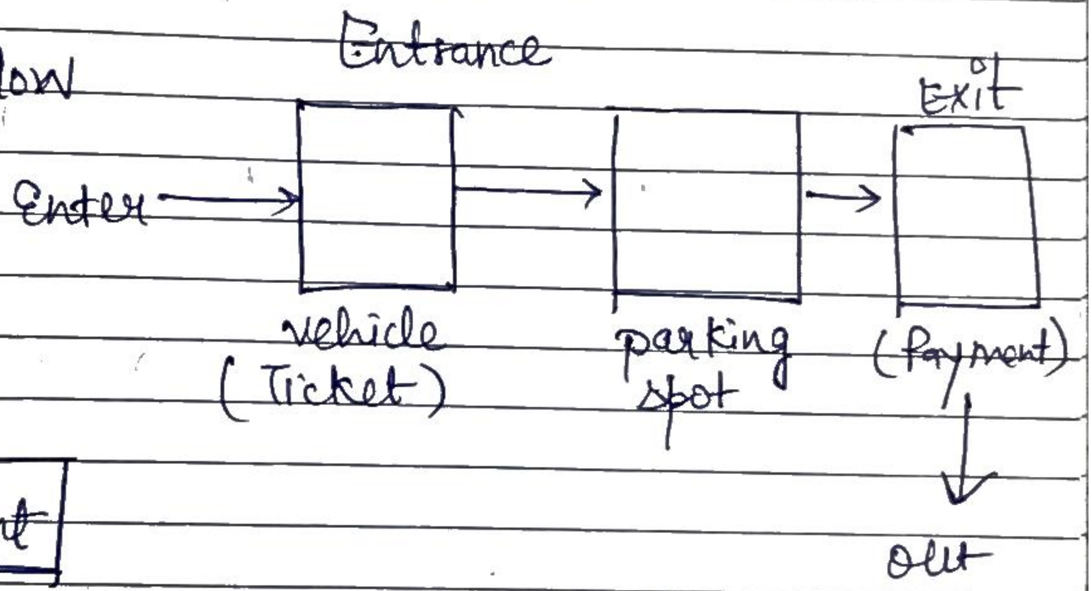
Design Parking Lot

Intial Steps :
↳ Requirement clarification
↳ Identify objects

Rough Flow
(High Level View)



- **Requirement**

- How many entrance and Exit in Parking lot
  Let say initially 1 entrance, 1 exit

But we have to build a system that is scalable

- Different types of parking spot
  - ↳ Two-wheeler] – Initial
  - ↳ Four wheeler]
  - ↳ Three wheeler] – May be added while scaling

- Payment charges
  - ↳ Hourly based charge] – For Scaling
  - ↳ Minute based]
  - ↳ Mixed (consider/initialy)

objects

- Vehicle
  - → Vehicle No
  - → Vehicle Type → (Enum 2,4 wheeler)
- Ticket
  - → Entry Time
  - → Parking slot
- Entrance Gate
  - → Find Parking space
  - → update Parking space
  - → Generate ticket
- Parking spot
  - → Parking spot id
  - → Parking isEmpty
  - → Parking type
  - → Parking Price
- Exit Gate
  - → cost calculation
  - → Recieve Payment
  - → Update Parking Spot

Follow-up Question :- 1. If we have multiple Entrance, the parking should be nearest to the entrance (optimisation)

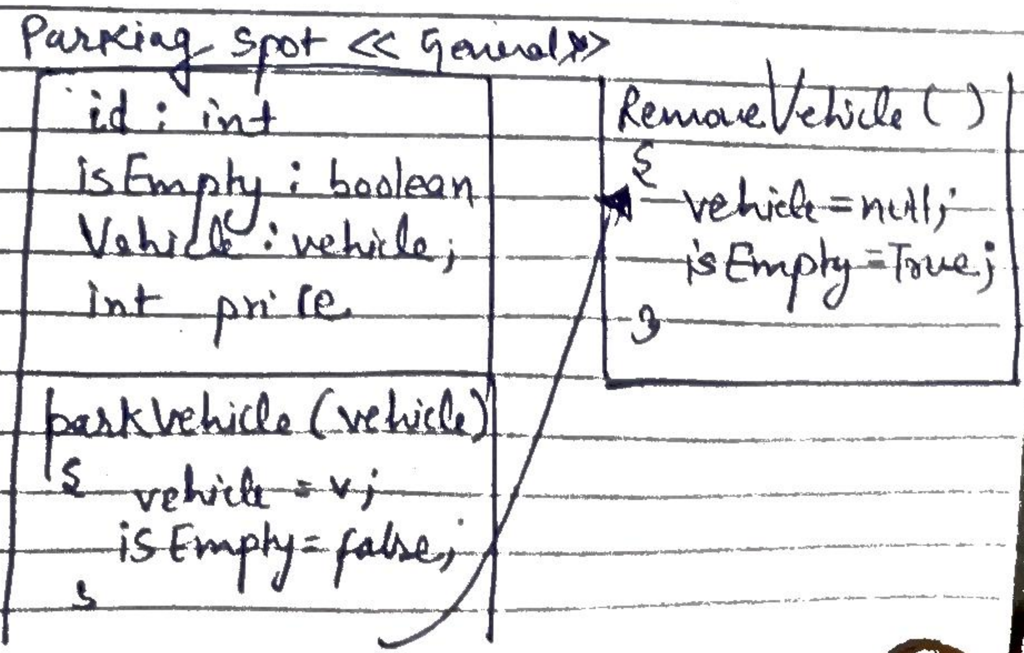
:- Do we have different floors for Parking ?
↳ NO (Initially)
but may be added for scalability

There are Two Approch

Top-down
Starting from Entrance gate

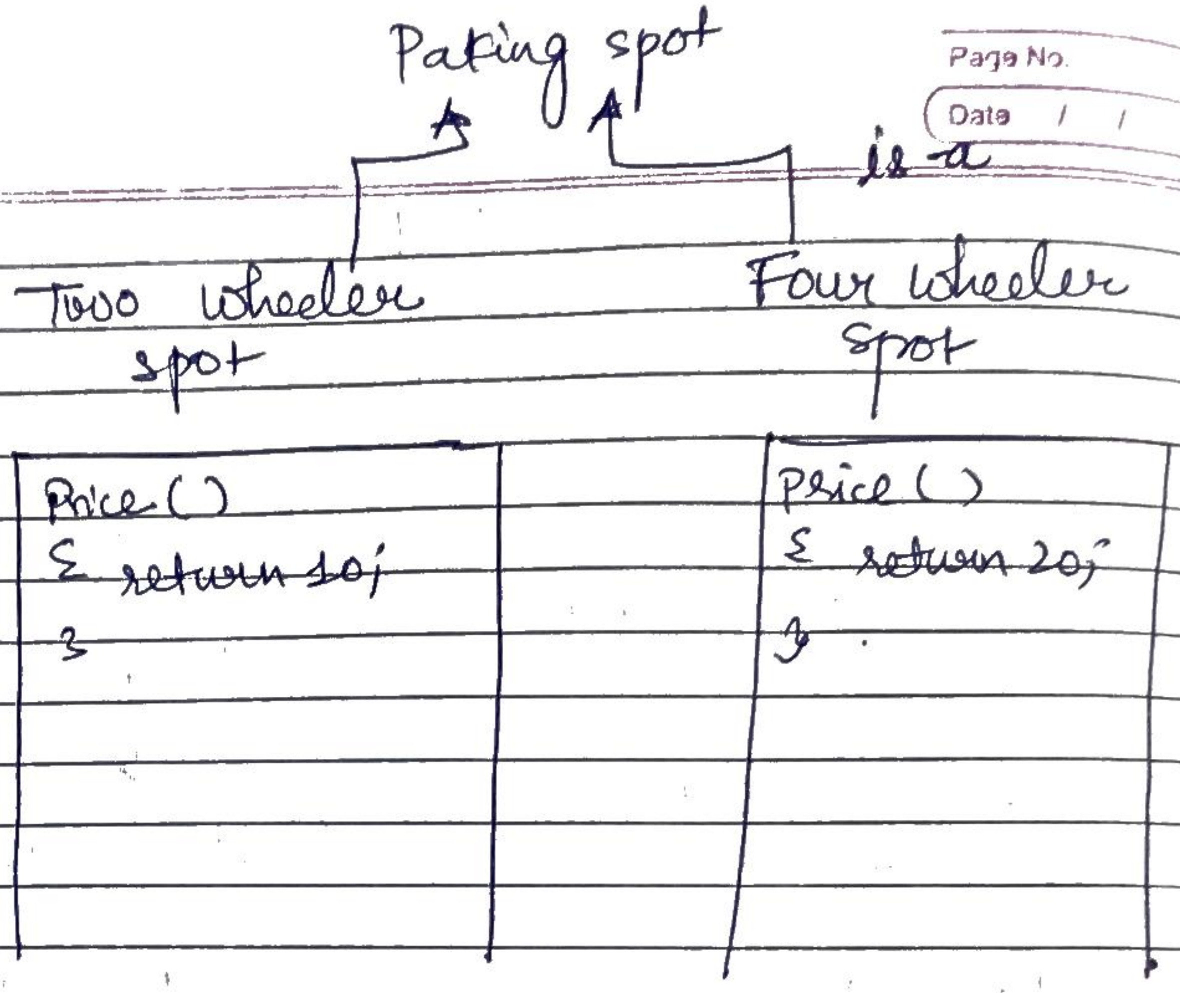
Bottom-up
Starting from making parking spot & then entrance gate & then exit gate and so on.

Solving with Bottom up.



```
Parking spot << General >>
  id : int
  isEmpty : boolean
  Vehicle : vehicle;
  int price.

  parkVehicle (vehicle)
  {
     vehicle = v;
     isEmpty = false;
  }

RemoveVehicle ( )
{
   vehicle = null;
   isEmpty = True;
}
```

Page No.
Date / /



```
Price()
{ return 10;
}
```

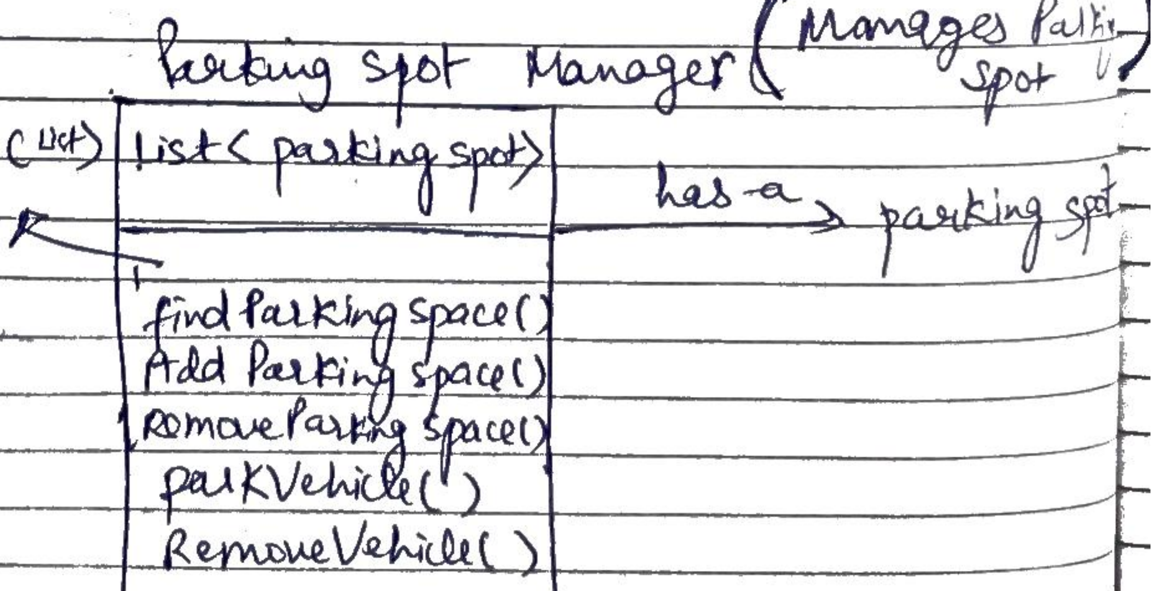
```
Price()
{ return 20;
}.
```

→ Now we can scale this for Heavy vehicle, handicaped people's vehicl.

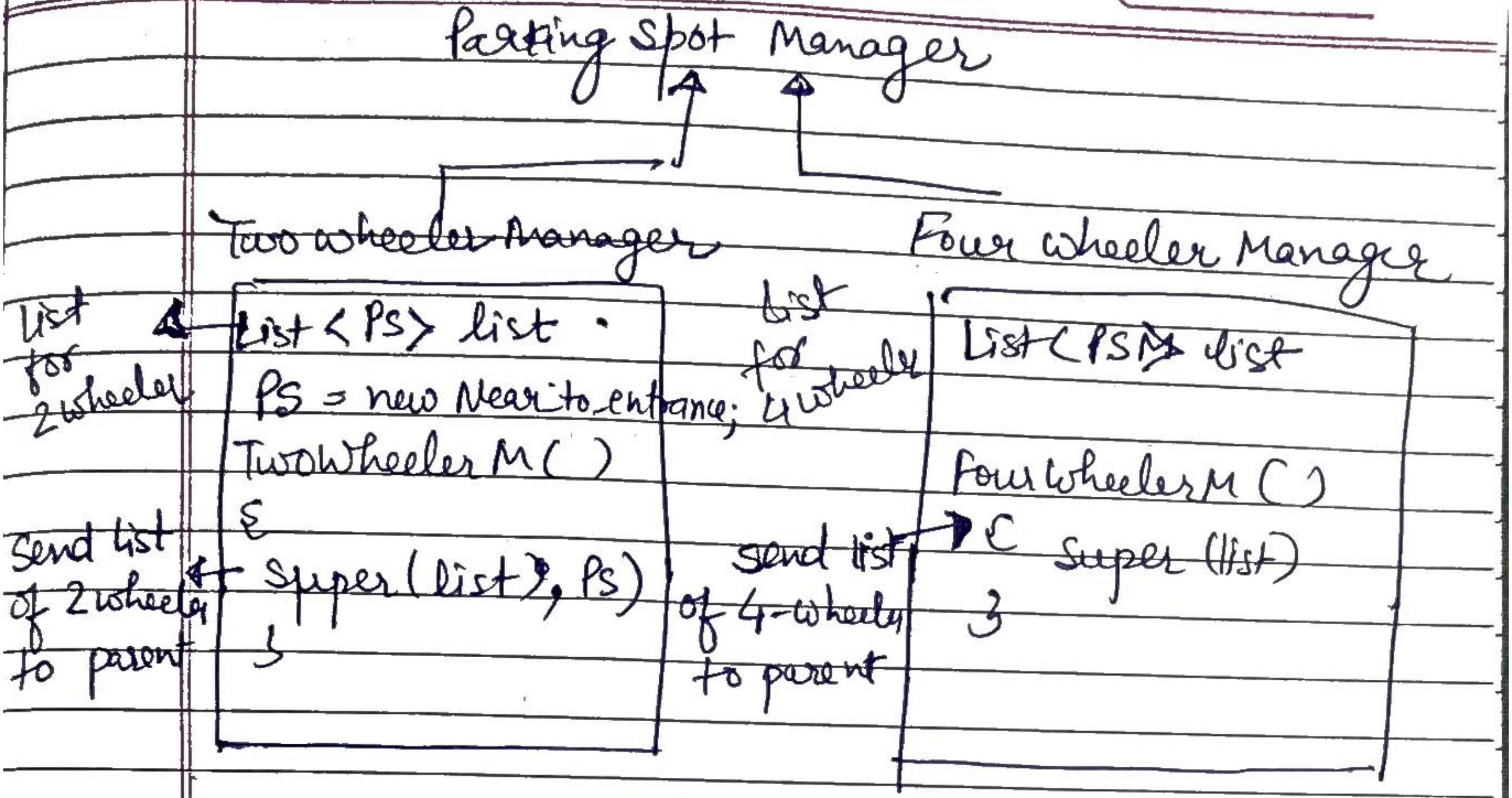
<u>Next</u> :- Now there can be many Parking Spot.

<u>constructor</u>

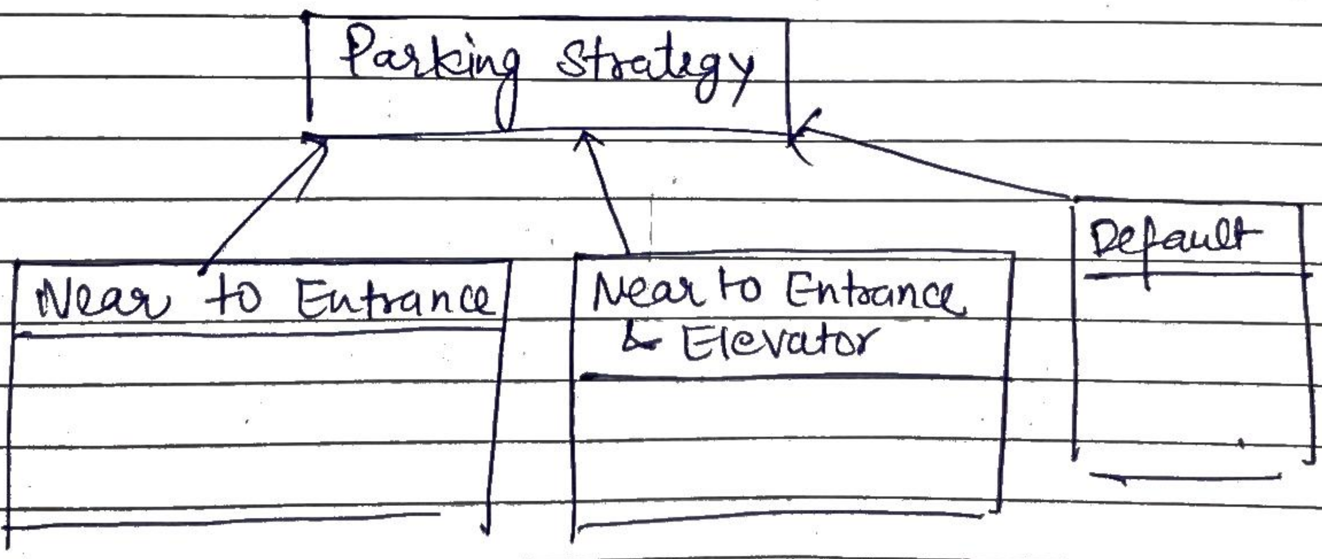
```
~~Parkingspot Manger (List)~~
{
  list = list)
}
```



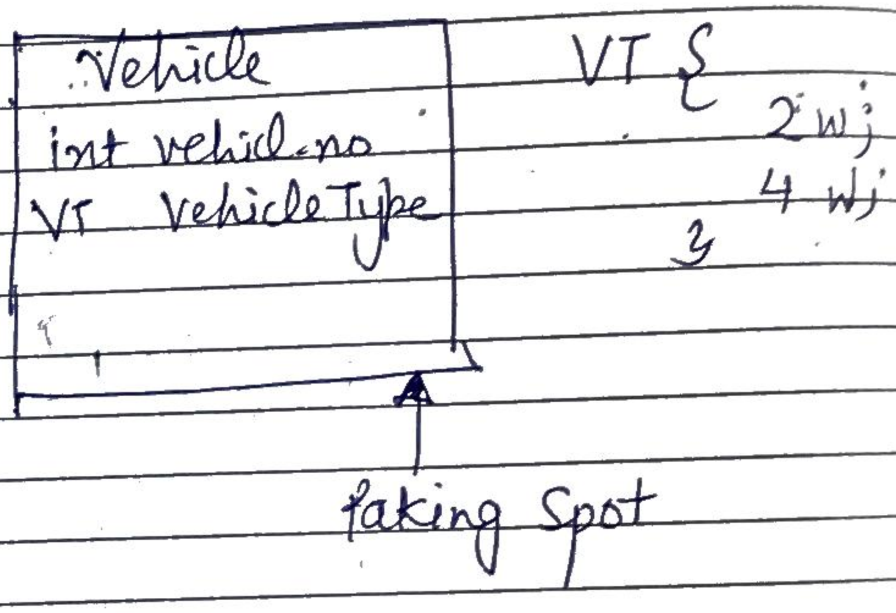
```
List< parking spot>

find Parking space()
Add Parking space()
Remove Parking space()
parkVehicle()
RemoveVehicle()
```



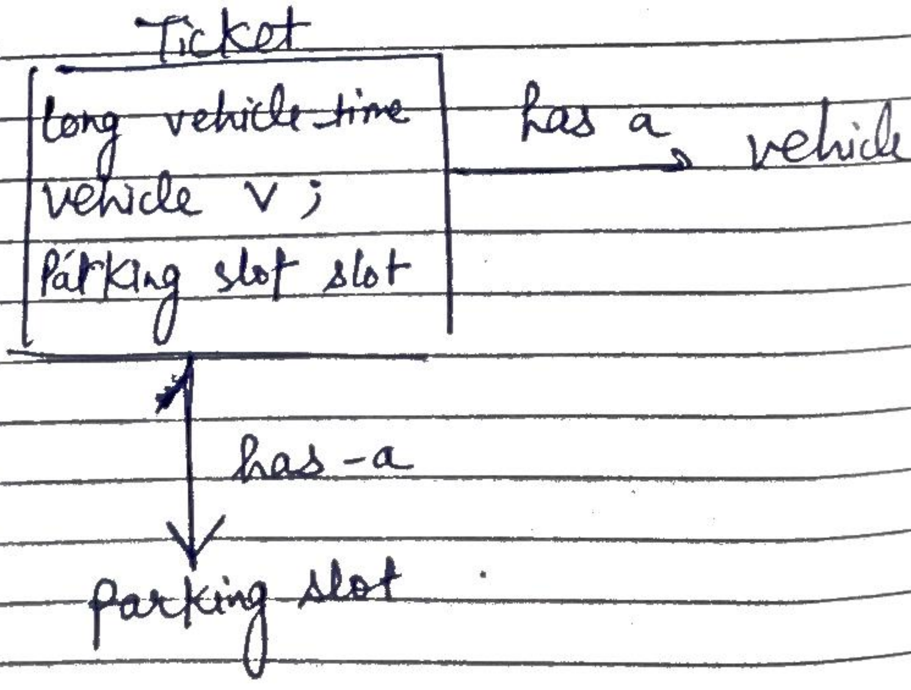
→ Now we can extend the Parking by introducing Parking Strategy



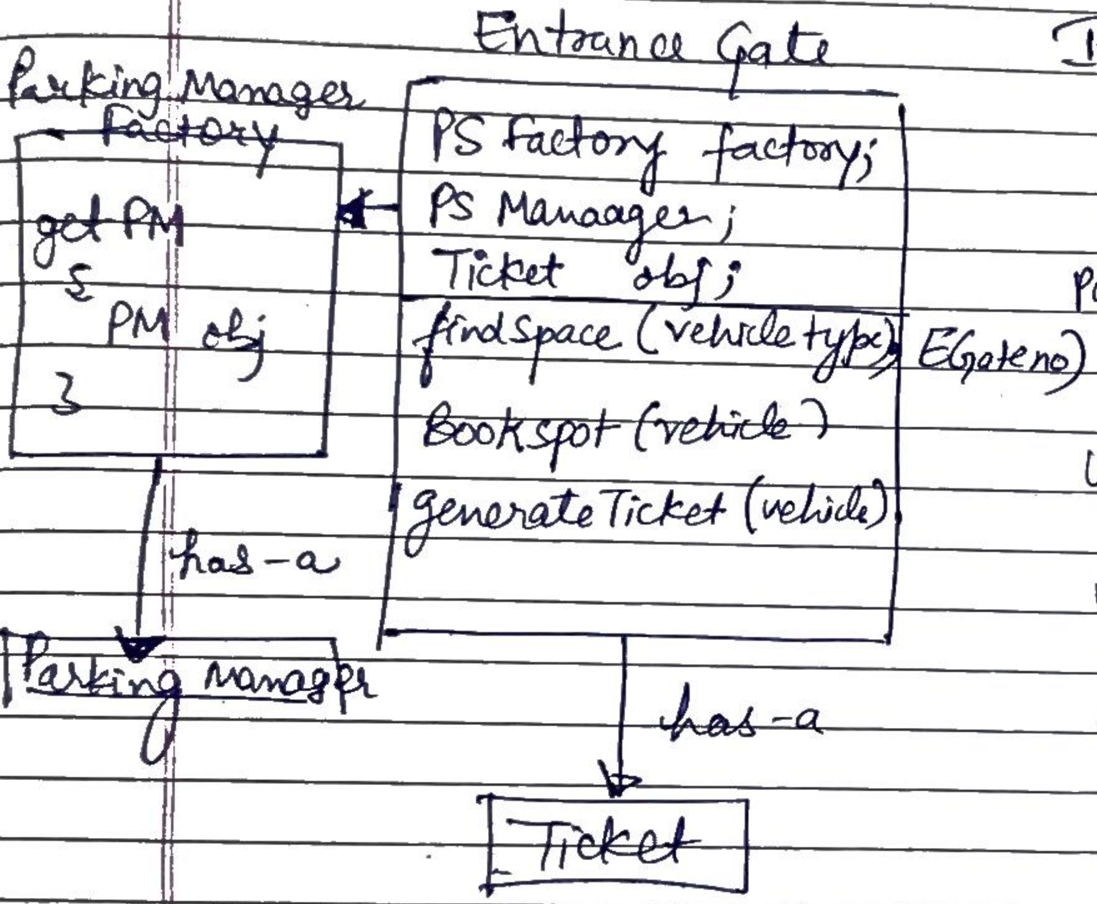
Now i have taken vehicle in parking spot so we need to create its object



Now let's create a ticket object

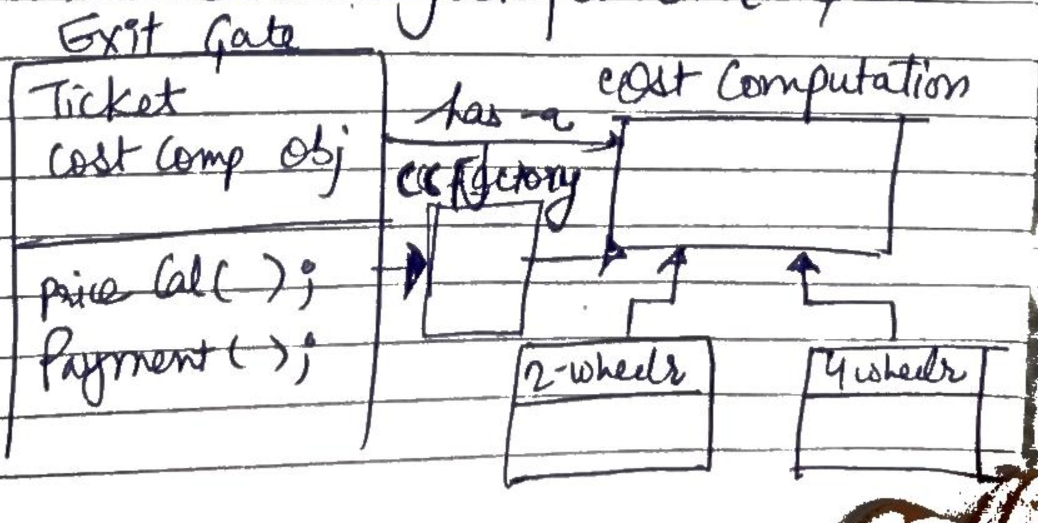


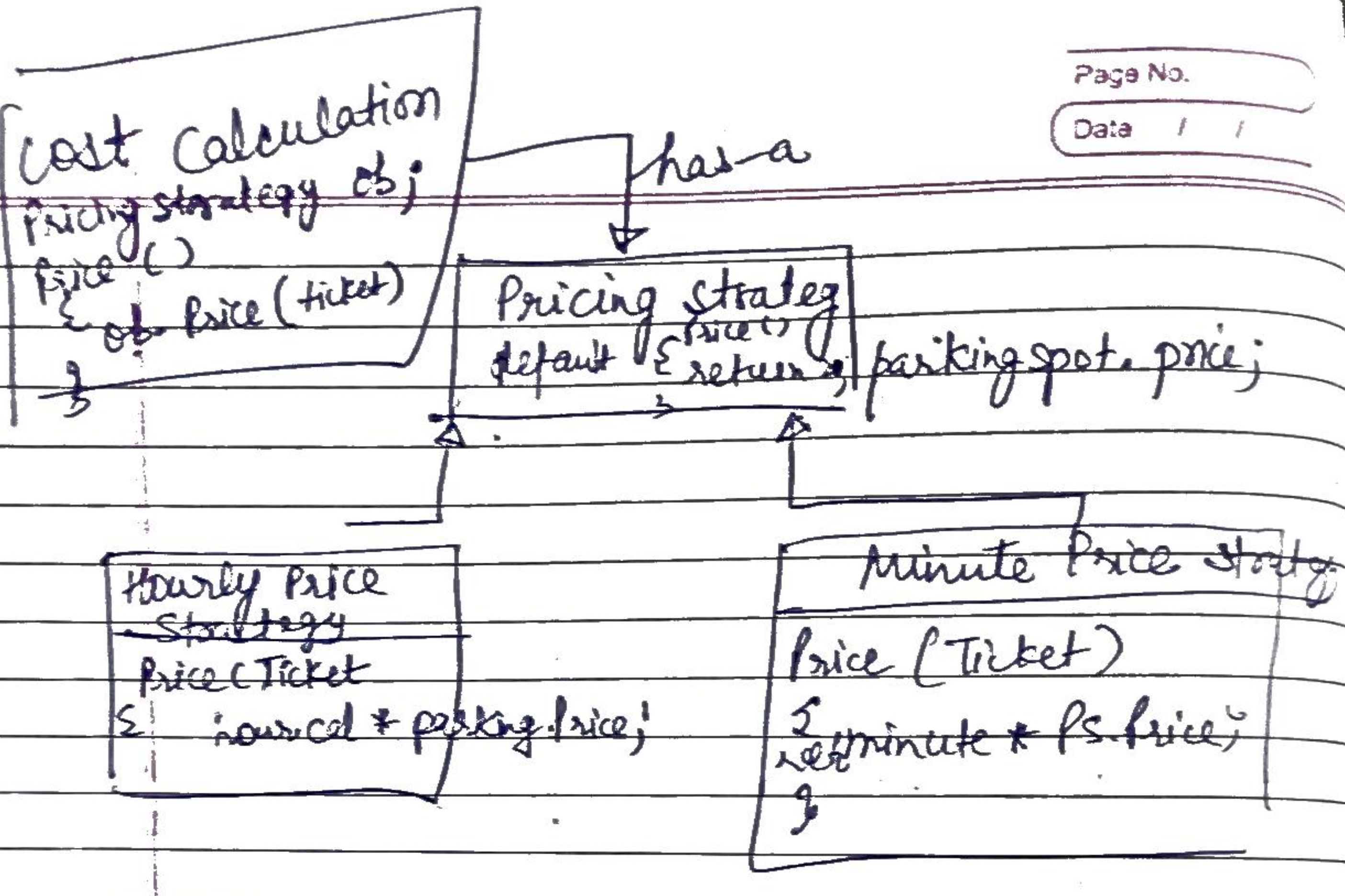
Now we will create an Entrance Gate



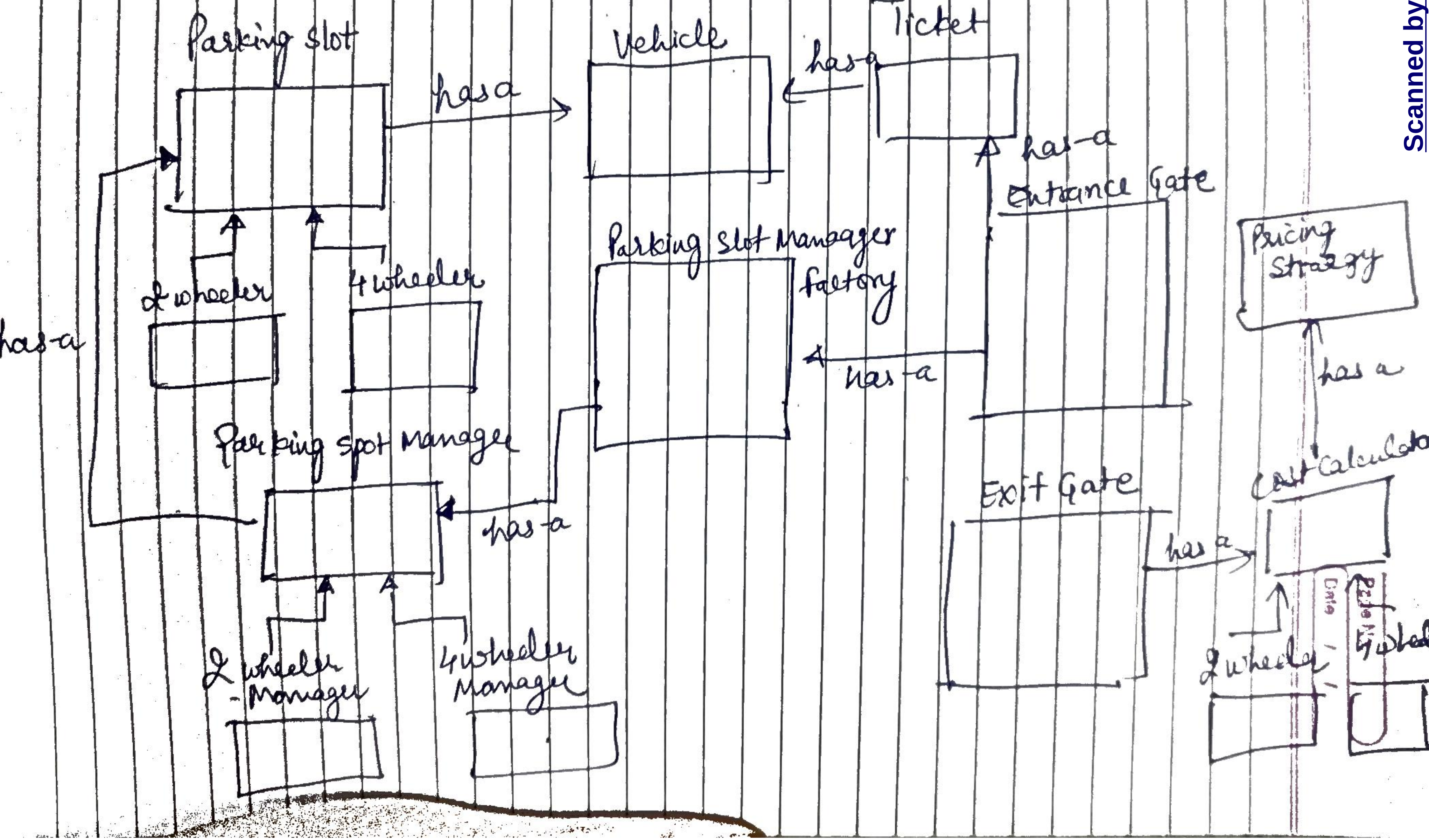
It has to find parking space
↓
For this we have Parking slot manager
↓
For this we can use Factory design Pattern since we have two ~~open~~ obj based on condition of vehicle type
↓
we can get parking slot based info from either 2 wheeler Manger or 4-wheeler Manager based on vehicle type.

→ Now we will create object for Exit Gate




Cost Calculation
Pricing strategy obj
price ()
{ obj.Price (ticket)
}
has-a
Pricing Strateg
default { price() return }; parkingspot.price;
Hourly Price Strategy
Price (Ticket
{ hours() * parking.Price;
Minute Price Strategy
Price (Ticket)
{ return minute * Ps.Price;
}

HIGH LEVEL VIEW


Parking slot
has a
Vehicle
has-a
Ticket
has-a
Entrance Gate
2 wheeler
4 wheeler
has-a
Parking Slot Manager factory
has-a
Pricing Strategy
has a
Parking spot Manager
has-a
Exit Gate
has a
Cost Calculator
2 wheeler Manager
4 wheeler Manager
2 wheeler
4 wheel