**CSCSI262 ASSIGNMENT 3: HONEYPOT EVENT MODELLER**

***Members:***

* Huy Quan Ha (studentID: 5204537)
* Michael Johnson (studentID: 5777471)

**I/INITIAL INPUT**

1. **STORING VEHICLE TYPES AND STATISTICS**

* Class VehicleType: represent vehicle type with the following attributes:
  + String name: name of the type (bus, taxi…)
  + Boolean canPark: whether this type is able to park or not (0 false, 1 true)
  + String regFormat: the plate format of this type
  + Int volumeWeight
  + Int speedWeight
* Class Stat: represent the statistics corresponding to each type
  + String name: the same with VehicleType
  + Double numMean: the number mean taken from Stats.txt
  + Double numStdDev: the number standard deviation taken from Stats.txt
  + Double speedMean: the speed mean taken from Stats.txt
  + Double speedStdDev: the speed standard deviation taken from Stats.txt
* Class Traffic, which is our main class, which performs all the logging, analysis and alert action, will contain a HashMap<String,VehicleType> and a HashMap<String,Stat> that maps the name of the vehicle type/stat with the corresponding VehicleType/Stat object.

1. **POTENTIAL INCONSISTENCIES**

* When reading in the vehicle types within vehicles.txt - if EOF is reached and the number of vehicle types read in does not match the number specified then an inconsistency exception is raised.
* The number of vehicle types in vehicles.txt must also match the number of vehicle types in stats.txt. This is also the same case for the vehicle types - their names must match.
* In the stats.txt file the length and max speed must be greater than zero. The parking spaces available, volume mean and speed mean as well as their respective standard deviations must be all greater than or equal to zero.

**II/ACTIVITY ENGINE AND THE LOGS**

1. **EVENT GENERATION PROCESS**

* First, class Event is created which contains various information we need in order to generate, keep track, and log the event:
  + String vehicleType: this will be the same with VehicleType’s name
  + String plate: the particular plate for the related Vehicle of a VehicleType
  + EventType eType: an enum, which can take on 5 different values: ARRIVAL, DEPART\_SIDE, DEPART\_END,PARK, MOVE
  + Int d,m: day and minute respectively, representing the time the event has taken place
  + ArrivalEvent and ParkEvent are 2 special Events that extend the base class. The reason is because we need another extra attribute called arrivalSpeed for ArrivalEvent, and park (to represent whether the vehicle is currently parking or not) for ParkEvent
* Determine the event probability distribution:
  + Among 5 different types of event, EventType.MOVE and EventType.DEPART\_END does not need to be randomly generated. During the process, if a vehicle is not currently parking, of course it will be moving on the road, so that’s an event we need to generate every minute for every vehicle that is not currently parking. When the vehicle travels a distance equals the length of the road, the DEPART\_END event needs to be generated because the vehicle will leave the road, but it’s a deterministic event
  + With the remaining 3, the probability is: DEPART\_SIDE 5%, PARK 5% and ARRIVAL 90%
* Generate event:
  + At the beginning of each day, we clear any vehicles that remain from the previous day (if any), then use the getGaussianNumber() method of Stat to generate a number consistent with the Gaussian distribution given in the file for every vehicle type. This will be our bag of vehicles for the day, so each time a new Arrival Event is to be generated, we take a vehicle randomly from the bag, and once we ran out of vehicles we don’t generate the Arrival Event anymore. During this step, we also take care of generating a unique plate for each vehicle by using a HashSet to store all the plates used so far, in order to make sure we can use the plate as a form of identity for a vehicle.
  + For each minute in the day, if there’s currently no vehicle in the system, we have to generate an Arrival Event, this is a corner case where we can’t generate anything else. If the number of remaining vehicles in our bag is 0, or if the minute is over 23:00, then we have to break out of the loop because these are the terminating conditions.
  + While we have some vehicles in the system, for each of them, we test if it is parked or not to generate a new Move Event on the vehicle. We change the current speed by a small delta (-1.0 to 1.0km/h change in speed), and adds the distance travelled in that minute to the total distance the vehicle have travelled. If this total distance >= length of the road, we generate a DEPART\_END Event
  + For each vehicle, we now generate a random event among ARRIVAL, DEPART\_SIDE and PARK.
    - If the event is ARRIVAL, we call the getGaussianSpeed() method in Stat class to generate an arrival speed for the new vehicle consistent with the given normal distribution in the file. Here we check that both of these conditions are true before generating the event: m<=1380 (23:00 of the day), and we still have remaining vehicles in the “bag”
    - If the event is DEPART\_SIDE, we simply remove the vehicle from the system
    - If the event is PARK, we check the current status of the vehicle (isParked attribute in Vehicle class). If the vehicle is parking, we set the status to false, and generate another initial speed for the vehicle by calling getGaussianSpeed() to make it move again. Otherwise if the vehicle is currently moving, we change its parking status to true provided there’re still enough parking spaces, then we decrease the number of parking spaces by 1.
  + Any event generated will be recorded in an ArrayList<Event> events, which will later be printed out to the log file.

1. **LOG FILE FORMAT**

* Name: log.txt
* The first line is the number of days we record the log for
* Each subsequent line is of the form: VehicleType,Plate,EventType,day,minute (comma separated)
* Except for Arrival Event and Park Event, which we specified above are special events because they have some extra attributes:
  + Arrival Event: VehicleType,Plate,ARRIVAL,ArrivalSpeed,day,minute
  + Park Event: VehicleType,Plate,PARK,StartPark/StopPark,day,minute (depending on whether the vehicle is currently parking or moving)
* With this file format, we can easily read in the number of days the log is recorded, then for each subsequent line to get the type, the plate, the event type, and the time the event happened by separating all the fields around the comma. With special type of events such as Arrival and Park, by including the Event Type in the log, we can customize our action to handle those types of events, taking extra attributes like arrival speed if needed.

1. **POSSIBLE ALARMS**

* Vehicle Arrival Event is only up to 23:00 every day, so we raise an alarm if it’s already 23:00 and no more vehicle events will be generated
* Vehicle Arrival is also possible if there’re remaining vehicles in the “bag” of the same type, so if there’re no more vehicles of that type, we raise an alarm and don’t generate a new vehicle arrival
* When the number of vehicles in the system is 0, we also raise an alarm and generate only the vehicle arrival event (because we have no vehicles in the system to generate any other type of events). If no remaining vehicles are left, or if it is already over 23:00, we also raise and alarm saying we’re done for the day, and move to the next day without wasting time looping in the current day.

**III/ANALYSIS ENGINE**

1. **DAILY TOTAL FILE**

* Name: dailyTotals.txt
* Format: Day:VehicleType:TotalNum:AverageSpeed (colon separated), in which:
  + Day is the day number that we record the totals for a vehicle type
  + TotalNum is the total number of vehicles of that type arriving at the road that day
  + AverageSpeed is the mean speed of the above vehicles of that type arriving at the road (we only calculate the average speed by taking into account the arrival speeds of the vehicles)

1. **POSSIBLE ANOMALIES IN READING LOGS**

* If the log.txt file has been modified by a user prior to the analyze function, the format will have to be consistent.
* If an old log.txt file is currently open on the user’s system prior to the analyze function, an old log file may be analyzed instead of the current log file.

1. **POSSIBLE ANOMALIES IN DETERMINING STATISTICS**

* Over all days, in the case of a volume count of a vehicle type being 0, then it must mean that the speed average of that vehicle must also be 0. This is the same for their respective standard deviations.
* Since their is the case of a standard deviation being 0, it means that we must check the standard deviation is greater than zero when determining the anomalies.