**SMART PARKING MANAGEMENT SYSTEM**

**PROJECT DOCUMENT**

**Group Members (Group 27)**

Atul M - 2022AAPS1362G

Ansh Garg - 2022A8PS1360G

Nikhil Mahesh Tamboli -2021B2AA1603G

Sohan Palekar - 2022AAPS0554G

**Project Structure**

* Header
  + main
    - Driver.java [Driver class() and MenuFunction class()]
  + management
    - ParkingManagement.java [ParkingManagement class()]
    - ParkingSpot.java [ParkingSpot class()]
    - User.java [User class()]
  + vehicles
    - Vehicle.java [Vehicle class()]
    - TwoWheeler.java [TwoWheeler class()]
    - FourWheeler.java [FourWheeler class()]
    - Bike.java [Bike class()]
    - Car.java [Car class()]
  + exceptions
    - InvalidInputException.java [InvalidInputException class()]
    - UserAlreadyExistsException.java[UserAlreadyExistsException class()]
  + Utils
    - FileLogger.java[FileLogger class()]
    - ReservationSystem.java[Reservation class()]
    - PaymentSystem.java[PaymentSystem class()]

### 

### 

### **ParkingSpot**

Manages an individual parking spot, including reservations, vehicle assignment, and availability status.

**Features:**

* Stores attributes: spotId, isAvailable, type (Two Wheeler/Four Wheeler), reservedBy, parkedVehicle, and entryTime.
* Allows reserving a spot for a User and marking it available or unavailable.
* Manages the vehicle currently parked and tracks the vehicle’s entry time.
* Provides getters and setters for full access and control over the parking spot's status.

### **Payment System**

The Payment System interface handles cost calculations for parking usage, reservations, and cancellations in the parking management system.

**Features:**

* Fixed charges:  
  + Reservation: ₹10
  + Cancellation: ₹5
* Parking cost based on:  
  + Vehicle type (Two Wheeler or Four Wheeler)
  + Time parked (current time - entry time)
* Different rates for vehicles:  
  + Bike: ₹15 per time unit
  + Car: ₹25 per time unit

**Reservation System**

It attempts to reserve a parking spot based on vehicle type. It first checks if the vehicle is already parked and prevents duplicate bookings. For a bike, it searches for an available "Two Wheeler" spot, while for a car, it looks for a "Four Wheeler" spot. If a suitable spot is found, it is reserved, and the method returns the booked parking spot; otherwise, it returns null. It also supports cancelling a user's reservation by searching for the corresponding parking spot. If a match is found, the spot is marked as available again and returned; otherwise, it returns null.

## **Features:**

* Book a parking spot for a user’s vehicle.
* Cancel an existing reservation made by a user.
* Supports different vehicle types:

### **FileLogger**

It provides logging functionality for major actions performed in the parking system, such as reserving, parking, cancelling, and checking out.

**Features:**

* Defines constants:  
  + LOG\_FILE: The log file's name where actions are recorded (log.txt).
* Provides a centralized private logging method:  
  + log(String action, String details):  
     Writes a formatted log entry with a timestamp, action type, and details to the log file.
* Offers public static methods to log specific actions:  
  + logReserve(String userId, String userName, String spotId): Logs a reservation action.
  + logCancel(String userId, String userName, String spotId): Logs a reservation cancellation action.
  + logPark(String userId, String userName, String spotId): Logs when a vehicle is parked.
  + logCheckout(String userId, String userName, String spotId): Logs when a vehicle checks out.
* Handles file writing with automatic resource management:  
  + Uses FileWriter to write to a file.
  + Catches and prints any IOException during file operations.

### 

### **User**

Represents a user account in the Smart Parking Management System, handling personal information, associated vehicles, and wallet balance.

**Features:**

* Stores essential user information:  
  + userId: A unique ID is generated for each user.
  + name, email, password: Basic user credentials.
  + vehicles: Array to store up to 5 vehicles.
  + balance: User's wallet balance.
* Constructor:  
  + Initializes user data and automatically generates a unique user ID.
* Vehicle Management:  
  + Adds a vehicle to the user's account (up to 5 vehicles).
  + Automatically sets the vehicle’s owner name when a vehicle is added.
* Balance Management:  
  + Retrieves the user's current balance.
  + Provides overloaded setBalance() methods to update balance using Double, Integer, or int types.

### **Vehicle**

It is an abstract base class representing general vehicles in the parking system. It stores details like license plate, vehicle type, and owner information. It also tracks rwhether the car is currently parked. The class provides essential getters and setters for its attributes and supports flexibility for future specific vehicle types (Car, Bike, etc.).

**Features:**

* Stores essential attributes:  
  + licensePlate: Unique license plate number of the vehicle.
  + type: Type of the vehicle (Car, Bike, etc.).
  + owner: Name of the vehicle's owner (optional).
  + isParked: Parking status of the vehicle.
* Allows managing vehicle information:  
  + Retrieve or update owner details.
  + Check and update parking status (parked/unparked).

### **FourWheeler**

It represents a general four-wheeled vehicle in the parking system and serves as a parent class for specific types like Car. It extends the Vehicle class and adds an extra attribute to specify the type of four-wheeler (e.g., Car).

**Features:**

* Inherits essential attributes and behavior from Vehicle.
* Stores additional attribute:  
  + fourWheelerType: Specific type of the four-wheeler (e.g., Car).
* Provides constructors for:  
  + Four-wheeler with license plate and owner.
  + Four-wheeler with a license plate only.
* Allows retrieving the four-wheeler type.

### 

### **TwoWheeler**

It represents a general two-wheeled vehicle in the parking system and serves as a parent class for specific types like Bike. It extends the Vehicle class and introduces an attribute to specify the type of two-wheeler (e.g., Bike).

**Features:**

* Inherits essential attributes and behavior from Vehicle.
* Stores additional attribute:  
  + TwoWheelerType: Specific type of the two-wheeler (e.g., Bike).
* Provides constructors for:  
  + Two-wheeler with license plate and owner.
  + Two-wheeler with a license plate only.
* Allows retrieving the two-wheeler type.

### **Bike**

It represents a two-wheeler vehicle type specifically for bikes in the parking system. It inherits from the TwoWheeler class and automatically sets the vehicle type as "Bike." It supports both owner-assigned and ownerless bike creation.

**Features:**

* Inherits essential attributes and behavior from TwoWheeler.
* Provides constructors for:  
  + Bike with license plate and owner.
  + Bike with license plate only (owner is optional).

### **Car**

It represents a four-wheeler vehicle type specifically for cars in the parking system. It inherits from the FourWheeler class and automatically sets the vehicle type as "Car." It offers flexibility by allowing car creation with or without specifying the owner.

**Features:**

* Inherits essential attributes and behavior from FourWheeler.
* Provides constructors for:  
  + Car with license plate and owner.
  + Car with license plate only.

### **ParkingManagement**

It represents the core system managing users, vehicles, parking spots, payments, and reservations.

and file logging for the parking system.

**Features:**

* Manages parking operations:  
  + Reserves parking spots for users based on vehicle type and balance.
  + Cancels existing parking spot reservations.
  + Park vehicles in reserved or available spots.
  + Checks out parked vehicles and calculates payments.
* Manages users:  
  + Adds new users (with email uniqueness validation).
  + Removes users from the system.
* Manages parking spots:  
  + Adds new parking spots.
  + Removes existing parking spots.
  + Finds suitable parking spots matching the vehicle type.
* Handles payment processing:  
  + Deducts appropriate charges for reservation, cancellation, and parking.
  + Verifies user balance before processing payments.
  + Calculates total payment based on parking duration.
* Integrates logging functionality:  
  + Log reservations, cancellations, parking actions, and checkouts through the FileLogger interface.

**Driver**

Controls the flow of the Parking Management System, providing the main menu and user interaction.

**Features:**

* Initializes sample data and displays the main menu for login, registration, or exit.
* Handles user login and registration, verifying credentials, and adding new users.
* Provides a user menu with options for parking, reserving spots, checking out, adding vehicles, and managing balance.
* Allows users to park, reserve spots, cancel reservations, and check out vehicles.
* Facilitates adding balance, viewing balance, and managing registered vehicles.
* Ensures input validation and error handling throughout the user interactions.

**NOTE : .**java files have been added in the zip along with the rubric table and the question.