# Parking Lot Design by Think Software

* Clarify all requirements.
* Identify Design Pattern
* Tacking Concurrency
* Problem Solving Skills
* Not solving as Distributed System Design
* System should be reusable so that It can be used by just changing the configuration.

## Requirements

* Big Parking Lots
* Multiple Entrances & Exits
* Ticket and Spot attached at entrance.
* Parking spot should be nearest to entrance.
* Limit/Capacity -> vehicles should not be allowed more than spots.
* Parking System – Bug Spot, Compact, Motorcycle.
* Hourly Rate
* Payment by Cash and Card.
* Monitoring System (Cars entering and Exiting)

Do not discuss about panel showing free spot at each floor. This requirement conflicts the parking spot defined at entrance on the ticket.

Color of the vehicle is not required.

Design Patterns: -

* Creational
* Structural
* Behavioral

Design Approach

* Top Down Design – Component -> Sub Component
* Bottom Up Design – Sub Component -> Component: Used in OOD.

Actors/Objects

* Parking Lot System
* Entry/Exit Terminals
  + Printers
  + Payment System
* Parking Spot
* Ticket
* Database
* Monitoring System

Vehicle class is not required. Parking Lot systems will be using vehicle parking spot but it will not be using vehicle class.

Parking Spot

Enum Parking Spot

{

Handicapped,

Compact,

Larger,

Motorcylce,

}

This has flaw. New Parking Spot is added then it will require code changes at multiple places and violated OCP Open for extension, closed for modification.

|  |
| --- |
| Parking Spot (Abstract)  +id  + reserve  Large Parking Spot  Compact  Handicap parking Spot |

|  |
| --- |
| Terminal  +getId()  Exit Terminal  AcceptTicket(Ticket)  EntryTerminal  getTicket(ParkingSpot Type) |

Class ParkingAssignmentStrategy (Abstract)

+ GetPrkingSpot(Terminal terminal)

+ Release Parking Spot (ParkingSpot spot)

Class ParkingSpotNearEntranceStrategy: ParkingAssignmentStrategy

\*Strategy Pattern.

\*4 MinHeaps for all parking spot for each terminal.

\*1 set to maintain for available and 1 set for not available.

\* remember to handle concurrency when sets are updated.

\* K LogN which K is number of min heaps and N is number of parking lots

\* 4 Elevators in Parking Lot --- How to handle it?

PaymentProcess (Abstract)

+ process (Payment)

CreditCardPaymentProcess

CashPaymentProcess

\*strategy Design Pattern

\* In future, Wallets can also be part

Tariff Calculator

+ CalculateTariff (ParkingSpot Time)

Extended Tariff calculation for weekdays, weekends

Logger

+LogMessage()

\*Observer Pattern

Design of Parking Lot System

* Parking Lot Class
  + Singleton
  + Consists of multiple objects (Terminals, Parking Spot)
  + Factory Design Pattern to instantiate all those object.
  + Can take configuration object (Printer Settings, Parking Spot, Payment Process etc)
    - Abstract Factory Pattern