Parking Lot System Design

System Overview

The parking lot system manages vehicle parking, entry/exit operations, payment processing, and space allocation across different vehicle types and parking zones.

Core Entity Classes

1. ParkingLot

```java
public class ParkingLot {
    private String id;
    private String name;
    private Address address;
    private List<ParkingFloor> floors;
    private List<EntryPoint> entryPoints;
    private List<ExitPoint> exitPoints;
    private ParkingRate parkingRate;
    private int totalCapacity;
    private int availableSpots;

    public boolean isFull();
    public ParkingSpot findAvailableSpot(VehicleType vehicleType);
    public boolean assignVehicleToSpot(Vehicle vehicle, ParkingSpot spot);
    public boolean releaseSpot(ParkingSpot spot);
    public double calculateParkingFee(ParkingTicket ticket);
}
```

2. ParkingFloor

```java
public class ParkingFloor {
    private String floorId;
    private int floorNumber;
    private List<ParkingSpot> parkingSpots;
    private Map<VehicleType, Integer> spotsCount;
    private ParkingDisplayBoard displayBoard;

    public boolean isFull();
    public ParkingSpot findAvailableSpot(VehicleType vehicleType);
    public void updateDisplayBoard();
}
```

3. ParkingSpot

java

```
public abstract class ParkingSpot {
    private String spotId;
    private String spotNumber;
    private VehicleType vehicleType;
    private ParkingSpotStatus status;
    private Vehicle parkedVehicle;

    public boolean isAvailable();
    public boolean assignVehicle(Vehicle vehicle);
    public boolean removeVehicle();
}

public class CompactSpot extends ParkingSpot {}
public class LargeSpot extends ParkingSpot {}
public class MotorcycleSpot extends ParkingSpot {}
public class ElectricSpot extends ParkingSpot {}
public class HandicappedSpot extends ParkingSpot {}
```

4. Vehicle

java

```
public abstract class Vehicle {
    private String licensePlate;
    private VehicleType type;
    private String color;
    private String model;

    public abstract boolean canFitInSpot(ParkingSpot spot);
}

public class Car extends Vehicle {}
public class Truck extends Vehicle {}
public class Motorcycle extends Vehicle {}
public class Van extends Vehicle {}
```

5. ParkingTicket

```java
public class ParkingTicket {
    private String ticketId;
    private String licensePlate;
    private Date entryTime;
    private Date exitTime;
    private ParkingSpot assignedSpot;
    private double totalCost;
    private TicketStatus status;
    private PaymentInfo paymentInfo;

    public double calculateTotalCost();
    public void updateExitTime();
}
```

6. Payment System

```java
public abstract class Payment {
    private String paymentId;
    private double amount;
    private Date paymentDate;
    private PaymentStatus status;

    public abstract boolean processPayment();
}

public class CashPayment extends Payment {}
public class CreditCardPayment extends Payment {}
public class DigitalWalletPayment extends Payment {}
```

7. Entry/Exit Points

```java
public class EntryPoint {
    private String entryId;
    private ParkingTicket generateTicket(Vehicle vehicle);
    private ParkingSpot assignSpot(Vehicle vehicle);
}

public class ExitPoint {
    private String exitId;
    private double calculateFee(ParkingTicket ticket);
    private boolean processPayment(Payment payment);
    private void releaseSpot(ParkingSpot spot);
}
```

8. Supporting Classes

```java
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;
    private String country;
}

public class ParkingRate {
    private Map<VehicleType, Double> hourlyRates;
    private double maxDailyRate;
    private double penaltyRate;

    public double calculateCost(VehicleType type, long hours);
}

public class ParkingDisplayBoard {
    private Map<VehicleType, Integer> availableSpots;

    public void updateAvailability(VehicleType type, int count);
    public void displayMessage(String message);
}
```

Enums

```java
public enum VehicleType {
    MOTORCYCLE, CAR, VAN, TRUCK, ELECTRIC
}

public enum ParkingSpotStatus {
    AVAILABLE, OCCUPIED, RESERVED, OUT_OF_ORDER
}

public enum TicketStatus {
    ACTIVE, PAID, LOST, EXPIRED
}

public enum PaymentStatus {
    PENDING, COMPLETED, FAILED, CANCELLED, REFUNDED
}
```

Key System Features

Core Functionality

- **Multi-floor support** with different vehicle types
- **Dynamic spot allocation** based on vehicle size
- **Real-time availability tracking**
- **Flexible payment options** (cash, card, digital wallet)
- **Time-based pricing** with penalty handling

Advanced Features

- **Reserved parking** for VIP/handicapped users
- **Electric vehicle charging** spots
- **Lost ticket handling** with penalty fees
- **Display boards** showing real-time availability
- **Maximum daily rate** capping

System Constraints

- Each spot can accommodate only compatible vehicle types
- Larger vehicles can use smaller spots if needed (Car → Compact)
- Electric spots reserved for electric vehicles
- Handicapped spots have special access requirements

Usage Workflow

1. **Entry Process**
    - Vehicle arrives at entry point
    - System finds available spot based on vehicle type
    - Generate parking ticket with spot assignment
    - Vehicle parks in assigned spot
2. **Exit Process**
    - Present ticket at exit point
    - Calculate parking fee based on duration
    - Process payment
    - Release parking spot
    - Update availability counters
3. **Payment Processing**
    - Support multiple payment methods
    - Handle payment failures gracefully
    - Generate receipts for completed transactions

This design provides a scalable, maintainable parking lot management system that can handle various vehicle types, payment methods, and operational scenarios.