff..f..f.fff.fff......................................................
........................................................................
............................................................................
.....................................................................fffffff
fffffffff.f..ffff...........................................................
........................................................................
............................................................................
.....................................................................fffffff
ffffffffffff................................................................
........................................................................
.......................................................ffff.................
................fff...f.fffffff.....fff.fffff.f..f.ff..fffffffffffffffffffff
```

```
ff..........................................................................
........................................................................
..................................fffffffffffffffffffffffffff...............
.......fff..ffffffff.f..f.........f.fffff.ff..ff...ff.....f...f.ffffffffffff
ff..........................................................................
........................................................................
...............................ffffffffffffff.ffffffffff.ff.................
.....fffffffffff..f.f...ff.ff.fff.f.fff...fffffffffffff.fffff..fff..........
............................................................................
........................................................................
..........................f..ff..ffffffffffff..........................fffff
ffffffffff.ffffffff.......f...f..fffffffffffffffffffffff........ffffffff....
......fff...................................................................
........................................................................
...................ffffffffffffff.........fffffff..ffffffffff...ff...f.fffff
fffffff..ff...f.ff.f.f.ffff.f.ffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffff...................................
........................................................................
......................fffffff............fffffffffffffffffffffffffffffffffff
fffffffff......f........fff.fffffffffffffffffffffff.f.ffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffff....................................
........................................................................
...................ffffffff.......................f.....f.f...fffffffffff.ff
ffff..f.f....fff...f.fffff.fff...ff..f.ffff.ffffffffffffffff....f..fffffffff
fffffffffffffffffffffffffffffffff...........................................
........................................................................
..................ff......................................ffffff....ffffffff
fff..f....ffffffffffff.ffff.f..f.....f.f.fff.ffffffffffffff.f.ffffffffffffff
ffffffffffffffffffffffffffff................................................
........................................................................
...................................fffffffff..f..f.....fffffffffffffffffffff
ffffffffffffffffffff....ffff....f.f.ffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffff.......................................................
........................................................................
.....................................................ffffffffffffff..f..ffff
fffffffff.ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffff............................................................
........................................................................
.........................................................fff.........ffff.ff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

```
ffffffffffffffffffffffff....................................................
........................................................................
.................................................ffffffffffffffff.ff.......f
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffff..f.....................................
........................................................................
.................................................ff.....ffffff.f........ffff
f.ffff.ffffffffff......fffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffff.............................
........................................................................
...........................................ffffffffffffffffffff..........fff
fff...................ffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffff...........................
........................................................................
......................fffffffffffffffffffffffffffffffffffffffffffffff.......
..............fffffffffffffffffffffff.f.ffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffff......................
........................................................................
..........fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.
..............ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.................
........................................................................
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffff.............fff.....................................ffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ff......................................................................
.fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffff..........................................................ffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffff..............................................................
.................fffffffffff.................ffffffffffffffffffffffff.......
.......................................................................fffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffff........................................................
............................................................................
............................................................................
.fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffff...................................................
............................................................................
....................................f.f.....................................
```

```
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffff................................................
............................................................................
............................ffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffff...........................................
............................................................................
...............................fffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffff...fffffffffffffffffffffffffff.............
............................................................................
.........................ffffffffffffffff.f.ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff..
............................................................................
.............................................fffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
............................................................................
...................................fffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff..
............................................................................
...........fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.....
............................................................................
..........ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffff...............
............................................................................
..................fffffffffff...........ffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffff..........................
............................................................................
.............................f.............fffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff..........
............................................................................
................................ffffffffffffffffffffffffffffffffffffffffffff
```

```
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff............
............................................................................
..........................ffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffff.................................
............................................................................
...............ffffffffffffffffffffffffffffffffffffffffffffffffffffffff.....
................ffffffffffff.fffffff..............fffffffff.................
........................................................................
..........................................................................ff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff...........
............................................................................
........................................................................
......................................................fff................fff
fffffffffffffffffff.........................................................
............................................................................
........................................................................
......................................................f.....................
............................................................................
............................................................................
........................................................................
```

- The above is not really an art just a layout of how the points look like visualized on a `300*200` grid.
- Here it is on a smaller grid

```
  .......f..
..........
..........
..........
.....ff...
....fff...
ffffff....
ffffff....
ffffff....
fffffff...
ffffffff..
...ffffff.
..ffffffff
```

```
..fffffff.
.ffffffff.
.f........
```

- Kind of looks a bit like UK already doesn't it? I think perhaps the far off coordinate at the top is maybe some island off of Scotland?
- Asked ChatGPT about ASCII Art, Maps, and learned a lot of stuff about bounding boxes, converting pixels to ascii grids etc
- Here is a docstring ascii art of UK created by ChatGPT

```
               .-"      "-.
            .-"              "-.
         .-"                    "-.
        |                        |
        |     Scotland           |
         \                      /
          \                   _/
           |      N. Ireland  /
           |                /
          /    England    /
         /               /
      .-"              ."
     /    Wales       /
    |                /
    |               /
     \            ."
      "-.      .-"
         "-._.-"
```

- 2PM: I have been laid off so I realized that I will take my time with this project and try to add some creative elements if possible
- 2:15PM: Started messing around with sorting
  - Very basic idea of sorting
  - Realized something, there's a point in here that is "off", like not, Scotland off but wayy wayy off.

```
........................................f.........
..................................................
..................................................
..................................................
```

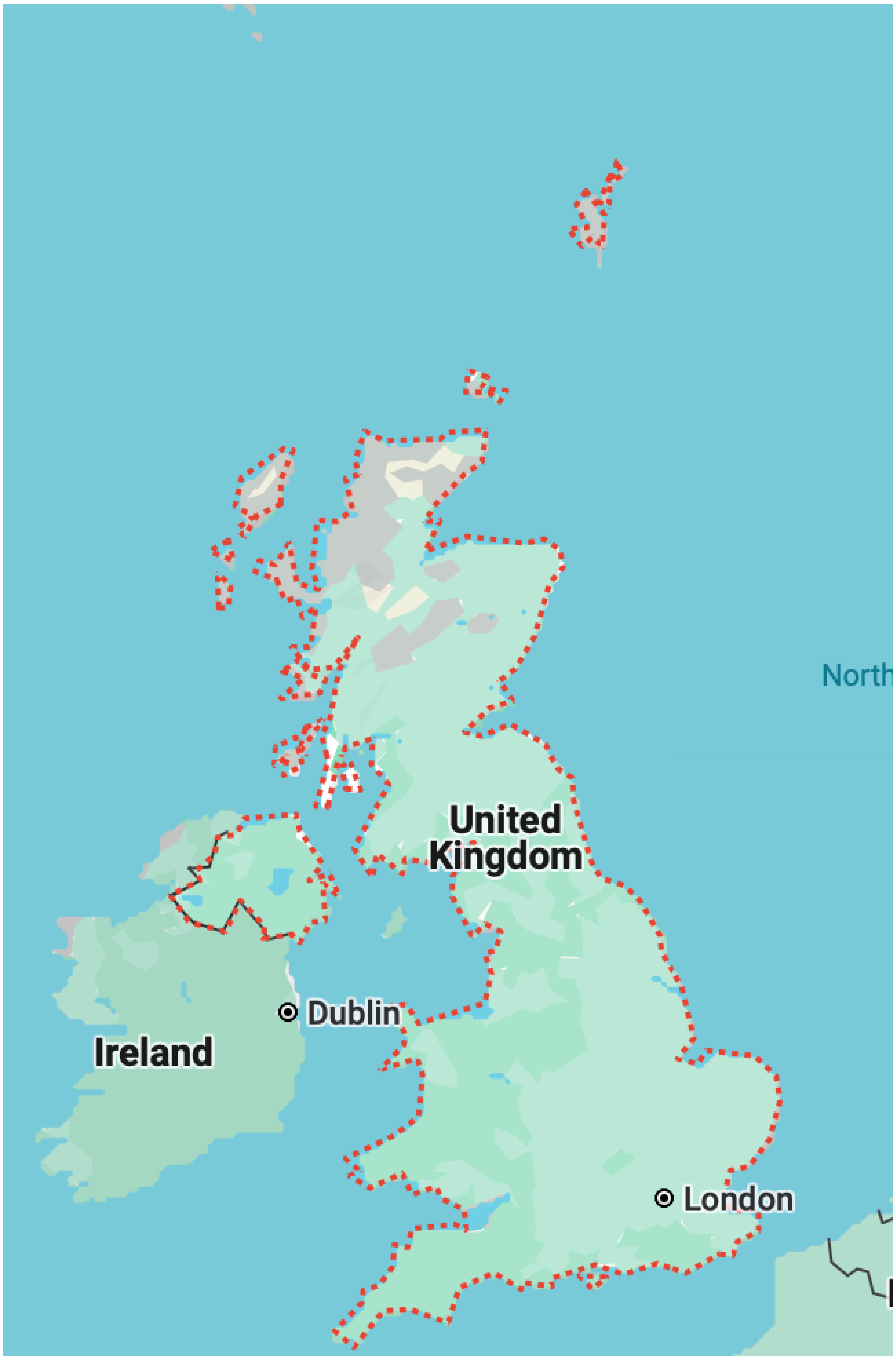
```
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
.................................ffff.............
..............................fffffff.............
................................fff...............
.......................ffffff.....................
.........f......fffffffffff.......................
.....fffff...fffffffffffff........................
...fffffffffffffffffffffffffffff..................
...ff.ffffffffffffffffffffffffff..................
...f.fffffffffffffffffffffffff....................
......f.fffffffffffffffffffff.....................
........fffffffffffffffffffffffff.................
...ffffffffff.ffffffffffffffffffff................
ffffffffffffffffffffffffffffffffffffff............
ffffffffffffff........ffffffffffffffffff..........
.........................fffffffffffffffff........
.................ffffffffffffffffffffffffffffff...
................ffffffffffffffffffffffffffffffffff
..............fffffffffffffffffffffffffffffffffff.
..............fffffffffffffffffffffffffffffffff...
```

```
.................fffffffffffffffffffffffffffffff..
..............ffffffffffffffffffffffffffffff......
........ff..ffffffffffff..........................
........f.........................................
```

- If you see this grid has this lone point up top which is "off"
- Googled to check what are the rational boundaries of UK
- It seems UK lies somewhere between 49 to 59 Long and -2 to 8 Lat,
- But then I saw something interesting on Google Maps

```

```


North
United Kingdom
Dublin
Ireland
London

- Do you see the little red dots up top? Those I think are the Shetland Islands and I believe this data includes the co-ordinates from the Shetland Islands.

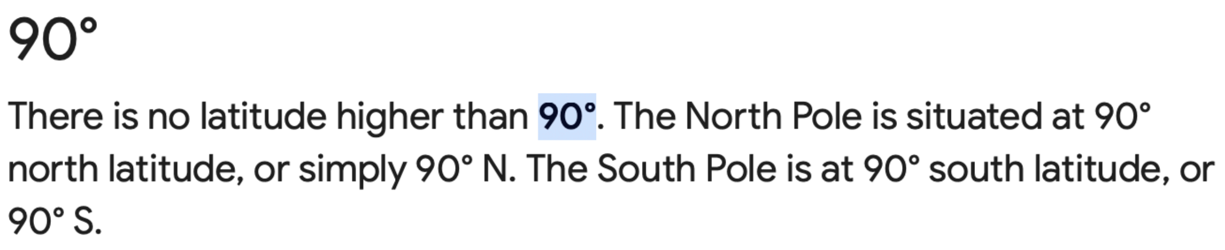
3PM: **New Realization**

- Oops looks like there's some *really really* weird points in here
- Even accounting for Shetlands Lats and Longs.

```
              id  postcode   latitude  longitude
1467109  1538687  BN91 9AA  99.999999        0.0
1534108  1608927  BD98 1GA  99.999999        0.0
1534109  1608928  BD98 1GB  99.999999        0.0
1534110  1608929  BD98 1GD  99.999999        0.0
1534111  1608930  BD98 1GG  99.999999        0.0

Panda's output.
```

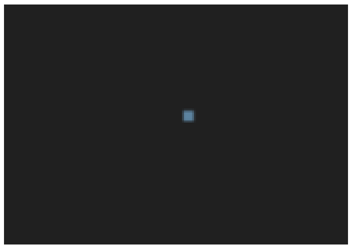
What the hell is going on?



90°

There is no latitude higher than 90°. The North Pole is situated at 90° north latitude, or simply 90° N. The South Pole is at 90° south latitude, or 90° S.

Well, well, well, the postcodes are *valid* btw.

But, the lats longs are incorrect. 

Here is what you see if you try to draw the plot of just these 540 coords.

Found the culprit!

Alright, time to just skip this and try again.

```
if(lat == 99.99999) continue;
```

added this line in my Java ascii art creator

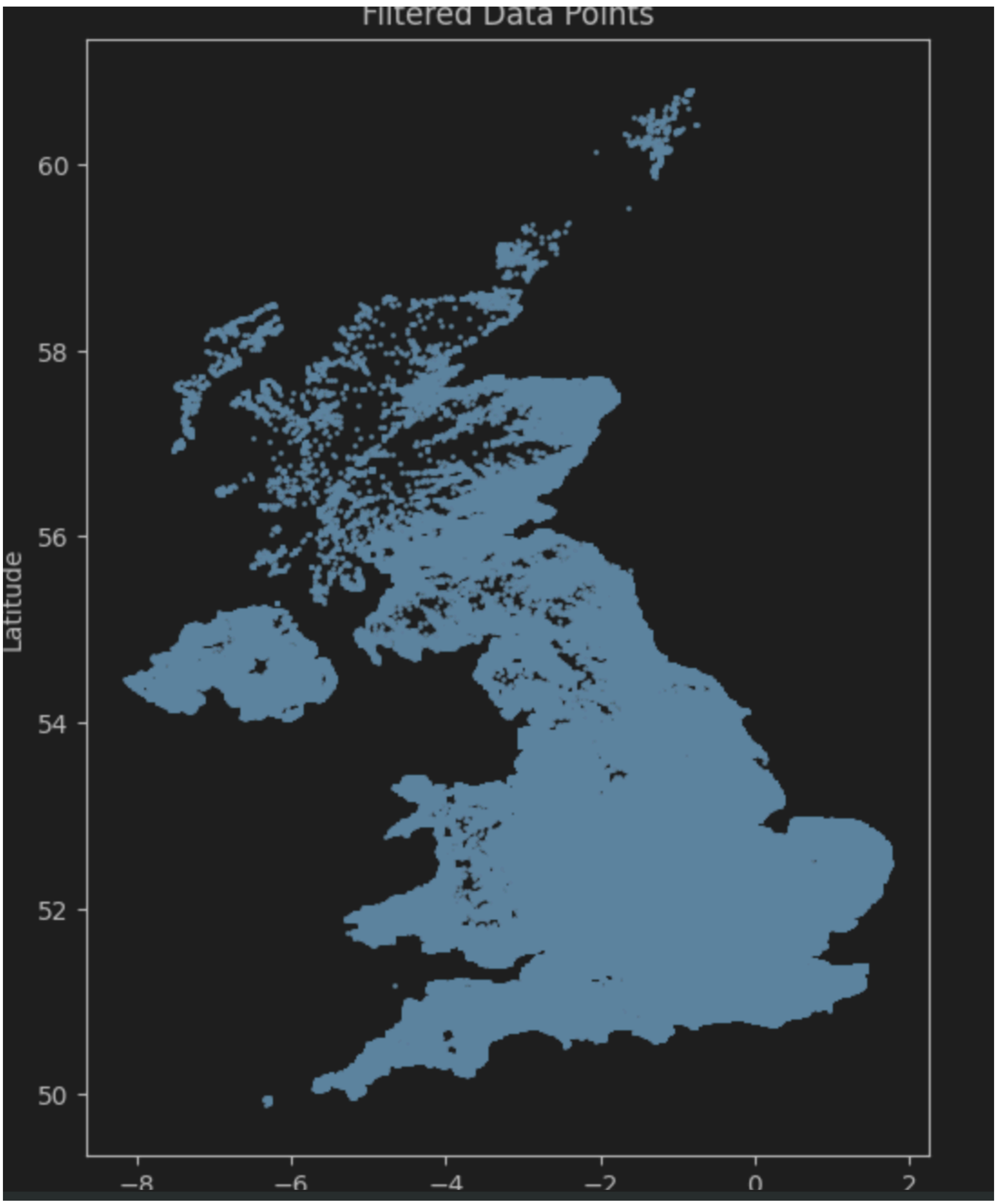
```
................................................................................fff.............................
...............................................................................fff..............................
............................................................................fffff..............................
.........................................................................ffffff................................
...........................................................................fffffff.............................
........................................................................f.ffffff...............................
.........................................................................ffffff................................
...................................................................f.......fff................................
............................................................................ff.................................
............................................................................f..................................
............................................................................f..................................
................................................................................................................
................................................................................................................
.........................................................................f......................................
................................................................................................................
.........................................................ff....f...............................................
.........................................................ff..ff................................................
........................................................ff.fff.................................................
......................................................ffffffff.................................................
......................................................ffffff.f.................................................
......................................................fffffff..................................................
......................................................fffff....................................................
.......................................................f.ff....................................................
.................................................fffffff.......................................................
.....................................ff..fffffffffffff.......................................................
...................fff...........fff.ffff.f..f..ffffff.......................................................
...............ffff.f..........fff.....f.f....f..ffff.......................................................
.............fff..ffff........ffffff...ff.fff..fffff........................................................
...........ffffffffff........fffff..f.f......f..ff..........................................................
...........f.f.ffff..........ff..fff..ff.fff.fff.............................................................
...........ff..f.ff..........ffff..fffff.fffff...............................................................
.............fff...........ffffff.....fffffff.................................................................
...........fffff.........ffff.ffff.......ffffff...............................................................
.........fffff.....f....fff.....f.ff.fffffff...fff...fff.....ff..........................................
.......fffff......fff...f..fff..ffff.ffffff..fffffffffffffffffff........................................
.......ffff.....f.fffffffff.fff..ffffffffffffffffffffffffffffffff........................................
........fff...ffffff.fff.ffff.f..fffffffffff.fffffffffffffff........................................
```

```
.......fff....ffffffff.fffff.....fffffffff..fffffffffffffffff.....................................
.......ff........ffffffffff.....f.ffff.fffffffff.fffffffffff......................................
.......ff.........ffffffff.f..f..ffff.....fff..f.fffffffffff.......................................
.......ff.......f....fff.fff..ff.ff.....fff.f....fffffffffff........................................
......ff..........f..f.ffff..f..ff...fff.f....ffffff..fffffff........................................
......f.............f..fff.f..fffff...ff..f..f.f.f..ffffffff.........................................
...................f...fff..ffff...f...fffff..ffffffffffff........................................
...................fffffff.ffffff..ffffffffffffffffffffff..........................................
...............ff.fffffffff.ffff.f...ffffffffffffffffffff............................................
...........ffff...fffffff.ffff.f.ff.ff.ffffffffffffffff.............................................
...........fff....ffff..f.fffffffff..fff.ffffffffffff................................................
.................fffff.fffff.fff..ffffffffffffffffffff...............................................
..................ff.....ffffffff.ffffffffffffffffffff..............................................
.........................ffffffffffffffffffffffffffffff..............................................
...................f.....ffffffffffffffffffffffffff..fff..............................................
...................f...ffffffffffffffffffffffffffffffffff..........................................
....................f.ff.fffffffffffffffffffffffffffffffffff.......................................
.................ffffff..ffffffffffffffffffffffffffffff.fffffff......................................
................ffff.f..fffff..ffffffffffffffffffffffffffffffff....................................
..................fff...f.fffff..ffffffffffffffffffffffffffffffff..................................
.........................f.f.fff...fffffffffffffffffffffffffffffff..................................
.........................fff.fff..ffffffffffffff.fffffffffffffff..................................
.........................ff.......fffffffffffffffffffffffffffffff..................................
...............ffffff....f......fffffffffffffffffffffffffffffffff.................................
............ffffffffff.........fffff...fffffffffffffffffffffffffff.................................
.......ffffffffffff.ff.........ffffff.ffffffffffffffff.ffffffffffff................................
.......ffffffffffffffff......fffffffffffffffffffffffffffffffffffff................................
.......ffffffffffffffffff....ffffffffffffffff.fffffffffffffffffffff...............................
......fffffffffffffffffff.....fffffffffffff...fffffffffffffffffffff...............................
...ffffffffffffffffffffffff.....ff..fff........fffffffffffffffffffffff..............................
...fffffffffffff.fffffffff.....f............ffffffffffffffffffffffffffff...........................
ffffffffffffffffffffffffffff.................fffffffffffffffffffffffffffffff........................
ffffffffffffffffffffffffffff.................fffffffffffffff.ffffffffffffff.......................
fffffffffff.ffffffffffffff...................fffffffffffffffffffffffffffffff......................
..fffffffff.fffffffffffff.....................fffffffffffffffffffffffffffffff.....................
....ffffff....ffffffffff.........................ffffffffffffffffffffffffffffffff...................
...............ffffffff.........................fff.ffffffffffffffffffffffffffff...................
..................................................fffffffffffffffffffffffffffff....................
.................................................fffffffffffffffffffffffffffffff...................
.................................................ffffffffffffffffffffffffffffffff..................
```

```
.....................................................ffffffffffffffffffffffffffffffff..................
....................................................fffffffffffffffffffffffffffffffff..................
....................................................fffffffffffffffffffffffffffffffff..................
....................................ffff..........fffffffffffffffffffffffffffffffffff.................
...................................ffffffffffffffffffffffffffffffffffffffffffffffffff................
....................................fffffffffffffffffffffffffffffffffffffffffffffffff................
.....................................ffffffffffffffffffffffffffffffffffffffffffffffff................
......................................fffffffffffffffffffffffffffffffffffffffffffffff................
....................................ffffffffffffffffffffffffffffffffffffffffffffffff...fffffffff.....
...................................ffffffffffffffffffffffffffffffffffffffffffffffffff.ffffffffffff...
..................................ffff...fffffffffffffffffffffffffffffffffffffffffffffffffffffffff..
..........................................ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.
..........................................ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.
..........................................fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
..........................................ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.
........................................fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.
......................................fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff..
...................................fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff..
...............................fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff...
..............................ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff....
.............................ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff.....
.............................fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff......
..............................ffffff...ffffffffffffffffffffffffffffffffffffffffffffffffffff.........
................................ff.....ffffffffffffffffffffffffffffffffffffffffffffffffffff.........
........................................f....ffffffffffffffffffffffffffffffffffffffffffffff..........
...............................................fffff.fffffffffffffffffffffffffffffffffffffffffffff....
.....................................................fffffffffffffffffffffffffffffffffffffffffffff....
........................................fffffffffffffffffffffffffffffffffffffffffffffffffffffffff....
...................................f....fffffffffffffffffffffffffffffffffffffffffffffffffffffffff....
.....................................ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff......
.....................................ffffffffffffffffffffffffffffffffffffffffffffffffffffffff........
....................................ffffffffffffffffffffffffffffffffffffffffffffffffffffff...........
....................................fffffffffffffffffffffffffffffffffffffffffff..fffff...............
..................................fffffffffffffffffffffffffffffff.ffffff..............................
................................ffffffffffffffffff.......fffffff....fff...............................
................................fffffffffff.ffff.........ff............................................
...............................fffffffffffffffff.......................................................
............................ffffffffff.ffffffff.......................................................
.........................fffffffff........fff..........................................................
........................ffffffff......................................................................
```

```
........................fff.fff.....................................................................
..................f..........f......................................................................
..................f.................................................................................
```

Now it's starting to resemble a lot like UK. Here is my Python plot for comparison



Here is the very bare bones unoptimized Java code.

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UKPostcodeAsciiArt {

    // Simple holder for lat/lon
    static class LatLon {
        double lat;
        double lon;
        LatLon(double lat, double lon) {
            this.lat = lat;
            this.lon = lon;
        }
    }

    public static void main(String[] args) {

        // 1. Load data
        List<LatLon> points = new ArrayList<>();
        String csvFilePath = "ukpostcodes.csv";  // Adjust path as needed

        if (extractPointsFromCsv(csvFilePath, points)) {
            return;
        }

        // >>> Sort the points: North to South (desc lat) and West to East
(asc lon) <<<
        // North to South means larger lat first, so descending lat.
// West to East means smaller lon first, so ascending lon.
points.sort((p1, p2) -> {
            // compare lat in descending order
            int lonComp = Double.compare(p2.lon, p1.lon);
            if (lonComp != 0) {
                return lonComp;
            }
            // if lat is the same, compare lon in ascending order
            return Double.compare(p1.lat, p2.lat);
        });
        // 3. Determine bounding box
        double minLat = Double.MAX_VALUE;
        double maxLat = -Double.MAX_VALUE;
        double minLon = Double.MAX_VALUE;
        double maxLon = -Double.MAX_VALUE;

        for (LatLon ll : points) {
            if (ll.lat < minLat) minLat = ll.lat;
            if (ll.lat > maxLat) maxLat = ll.lat;
            if (ll.lon < minLon) minLon = ll.lon;
            if (ll.lon > maxLon) maxLon = ll.lon;
        }

        // 4. Choose ASCII grid size (adjust as needed)
        int gridWidth = 80;   // x-axis
        int gridHeight = 100;  // y-axis

        // 5. Initialize ASCII grid with background char '.'        char[][]
asciiGrid = new char[gridHeight][gridWidth];
        for (int r = 0; r < gridHeight; r++) {
            for (int c = 0; c < gridWidth; c++) {
                asciiGrid[r][c] = '.';
            }
        }

        // 6. Map each lat/lon to the grid
        double latRange = maxLat - minLat;
        double lonRange = maxLon - minLon;

        for (LatLon ll : points) {
            // row: larger lat should map to row 0 (top),
            // so we do (maxLat - ll.lat) / latRange
            int row = (int) ((maxLat - ll.lat) / latRange * (gridHeight -
1));

            int col = (int) ((ll.lon - minLon) / lonRange * (gridWidth -
1));

            // Bounds checking
            if (row >= 0 && row < gridHeight && col >= 0 && col < gridWidth)
{
                asciiGrid[row][col] = 'f';
            }
        }
        // 7. Print out the ASCII map
        // Note: row 0 is top row, row gridHeight-1 is bottom        for
(int r = 0; r < gridHeight; r++) {
            StringBuilder sb = new StringBuilder();
            for (int c = 0; c < gridWidth; c++) {
                sb.append(asciiGrid[r][c]);
            }
            System.out.println(sb.toString());
        }

        System.out.println("ASCII Art Map generated with " + points.size() +
" points.");
    }

    private static boolean extractPointsFromCsv(String csvFilePath,
List<LatLon> points) {
        try (BufferedReader br = new BufferedReader(new
FileReader(csvFilePath))) {
            String line;
            // Skip header (assuming the first line is a header)
            br.readLine(); // e.g. "id,postcode,latitude,longitude"

            while ((line = br.readLine()) != null) {
                // Example line: "1,AB10 1XG,57.14416516,-2.114847768"
                String[] tokens = line.split(",");
                if (tokens.length < 4) {
                    continue; // skip if not enough columns
                }

                // parse lat/lon
                try {
                    double lat = Double.parseDouble(tokens[2]);
                    double lon = Double.parseDouble(tokens[3]);
                    if(lat == 99.999999) continue;
                    points.add(new LatLon(lat, lon));
                } catch (NumberFormatException e) {
                    // skip invalid data
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            return true;
        }
        return false;
    }
}
```

- 4:49PM **Took another break from this project**
- Then I saw that a lot of img2ascii converters try to make use of something called pixel brightness which they extract from a total of the RGB values. (there's some color to brightness formula that involves multiplying certain constants with r,g,b pixel values)
- Here it is in Rust

```
#[inline]
fn brightness(pixel: &RGB8) -> f32 {
    0.3 * pixel.r as f32 + 0.59 * pixel.g as f32 + 0.11 * pixel.b as f32
}
```

While I had absolutely no plan to yet to think about this Ascii art as video I did try my hand at using "density"

- A lot of this code is tested by me but generated via ChatGPT so I hope that's okay

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UKPostcodeAsciiArt {

    // Simple holder for lat/lon
    static class LatLon {
        double lat;
        double lon;

        LatLon(double lat, double lon) {
            this.lat = lat;
            this.lon = lon;
        }
    }

    public static void main(String[] args) {
        // 1. Load data
        List<LatLon> points = new ArrayList<>();
        String csvFilePath = "ukpostcodes.csv";  // Adjust path as needed

        if (extractPointsFromCsv(csvFilePath, points)) {
            return;
        }

        // >>> Sort the points: North to South (desc lat) and West to East
(asc lon) <<<
        // North to South means larger lat first, so descending lat.
// West to East means smaller lon first, so ascending lon.
points.sort((p1, p2) -> {
            // compare lat in descending order
            int lonComp = Double.compare(p2.lon, p1.lon);
            if (lonComp != 0) {
                return lonComp;
            }
            // if lat is the same, compare lon in ascending order
            return Double.compare(p1.lat, p2.lat);
        });

        // 2. Load the image (if you still want to reference an actual map
file)
        try {
            File imgFile = new File("uk_map.png");
            BufferedImage img = ImageIO.read(imgFile);
            // (You might do something with the image here if desired)
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        // 3. Determine bounding box
        double minLat = Double.MAX_VALUE;
        double maxLat = -Double.MAX_VALUE;
        double minLon = Double.MAX_VALUE;
        double maxLon = -Double.MAX_VALUE;

        for (LatLon ll : points) {
            if (ll.lat < minLat) minLat = ll.lat;
            if (ll.lat > maxLat) maxLat = ll.lat;
            if (ll.lon < minLon) minLon = ll.lon;
            if (ll.lon > maxLon) maxLon = ll.lon;
        }

        // 4. Choose ASCII grid size (adjust as needed)
        int gridWidth = 80;   // x-axis
        int gridHeight = 120;  // y-axis

        // 5. Initialize ASCII grid with background char '.'        char[][]
asciiGrid = new char[gridHeight][gridWidth];
        for (int r = 0; r < gridHeight; r++) {
            for (int c = 0; c < gridWidth; c++) {
                asciiGrid[r][c] = '.';
            }
        }

        // 6. Map each lat/lon to the grid
        double latRange = maxLat - minLat;
        double lonRange = maxLon - minLon;

        for (LatLon ll : points) {
            // Adjust longitude scaling for latitude
            double adjustedLonRange = lonRange *
Math.cos(Math.toRadians(ll.lat));

            // Map to grid
            int row = (int) ((maxLat - ll.lat) / latRange * (gridHeight -
1));
            int col = (int) ((ll.lon - minLon) / adjustedLonRange *
(gridWidth - 1));

            // Bounds checking
            if (row >= 0 && row < gridHeight && col >= 0 && col < gridWidth)
{
                asciiGrid[row][col] = 'f'; // Basic fill
            }
        }

// Apply density-based symbol variation
        int[][] densityGrid = new int[gridHeight][gridWidth];
        for (LatLon ll : points) {
            int row = (int) ((maxLat - ll.lat) / latRange * (gridHeight -
1));
            int col = (int) ((ll.lon - minLon) / lonRange * (gridWidth -
1));
            if (row >= 0 && row < gridHeight && col >= 0 && col < gridWidth)
{
                densityGrid[row][col]++;
            }
        }
        for (int r = 0; r < gridHeight; r++) {
            for (int c = 0; c < gridWidth; c++) {
                int density = densityGrid[r][c];
                if (density == 0) {
                    asciiGrid[r][c] = '.'; // Background
                } else if (density < 5) {
                    asciiGrid[r][c] = 'f'; // Low density
                } else if (density < 20) {
                    asciiGrid[r][c] = 'F'; // Medium density
                } else {
                    asciiGrid[r][c] = '#'; // High density
                }
            }
        }

        // 7. Print out the ASCII map
        // Note: row 0 is top row, row gridHeight-1 is bottom        for
(int r = 0; r < gridHeight; r++) {
            StringBuilder sb = new StringBuilder();
            for (int c = 0; c < gridWidth; c++) {
                sb.append(asciiGrid[r][c]);
            }
            System.out.println(sb.toString());
        }

        System.out.println("ASCII Art Map generated with " + points.size() +
" points.");
    }

    private static boolean extractPointsFromCsv(String csvFilePath,
List<LatLon> points) {
        try (BufferedReader br = new BufferedReader(new
FileReader(csvFilePath))) {
            String line;
            // Skip header (assuming the first line is a header)
            br.readLine(); // e.g. "id,postcode,latitude,longitude"

            while ((line = br.readLine()) != null) {
                // Example line: "1,AB10 1XG,57.14416516,-2.114847768"
                String[] tokens = line.split(",");
                if (tokens.length < 4) {
                    continue; // skip if not enough columns
                }

                // parse lat/lon
                try {
                    double lat = Double.parseDouble(tokens[2]);
                    double lon = Double.parseDouble(tokens[3]);
                    if (lat == 99.999999) continue;
                    points.add(new LatLon(lat, lon));
                } catch (NumberFormatException e) {
                    // skip invalid data
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            return true;
        }
        return false;
    }
}
```

Feel free to ask me about anything in the code. AI can only generate dumb code but it is up to us to make meaning out of it.

Current benchmarks
-java UKPostcodeAsciiArt 1.19s user 0.16s system 161% cpu 0.834 total

- 5:30PM **Realized Sorting makes no difference on this code

```
........................................................ffF.............
........
......................................................ffFff.............
........
....................................................fFFFF...............
........
.....................................................f#Fffff............
........
...................................................ffFFFFf..............
........
...................................................fFF##F...............
........
................................................f.....##f...............
........
......................................................FF................
........
......................................................F.................
........
........................................................................
```

```
........
........................................................................
........
...................................................f....................
........
.............................................f..........................
........
.........................................F..ff..........................
........
........................................ffffF...........................
........
......................................###FffF...........................
........
......................................####Fff...........................
........
......................................fFf#F.............................
........
.......................................F.F..............................
........
...................................f#fFFF...............................
........
...............F........f.FFfFFFfFFF###FF...............................
........
.............FFF........FF.f.Fff.f..fFF##...............................
........
..........fFFF#F.......Ffff..Ffff..f.F#F................................
........
........fFFFf###f.....FFFf.f.ff..f..fF..................................
........
........f.FfF#f......ff#fff..F.ffff#f...................................
........
........ff.F.F.......fFff.fff#FFF##.....................................
........
........ff#F.......f#FF#F...ff#F#F#.....................................
........
.......FfFF.......F#F..fFf.f.fFF##F...#.................................
........
.....FFFf.....FF..fFfff..ffFf####f.F##############f.....................
........
.....FFF....Ff#FffffFfFfF.ff#########FF############.....................
........
......#F..fF#F##ffF.F#f.f.ffF####ffFff#############.....................
........
.....fFf.....FFfFff###f...FFF#FFfF#F#F#fFF########......................
........
.....fF.......f.fF#FFFf...ffF#f.f.##FF#f#########.......................
```

```
........
......#......f.ffFF.ff.fffF#....##Ff...FF########.......................
........
.....#f.......f..f#ff..f.ff..fFff....#FF#ff#####f.......................
........
....Ff..........f.Ffffff#FFf..fF.f..ff.fffFF####........................
........
...............ff.Fff..f#...f..ffF#.FFFfFF#####.........................
........
............ff.f#fFfFff###f.fFf#F###F#F#######..........................
........
.........fFff.fFFFFFfF#ff.ff.fFF##FF#########...........................
........
.........FF...ffFF.Ff###FFfFff#fffF#########............................
........
..............FFFfff##.FFf.FFf#fF###########............................
........
...................f#FFf#Ff#FF###############...........................
........
...............F....FFfFFF#f#############...............................
........
...............f..ffF#FFf#####################..........................
........
................FFf.F#F#####################FF###.......................
........
.............F##fF..FFf.#############F#####FF#####......................
........
.............Ff#F..FffFfFf####F####################.f...................
........
..............ff...F.#F##..####F#F#F#FFf#############...................
........
...................F#fFF#..#####ffF#fffFF#####FFF####...................
........
...............f...fF.....####F#F##Ff#FFf###FFFF#####...................
........
...........#####F...f.....##FffFf###F##FFFFF#F#######f..................
........
.........######F#........##FFff##F######F##Ff#########..................
........
......############.....f#FfF##ff######################F.................
........
.....#####F########f...F###F#########F#################.................
........
....F####f##########....##fF######F..#######F##########.................
........
..F##################....#F.F##.....#########fF#########................
```

```
........
.f###########F########..............###########f###########f............
........
######################..............##FF##F####F#############...........
........
F#######FF############...............#########F###############..........
........
..######F.###########................F########F################.........
........
...f####...f#######...................#########F#f###############.......
........
............##FF##.....................F.###FF##################........
........
........................................########################........
........
........................................#########################.......
........
........................................##########################......
........
........................................#########################Ff.....
........
.............................fF.........##########F################.....
........
...........................#####f.##################################....
........
...........................F########################################....
........
.............................####F##################################....
........
.............................######################################.....
........
...........................f######f###############################..####
#####...
..........................f###.f#F##F###################################
######f.
................................####F###################################
#######.
................................########################################
#######.
................................##fF####################################
#######f
................................###F####################################
#######.
..............................#####.####################################
#######.
...........................F######FF####################################
```

```
######..
........................F###########F###################################
######..
......................F#################################################
####f...
......................f#################################################
####....
......................f#################################################
#.......
........................f##F..##########################################
F.......
...............................F...######F##############################
........
....................................####.###############################
#####...
.................................F#FFf..f###############################
#####...
...........................f...#########################################
#####...
............................f#f#########################################
###f....
............................############################################
#.......
............................###########################################.
........
...........................##############################F##.....###....
........
.........................########F#####.....######..f###................
........
........................##############.......#..........................
........
.......................###############..................................
........
......................########.######...................................
........
...................#######F.......FF....................................
........
...................######...............................................
........
..............#........#................................................
........
..............f.........................................................
........
ASCII Art Map generated with 1738108 points.
java UKPostcodeAsciiArt  0.70s user 0.11s system 166% cpu 0.488 total
```

**New benchmark already 40% faster by just combining the two loops to calculate densityGrid and asciiGrid into one loop and removing sorting.

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UKPostcodeAsciiArt {

    // Simple holder for lat/lon
    static class LatLon {
        double lat;
        double lon;
        LatLon(double lat, double lon) {
            this.lat = lat;
            this.lon = lon;
        }
    }

    public static void main(String[] args) {

        // 1. Load data
        List<LatLon> points = new ArrayList<>();
        String csvFilePath = "../../../ukpostcodes.csv";  // Adjust path as
needed

        if (extractPointsFromCsv(csvFilePath, points)) {
            return;
        }


        // 3. Determine bounding box
        double minLat = Double.MAX_VALUE;
        double maxLat = -Double.MAX_VALUE;
        double minLon = Double.MAX_VALUE;
        double maxLon = -Double.MAX_VALUE;

        for (LatLon ll : points) {
            if (ll.lat < minLat) minLat = ll.lat;
            if (ll.lat > maxLat) maxLat = ll.lat;
            if (ll.lon < minLon) minLon = ll.lon;
            if (ll.lon > maxLon) maxLon = ll.lon;
        }

        // 4. Choose ASCII grid size (adjust as needed)
        int gridWidth = 80;   // x-axis
        int gridHeight = 100;  // y-axis

        // 5. Initialize ASCII grid with background char '.'        char[][]
asciiGrid = new char[gridHeight][gridWidth];
        int[][] densityGrid = new int[gridHeight][gridWidth];
        for (int r = 0; r < gridHeight; r++) {
            for (int c = 0; c < gridWidth; c++) {
                asciiGrid[r][c] = '.';
            }
        }

        // 6. Map each lat/lon to the grid
        double latRange = maxLat - minLat;
        double lonRange = maxLon - minLon;

        for (LatLon ll : points) {
            // row: larger lat should map to row 0 (top),
            // so we do (maxLat - ll.lat) / latRange            int row =
(int) ((maxLat - ll.lat) / latRange * (gridHeight - 1));
            int col = (int) ((ll.lon - minLon) / lonRange * (gridWidth -
1));

            // Bounds checking
            if (row >= 0 && row < gridHeight && col >= 0 && col < gridWidth)
{
                asciiGrid[row][col] = 'f';
                densityGrid[row][col]++;
            }
        }

        for (int r = 0; r < gridHeight; r++) {
            for (int c = 0; c < gridWidth; c++) {
                int density = densityGrid[r][c];
                if (density == 0) {
                    asciiGrid[r][c] = '.'; // Background
                } else if (density < 5) {
                    asciiGrid[r][c] = 'f'; // Low density
                } else if (density < 20) {
                    asciiGrid[r][c] = 'F'; // Medium density
                } else {
                    asciiGrid[r][c] = '#'; // High density
                }
            }
        }

        // 7. Print out the ASCII map
        // Note: row 0 is top row, row gridHeight-1 is bottom        for
(int r = 0; r < gridHeight; r++) {
            StringBuilder sb = new StringBuilder();
            for (int c = 0; c < gridWidth; c++) {
                sb.append(asciiGrid[r][c]);
            }

            System.out.println(sb.toString());
        }


        System.out.println("ASCII Art Map generated with " + points.size() +
" points.");
    }


private static boolean extractPointsFromCsv(String csvFilePath, List<LatLon>
points) {
    try (BufferedReader br = new BufferedReader(new
FileReader(csvFilePath))) {
        String line;
        // Skip header (assuming the first line is a header)
        br.readLine(); // e.g. "id,postcode,latitude,longitude"

        while ((line = br.readLine()) != null) {
            // Example line: "1,AB10 1XG,57.14416516,-2.114847768"
            String[] tokens = line.split(",");
            if (tokens.length < 4) {
                continue; // skip if not enough columns
            }

            // parse lat/lon
            try {
                double lat = Double.parseDouble(tokens[2]);
                double lon = Double.parseDouble(tokens[3]);
                if(lat == 99.999999) continue;
                points.add(new LatLon(lat, lon));
            } catch (NumberFormatException e) {
                // skip invalid data
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
        return true;
    }
    return false;
}
}
```

And here is the more finalized version of it

Final - 1 version (I also wanted to try color next)

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UKPostcodeAsciiArt {

    // Simple holder for lat/lon
    static class LatLon {
        double lat;
        double lon;

        LatLon(double lat, double lon) {
            this.lat = lat;
            this.lon = lon;
        }
    }

    public static void main(String[] args) {
        // 1. Load data
        List<LatLon> points = new ArrayList<>();
        String csvFilePath = "../../../ukpostcodes.csv";  // Adjust path as
needed

        // If there's an error reading CSV or no points, exit early
if (extractPointsFromCsv(csvFilePath, points)) {
            return;
        }

        // 3. Determine bounding box for min/max lat/lon
        double[] bounds = computeBounds(points);
        double minLat = bounds[0];
        double maxLat = bounds[1];
        double minLon = bounds[2];
        double maxLon = bounds[3];

        // 4. Choose ASCII grid size (adjust as needed)
        int gridWidth = 80;   // x-axis
        int gridHeight = 100; // y-axis

        // 5. Initialize the ASCII and density grids        char[][]
asciiGrid = initializeAsciiGrid(gridHeight, gridWidth);
        int[][] densityGrid = new int[gridHeight][gridWidth];

        // 6. Map each lat/lon to the grid
        fillDensityGrid(points, asciiGrid, densityGrid, minLat, maxLat,
minLon, maxLon);

        // 7. Convert density values to characters in the ASCII grid
        convertDensityToCharacters(asciiGrid, densityGrid);

        // 8. Print out the ASCII map
        printAsciiGrid(asciiGrid);

        System.out.println("ASCII Art Map generated with " + points.size() +
" points.");
    }

    /**
     * Reads CSV file and populates the list of points (lat/lon).     *
Returns true if there's any IO error (or if file reading fails).     */
private static boolean extractPointsFromCsv(String csvFilePath, List<LatLon>
points) {
        try (BufferedReader br = new BufferedReader(new
FileReader(csvFilePath))) {
            String line;

            // Skip header (assuming the first line is a header)
            br.readLine(); // e.g. "id,postcode,latitude,longitude"

            while ((line = br.readLine()) != null) {
                // Example line: "1,AB10 1XG,57.14416516,-2.114847768"
                String[] tokens = line.split(",");
                if (tokens.length < 4) {
                    continue; // skip if not enough columns
                }
                // parse lat/lon
                try {
                    double lat = Double.parseDouble(tokens[2]);
                    double lon = Double.parseDouble(tokens[3]);
                    if (lat == 99.999999) {
                        // skip sentinel/bogus data
                        continue;
                    }
                    points.add(new LatLon(lat, lon));
                } catch (NumberFormatException e) {
                    // skip invalid data
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            return true;
        }
        return false;
    }

    /**
     * Computes the bounding box [minLat, maxLat, minLon, maxLon] for the
given points.     */    private static double[] computeBounds(List<LatLon>
points) {
        double minLat = Double.MAX_VALUE;
        double maxLat = -Double.MAX_VALUE;
        double minLon = Double.MAX_VALUE;
        double maxLon = -Double.MAX_VALUE;

        for (LatLon ll : points) {
            if (ll.lat < minLat) minLat = ll.lat;
            if (ll.lat > maxLat) maxLat = ll.lat;
            if (ll.lon < minLon) minLon = ll.lon;
            if (ll.lon > maxLon) maxLon = ll.lon;
        }
        return new double[]{minLat, maxLat, minLon, maxLon};
    }

    /**
     * Initializes a 2D char array with '.' to represent empty background.
*/    private static char[][] initializeAsciiGrid(int rows, int cols) {
        char[][] grid = new char[rows][cols];
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                grid[r][c] = '.';
```

```java
            }
        }
        return grid;
    }

    /**
     * Fills the density grid by mapping each point's lat/lon     * to its
corresponding row/col in the ASCII grid.     */    private static void
fillDensityGrid(List<LatLon> points,
                                        char[][] asciiGrid,
                                        int[][] densityGrid,
                                        double minLat,
                                        double maxLat,
                                        double minLon,
                                        double maxLon) {

        int rows = asciiGrid.length;
        int cols = asciiGrid[0].length;

        double latRange = maxLat - minLat;
        double lonRange = maxLon - minLon;

        for (LatLon ll : points) {
            // row: larger lat should map to row 0 (top),
            // so we do (maxLat - ll.lat) / latRange            int row =
(int) ((maxLat - ll.lat) / latRange * (rows - 1));
            // col: smaller lon should map to col 0 (left),
            // so we do (ll.lon - minLon) / lonRange            int col =
(int) ((ll.lon - minLon) / lonRange * (cols - 1));

            // Bounds checking
            if (row >= 0 && row < rows && col >= 0 && col < cols) {
                asciiGrid[row][col] = 'f';
                densityGrid[row][col]++;
            }
        }
    }

    /**
     * Converts the density counts into character symbols:     *   '.' for
zero density,     *   'f' for low density,     *   'F' for medium density,
*   '#' for high density.     */    private static void
convertDensityToCharacters(char[][] asciiGrid, int[][] densityGrid) {
        int rows = asciiGrid.length;
        int cols = asciiGrid[0].length;
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                int density = densityGrid[r][c];
                if (density == 0) {
                    asciiGrid[r][c] = '.'; // Background
                } else if (density < 5) {
                    asciiGrid[r][c] = 'f'; // Low density
                } else if (density < 20) {
                    asciiGrid[r][c] = 'F'; // Medium density
                } else {
                    asciiGrid[r][c] = '#'; // High density
                }
            }
        }
    }

    /**
     * Prints the ASCII grid rows top-to-bottom.     */    private static
void printAsciiGrid(char[][] asciiGrid) {
        for (int r = 0; r < asciiGrid.length; r++) {
            System.out.println(new String(asciiGrid[r]));
        }
    }
}
```

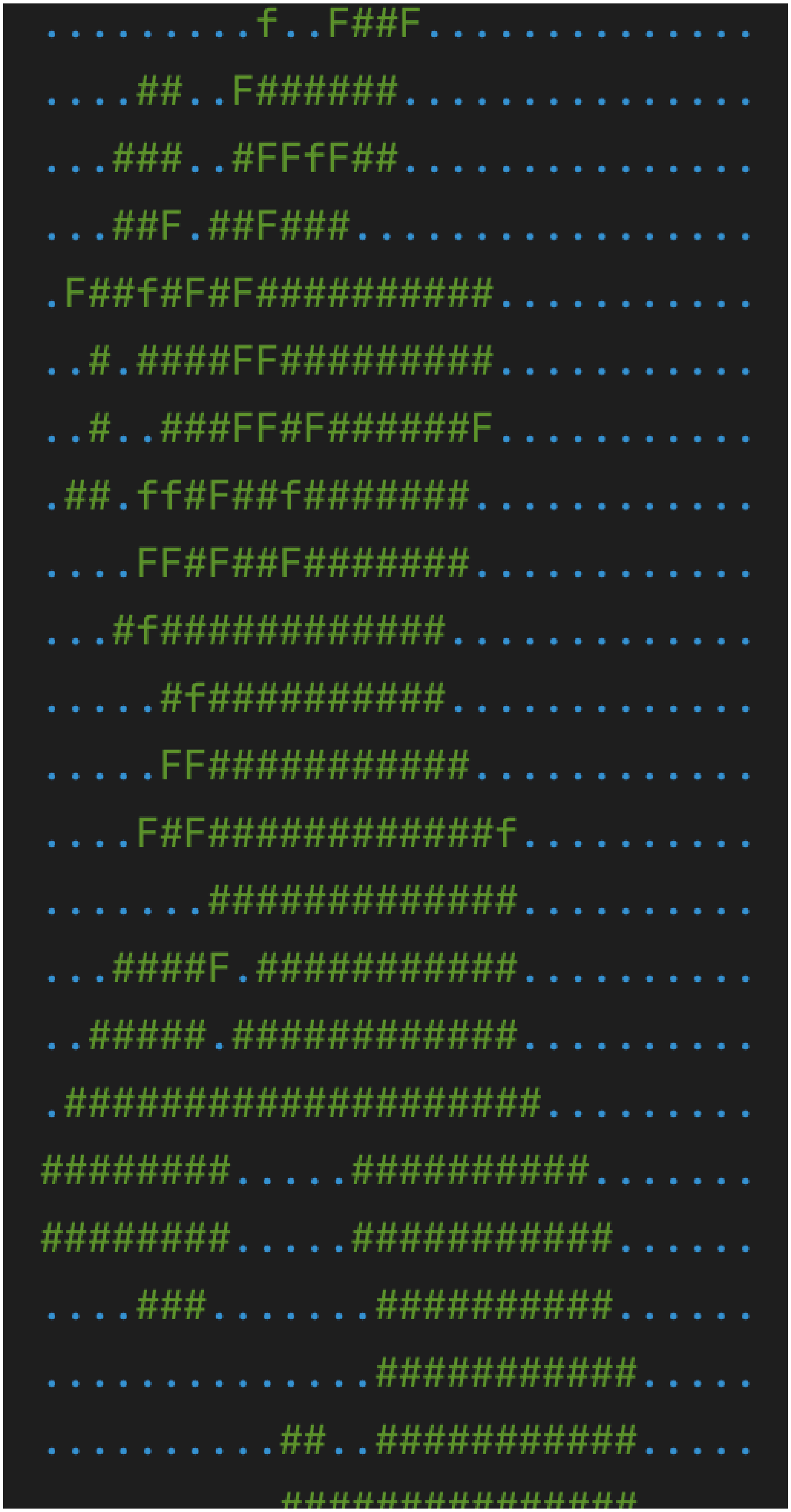
Above is clean code and gives slightly less fast output

java UKPostcodeAsciiArt 0.74s user 0.13s system 171% cpu 0.508 total

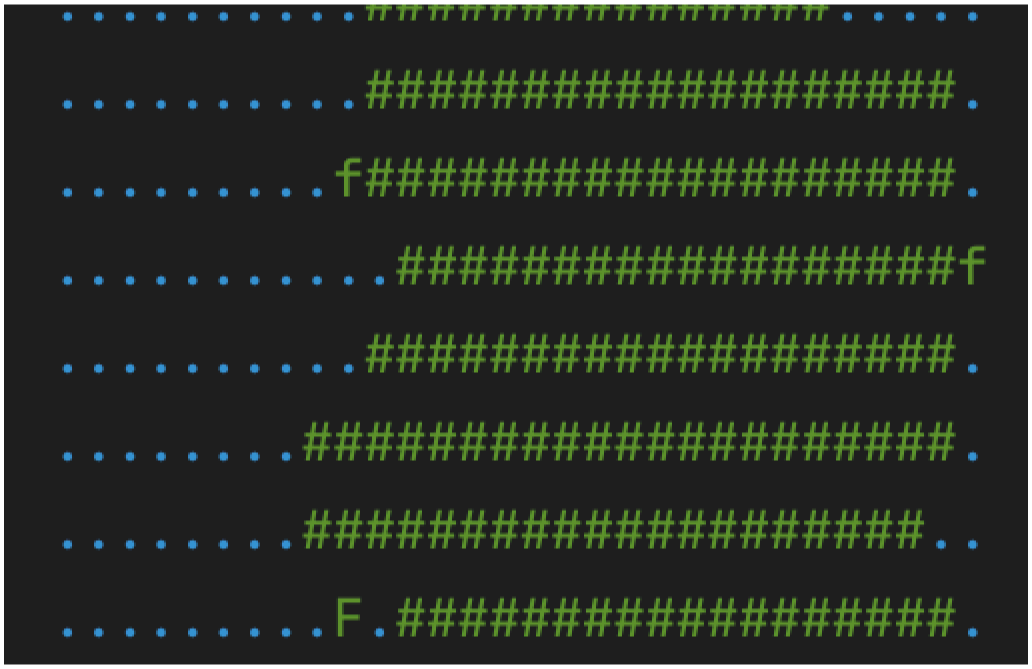
So much for clean code I guess :P

6PM And it worked!

Here is UK in all its shining ASCII 24bit glory.



```
.........f..F##F..............
....##..F######...............
...###..#FFfF##...............
...##F.##F###.................
.F##f#F#F##########...........
..#.####FF#########...........
..#..###FF#F######F...........
.##.ff#F##f#######............
....FF#F##F#######............
...#f############.............
.....#f##########.............
.....FF###########............
....F#F############f..........
.......#############..........
...####F.###########..........
..#####.############..........
.####################.........
########.....##########.......
########.....###########......
....###.......##########......
..............###########.....
..........##..###########.....
          ###############
```



```
..........###############.....
..........###################.
.........f###################.
...........##################f
..........###################.
........#####################.
........####################..
.........F.##################.
```

Maybe it would be better for you to run the code on your machine because Markdown really doesn't like colored CLI output

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UKPostcodeAsciiArt {

    // ANSI color codes
    private static final String ANSI_RESET = "\u001B[0m";
    private static final String ANSI_BLUE = "\u001B[34m";
    private static final String ANSI_GREEN = "\u001B[32m";

    // Simple holder for lat/lon
    static class LatLon {
        double lat;
        double lon;

        LatLon(double lat, double lon) {
            this.lat = lat;
            this.lon = lon;
        }
    }

    public static void main(String[] args) {
        // 1. Load data
        List<LatLon> points = new ArrayList<>();
        String csvFilePath = "ukpostcodes.csv";  // Adjust path as needed

        // If there's an error reading CSV or no points, exit early
if (extractPointsFromCsv(csvFilePath, points)) {
            return;
        }

        // 2. Sort the points:
        //    North to South => descending lat        //    West to East
=> ascending lon        //    (We’re preserving the original code’s
comparator logic)        sortPoints(points);

        // 3. Determine bounding box for min/max lat/lon
        double[] bounds = computeBounds(points);
        double minLat = bounds[0];
        double maxLat = bounds[1];
        double minLon = bounds[2];
        double maxLon = bounds[3];

        // 4. Choose ASCII grid size
        int gridWidth = 80;   // x-axis
        int gridHeight = 120; // y-axis

        // 5. Initialize the ASCII and density grids        char[][]
asciiGrid = initializeAsciiGrid(gridHeight, gridWidth);
        int[][] densityGrid = new int[gridHeight][gridWidth];

        // 6. Map each lat/lon to the grid
        fillDensityGrid(points, asciiGrid, densityGrid, minLat, maxLat,
minLon, maxLon);

        // 7. Convert density values to characters in the ASCII grid
        convertDensityToCharacters(asciiGrid, densityGrid);

        // 8. Print out the ASCII map in color
        printAsciiGridWithColor(asciiGrid);

        System.out.println("ASCII Art Map generated with " + points.size() +
" points.");
    }

    /**
     * Reads CSV file and populates the list of points (lat/lon).     *
Returns true if there's any IO error (or if file reading fails).     */
private static boolean extractPointsFromCsv(String csvFilePath, List<LatLon>
```

```java
points) {
        try (BufferedReader br = new BufferedReader(new
FileReader(csvFilePath))) {
            String line;

            // Skip header (assuming the first line is a header)
            br.readLine(); // e.g. "id,postcode,latitude,longitude"

            while ((line = br.readLine()) != null) {
                // Example line: "1,AB10 1XG,57.14416516,-2.114847768"
                String[] tokens = line.split(",");
                if (tokens.length < 4) {
                    continue; // skip if not enough columns
                }

                // parse lat/lon
                try {
                    double lat = Double.parseDouble(tokens[2]);
                    double lon = Double.parseDouble(tokens[3]);
                    if (lat == 99.999999) {
                        // skip sentinel/bogus data
                        continue;
                    }
                    points.add(new LatLon(lat, lon));
                } catch (NumberFormatException e) {
                    // skip invalid data
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            return true;
        }
        return false;
    }

    /**
     * Sorts the points according to the original logic:     *   - Compare
lon in descending order first     *   - If lon is the same, compare lat in
ascending order     */    private static void sortPoints(List<LatLon>
points) {
        points.sort((p1, p2) -> {
            int lonComp = Double.compare(p2.lon, p1.lon);
            if (lonComp != 0) {
                return lonComp;
            }
            // if lon is the same, compare lat in ascending order
            return Double.compare(p1.lat, p2.lat);
        });
    }

    /**
     * Computes the bounding box [minLat, maxLat, minLon, maxLon] for the
given points.     */    private static double[] computeBounds(List<LatLon>
points) {
        double minLat = Double.MAX_VALUE;
        double maxLat = -Double.MAX_VALUE;
        double minLon = Double.MAX_VALUE;
        double maxLon = -Double.MAX_VALUE;

        for (LatLon ll : points) {
            if (ll.lat < minLat) minLat = ll.lat;
            if (ll.lat > maxLat) maxLat = ll.lat;
            if (ll.lon < minLon) minLon = ll.lon;
            if (ll.lon > maxLon) maxLon = ll.lon;
        }
        return new double[]{minLat, maxLat, minLon, maxLon};
    }

    /**
     * Initializes a 2D char array with '.' to represent empty background.
*/    private static char[][] initializeAsciiGrid(int rows, int cols) {
        char[][] grid = new char[rows][cols];
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                grid[r][c] = '.';
            }
        }
        return grid;
    }

    /**
     * Fills the density grid by mapping each point's lat/lon     * to its
corresponding row/col in the ASCII grid.     */    private static void
fillDensityGrid(List<LatLon> points,
                                        char[][] asciiGrid,
                                        int[][] densityGrid,
                                        double minLat,
                                        double maxLat,
                                        double minLon,
                                        double maxLon) {

        int rows = asciiGrid.length;
        int cols = asciiGrid[0].length;

        double latRange = maxLat - minLat;
        double lonRange = maxLon - minLon;

        for (LatLon ll : points) {
            // row: larger lat => row 0 (top),
            // so we do (maxLat - ll.lat) / latRange            int row = 
(int) ((maxLat - ll.lat) / latRange * (rows - 1));
            // col: smaller lon => col 0 (left),
            // so we do (ll.lon - minLon) / lonRange            int col = 
(int) ((ll.lon - minLon) / lonRange * (cols - 1));

            // Bounds checking
            if (row >= 0 && row < rows && col >= 0 && col < cols) {
                asciiGrid[row][col] = 'f';
                densityGrid[row][col]++;
            }
        }
    }

    /**
     * Converts the density counts into character symbols:     *   '.' for 
zero density,     *   'f' for low density,     *   'F' for medium density,
*   '#' for high density.     */    private static void 
convertDensityToCharacters(char[][] asciiGrid, int[][] densityGrid) {
        int rows = asciiGrid.length;
        int cols = asciiGrid[0].length;

        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                int density = densityGrid[r][c];
                if (density == 0) {
                    asciiGrid[r][c] = '.'; // Water / background
                } else if (density < 5) {
                    asciiGrid[r][c] = 'f'; // Low density
                } else if (density < 20) {
                    asciiGrid[r][c] = 'F'; // Medium density
                } else {
                    asciiGrid[r][c] = '#'; // High density
                }
            }
        }
    }

    /**
```

```
     * Prints the ASCII grid rows top-to-bottom, coloring:     * - '.' (dot)
as blue (water)     * - everything else as green (land)     */    private
static void printAsciiGridWithColor(char[][] asciiGrid) {
        for (int r = 0; r < asciiGrid.length; r++) {
            StringBuilder sb = new StringBuilder();
            for (int c = 0; c < asciiGrid[r].length; c++) {
                char ch = asciiGrid[r][c];
                if (ch == '.') {
                    // Water in blue
                    sb.append(ANSI_BLUE).append(ch).append(ANSI_RESET);
                } else {
                    // Land in green
                    sb.append(ANSI_GREEN).append(ch).append(ANSI_RESET);
                }
            }
            System.out.println(sb);
        }
    }
}
```

-> Pushed code to github in 3 different branches
- Main
- Density
- Colored

https://github.com/yugonline/sedaiasciiart

-> Also added Python data analysis done on these points

python analysis can be found on /pythonanalysis branch for which I have also created a pull request on Github.

https://github.com/yugonline/sedaiasciiart/pull/1/files?short_path=e0a59e9#diff-e0a59e9d8b3333abe6c1da199b254d4a9a3415700fc1671715a3e92573cb68c7
Make sure you click on `Display rich diff`

6:30PM

**COOL STUFF I DIDNT GET TO TRY

- Using an existing outline of UK map and then "filling" in these coordinates on top of it
- Using a panorama or a street view gif of UK and "asciifying" the street view to ASCII art with a background sound
    - I was day dreaming about making a video of Harry Styles walking in the streets of ASCII UK London while the song Watermelon plays in the background

- Maybe I will try it over the weekend!!

Thank You and Have a Great Weekend!