CS 106 Winter 2019

Lab 11: Trees

Due: Wednesday, April 3rd, 11:59pm

SAVE your sketch as “L11Q01”.

Question 1 Room Schedule

The University of Waterloo's Open Data API makes it possible to look up the weekly schedule of classes held in any given classroom. The API returns a JSON object containing a list of each course scheduled in the room, along with the days of the week and times of day that the course is held. We can use that information to figure out when the room is available and when it's free.

In this lab you will process a JSON object stored in a downloaded text file, containing the full schedule information for STC 0040. In a first step, you will print out text information about all scheduled courses in the room. Solving this step alone will earn you 70% of the marks for the lab. For the remaining marks, you must produce a visualization of the weekly schedule in a standard calendar arrangement.

1. The file classroom.json is provided. It was acquired using the UW API. Look through the file using a text editor, to a get a sense of how the data is laid out.
2. Add a `setup()` function to the sketch that loads the JSON file into Processing, storing it as a value of type`JSONObject`.
3. Iterate over all of the individual course offerings in the JSON file. For each one, print out (to the console) the subject and catalog number of the course, followed by the weekdays and start and end times. Format all this information on a single line. your output should look like this:

```
CHEM 357: MWF, 11:30-12:20
ENBUS 112: W, 14:30-16:20
FINE 112: TTh, 10:00-11:20
PLAN 203: TTh, 14:30-15:50
PLAN 233: T, 12:30-14:20
PLAN 261: MW, 10:00-11:20
SE 102: F, 13:30-14:20
STAT 202: MWF, 12:30-13:20
STAT 202: M, 14:30-15:20
```

4. To complete this step, you'll need to "drill down" into the JSON, using a combination of JSON get methods, local variables, and a loop over a JSON array. Every line in the output will rely on accessing five fields stored inside the information about a single course: `subject`, `catalog_number`, `weekdays`, `start_time`, and `end_time`.

Completing Steps 1–4 will earn you 70% of the marks available for this lab. If you want to earn all the marks, you'll need to do more:

5. You'll need a helper function that can take as an argument a string describing weekdays, like `"MWF"` or `"TTh"`, and determine which of the five weekdays are actually in that string. Ideally, the function will return an array of five boolean values, one for each weekday that appears in the passed-in string. This function is a bit fussy to write so here it is. Add it to the sketch.

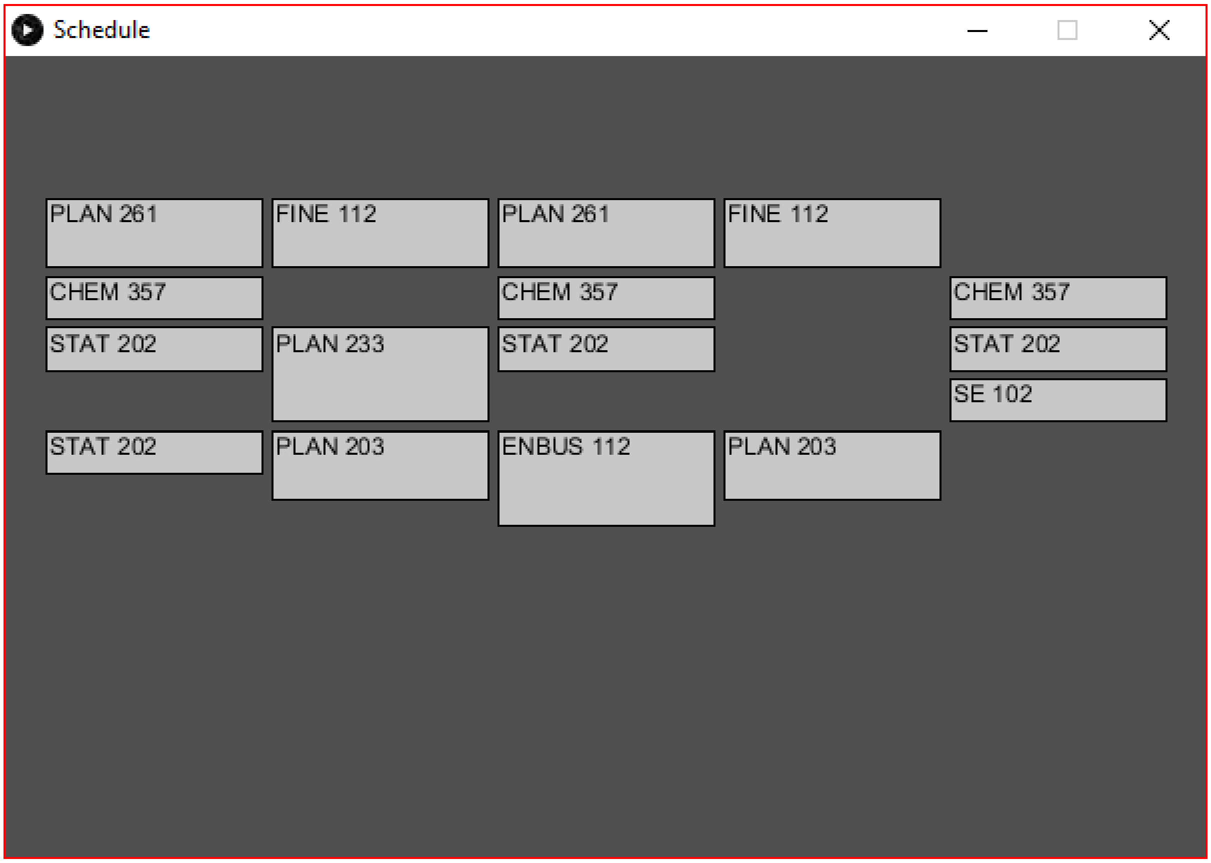
```
getDays( String str )
{
  boolean[] res = { false, false, false, false, false };

  int idx = 0;
  while ( idx < str.length() ) {
    char c = str.charAt( idx );
    if ( c == 'M' ) {
      res[0] = true;
    } else if ( c == 'W' ) {
      res[2] = true;
    } else if ( c == 'F' ) {
      res[4] = true;
    } else if ( c == 'T' ) {
      if ( (idx < str.length() - 1) && (str.charAt( idx + 1 ) == 'h') ) {
        res[3] = true;
        ++idx;
      } else {
        res[1] = true;
      }
    }
    ++idx;
  }
  return res;
}
```

   You'll use this function by giving it the contents of the `weekdays` fields inside the JSON file. So, for example, `getDays( "MWF" )` would return the array `{ true, false, true, false, true }`.

6. You'll also want a helper function that takes a string representing a time of day as an argument, and turns it into an integer representing the number of minutes from midnight until that time. Call the function `getTime()`. So, for example, `getTime( "08:30" )` should return 8×60+30 = 510, and `getTime( "14:20" )` should return 14×60+20 = 860. Use your text processing skills to write this function.

7. Every course in the JSON file is held at certain times on certain days. Create a visualization of a weekly calendar grid showing the use of the room. Set the size of the sketch window to be at least 600×400, and choose colours for the background, the schedule blocks, and text. Now, for every course, extract its name, along with the weekdays and start and end times. Use `getDays()` to figure out which days the course is held, and use `getTime()` to parse the start and end times into minutes since midnight. For each day that the course is held, draw a rectangle. The rectangle's horizontal position in the sketch window is determined by the weekday (so the visualization should have five columns). Vertically, the rectangle should start at the class's start time and run until its end time. Use `map()` to fit a reasonable portion of the day (say, 8:00am to 10:00pm) into the sketch window's height. Put the subject and catalog number of the course inside the rectangle. When you're done, the grid for STC 0040 should look something like this:



PLAN 261	FINE 112	PLAN 261	FINE 112	
CHEM 357		CHEM 357		CHEM 357
STAT 202	PLAN 233	STAT 202		STAT 202
				SE 102
STAT 202	PLAN 203	ENBUS 112	PLAN 203	

Note that the gaps between consecutive classes are correct: they reflect the 10-minute class change breaks in the schedule.

Of course, you can experiment with all sorts of enhancements to this visualization. Try using different colours for different courses. Try popping up more information about a course when the user hovers over one of the boxes in the schedule.

Submission

Submit all sketch directories from this lab as one ZIP file called L11.zip to the lab dropbox on Learn.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will receive a mark of 0.