Parking Lot System - Low-Level Design (LLD)

Core Requirements

- Different **vehicle types** (Car, Bike, Truck) with different spot sizes.
- Different **parking spot types** (Compact, Large, Handicapped, Electric).
- **Multiple floors**, each with a set of parking spots.
- **Parking strategies** (Nearest available, Handicapped priority).
- **Payment system** (Hourly-based, flat fee for some vehicles).
- **Entry/Exit gates** for vehicles.

Class Design

Key Classes & Their Responsibilities

1. `Vehicle` **(Abstract Class)**
    - Properties: `licensePlate`, `vehicleType` (Car, Bike, Truck).
    - Subclasses: `Car`, `Bike`, `Truck`.
2. `ParkingSpot`
    - Properties: `spotId`, `spotType` (Compact, Large, Handicapped, Electric), `isOccupied`, `vehicle`.
    - Methods: `assignVehicle()`, `removeVehicle()`.
3. `Floor`
    - Properties: `floorNumber`, `list<ParkingSpot>`.
    - Methods: `addSpot()`, `findAvailableSpot()`.
4. `ParkingLot` **(Singleton)**
    - Properties: `list<Floor>`, `entranceGates`, `exitGates`.
    - Methods: `parkVehicle()`, `unparkVehicle()`, `calculateFee()`.
5. `Ticket`
    - Properties: `ticketId`, `entryTime`, `vehicle`, `assignedSpot`.
6. `Payment`
    - Methods: `processPayment()`, `calculateFee()` (hourly-based or flat rate).
7. `ParkingStrategy` **(Interface)**
    - Methods: `findSpot()`.
    - Implementations: `NearestFirstStrategy`, `HandicappedFirstStrategy`.

Flow of Operations

A. Vehicle Entry (Parking)

1. **Vehicle arrives** at the entrance gate.
2. **System checks** for available spots based on vehicle type.
3. **Parking strategy** (Nearest/Handicapped) selects the best spot.
4. **Ticket is generated** with entry time and assigned spot.
5. **Spot is marked as occupied**.

B. Vehicle Exit (Unparking & Payment)

1. **Driver presents ticket** at the exit gate.
2. **System calculates fee** based on duration (hourly rate).
3. **Payment is processed** (cash/card/digital).
4. **Spot is freed** for new vehicles.
5. **Ticket is marked as paid**.

Key Design Considerations

a) Parking Strategies

- **Nearest Available**: Assigns the closest available spot to the entrance.
- **Handicapped Priority**: Prioritizes spots for disabled drivers.
- **Electric Vehicle Priority**: Assigns spots with charging stations.

b) Payment Handling

- **Hourly rate** for cars/trucks.
- **Flat fee** for bikes in some cases.
- **Different pricing** for different floors (premium spots).

c) Scalability & Extensibility

- **Singleton** `ParkingLot` ensures a single global point of control.
- **Strategy Pattern** allows easy addition of new parking algorithms.
- **Factory Pattern** can be used for creating different vehicle types.

FInal Features :

- **Multi-floor parking** with automatic slot generation (1A-1Z, 2A-2Z, etc.)
- **Real-time slot availability** tracking
- **Different vehicle types** support (Cars, Bikes, Trucks)
- **Multiple spot types** (Compact, Large, Handicapped, Electric)
- **Ticket-based system** with automatic fee calculation ($10/hour)
- **Singleton design pattern** for single parking lot instance
- **Smart pointers** for automatic memory management

System Architecture

```
ParkingLot (Singleton)
├── Floors (1-N)
│   ├── ParkingSpots (A-Z per floor)
│   │   ├── Spot ID (e.g., 1A, 2B)
│   │   ├── Spot Type
│   │   └── Occupancy Status
├── Tickets
│   ├── Entry Time
│   ├── Vehicle Info
│   └── Assigned Spot
└── Payment System
    └── Hourly Rate Calculation
```