Introduction

The goal of Task 4 was to create an implementation for a "Transportation Company" that manages a list of vehicles. This company implements the provided `LegalEntity` interface, which includes methods for obtaining the company's address and VAT number. Additionally, the `TransportationCompany` class has methods for managing the list of vehicles, similar to the functionalities implemented in Tasks 2 and 3.

Class Descriptions

1. LegalEntity.java

Functionality:
The `LegalEntity` interface defines the basic structure for any legal entity by providing two methods: `getAddress()` and `getVatNumber()`. These methods are intended to return the address and VAT number of the legal entity, respectively.

Goal:
To standardize the representation of any legal entity by enforcing the implementation of methods to get the address and VAT number.

```
1   package finalexam.task4;
2
    1 usage   1 implementation
3   public interface LegalEntity {
        no usages   1 implementation
4       String getAddress();
        no usages   1 implementation
5       String getVatNumber();
6   }
```

TransportationCompany.java

Functionality:
The `TransportationCompany` class implements the `LegalEntity` interface and manages a list of vehicles. It provides methods to add and remove vehicles from the list, as well as save the list to a file and load it from a file. The class also includes methods to get the address and VAT number of the company.

Goal:
To model a transportation company that manages its fleet of vehicles, with capabilities to persist the vehicle list to a file and retrieve it when needed.

```java
package finalexam.task4;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

2 usages
public class TransportationCompany implements LegalEntity {
    2 usages
    private String address;
    2 usages
    private String vatNumber;
    6 usages
    private List<Vehicle> vehicles;

    1 usage
    public TransportationCompany(String address, String vatNumber) {
        this.address = address;
        this.vatNumber = vatNumber;
        this.vehicles = new ArrayList<>();
    }

    no usages
    @Override
    public String getAddress() {
        return address;
    }

    no usages
    @Override
    public String getVatNumber() {
        return vatNumber;
```

```java
    @Override
    public String getVatNumber() {
        return vatNumber;
    }
    2 usages
    public void addVehicle(Vehicle vehicle) {
        vehicles.add(vehicle);
    }

    1 usage
    public boolean removeVehicle(Vehicle vehicle) {
        return vehicles.remove(vehicle);
    }

    3 usages
    public List<Vehicle> getVehicles() {
        return vehicles;
    }

    1 usage
    public void saveVehiclesToFile(String filename) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(vehicles);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    1 usage
    public void loadVehiclesFromFile(String filename) {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            vehicles = (List<Vehicle>) ois.readObject();
```

```java
    1 usage
    public void loadVehiclesFromFile(String filename) {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            vehicles = (List<Vehicle>) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}

```

Vehicle.java

Functionality:
The `Vehicle` class represents a vehicle in the transportation company's fleet. It includes fields for the model, license plate, and capacity of the vehicle. The class also provides getter and

setter methods for these fields, as well as a `toString()` method for representing the vehicle as a string.

Goal:
To model the details of a vehicle in the transportation company's fleet.

```java
package finalexam.task4;

import java.io.Serializable;

12 usages
public class Vehicle implements Serializable {
    4 usages
    private String model;
    4 usages
    private String licensePlate;
    4 usages
    private int capacity;

```

```
    2 usages
    public Vehicle(String model, String licensePlate, int capacity) {
        this.model = model;
        this.licensePlate = licensePlate;
        this.capacity = capacity;
    }

    no usages
    public String getModel() {
        return model;
    }

    no usages
    public void setModel(String model) {
        this.model = model;
    }

    no usages
    public String getLicensePlate() {
        return licensePlate;
    }

    no usages
    public void setLicensePlate(String licensePlate) {
        this.licensePlate = licensePlate;
    }

    no usages
    public int getCapacity() {
        return capacity;
    }
```

```
    }

    no usages
    public void setModel(String model) {
        this.model = model;
    }

    no usages
    public String getLicensePlate() {
        return licensePlate;
    }

    no usages
    public void setLicensePlate(String licensePlate) {
        this.licensePlate = licensePlate;
    }

    no usages
    public int getCapacity() {
        return capacity;
    }

    no usages
    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    @Override
    public String toString() {
        return "Vehicle [model=" + model + ", licensePlate=" + licensePlate + ", capacity=" + capacity + "]";
    }
}
```

CompanyTester.java

Functionality:
The `CompanyTester` class is a test class that demonstrates the functionalities of the `TransportationCompany` class. It creates a `TransportationCompany` instance, adds vehicles to it, displays the list of vehicles, saves the vehicle list to a file, removes a vehicle, displays the list again, and finally loads the vehicle list from the file.

Goal:
To test and demonstrate the functionalities of the `TransportationCompany` class and its interaction with the `Vehicle` class.

```java
package finalexam.task4;

public class CompanyTester {
    public static void main(String[] args) {
        TransportationCompany company = new TransportationCompany( address: "123 Main St",  vatNumber: "VAT123456");

        Vehicle vehicle1 = new Vehicle( model: "Truck",  licensePlate: "ABC123",  capacity: 10000);
        Vehicle vehicle2 = new Vehicle( model: "Van",  licensePlate: "XYZ789",  capacity: 2000);

        company.addVehicle(vehicle1);
        company.addVehicle(vehicle2);

        System.out.println("Vehicle List:");
        for (Vehicle v : company.getVehicles()) {
            System.out.println(v);
        }

        company.saveVehiclesToFile( filename: "vehicles.dat");

        company.removeVehicle(vehicle1);

        System.out.println("Vehicle List after removal:");
        for (Vehicle v : company.getVehicles()) {
            System.out.println(v);
        }

        company.loadVehiclesFromFile( filename: "vehicles.dat");

        System.out.println("Vehicle List after loading from file:");
        for (Vehicle v : company.getVehicles()) {
            System.out.println(v);
        }
    }

    company.loadVehiclesFromFile( filename: "vehicles.dat");

    System.out.println("Vehicle List after loading from file:");
    for (Vehicle v : company.getVehicles()) {
        System.out.println(v);
    }
}
```