Sprint 1: Build Core Functions

Implementation of user stories

Project Group 8: Yao-Fu Yang, Qiushi Li, and Rong Chang
Trello board: https://trello.com/b/61ULBOVX/grocery-store-data-support
Git: https://github.ccs.neu.edu/yangyaof/cs5500_sum2020_group8

User stories

As a store manager, I need to know the patterns of customers shopping, so that I can plan operations to reduce customer waiting time, specifically:

- I need to know the pattern of **customer entry time** and **shopping duration** on each day to analyze the store occupancy at hourly intervals per day.
- I need to differentiate the **patterns of customer volumes** over the week including **weekdays** and **weekends** to analyze the staffing requirements over the week.
- I need to analyze the **changes** in customer volume and shopping duration for all the days that are affected by a **federal holiday**.
- I need to analyze the **changes** in customer volume and shopping duration on weekends due to environmental factors such as **weather**.
- I need to analyze changes in customer volume and shopping duration for all the days affected by special discounts, such as **senior discount times** on Tuesdays.

Input

The user needs the data to plan store operations under a COVID-19 scenario so that the store can provide services more efficiently to optimize staffing, minimize customer waiting time, in order to give customers better shopping experience.

Therefore, the data aims to (1) observe the customer visiting **distribution** over a day and over a week, and (2) records the length of **shopping time** at different times on different days.

We created a series of Java classes as depicted in the UML class diagram. Both the entry time and shopping duration were randomly generated by following the visiting pattern provided by the user. The day of the week and holiday identity were retrieved from the 2020 year calendar.

In our controller class, PilotSim, we have specified two constants reflecting the month and the number of days in a month (MONTH, DAY_IN_MONTH). This supports modifications to simulate data for different full months in the year 2020. We have made preparations to accommodate data requests for as early as 2016, but currently only support simulations in 2020.

Output

Our first attempt of data provided the shopping pattern within a range of one month, from May 01, 2020 to May 31, 2020.

Parameters (Column titles in csv file)
1. Visit ID.
2. Entry time: the date and time that a customer enters the store.
3. Leave time: the date and time that a customer leaves the store.
4. Shopping duration: shopping time (in minutes).
5. Day of the Week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday.
6. Holiday type: Indicates any holiday modifiers that may change the daily customer volume. Values are set to be none, holiday, day before holiday, week before holiday.
7. Day description: Additional descriptions of the day of this visit. Values are set to be regular weekday, regular weekend, senior discount weekday, nice weather weekend.

Features
1. The data allows the user to analyze the traffic of visits per day on weekdays and weekends.
2. The parameters entry time and shopping duration for each visit may be used by the user to analyze the peak hours during a day.
3. The holiday information and additional day descriptions provide additional parameters for the user to consider. This stages us for implementing a more sophisticated filtering and sorting mechanism to allow the user to quickly analyze the data based on these parameters.

HolidayType
- IS_HOLIDAY
- DAY_BEFORE_HOLIDAY
- WEEK_TO_HOLIDAY
- NON_HOLIDAY

Weather
- Weather(Boolean, Double)
- getWasNiceWeather() Boolean
- getAverageTemperature() Double

HolidayRelatedDates
- HolidayRelatedDates()
- getHolidaysByYear(int) HashMap<HolidayType, List<LocalDate>>

DateTime
- DateTime(LocalDateTime, Weather, HolidayType)
- getLocalDateTime() LocalDateTime
- setLocalDateTime(LocalDateTime) void
- getWeather() Weather
- setWeather(Weather) void
- isHoliday() HolidayType
- setHoliday(HolidayType) void

Visit
- Visit()
- getVisitID() String
- setVisitID(String) void
- getEntryTime() DateTime
- setEntryTime(DateTime) void
- getLeaveTime() DateTime
- setLeaveTime(DateTime) void
- getTotalTime() Integer
- setTotalTime(Integer) void

HolidayDeterminer
- getHolidayInfo(LocalDate) HolidayType

«create»

Constant
- Constant()
- getAmountOfCustomers() Map<String, Integer>
- getEntryTimeDist() Map<String, double[]>
- getDurationTimeDist() Map<String, Map<String, double[]>>

RandomGenerator
- generateEntryData(int, double[]) LocalDateTime
- generateDuration(double[]) int

Day
- Day()
- addVisit(Visit) void
- getVisits() List<Visit>

Util
- Util()
- findWeather(LocalDateTime) Weather
- isNiceWeather(String) boolean

DistributionDeterminer
- getDailyVolume(LocalDate, Constant, Util) int
- getEntryTime(int, LocalDate, Constant, Util) LocalDateTime
- getDurationDistribution(LocalDateTime, Constant, Util) double[]

CsvGenerator
- writeToCSV(List<Day>) void

«create»

PilotSim
- main(String[]) void

Powered by yFiles