Comprehensive Report for Task 4

Introduction

The task involves creating a management system for an organization named "Cleaning Service". The system includes functionalities for managing a list of employees, including adding, deleting, saving, and loading employee records. The project demonstrates the use of Java interfaces, serialization, and basic file I/O operations. The system is designed with three main classes: `CleaningService`, `Employee`, and `CleaningServiceTester`. Additionally, it implements an interface named `LegalEntity`.

1. LegalEntity Interface

Purpose:
The `LegalEntity` interface defines the blueprint for any class that represents a legal entity with an address and a VAT number. It enforces the implementation of two methods:

- **getAddress()**: Returns the address of the legal entity.
- **getVatNumber()**: Returns the VAT number of the legal entity.

Functionality:
This interface ensures that any implementing class will provide these essential details, which are common attributes of any legal entity.

2. Employee Class

Purpose:
The `Employee` class represents an employee with a name, surname, and employee ID. This class is serializable, allowing instances to be saved to and loaded from files.

Functionality:

- **Attributes**: Stores the name, surname, and employee ID.
- **Serialization**: Implements `Serializable` to allow the object to be serialized.
- **Equality and Hashing**: Overrides `equals` and `hashCode` methods to ensure proper comparison and hashing based on the `employeeId`.
- **String Representation**: Overrides `toString` method to provide a readable representation of an `Employee` object.

3. CleaningService Class

Purpose:
The `CleaningService` class implements the `LegalEntity` interface and manages a list of employees. It provides methods for adding, deleting, saving, and loading employees.

Functionality:

- **Attributes**: Stores the address, VAT number, and a list of employees.
- **Employee Management**:
  - `addEmployee(Employee e)`: Adds an employee to the list.
  - `deleteEmployee(Employee e)`: Removes an employee from the list if present.
  - `getEmployees()`: Returns a copy of the current employee list.
- **File Operations**:
  - `saveEmployeesToFile(String fileName)`: Saves the current list of employees to a file.
  - `loadEmployeesFromFile(String fileName)`: Loads a list of employees from a file.

4. CleaningServiceTester Class

Purpose:
The `CleaningServiceTester` class is a tester class containing the `main` method to demonstrate the functionalities of the `CleaningService` class.

Functionality:

- **Instantiation**: Creates an instance of `CleaningService` with a specified address and VAT number.
- **Adding Employees**: Demonstrates adding `Employee` objects to the `CleaningService`.
- **Saving to File**: Saves the current list of employees to a file named `employees.dat`.
- **Loading from File**: Loads the list of employees from `employees.dat` and prints them.
- **Deleting Employees**: Demonstrates removing an `Employee` from the `CleaningService`.

Conclusion

The Cleaning Service Management System provides an effective way to manage employees within an organization. It leverages Java's interfaces, serialization, and file I/O capabilities to offer functionalities like adding, deleting, saving, and loading employee records. This task not only demonstrates object-oriented programming principles but also showcases practical usage of Java's core features.