PARKING LOT
Case Study

Team Members:

1.Abhay Krishna

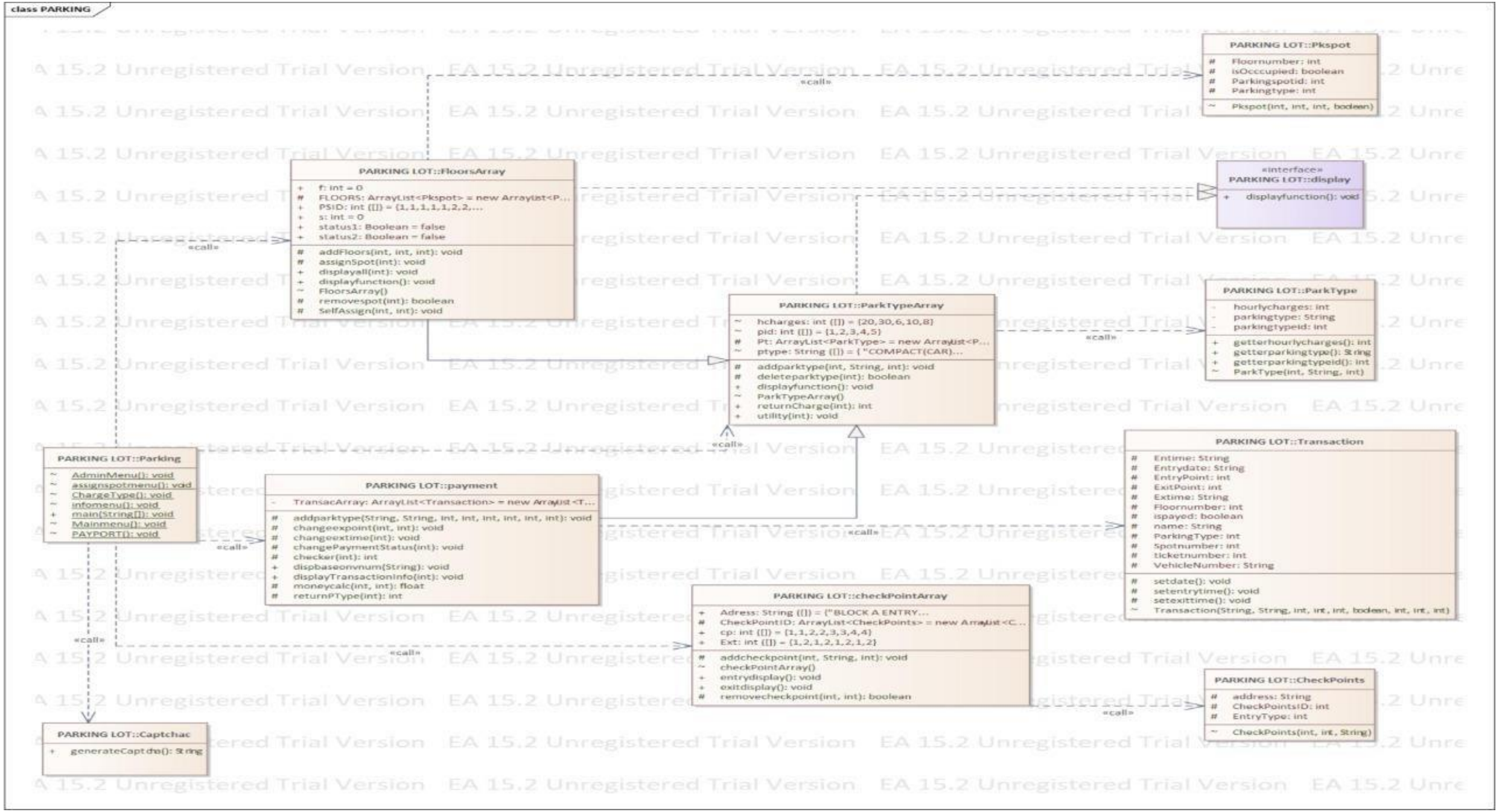
2.Aravind Kumar

3.Sirish Sekhar

4.Sri Vibhav J

Failed Assumption we made at first:

1) At first, we thought 3 floors each categorized according to the weight of the vehicles where in heavier vehicles will be filled in the lower floors and lighter vehicles in the top. But then we changed such that we can park all types of vehicles in all floors
2) We first made the 3 different array list for each floor but then we changed that because it was hard to connect each floor for payment method.
3) Initially we didn't use any interfaces nor did we add any access modifiers in our code making it very inconvenient as we had many methods with same name but different body
4) Initially we didn't keep any restricted access in our code, and anyone could access any class. But later we added utilities to prevent this.
5) At first, we made public array for the payment method but then we changed that to private & protected to keep the payment details confidential and we also kept a password for that which is only accessible by the admin.
6) Initially we cannot extend any floors, vehicle types or anything else but the we added methods which can extend the required things in future.
7) Initially we didn't keep any multiple exit points nor entry points.
8) Initially we forgot to keep any charging point for electric vehicles. But then we fixed the issue by adding the necessary function required.


class PARKING
PARKING LOT::Pkspot
Floornumber: int
isOccupied: boolean
Parkingspotid: int
Parkingtype: int
~ Pkspot(int, int, int, boolean)
«call»
PARKING LOT::FloorsArray
+ f: int = 0
# FLOORS: ArrayList<Pkspot> = new ArrayList<P...
+ PSID: int [[]] = {1,1,1,1,1,2,2,...
+ s: int = 0
+ status1: Boolean = false
+ status2: Boolean = false
# addFloors(int, int, int): void
# assignSpot(int): void
+ displayall(int): void
+ displayfunction(): void
~ FloorsArray()
# removespot(int): boolean
# SelfAssign(int, int): void
«interface»
PARKING LOT::display
+ displayfunction(): void
«call»
PARKING LOT::ParkTypeArray
~ hcharges: int [[]] = {20,30,6,10,8}
~ pid: int [[]] = {1,2,3,4,5}
# Pt: ArrayList<ParkType> = new ArrayList<P...
~ ptype: String [[]] = { "COMPACT(CAR)...
# addparktype(int, String, int): void
# deleteparktype(int): boolean
+ displayfunction(): void
~ ParkTypeArray()
+ returnCharge(int): int
+ utility(int): void
PARKING LOT::ParkType
- hourlycharges: int
- parkingtype: String
- parkingtypeid: int
+ getterhourlycharges(): int
+ getterparkingtype(): String
+ getterparkingtypeid(): int
~ ParkType(int, String, int)
«call»
PARKING LOT::Parking
~ AdminMenu(): void
~ assignspotmenu(): void
~ ChargeType(): void
~ infomenu(): void
+ main(String[]): void
~ Mainmenu(): void
~ PAYPORT(): void
«call»
PARKING LOT::payment
- TransacArray: ArrayList<Transaction> = new ArrayList<T...
addparktype(String, String, int, int, int, int, int, int): void
changeexpoint(int, int): void
changeextime(int): void
changePaymentStatus(int): void
checker(int): int
+ dispbaseonvnum(String): void
+ displayTransactionInfo(int): void
# moneycalc(int, int): float
# returnPType(int): int
«call»
PARKING LOT::Transaction
Entime: String
Entrydate: String
EntryPoint: int
ExitPoint: int
Extime: String
Floornumber: int
ispayed: boolean
name: String
ParkingType: int
Spotnumber: int
ticketnumber: int
VehicleNumber: String
setdate(): void
setentrytime(): void
setexittime(): void
~ Transaction(String, String, int, int, int, boolean, int, int, int)
«call»
«call»
PARKING LOT::checkPointArray
+ Adress: String [[]] = {"BLOCK A ENTRY...
# CheckPointID: ArrayList<CheckPoints> = new ArrayList<C...
+ cp: int [[]] = {1,1,2,2,3,3,4,4}
+ Ext: int [[]] = {1,2,1,2,1,2,1,2}
# addcheckpoint(int, String, int): void
~ checkPointArray()
+ entrydisplay(): void
+ exitdisplay(): void
# removecheckpoint(int, int): boolean
«call»
PARKING LOT::CheckPoints
address: String
CheckPointsID: int
EntryType: int
~ CheckPoints(int, int, String)
PARKING LOT::Captchac
+ generateCaptcha(): String

Final Outcome:

In total we made 9 classes. They are

1.**ParkType:**

This class consists of parking type(string), parking ID (string), Hourly charges(int).

2.**CheckPoints:**

This class consists of Checkpoint's ID (int), Entry type(int), Address (String)(takes the name of the entry point)

3.**PkSpot:**

This class consists of ParkingSpotID, Floornumber, ParkingType, isOccupied (Boolean).

4.**Transaction:**

This class consists of Ticketno, Name, VehicleNo, EntryDate, EntryTime, ExitTime

ParkingType, EntryPoint, ExitPoint, FloorNo, SpotNo, isPayed

5.**ParkTypeArray:**In this class we made 5 different parking IDs based on vehicle type and assigned their hourly charges. We also created methods which add and delete the parking types based on the future requirement's and a display board which shows vehicle type & it's ID.

6.**CheckPointArray:**

In this class we created different checkpoints and their exits. We also created methods like AddCheckPoints and RemoveCheckPoints which add & remove checkpoints in future if required. And 2 display methods which display details about entry and exit.

7.**FloorsArray extending ParkTypeArray:**

In this we created 25 spots for 3 floors, each floor consists of 5 spots for a particular vehicle type. We also created AddFloors method which adds new floors in future if required. And some display functions which display information about vehicles in a floor. We made 2 methods which helps in assigning spots to user or user can choose his parking spot.

8.**Payment extending ParkTypeArray:**

In this class we created methods through which can check payment status, if it is paid he can exit else he can choose cash/card payment methods and can pay the charge displayed on the screen. We also created display function for the Receipt which shows all the information regarding customer and his payment details. We also made all the methods protected in this class.

9.**Captcha:**

In this class we created method which generates new captcha every time which helps during the time of payment through cards.

INDIVIDUAL CONTRIBUTIONS

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.