

Parking Spot design

* Requirement:

There are 4 entry points and 4 exit points . The car should be given a parking spot which is nearest to the entry point

Actors

* Parking lot system
* Entry/Exit point
* Payment processor at exit
* Printing tickets at entry
* Parking spot
* Ticket
* Database
* Monitoring system

NOTE : The reason why we are not keeping a Vehicle class is because we don’t need Vehicle class and it behaviour . We just need Parking spot to accommodate different types of vehicles

i.e Handicap spot, Compact/sedan vehicle , Large / SUV Vehicle , Bikes

Bottom up approach

* **Parking spot**

CASE 1 : Lets say we use enums to start with like

ParkingSpotEnum {

Handicap,sedan,suv,bike;

}

This is violating OPEN/CLOSE design principle. Because if at all we want to add a new Vehicle type like Truck we will need to add a new enum and then change a well tested code and handle this enum .  
  
Open close principle says when a new feature has to be added existing and well testing class should not be modified

CASE 2: Correct way would be to create a parent abstract class as Parking spot with **attributes like id and reserve**(REASON for abstract is no one can create instance of the abstract class) and then **other types of parking spot extending it**

* **Parking Ticket**

This should be a class with attributes like

Id

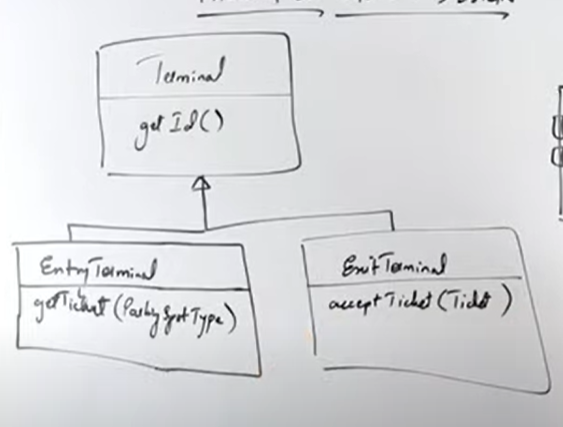
parkingSpotId

parkingSpotType

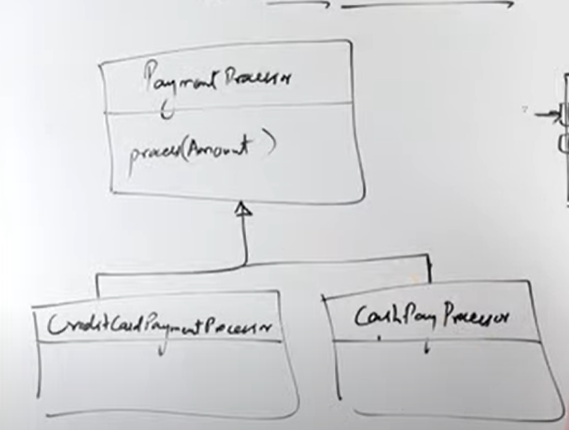
issueTime

* **Terminal**

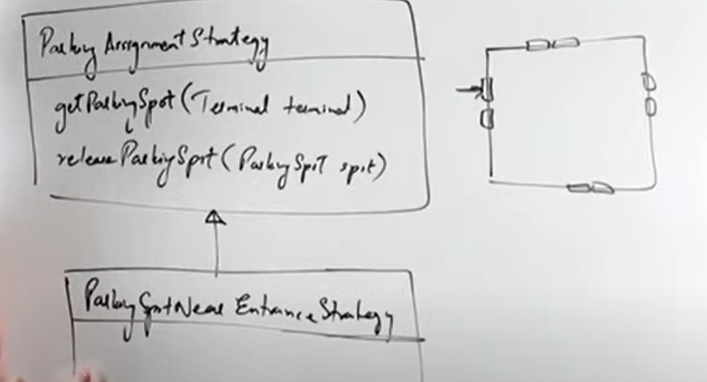
All are interfaces/abstract classes



* **Payment Processor**



* **Parking Arragement Strategy**



To implement the parking strategy we will need to use **MIN HEAP**

**Each entry point will have their own min heap which stores all the slots clossest to the entry**

**We would also require 2 sets**

**1 🡪 Available slot**

**2 🡪 Reserved slot**

**If a car is given a slot it should be removed from the available and shifted to reserved**

* **Tariff Calculator class**

**Takes in issue time and parking slot type and returns the amount to be paid**

* **Logger class for monitoring**