CS5500 Summer 2020

Amie Kong

# Building Code:

1) Fork the given github repo.

2) Setting up the project :

   a) You will need the following software to run the project:
      i) Text Editor like IntelliJ / Eclipse ( We used IntelliJ)
      ii) MySQL, MySQL Workbench
      iii) Postman / Web Browser to visualize the GET queries

   b) Start running the MySQL server. On MySQL workbench (or any other software), execute the following statements to start building the database. This needs to be done only once, when the application is run for the first time.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=1;
 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=1; SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DAT
E,NO_ZERO
_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE SCHEMA IF NOT EXISTS `CustomerSimulator` DEFAULT
CHARACTER SET utf8 ;
 USE `CustomerSimulator` ;

Drop table if exists SimulationMetadata;
 Create table SimulationMetadata (SNo INT AUTO_INCREMENT,
GenerationDateTime VARCHAR(255), InputParamatersFileName
VARCHAR(255), GeneratedTableName VARCHAR(255), PRIMARY KEY
(SNo));
```

   c) Open the project directory in the terminal and build the project using :

```
Compile the project: mvn clean install

Run it using: mvn spring-boot:run
```

3) In postman or your browser, you can now execute the following to see the results : (prepend all the endpoints with localhost:8080/

**a) /customer**

**b) /customers**

**c) /customer-generation (POSTMAN)**

To generate the customers - this can only be run in postman and needs a json file as an input.

**d) /busiest-senior-hour (BROWSER)**

Brings up the page to enter the date and weather on the date for which the busiest senior hour is needed.

**e)**

**/comparison/date1/{date1}/weather1/{weather1}/timeSlot1/{timeslot1}/date2/{date2}/weather2/{ weather2}/timeSlot2/{timeslot2} (POSTMAN or BROWSER)**

Compare the number of people visiting the store on any two given time slots, on given dates and weathers

**f) /customer/date/{date}/weather/{weather}/id/{customerId} (POSTMAN or BROWSER)**

Get one single customer details, including age and time spent in the store

**g) /customer-count/date/{date}/weather/{weather}/(POSTMANorBROWSER)**

Get the number of customers in the store on any date

**h) /busiest-hour/date/{date}/weather/{weather}(POSTMANorBROWSER)**

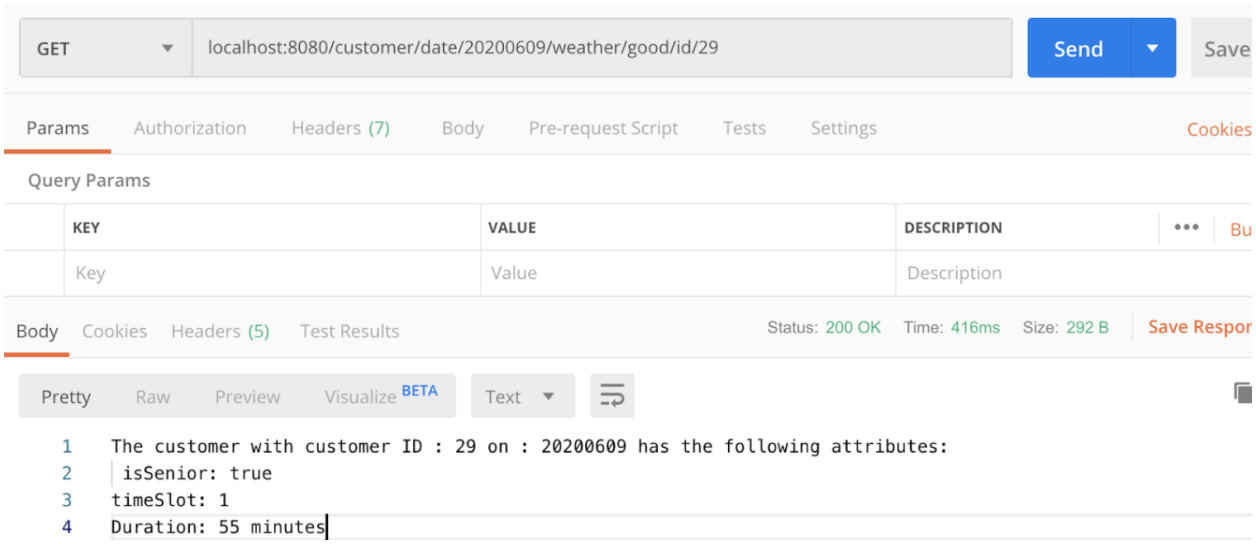
Get the busiest hour on a given date

**i) /generate-customers (POSTMAN)**

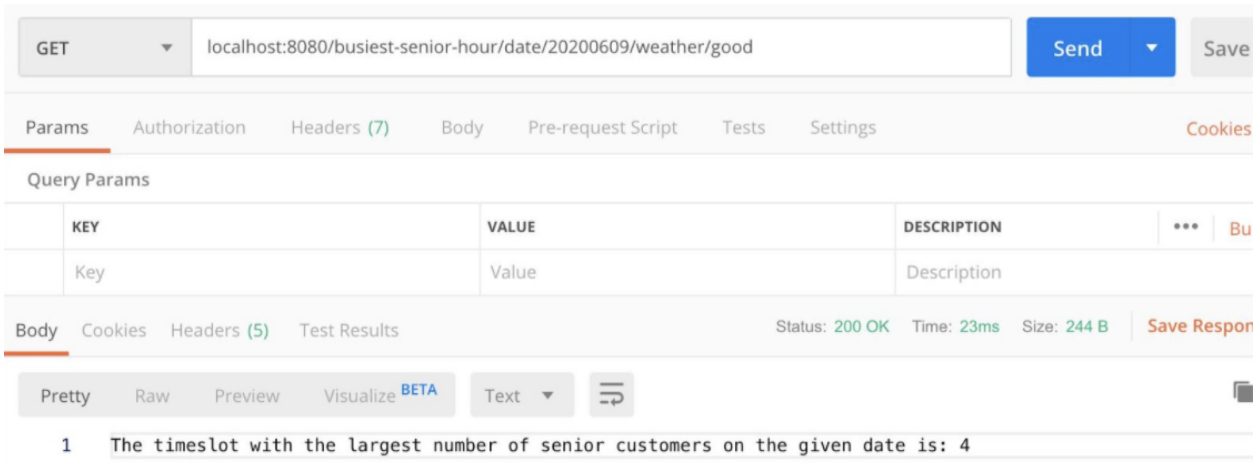
Takes a json file as input and generates the customers.

# Demo:

**Get amount of time spent by a user in the store along with details:**




**Output for hour with most senior customers:**

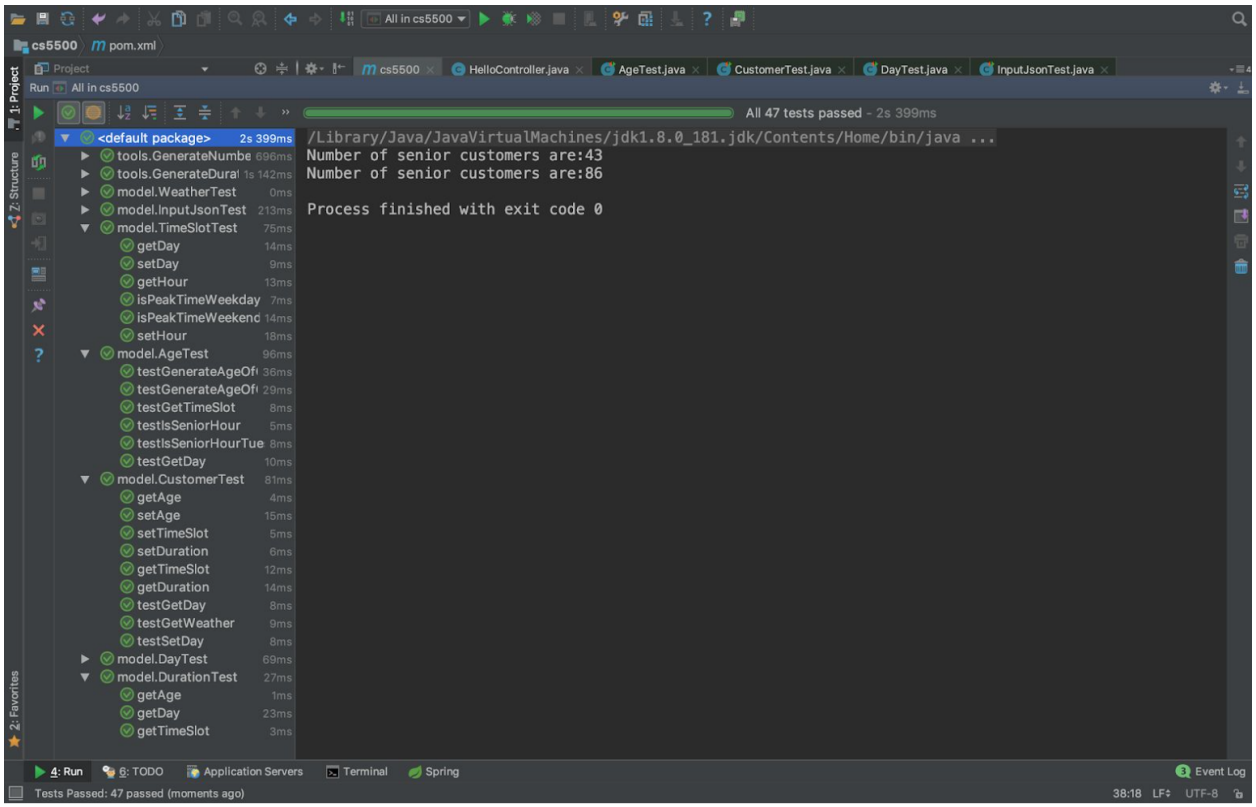



**Assumptions :**

1) The weather is currently only quantified as good weather or bad weather.
2) During lunch time and dinner time, the store will have 30% more customers grabbing food.
3) The weekends short visits (when weather is nice) are assumed to be about 20 minutes.
4) The age of customers is currently only quantified as senior or not senior. Normally the probability of a customer being a senior is 40%, but during the senior hour, the probability will increase to 85%.
5) The holidays are U.S. holidays and are determined using an external library.
6) It is assumed that the customer will input a “valid input”.

# Test and Code Metrics:

*Unit Tests*



*CodeMR Report*

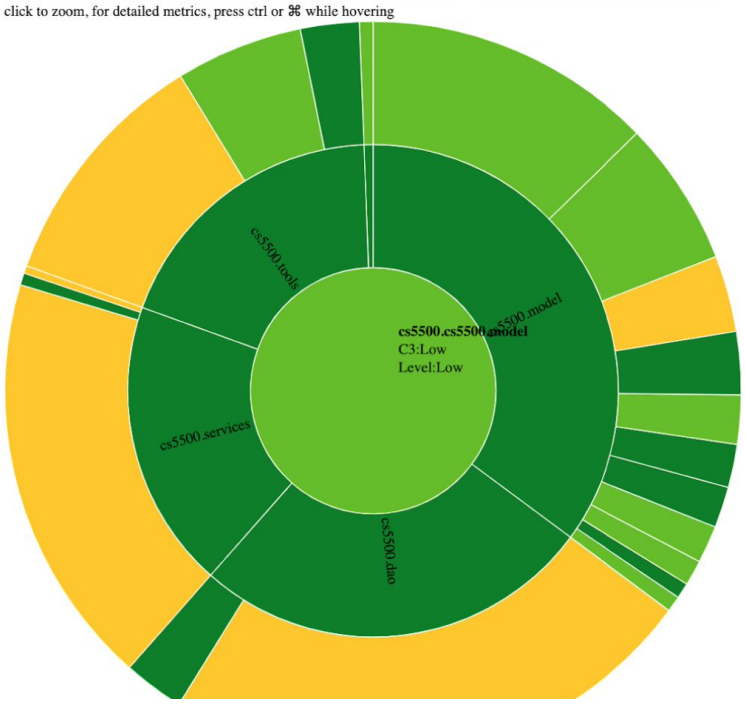
Metric Values in Sunburst Chart

Select metric for class, package and the project

Click to zoom into the packages

**Class Metric** C3 | **Package Metric** C3 | **Project Metric** Class Lines of Code

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering

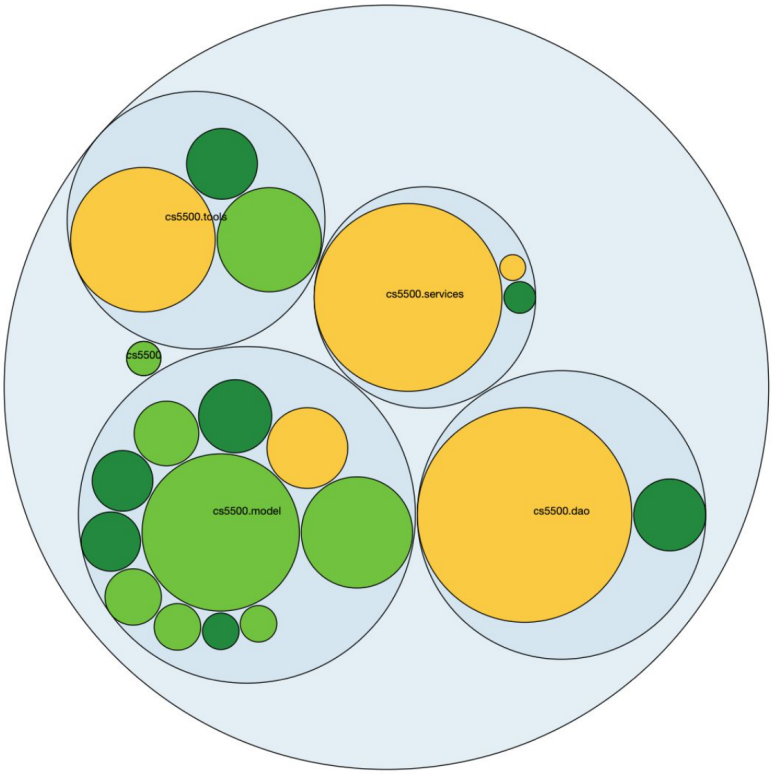


## Metric Values by Packages

Select class metric to visualize

**Metric Selection**

C3 click to zoom, for detailed metrics, press ctrl or ⌘ while hovering

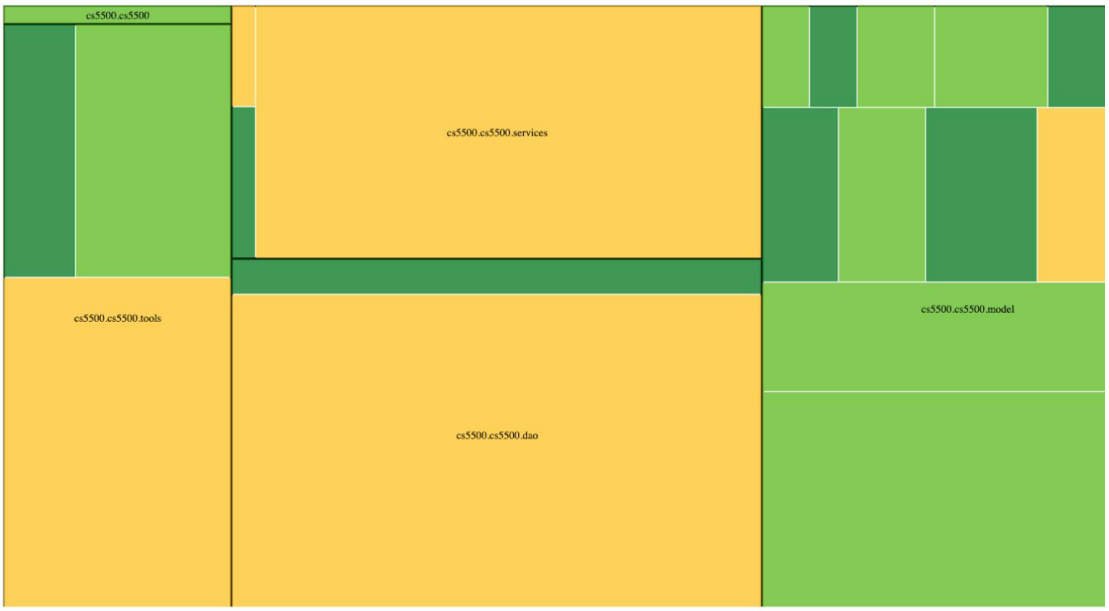



## Metric Values in Treemap Chart

Select class metric to visualize

**Metric Selection**

C3 click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



List of all classes (#20)

| ID | CLASS | COUPLING | COMPLEXITY | LACK OF COHESION | SIZE | LOC | COMPLEXITY | COUPLING | LACK OF COHESION | SIZE |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | GenerateCSV | | | | | 125 | low-medium | medium-high | low | low-medium |
| 2 | GenerateCSVContro... | | | | | 211 | medium-high | low-medium | low-medium | low-medium |
| 3 | GenerateDuration | | | | | 65 | low | low-medium | low | low-medium |
| 4 | Customer | | | | | 39 | low | low-medium | medium-high | low |
| 5 | CSVGeneratorDAO | | | | | 276 | medium-high | low | low | low-medium |
| 6 | URLNotFoundException | | | | | 4 | medium-high | low | low | low |
| 7 | InputJson | | | | | 148 | low-medium | low | low | low-medium |
| 8 | Day | | | | | 74 | low-medium | low | low | low-medium |
| 9 | HourlyTimeSlot | | | | | 25 | low-medium | low | low | low |
| 10 | DailyAverageNumCu... | | | | | 13 | low-medium | low | low | low |
| 11 | NameOfDay | | | | | 8 | low-medium | low | low | low |
| 12 | Cs5500Application | | | | | 7 | low-medium | low | low | low |
| 13 | Age | | | | | 32 | low | low | low | low |
| 14 | ConnectionManager | | | | | 31 | low | low | low | low |
| 15 | GenerateNumberOfC... | | | | | 30 | low | low | low | low |
| 16 | TimeSlot | | | | | 22 | low | low | low | low |
| 17 | Constants | | | | | 21 | low | low | low | low |
| 18 | Duration | | | | | 19 | low | low | low-medium | low |
| 19 | Weather | | | | | 8 | low | low | low | low |
| 20 | HelloController | | | | | 6 | low | low | low | low |

Metric Tables

Detailed metric tables

Classes with high coupling, high complexity, low cohesion (#0)

Classes with high coupling, high complexity (#0)

Classes with high coupling (#0)

Classes with high complexity (#0)

List of all classes (#20)