Parking Lot

Design and implement a car parking system in Core Java that can accommodate different types of vehicles and supports a configurable cost strategy. The system should support the following functionalities:

1. Initialize the parking lot with a given number of floors and car spaces per floor.
2. Add vehicle details to the parking lot, including the vehicle type (car, sports car, truck, bus), registration number and any other necessary attributes.
3. Remove a vehicle from the parking lot based on the registration number or parking slot number.
4. Check the availability of vehicle spaces on a specific floor for a given vehicle type.
5. Display the current status of the parking lot, including the occupied and available vehicle spaces on each floor for each vehicle type.
6. Configure a cost strategy for parking based on the initial configuration. The cost strategy should be able to vary the cost based on the vehicle type and support different currencies.

You should implement the following classes:

1. ParkingLot: Represents the parking lot and manages vehicle spaces.
2. Vehicle: Represents a vehicle with attributes such as vehicle type, registration number, color, and any other necessary attributes.
3. Floor: Represents a floor in the parking lot with attributes like floor number, capacity for each vehicle type, and vehicle spaces.
4. VehicleSpace: Represents a vehicle space with attributes such as space number, availability, vehicle type, and any other necessary attributes.
5. CostStrategy: Represents the cost strategy for parking, allowing for varying costs based on the vehicle type and supporting different currencies.

You are free to design and implement any additional classes, methods, or data structures as needed to solve the problem.
Instructions:

1. Write the Java code to implement the car parking system.
2. Implement the required functionalities mentioned above.
3. Use a main class to demonstrate the usage of the parking lot and its methods.
4. Test your code with multiple scenarios and edge cases.
5. Code should be generic and we should be able to use it for 10000 cars parking lots with multiple floors.
6. Document your code and provide explanations where necessary.
7. You have 3 hours to complete this assessment.

Please organize your code into appropriate packages and classes. Your main class should include the following methods:

1. Init : Initializes the parking lot with the given number of floors and vehicle spaces per floor for each vehicle type.

2. addVehicle : Adds a vehicle of the specified type with the given registration number and color to the parking lot.
3. removeVehicle : Removes the vehicle with the specified registration number from the parking lot.
4. checkAvailability : Checks the availability of vehicle spaces on the specified floor for the given vehicle type.
5. displayStatus(): Displays the current status of the parking lot, including the occupied and available vehicle spaces on each floor for each vehicle type.
6. configureCostStrategy : Configures the cost strategy for parking based on the initial configuration.

The CostStrategy class should include methods for calculating and retrieving the cost based on the vehicle type and currency.

Cost strategy example

Flat cost structure

- Bike per hour ₹ 10
- Car / Jeep per hour ₹ 20
- Bus / Truck per hour ₹ 30

Use case which should be covered

1. Initializing the Parking Lot for 2 Cars.
2. Enter 2 card details with timestamp value , should generate a token id for each car
3. This token can be used to take the car out of the parking lot. We should be able to call a method where we pass the token id and timestamp value and the system should tell if the token is correct. If the token is correct we should be able to get car details along with the total amount for the parking fee.
4. Once the capacity is full, i.e. if we have 2 cars already in the parking lot, if we try to enter the next car. It should throw an error.
5. The code should be loosely coupled, i.e. if we wish to add a new type of vehicle or a new cost strategy, then the code change is minimum.

Submit your completed Java code along with any supporting documents or explanations.

Duration : 3 hours

Expectation :

1. The Core Java code project should include a main method. Please refrain from creating a Spring or Spring Boot application. We expect pure Java code. Additionally, do not utilize a database; you may use the Java Collection library.
2. There should be a way to initialize the configuration (total capacity, total floor, different type of vehicle, type of currency user, per vehicle per hour/per day charges).
3. Also we should be able to enter the vehicle in the car parking and should take out the car by either car number or token number whichever you prefer.
4. On full capacity, the system should print an error message which should be simple to understand.