PARKING LOT Case Study

Team Members Name:

SIRISH SEKHAR
ABHAY KRISHNA
SRI VIBHAV J
PANDRALA ARAVIND KUMAR

Discussions:

1. 1) There can be a membership based parking system for taking the parking tickets. By this the process becomes faster.
2. 2) Each floor is categorised according to the weight of the vehicles wherein heavier vehicles can be filled in the lower floors and lighter vehicles in the top.
3. 3) There can be a separate parking spot for the cycles and electric vehicles to promote environmental protection awareness and less parking fee would be charged for such vehicles.
4. 4) Credit cards would accept a few commonly used international currencies and the membership system would allow hassle free experience when it comes to giving out change, but it isn't mandatory
5. 5) Valet system could be implemented and charged extra based on requirement

How to design the system?

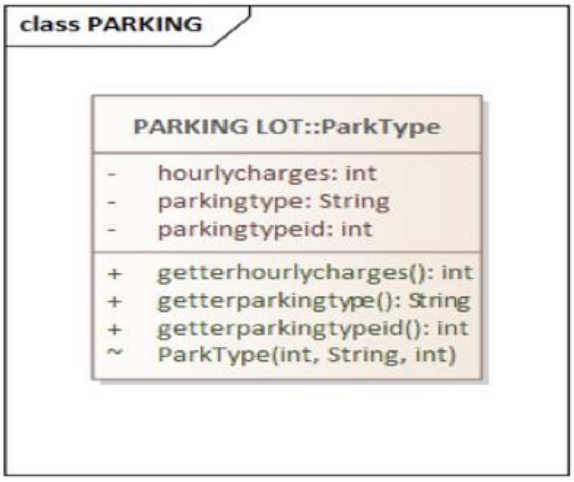
We have used a trial and error approach towards writing the code. We have gone through different approaches of solving an issue and picked a method easier for the user to access. We documented the code properly citing the functionality of each method and class. We have also extensively applied new approaches everytime we encountered an error.

In this process we incorporated new classes according to our convenience as the need to include new utilities increased.

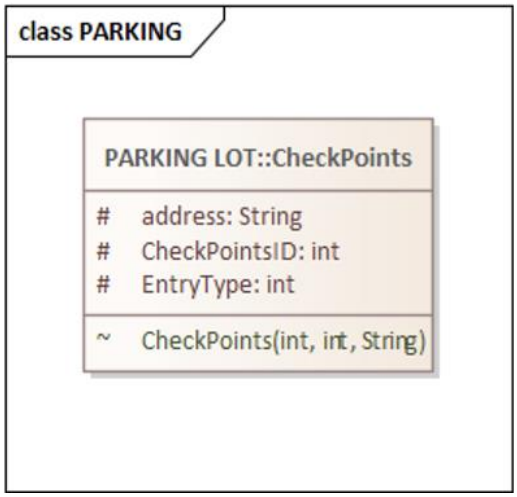
How many classes do we need?

1) We have written the code for 9 classes.

2) **ParkType** class : This class categorizes the parking vehicle types(i.e. Compact,Large,Handicapped,motorbike and electric car) and their Hourly charges.

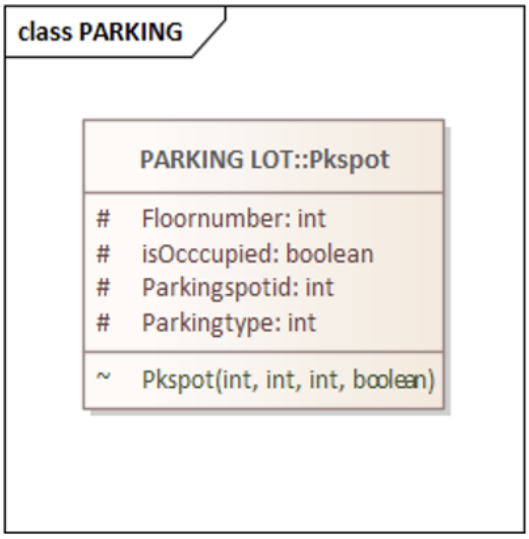


3) **CheckPoints** class : This class categorizes the entry and exit points of the Parking lot. We have designed 8 gates, 4 of which are for entry.

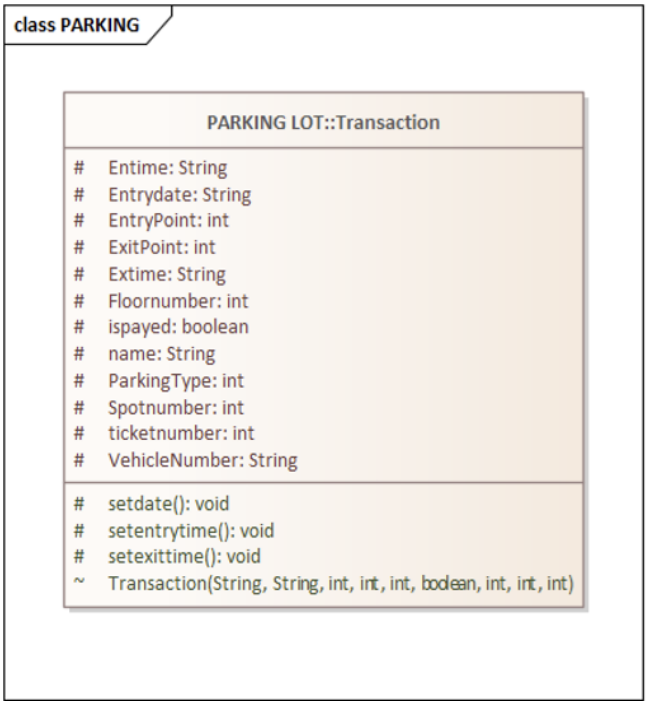


4) **PkSpot** class **:** This class categorizes the basic parking layout for each floor consisting of 5 terminals for the 5 types of vehicles. Each

terminal has 5 spots. So there are 25 spots per floor and 75 spots for the whole lot.

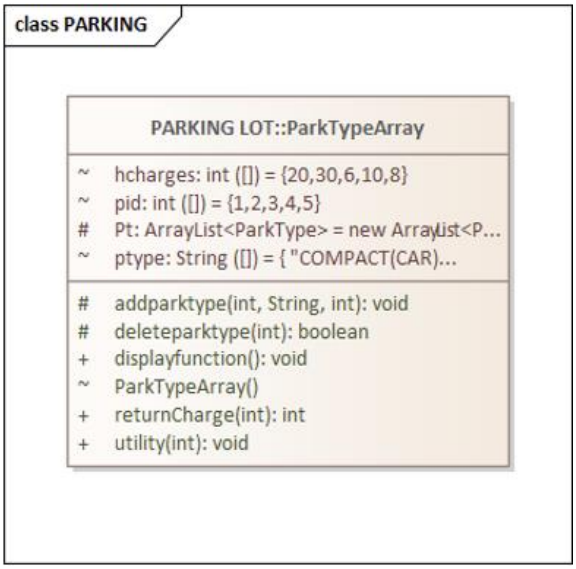


5) **Transaction** class : This class employs the transaction part of the code. It records the entry and exit time and calculates the parking bill accordingly.

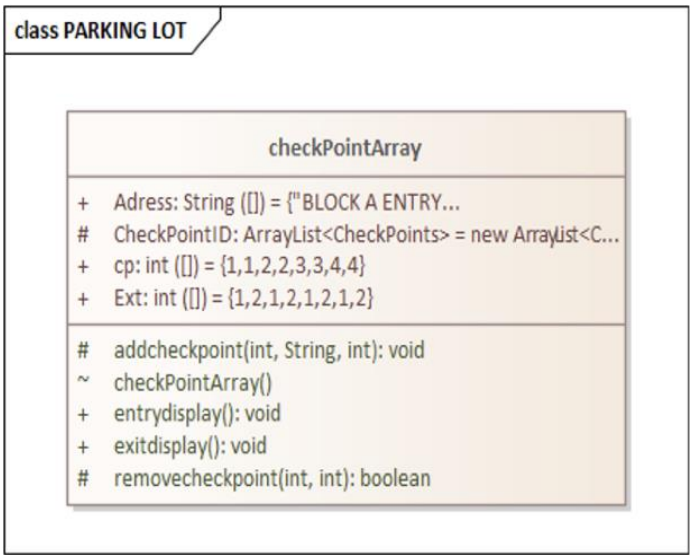


6) **ParkTypeArray** class : This class contains an Arraylist of type ParkType and initialises the object accordingly. The array classes are

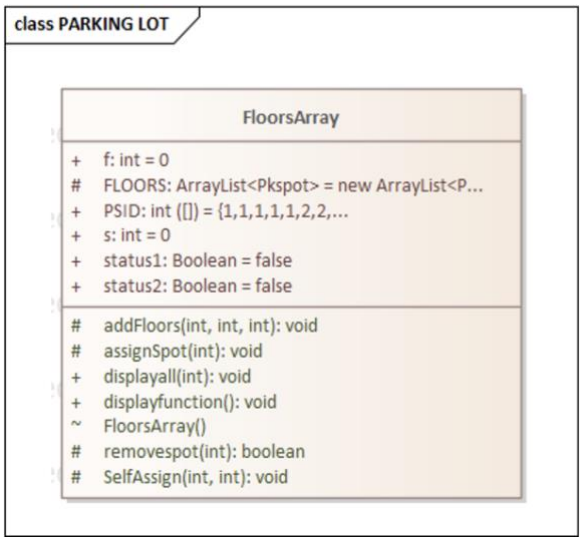
created for the convenience of Admin. This class helps for the dynamic allocation of extra vehicle types to tailor to the needs of admin.



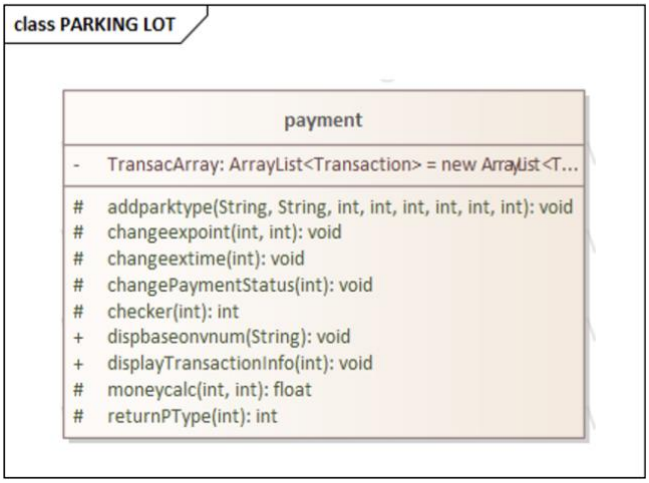
7) **CheckPointArray** class**:** This class contains the Arraylist of type CheckPoints and the constructor initialises the objects with the entry and exit points as defined by the admin. This class allocates extra entry and exit points and can be accessed by admin if required.



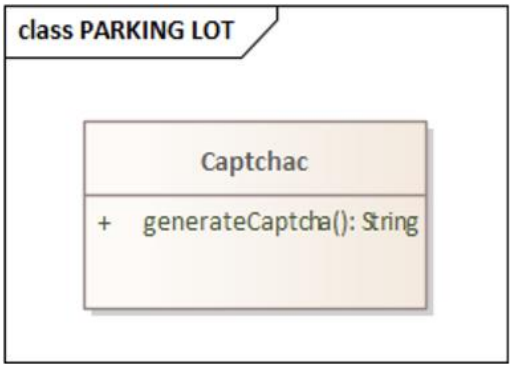
8. 8) **FloorsArray** class **:** This class is inherited from the ParkTypeArray and this class initialises the number of floors for the parking lot and assigns spots for each floor.



9) **Payment** class : This class is also inherited from the ParkTypeArray class and this class is utilised for the payment section of parking lot. This is devised to give both cash / card payment options.



10) **Captcha** class : This class generates a captcha randomly. This captcha can be used in the Card payment section.



Who is doing what work?

Abhay Krishna- code layout design, documentation, specific classes like captcha and suitable access specifiers.
Aravind- code layout design, documentation, class checkpoint and checkpoint array.
Sirish- UML diagram from java code, code layout design, class floors, ParkingType and all admin related utilities

Vibhav- code layout design, class (Transaction and payment) and all payment related functions.

What things are we adding that are not mentioned in the case study instructions?

The case study has been mentioned to include a static version of a hypothetical parking lot, but in reality, each parking lot has its own types and preferences. We have tried to add the functionality of adding as many parking floors and Vehicle type as required and an updatable menu which automatically reads the new inputs given by an admin so that he can tailor the parking lot according to his preference. The entire agenda of the admin section is to allow for the admin who uses the java code to make it according to his preference and simultaneously keeping the main functions intact.

What is the final design of the system (agreed by all)? (you can boxes and show the connections between them)