

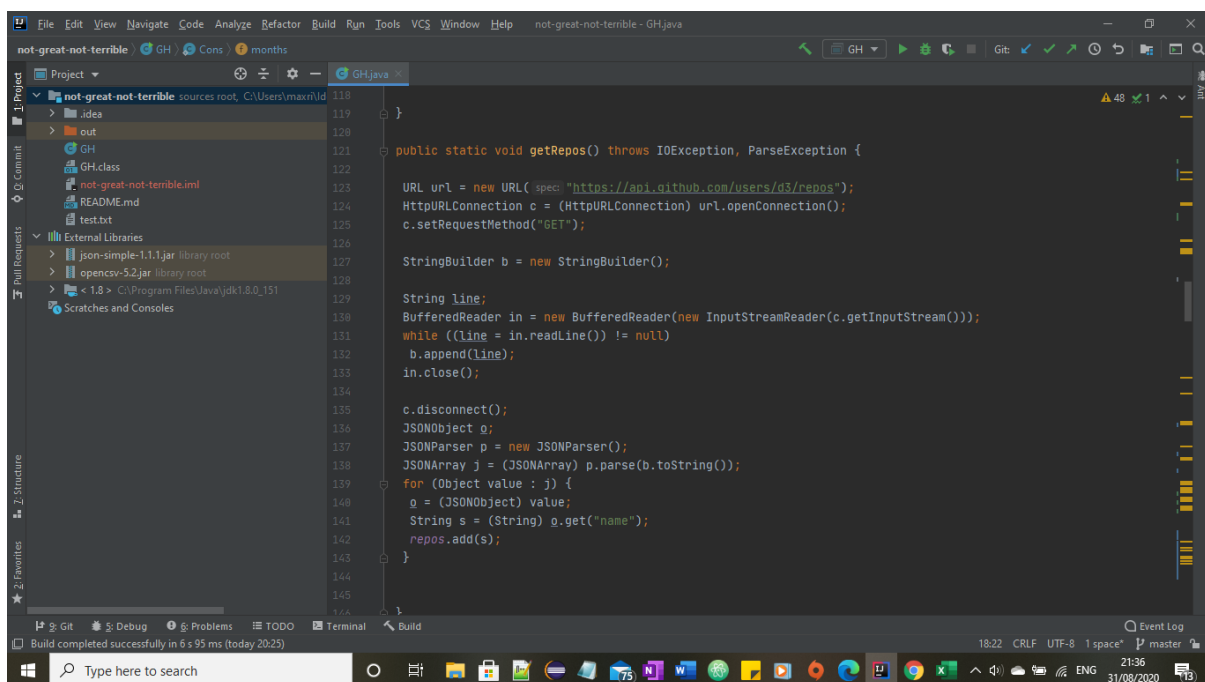
REPORT

Tools Used

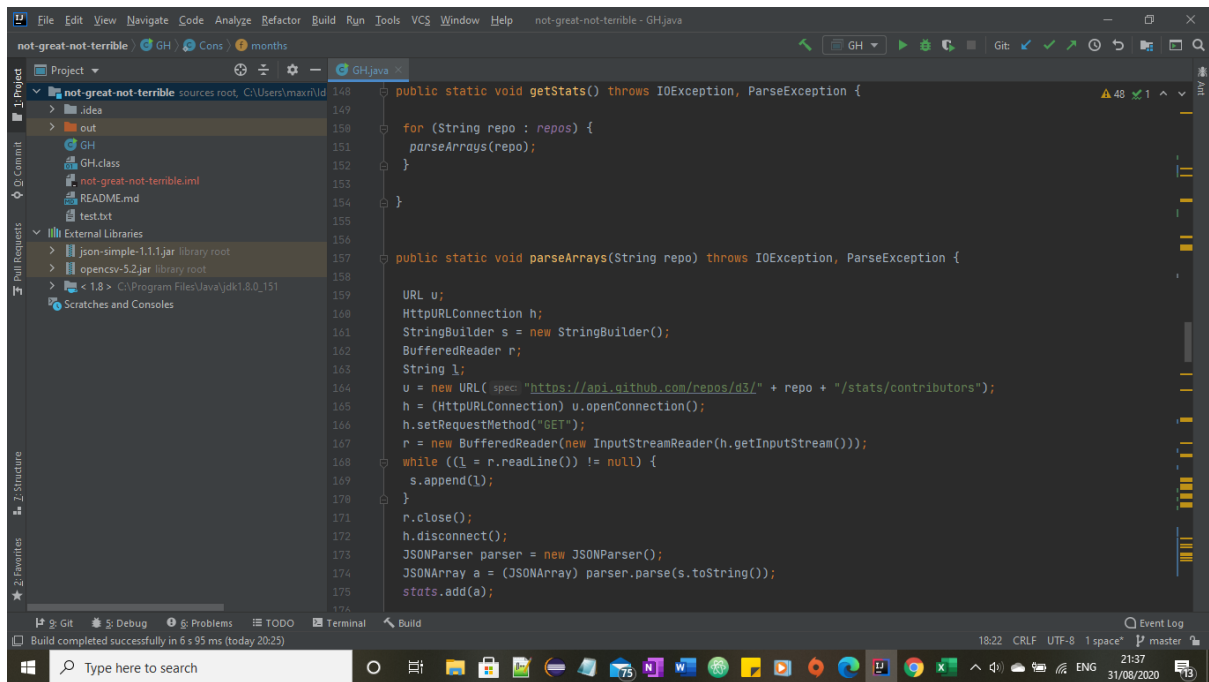
- IntelliJ IDE
- GitHub API v3
- Simple JSON jar file
- openCSV jar file
- d3 visualisation software

Hypothetical Outline of Functionality

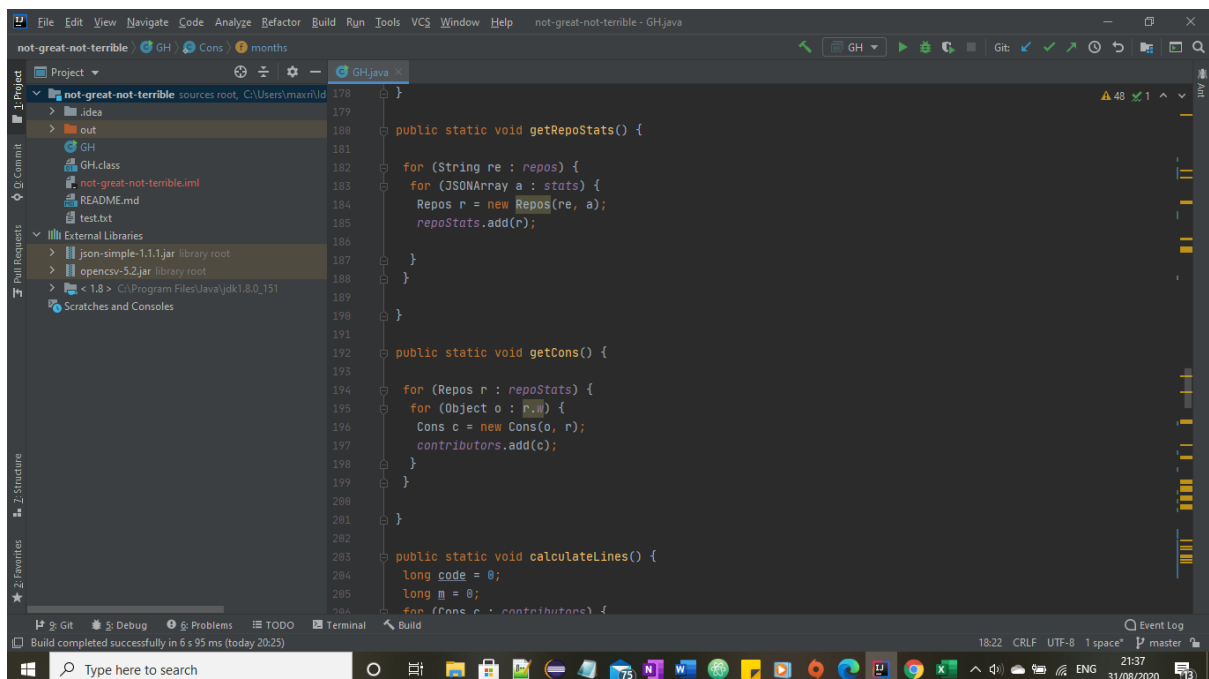
Firstly, the code calls the GitHub API requesting all repositories of which d3 is the owner, “GET”ting a JSONArray which contains JSONObject detailing each repository. Iterating through the JSONArray, a call is made to acquire the “name:” object in each JSONObject corresponding to the names of the repositories. Finally, these are converted to Strings and stored in a list called “repos”.



This list is then used when calling the GitHub API requesting the contributor stats for each repository. The results are JSONArrays, each of which is stored in a list called “stats”. Following this, a “Repos” object(Figure 1.1) is created which contains the name of the repository along with the JSONArray containing the stats for that array. The results are stored in the list “repoStats”.

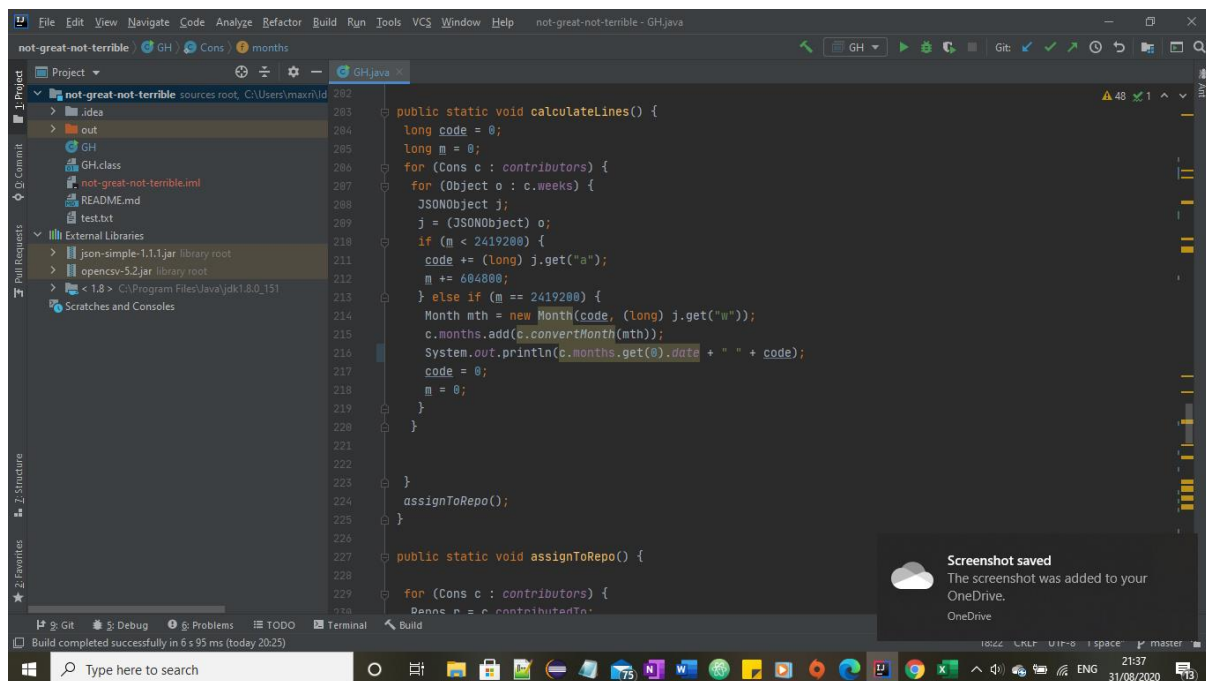


From this list, a “Cons” object(Figure 1.2) is created, each containing a JSONArray of weeks alongside the name of the contributor, and a Repo object that represents the repository they contributed to.



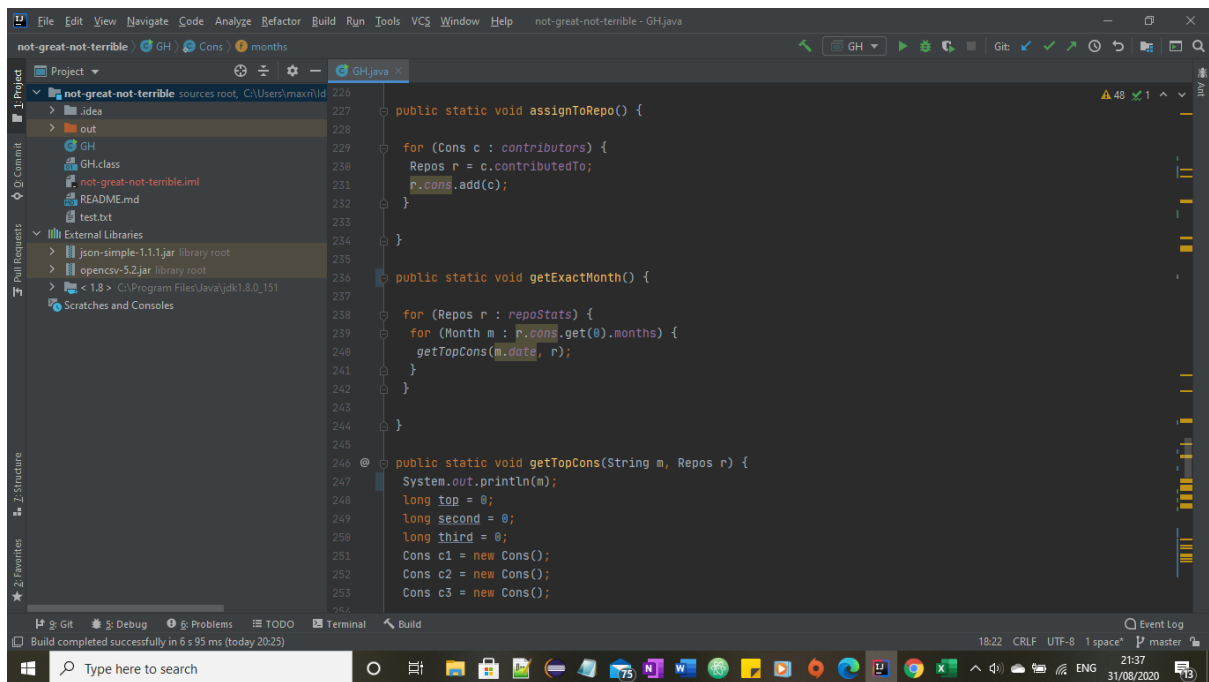
Now, for each cons object, their contributions for each month must be recorded. For this, I decided to stick with lines of code contributed per week. Therefore, to calculate each month, it involves counting “additions” for 4 weeks (4 weeks = 1 month). To do this, the system has to iterate through the JSONArray in each

Cons object in the list “cons”, getting the unix timestamp for each week as well as the “additions”, for each week. As the timestamp increments by 604,800 for each week, this number is continuously added to an int variable which starts at zero. Any time this variable reaches four times 604,800 (1 month), a new “Month” object(Figure 1.3) is created, which takes in the timestamp converted to a readable date as well as the lines of code calculated for those 4 weeks. The variable is then reset to zero to count the next month. Each month is added to a “months” list contained in each “Cons” variable.

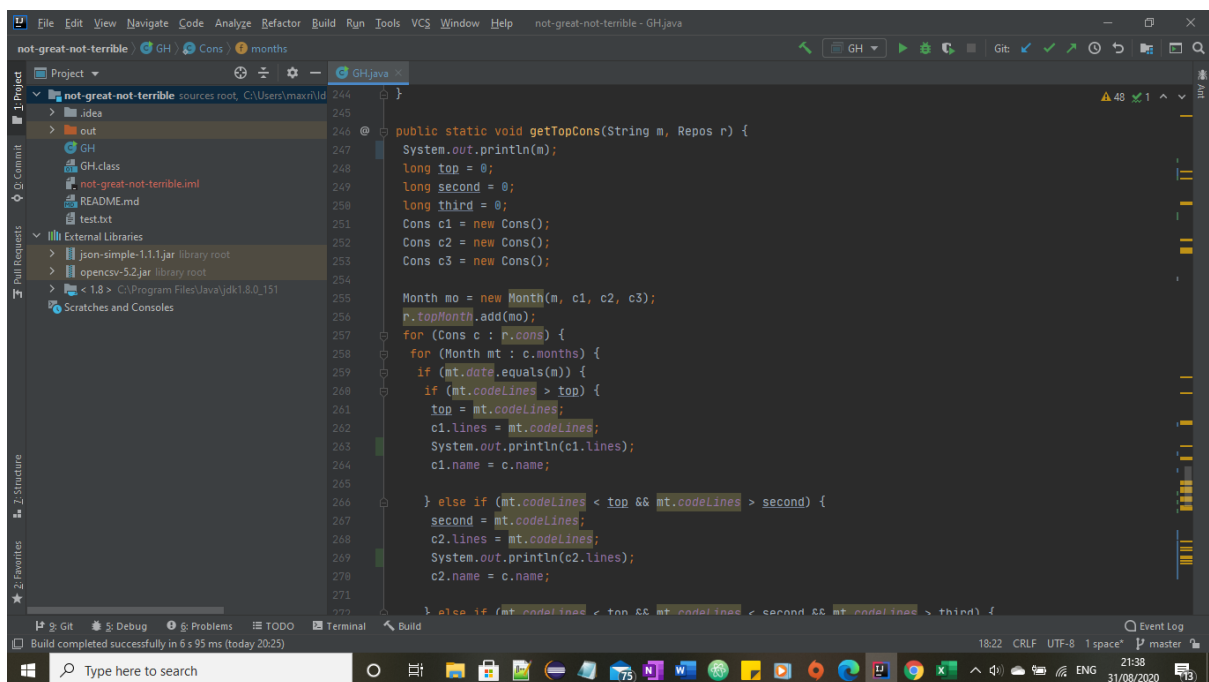


This left the calculations for top contributor for each month, to do. For this, a “contributor” list was created for each repository object, which contained all contributors to that repository. From that list the first contributor is selected, and their “months” list was iterated over to get each month of existence of each repository. Another method is called within this loop which calculates the top three contributors for each month in each repository. This method takes in a string for the date and a Repos object, and involves the creation of another Month object (Figure 1.4) which takes three Contributor objects as well as a date for each month calculated, which is added to a topMonth list contained in the Repos object. Three long variables are created for the top three contributions, each instantiated at zero. For each Repo object, its contributors list is iterated over, and in turn the months list for each Cons object is also iterated over to get their lines of code for each month. A series of if statements compare the lines of code to the long variables, replacing them if they are greater than the current values stored in the variables. A “lines” variable contained in each of the 3 Contributor objects within the Month variable is updated accordingly, as well as

a “username” variable, which is replaced by the name of the contributor who is one of the top 3 contributors.

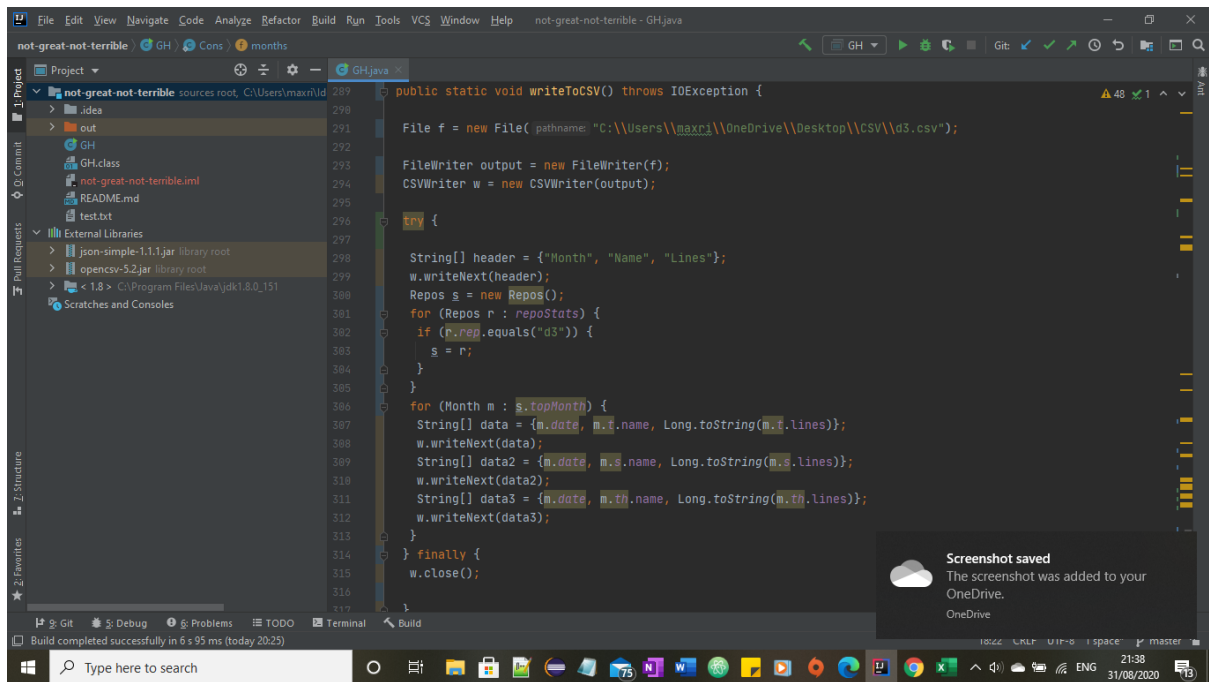


```
226 public static void assignToRepo() {
227
228
229     for (Cons c : contributors) {
230         Repos r = c.contributedTo;
231         r.cons.add(c);
232     }
233
234 }
235
236 public static void getExactMonth() {
237
238
239     for (Repos r : repoStats) {
240         for (Month m : r.cons.get(0).months) {
241             getTopCons(m.date, r);
242         }
243     }
244 }
245
246 public static void getTopCons(String m, Repos r) {
247     System.out.println(m);
248     long top = 0;
249     long second = 0;
250     long third = 0;
251     Cons c1 = new Cons();
252     Cons c2 = new Cons();
253     Cons c3 = new Cons();
254 }
```



```
246 public static void getTopCons(String m, Repos r) {
247     System.out.println(m);
248     long top = 0;
249     long second = 0;
250     long third = 0;
251     Cons c1 = new Cons();
252     Cons c2 = new Cons();
253     Cons c3 = new Cons();
254
255     Month mo = new Month(m, c1, c2, c3);
256     r.topMonth.add(mo);
257     for (Cons c : r.cons) {
258         for (Month mt : c.months) {
259             if (mt.date.equals(m)) {
260                 if (mt.codeLines > top) {
261                     top = mt.codeLines;
262                     c1.lines = mt.codeLines;
263                     System.out.println(c1.lines);
264                     c1.name = c.name;
265
266                 } else if (mt.codeLines < top && mt.codeLines > second) {
267                     second = mt.codeLines;
268                     c2.lines = mt.codeLines;
269                     System.out.println(c2.lines);
270                     c2.name = c.name;
271
272                 } else if (mt.codeLines < top && mt.codeLines < second && mt.codeLines > third) {
273                     third = mt.codeLines;
274                     c3.lines = mt.codeLines;
275                     System.out.println(c3.lines);
276                     c3.name = c.name;
277                 }
278             }
279         }
280     }
281 }
```

Once these calculations are complete, the data is then converted and written into a CSV file using opencsv. This is for the d3 visualisation.



The data visualisation involves html and css files for the website, and a javascript file for the chart to be created.

What went wrong?

The program works as intended, right up until the point where the top contributors are calculated and the data is written into a csv file. When it gets to this point, it incorrectly reads the month from the “getExactMonth” method, thus leading a lack of data in the csv file. An earlier attempt to create multiple csv files for each repository resulted in empty csv files and all of the data being written into only one of the files. Therefore I attempted to see if I could get the program to work for just one csv file, which was unsuccessful. Whilst I have been unable to pinpoint exactly the reason for the failure due to time constraints inflicted by my own actions, I have discovered that the failure occurs somewhere between the monthly contribution for each contributor being calculated and writing the data to csv files.

Visualisation

As the data could not be collected properly, the graph provided is a visualisation of what the finished product may have looked like. The data within the graph is random, intended to showcase my visions.

POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. [Don't show again](#) [Save As...](#)

Month	Name	Lines
1	1/1/2020	0
2	2/1/2020	0
3	3/1/2020	0
4	4/1/2020	0
5	5/1/2020	0
6	6/1/2020	0
7	7/1/2020	0
8	8/1/2020	0
9	9/1/2020	0
10	10/1/2020	0
11	11/1/2020	0
12	12/1/2020	0
13	1/1/2021	0
14	2/1/2021	0
15	3/1/2021	0
16	4/1/2021	0
17	5/1/2021	0
18	6/1/2021	0
19	7/1/2021	0
20	8/1/2021	0
21	9/1/2021	0

Data that the program currently outputs

FIGURES

```

49  m.date = dt;
50  return m;
51  }
52  }
53  }
54  //class for repositories
55  public static class Repos {
56
57      static String repo;
58      static JSONArray w;
59      static List<Cons> cons;
60      static List<Month> topMonth;
61
62
63      public Repos(String repo, JSONArray a) {
64
65          this.repo = repo;
66          this.w = a;
67          cons = new ArrayList<>();
68          topMonth = new ArrayList<>();
69
70      }
71
72
73      public Repos() {
74
75      }
76
77  }
  
```

Figure 1.1 – Repos class

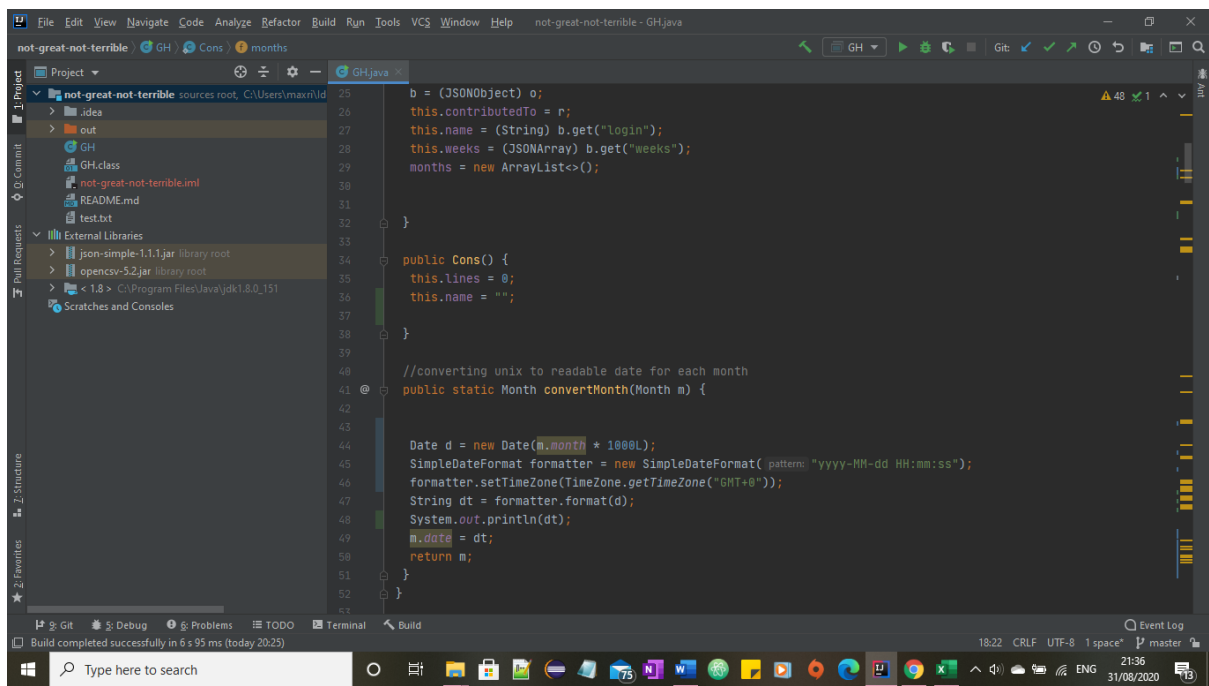


Figure 1.2 – Cons class

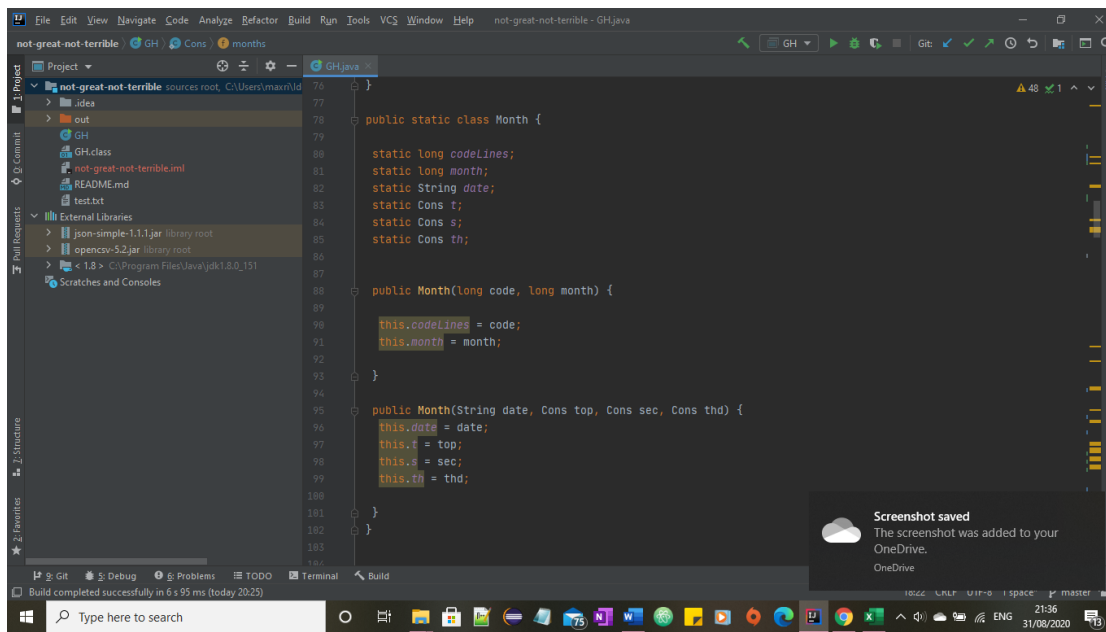


Figure 1.3 & 1.4 – Month class

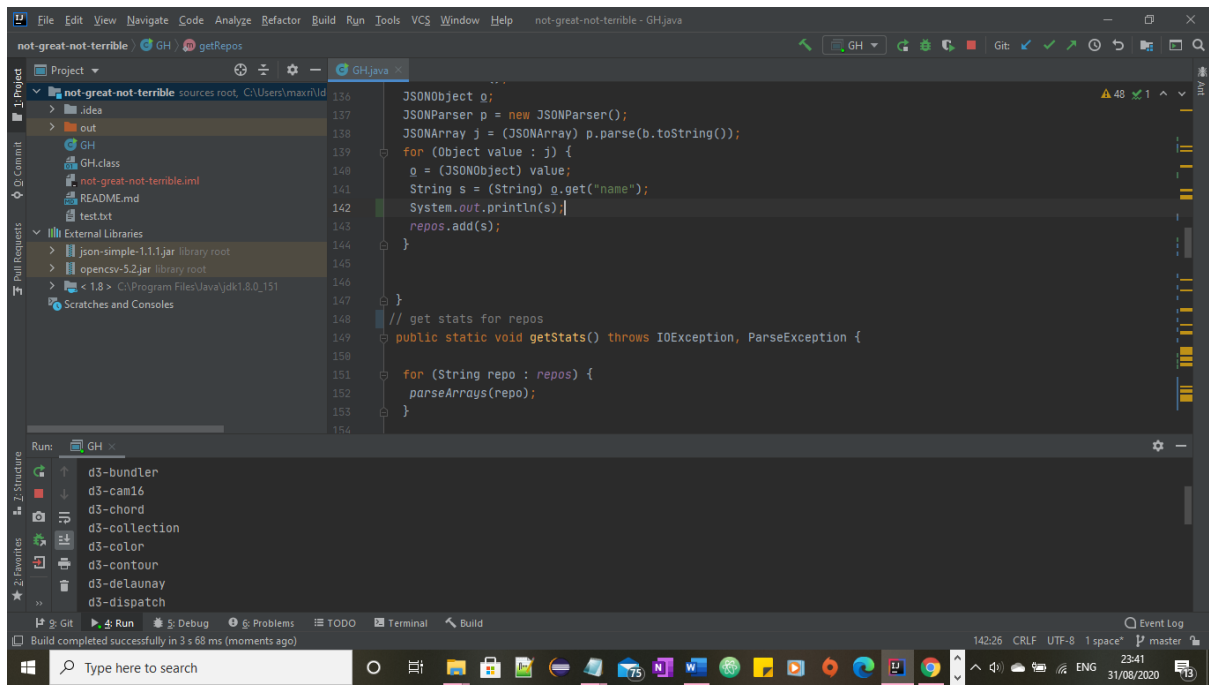


Figure 2.1 – each Repository collected by getRepos() printed to screen

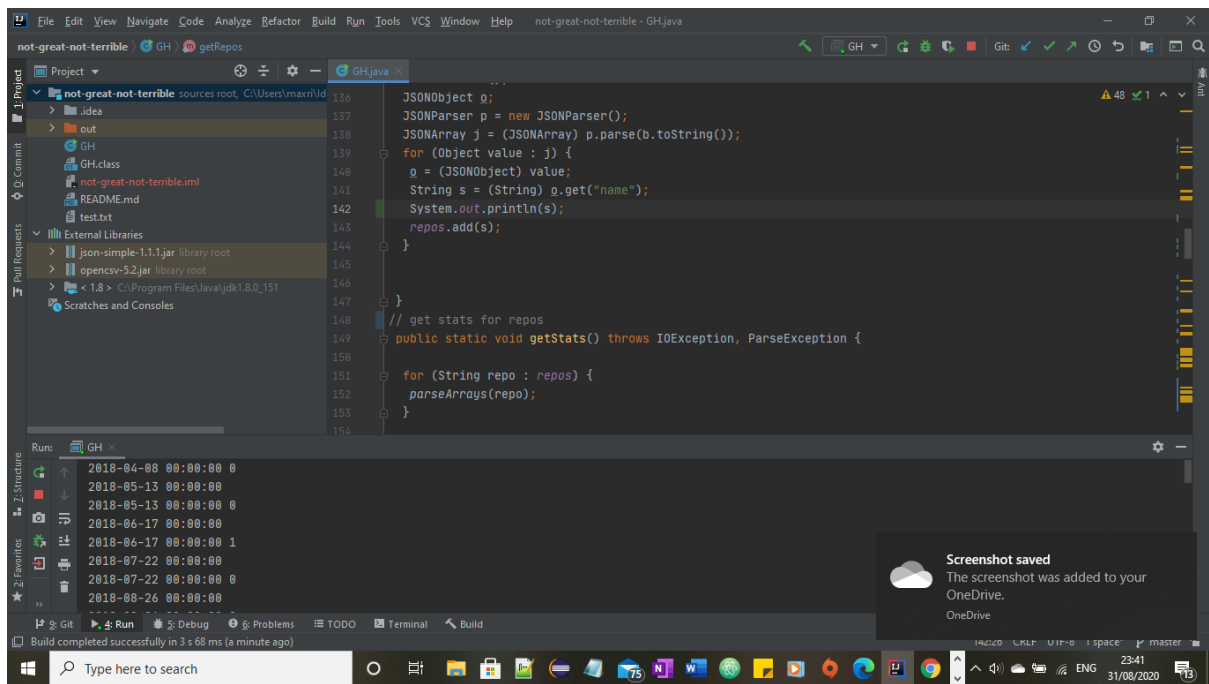


Figure 2.2 – Readable dates along with code for each month from calculateLines() being printed to screen