**Assignment: Implement a Parking Lot Management System in Java**

**Objective: Build a system to manage a parking lot using Java, with a focus on object-oriented principles and multithreading. This system should manage parking spots, allow vehicles to enter and exit, and handle concurrent operations like checking availability and retrieving vehicle details.**

**System Requirements:**

1. **ParkingSpot Class**
   * **Attributes:**
     + **spotId (int): Unique identifier for each spot.**
     + **spotType (String): Type of parking spot (e.g., "compact," "regular," or "large").**
     + **isOccupied (boolean): Indicates if the spot is currently occupied.**
     + **pricePerHour (double): Hourly rate for the spot.**
   * **Override Methods:**
     + **Override toString() to display parking spot details.**
     + **Override equals() and hashCode() to manage identity by spotId.**
2. **Vehicle Class**
   * **Attributes:**
     + **licensePlate (String): Unique identifier for each vehicle.**
     + **vehicleType (String): Type of vehicle (e.g., "motorcycle," "car," or "truck").**
3. **ParkingLotManager Class**
   * **Manages parking spots in a collection, such as HashMap<Integer, ParkingSpot> or ArrayList<ParkingSpot>.**
   * **Core Operations:**
     + **Add Parking Spot: Add a new parking spot to the lot. If a spot with the same spotId already exists, throw a SpotAlreadyExistsException.**
     + **Park Vehicle: Allow a vehicle to park in a specific spot. If the spot is occupied or unsuitable for the vehicle type, throw a SpotUnavailableException.**
     + **Remove Vehicle: Free up a parking spot. If the spot was not occupied, throw a SpotNotOccupiedException.**
     + **View Available Spots: Display all available parking spots with details.**
     + **Calculate Parking Fee: Calculate and display the total fee based on the duration the vehicle occupied the spot.**
4. **Custom Exception Handling**
   * **Define custom exceptions:**
     + **SpotAlreadyExistsException: Thrown when attempting to add a spot with an existing spotId.**
     + **SpotUnavailableException: Thrown when trying to park a vehicle in an unavailable or unsuitable spot.**
     + **SpotNotOccupiedException: Thrown when attempting to remove a vehicle from an empty spot.**
5. **Multithreading**
   * **Implement multithreading by creating separate Runnable classes for each core operation (AddParkingSpot, ParkVehicle, RemoveVehicle, ViewAvailableSpots).**
   * **Each operation should be executed in its own thread, and the main thread should use join() to ensure completion before proceeding.**

**Bonus Tasks:**

1. **Update Spot Details: Allow updating of spotType or pricePerHour for a parking spot.**
2. **Search by Spot Type: Enable searching for available spots based on type (e.g., find all "compact" spots).**
3. **Calculate Occupancy Rate: Periodically calculate the current occupancy rate of the parking lot in a separate thread.**