You are asked to write a **vehicle rental system** for a rental agency, which rents out different types of vehicles: **Cars, Motorcycles, and Trucks**.

The system should manage **vehicles, customers, and rental transactions**. Each type of vehicle has its own rules for calculating rental costs as follows:

- **Car**: rentalRate × days fName: customer's firstname.
- If days > 7 → apply **10% discount**.
- **Motorcycle**: rentalRate × days
  - Promotion: **first day free** if rented more than 1 day.
- **Truck**: rentalRate × days + surcharge × days
  - Surcharge = $\$20$/day if load capacity ≤ 3 tons, otherwise $\$50$/day.

The rental agency should be able to rent vehicles to customers, calculate rental costs, and track returns.

Here is a description of the classes:

1. **Vehicle (Abstract Class)** Serves as the parent class for all vehicles in the agency.
   - **Attributes:**
     - licensePlate: unique identifier for the vehicle.
     - brand: the vehicle brand (e.g., Toyota, Honda).
     - rentalRate: daily rental rate.
     - available: indicates if the vehicle is available for rent.
   - **Methods:**
     - Getters and setters for attributes.
     - isAvailable() and setAvailable(boolean status).
     - calculateRentalCost(int days): double (abstract) → to be implemented by subclasses with their specific cost rules.
2. **Car (Subclass of Vehicle)** Represents a car available for rent.
   - **Attributes:**
     - seats: number of seats in the car.
     - fuelType: type of fuel (e.g., petrol, diesel, electric).

- **Methods**:
  - Overrides calculateRentalCost(int days)
    - Base cost = rentalRate × days.
    - If days > 7 → apply **10% discount**.

3. **Motorcycle (Subclass of Vehicle)** Represents a motorcycle available for rent.
   - **Attributes**:
     - engineCC: engine size (e.g., 150, 600).
   - **Methods**:
     - Overrides calculateRentalCost(int days)
       - Base cost = rentalRate × days.
       - Promotion: **first day is free** if rented for more than 1 day.

4. **Truck (Subclass of Vehicle)** Represents a truck available for rent.
   - **Attributes**:
     - loadCapacity: maximum load capacity in tons.
   - **Methods**:
     - Overrides calculateRentalCost(int days) →
       - Base cost = rentalRate × days.
       - Add a surcharge per day:
         - ≤ 3 tons → +$20/day.
         - 3 tons → +$50/day.
         -

5. **Customer** Stores information about a customer who rents vehicles
   - **Attributes**:
     - fName: customer's first name.
     - lName: customer's last name.
     - driverLicenseNumber: customer's license number.
   - **Methods**:
     - Getters and setters for attributes.
     - getFullName()

6. **Rental** represents a rental transaction between a customer and a vehicle.

- **Attributes**:
  - customer: Customer who rented the vehicle.
  - vehicle: Vehicle which vehicle is rented.
  - days: rental duration.
  - active: rental status (active or completed).
- **Methods**:
  - startRental() marks the rental as active and vehicle unavailable.
  - endRental() marks the rental as completed and vehicle available.
  - getTotalCost() calls the vehicle's calculateRentalCost(days) to compute total rental cost.

7. **RentalAgency** manages all vehicles, customers, and rental transactions in the system.

- **Attributes**:
  - vehicles: ArrayList<Vehicle> all vehicles in the agency.
  - customers: ArrayList<Customer> all customers registered.
  - rentals: ArrayList<Rental> all rental transactions.
- **Methods**:
  - addVehicle(Vehicle v) add a vehicle to the fleet.
  - addCustomer(Customer c) add a customer.
  - rentVehicle(Customer c, Vehicle v, int days) → create a rental if available.
  - returnVehicle(Rental r) → complete a rental and return the vehicle.

8. **Rentable (Interface)** defines the basic operations that all rentable vehicles must implement.

- **Methods**:
  - rent(): Marks the vehicle as rented (sets availability to false).
  - returnVehicle(): Marks the vehicle as returned (sets availability to true).

1. **Write the required code of your classes and interfaces**.

- Implement all abstract methods in the Vehicle class, especially calculateRentalCost(int days), which must be overridden in Car, Motorcycle, and Truck.
- Ensure the Rentable interface methods (rent(), returnVehicle()) are implemented by all vehicles.

2. **Create a tester class** in which:
   - Create some customer objects and add them to the agency.
   - Create several vehicle objects (cars, motorcycles, trucks) and add them to the agency.
   - Create an ArrayList<Rental> and populate it with rentals for customers.
   - For each rental, specify the number of days and confirm the cost is computed correctly using the rules for each subclass.
   - Loop through the array list and print out the total cost of each rental.
3. **Write a method named maxRentalCost()** that receives an ArrayList<Rental> and returns the rental with the **maximum total cost**.
   - Invoke the method and print out the **name of the customer** and the **vehicle details** for that rental.

   public static Rental maxRentalCost(ArrayList<Rental> rentals)

Deliverables

To turn in the final project, please submit a compressed file containing:

- Your project folder named with your name.
- A file that contains **screenshots of sample runs** of the project.