Lab 8

Submission deadline: 2359, Wednesday, November 1, 2017.

As Wei Tsang is away until Sunday, you get extra time to work on this lab.

Setup

The skeleton code from Lab 8 is available on `cs2030-i` under the directory `~cs2030/lab08`. `BusSg.java` is the main file that you need to edit and fill in the blanks.

Task

For Lab 8, we are implementing a simple bus information system, an extension of the midterm question. The classes `BusStop` and `BusService` have been given to you, as well as the main class `LabEight` and a partially implemented `BusSg`.

You are still required to complete five methods in class `BusSg`. This time you have some constraints -- you are not allowed to use loops (`while`, `for`) and must process your data using `Stream`. If the functor `Optional` are involved, you must not use `get` to retrieval the value inside the `Optional`. You should use specific `Collector` as asked.

You are still required to

- follows the CS2030 Coding Style [../style/index.html]
- clearly documented with `javadoc` [../javadoc/index.html] (this has been done for you, for free!)

Provided Classes

BusStop

A `BusStop` encapsulates information about a bus stop. It has an `id` (e.g., "16181"), a location (longtitude and latitude -- irrelevant for this lab), and a human-friendly name (e.g., "Computer Ctr").

A `BusStop` also has a collection of `BusService` objects that serve the `BusStop`. Your tasks involve populating and processing this collection. To do so, you can call `addBusService` to add a bus service to the collection and call `getBusServices` to retrieve the collection of bus services as a stream.

BusService

A `BusService` encapsulates information about a bus service. Each bus service has a string `id` (e.g, "96") and a collection of `BusStop` objects, corresponding to the bus stops served by the given bus service. Your tasks involve populating and processing this collection. To do so, you can call `addBusStop` to add a bus stop, and `getBusStops` to retrieve the collection of bus stops as a stream.

Tasks

- `readBusStopsAndServices`: Complete this method, which takes in a filename, reads line by line, and enters the relationship of which bus service serves which bus stop to the hash maps `busStops` and `busServices`. The hash maps are of the class `HashMapO` which wraps about Java `HashMap` -- `HashMapO` supports `Optional` return type `get` and `Stream` return type in `entries`.
- `averageNumberOfBusesPerStop` and `averageNumberOfStopsPerBus`. These two code are similar. You should use `Collectors.averagingDouble` to implement them.
- `busesWithMostStops` and `stopsWithMostBuses`. You should use `Collectors.groupingBy` to implement them. Note that despite the dataset

given returning only one bus / one stop, in other dataset it is possible to have multiple answers -- thus the return type of these should be of `Stream` type.

Data Files

Three data files about the bus services and bus stops are provided.

- `bus-services.csv`
- `bus-stops-services.csv`
- `bus-stops.csv`

The code for reading them explains what they are. During grading, we may test with other datasets in the same format.

Running LabEight

To run LabEight,

```
java LabEight bus-stops.csv bus-services.csv bus-stops-services.c
```

You may want to create a shell script to automate the above command (or use up arrow in `bash`)

Grading

This lab contributes another 4 marks to your final grade (100 marks).

- 2 mark for `readBusStopsAndServices()`
- 1 mark for `averageNumberOfBusesPerStop` and `averageNumberOfStopsPerBus`
- 1 mark for `busesWithMostStops` and `stopsWithMostBuses`

You can get -0.5 mark deduction for serious violation of style. Note that "correct" here not only means it gives the correct output, but it should follows the constraints

of using `Stream`, `Optional`, and `Collectors`.

Submission

When you are ready to submit your lab, on `cs2030-i`, run the script

```
1  ~cs2030/submit08
```

which will copy all files matching `*.java` (and nothing else) from your `~/lab08` directory on `cs2030-i` to an internal grading directory. We will test compile and test run with a tiny sample input to make sure that your submission is OK.

You can submit multiple times, but only the most recent submission will be graded.

> **Warning**
> Make sure your code are in the right place -- it must be in subdirectory named `lab08`, directly under your home directory, in other words `~/lab08`. If you place it anywhere else, it will not get submitted.