## Description

Design a Parking Lot Management System which has the capability to issue parking tickets, allocate and manage parking slots.

## Features

1. Parking has 1 or more Entrances and Exits.
2. Each Parking has 1 or more Parking Slots
3. Parking slots are of 3 different types
   1. Cars
   2. Bikes
   3. Bicycles
4. Few car parking slots are reserved for
   1. Disabled People
   2. Pregnant Women
5. There can be 0 or more Parking Slots of each type
6. When a vehicle comes at the entrance, the following scenarios need to be handled.
   1. No slot is available and the vehicle is not allowed inside the parking lot
   2. A slot is available and the customer decides to use it. The customer enters the parking lot

1. Available Slot quantity is updated appropriately

1. Available Slot Selection can be governed by one of the following policies.
   1. Sequential
   2. Random Available
   3. More policies can be added later
2. When a vehicle comes at the Exit, following scenarios need to be handled
   1. Available Slot quantity is updated

## Requirements

1. Onboard a new Parking Lot with a list of Parking Slots with type and capacity.
2. Issue Parking Tickets to vehicles entering the Parking Lot
3. The following system view should be available
   1. Available slots of each type
   2. Vehicles-In per Vehicle Type per Parking Entrance
   3. Vehicles-Out per Vehicle Type per Parking Exit

## Other Notes

1. Do not use any database or NoSQL store, use in-memory data-structure for now.
2. Do not create any UI for the application.
3. Write a driver class for the demo purpose. This will execute all the commands at one place in the code and test cases.
4. Please prioritize code compilation, execution and completion.
5. Work on the expected output first and then add good-to-have features of your own.

## Expectations

1. Make sure that you have a working and demonstrable code
2. Make sure that code is functionally correct

## 3. Make sure concurrent requests are handled appropriately

1. Code should be modular and readable
2. Separation of concern should be addressed
3. Code should easily accommodate new requirements with minimal changes
4. Code should be easily testable

**Test cases**​:

The strings in the test cases are indicative. Feel free to take exact parameters into your methods.

1. add\_parking\_lot(“PVR Koramangala”, “[1 Car Parkings, 1 Bike Parking, 1 Bicycle Parking, 1 Disabled Parking, 1 Pregnant Women Parking]”, “2 Entry Gates”, “2 Exit Gates”)
2. is\_available(“PVR Koramangala”, “Bike”) -> True (queried from Entry Gate 1)
3. is\_available(“PVR Koramangala”, “Car”) -> True
4. is\_available(“PVR Koramangala”, “Bike”) -> True (queried from Entry Gate 2)
5. park\_vehicle(“PVR Koramangala”, “Car”, “Entry Gate 1”) -> True
6. park\_vehicle(“PVR Koramangala”, “Bike”, “Entry Gate 2”) -> True
7. park\_vehicle(“PVR Koramangala”, “Bike”, “Entry Gate 1”) -> False (This will fail because the spot was taken in the previous step)
8. is\_available(“PVR Koramangala”, “Car”) -> False
9. is\_available(“PVR Koramangala”, “Car”, “Disabled”) -> True (queried from Entry Gate 2)
10. park\_vehicle(“PVR Koramangala”, “Car”, “Disabled”) -> True
11. print\_all\_available\_slots(“PVR Koramangala”)

Total Available: 2

Car 1

Bike 0

Bicycle 1

1. print\_total\_in(“PVR Koramangala”, “Entry Gate 1”)

Car 1

# Bike 0

# Bicycle 0

# 13. print\_total\_out(“PVR Koramangala”, “Exit Gate 1”)

Car 0

Bike 0

Bicycle 0

1. unpark\_vehicle(“PVR Koramangala”, “Bike”, “Exit Gate 1”)
2. print\_total\_out(“PVR Koramangala”, “Exit Gate 1”)

Car 0

Bike 1

Bicycle 0