**Java Coursework -** highlighted text is level 1

**Types of vehicle**

* **Small cars**
  + 7-9 gallon tanks // variable (int) + method – getTankSize
  + 1 unit of space in queue // variable (double)
  + Probability of p per tick // variable (double) + method – getProbability + setProbability
  + If refill is done in under 5 mins, probability of 0.3 the driver will shop for 2-4 mins and spend extra £5-£10 // variable (static) + variable(double) + method - accessor
* **Motorbikes**
  + 5 gallon tanks // variable (int) + method – getTankSize
  + 0.75 units of space in queue // variable (double)
  + Probability of p per tick // variable (double) + method – getProbability + setProbability
  + Never go to shopping area // variable(static)
* **Family sedans**
  + 12-18 gallon tanks// variable (int) + method – getTankSize
  + 1.5 units of space in queue // variable (double)
  + Probability of q per ticks // variable (double) + method – getProbability + setProbability
  + If refill is done in under 10 mins, probability of 0.4 the driver will shop for 2-5 mins and spend an extra £8-£16 //variable (static) + variable(double) + method - accessor
* **Trucks**
  + 30-40 gallon tanks // variable (int) + method – getTankSize
  + 2 units of space in queue // variable (double)
  + If refilled in under 8 mins, will ALWAYS shop for 4-6 min and spend an extra £15-£20 //variable (static) + variable(double) + method - accessor
  + Unhappy drivers will let other truck drivers know //
  + Arrive with probability of t, which is initially t0= 0.02 //variable (static)
  + Unhappy drivers reduce t by 20% of its current value: t’=0.8t //
  + Happy drivers increase t by 5% of its current value, to the original value of t: t’=min{1.05t, t0}

**Pumps**

* Provide 1 gallon per tick of fuel //variable (static)
* Has a queue that can fit up to 3 units of space // variable (static)
* Customer always goes to least occupied queue // array
* If vehicle does not fit into any queue they leave
* Always fully top up their tanks
* One gallon = £1.20
* Vehicles stay in queue while driver is paying
* Vehicle starts topping up on next tick after it gets to the front of the queue. Driver goes to till on the next tick after it tops up

**Tills**

* Paying at tills takes 2-3 minutes
* Customer always goes to least occupied queue
* Their vehicle remains in pump queue while they are paying
* Includes a shopping area
  + Happy customers that refill quickly will look around the shop before going to till on the next tick

**Extra points**

* Must track how much money was earned and lost (due to vehicles not fitting)
* Write test classes for at least 5 classes
* Need to track money lost from missed sale i.e. if they didn’t fill up fast enough or didn’t shop
* Users should be able to set values of p, q, price of gallon, period of time simulation runs
* Should be run for 4 hours (1440 ticks)
* Results averaged over 10 different seeds for a random number generator
* Should use all the independent combinations of the values below:
  + P: 0.01, 0.02, 0.03, 0.04, 0.05.
  + Q: same as P
  + Pumps: 1, 2 and 4
  + Tills: 1, 2 and 4
  + With and without trucks
    - We should run separate studies with and without trucks; the owners of the garage are wondering if they should allow trucks or not, so should decide if it’s better to allow them or not.

**Design notes**

* Write small library of classes that can support similar scenarios (more types of vehicle, smarter queuing) without change to the library classes
* Deciding which vehicle will arrive, generate a random number between 0-1
  + Less or equal to p – small car
  + Between p and 2p – motorbike
  + Between 2p and 2p+q – sedan
  + Between 2p+q and 2p +q +t – trucks
  + Otherwise no vehicle

**Report**

* Clearly state if doing level 1 or level 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Small cars** | **motorbikes** | **Sedans** | **trucks** |
| **3** |  |  |  |
| **2** | **1** |  |  |
|  | **4** |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |