**A 6OOP Final Document Project**

<Delivery Agent Manager>

Presented to the School of Computing

Holy Angel University



Group 7

Manio, Marco Ruiz

Opiniano, Ezkylle

San Pedro, Adrian Elijah

October 20, 2024

**Project Title:** Delivery Agent Management

**Project Description:**

The Delivery Agent Management System is a console-based application designed to streamline the management of delivery agents in various industries, such as logistics, medical services, and food delivery. The system allows users to register, update, and delete agent details efficiently. Agents are categorized by the type of goods they deliver, such as documents, medical supplies, food, hazardous materials, and freight. The application also categorizes delivery vehicles into bikes, motorcycles, cars, vans, and trucks, facilitating efficient assignment of delivery agents.

**Scope**:

The code implements a **Delivery Agent Management System** using Java. Its scope includes:

1. **Agent Management**: Adding, updating, and removing delivery agents. The system stores each agent’s name, contact information, agent ID, vehicle type, goods type, and availability status.
2. **CSV Integration**: The system supports reading from and writing agent data to a CSV file, ensuring persistence across sessions.
3. **Filtering and Searching**: Users can filter agents by vehicle type and search by goods type, offering customized retrieval of agent data.
4. **User Interaction**: A user interface is provided for interacting with the system through a console-based menu.
5. **Error Handling**: Basic error handling is integrated, including handling invalid inputs, such as non-existent agent IDs or incorrect formats.

**Limitations:**

1. **Single Storage Format**: The system only supports CSV files for data storage. It lacks database integration, which would offer better scalability.
2. **Limited Input Validation**: Although there is some validation (e.g., for vehicle and goods types), the system does not perform deep validation on agent details such as contact numbers or the uniqueness of agent IDs.
3. **Console-Based Interface**: The user interface is limited to console input/output, lacking a graphical user interface (GUI) or web-based interaction, which limits ease of use and accessibility.
4. **Error Handling**: Exception handling is minimal and focused mainly on basic input errors. More complex exceptions (e.g., file read/write issues) are handled simply by printing error messages.
5. **Concurrency**: The system is not designed for concurrent use, meaning multiple users cannot manage agents simultaneously in real-time without risking data corruption.

**Code-base Structure**

/DeliveryAgentManagementSystem

│

├── Main.java # Entry point of the application.

├── DeliveryAgent.java # DeliveryAgent class, extending Person.

├── Person.java # Base class holding name and contact info.

├── DeliveryAgentManager.java # Manages delivery agents (add, update, remove, etc.).

├── CSVUtils.java # Utility class for reading and writing agent data from/to CSV.

├── ExceptionHandling.java # Handles different exceptions like invalid inputs.

├── UserInterface.java # Console-based interface for user interactions.

├── GoodsType.java # Class to validate and manage valid goods types.

├── VehicleType.java # Class to validate and manage valid vehicle types.

│

├── /Data

│ └── agents.csv # CSV file storing the delivery agent data.

│

└── README.md # Instructions on how to run and use the system.

**Main Class**

/\*\*

 \* Main Application for Delivery Agent Management System

 \*

 \* This program serves as the entry point for the Delivery Agent Management System.

 \* It provides a console-based interface that allows users to manage delivery agents,

 \* perform CRUD operations (Create, Read, Update, Delete), search/filter agents, and handle

 \* file operations for data persistence.

 \*

 \* Features:

 \* - Menu-driven system with options for adding, viewing, updating, and deleting agents

 \* - Filter agents by vehicle type and search by goods type

 \* - Data persistence using CSV file handling for loading and saving agent information

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/14/2024

 \*/

import java.util.List;

public class Main {

    public static void main(String[] args)

    {

        String filePath = "C:\\Users\\Ezkylle Opiniano\\Desktop\\SchoolStuff\\1st Semester\\oop\\finals\\Final OOP Group 7\\DeliveryAgentManagementSystem\\Data\\agents.csv"; // Update with your CSV file path

        DeliveryAgentManager agentManager = new DeliveryAgentManager(filePath);

        UserInterface userInterface = new UserInterface();

        boolean running = true;

        while (running)

        {

            userInterface.displayMenu();

            String choice = userInterface.getUserInput();

            switch (choice) {

                case "1": // Adding

                    userInterface.handleRegistration(agentManager);

                    agentManager.saveAgentsToCSV(); // Save after adding

                    break;

                case "2": // Updating

                    try

                    {

                        System.out.print("Enter Agent's ID to update: ");

                        int updateID = Integer.parseInt(userInterface.getUserInput());

                        if (agentManager.getAgentById(updateID) == null)

                        {

                            userInterface.displayErrorMessage("Agent ID not found.");

                            break;

                        }

                        userInterface.handleUpdate(agentManager, updateID);

                        agentManager.saveAgentsToCSV(); // Save after updating

                    }

                    catch (NumberFormatException e)

                    {

                        userInterface.displayErrorMessage("Invalid input. Please enter a valid number.");

                    }

                    break;

                case "3": // Deleting

                    try

                    {

                        System.out.print("Enter Agent's ID to delete: ");

                        int deleteID = Integer.parseInt(userInterface.getUserInput());

                        agentManager.removeAgent(deleteID);

                        agentManager.saveAgentsToCSV(); // Save after deleting

                    }

                    catch (NumberFormatException e)

                    {

                        userInterface.displayErrorMessage("Invalid input. Please enter a valid number.");

                    }

                    break;

                case "4": // Listing

                    List<DeliveryAgent> agents = agentManager.getAllAgents();

                    userInterface.displayAgentsList(agents);

                    break;

                case "5": // Filter Agents by Vehicle Type

                    System.out.print("Enter Vehicle Type to filter: ");

                    String vehicleType = userInterface.getUserInput();

                    List<DeliveryAgent> filteredAgentsByVehicle = agentManager.filterByVehicleType(vehicleType);

                    userInterface.displayAgentsList(filteredAgentsByVehicle);

                    break;

                case "6": // Search Agents by Goods

                    System.out.print("Enter Goods Type to search: ");

                    String goodsType = userInterface.getUserInput();

                    List<DeliveryAgent> searchedAgentsByGoods = agentManager.searchByGoodsType(goodsType);

                    userInterface.displayAgentsList(searchedAgentsByGoods);

                    break;

                case "7": // Save Agents to CSV

                    agentManager.saveAgentsToCSV();

                    System.out.println("Agents saved to CSV successfully.");

                    break;

                case "8": // Exit

                    running = false;

                    break;

                default:

                    System.out.println("Invalid choice. Try again.");

            }

        }

    }

}

**DeliveryAgent Class**

/\*\*

 \* DeliveryAgent Class

 \*

 \* This class represents a delivery agent within the Delivery Agent Management System.

 \* It encapsulates key attributes such as the agent's ID, name, contact number, availability status,

 \* the type of goods they handle, and the type of vehicle they use.

 \*

 \* It provides methods to update the agent's details and convert a CSV line into a DeliveryAgent object.

 \* The CSV data is expected to follow the format: agentID, name, contactNumber, goodsType, vehicleType, availability.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/14/2024

 \*/

public class DeliveryAgent extends Person

{

    private int agentID;

    private String vehicleType;

    private String goodsType;

    private boolean availability;

    public DeliveryAgent(int agentID, String name, String contactNumber, String vehicleType, String goodsType, boolean availability)

    {

        super(name, contactNumber);

        this.agentID = agentID;

        this.vehicleType = vehicleType;

        this.goodsType = goodsType;

        this.availability = availability;

    }

    public int getAgentID()

    {

        return agentID;

    }

    public String getVehicleType()

    {

        return vehicleType;

    }

    public void setVehicleType(String vehicleType)

    {

        this.vehicleType = vehicleType;

    }

    public String getGoodsType()

    {

        return goodsType;

    }

    public void setGoodsType(String goodsType)

    {

        this.goodsType = goodsType;

    }

    public boolean isAvailability()

    {

        return availability;

    }

    public void setAvailability(boolean availability)

    {

        this.availability = availability;

    }

    public void updateAgentDetails(String name, String contactNumber, String vehicleType, String goodsType, boolean availability) {

        setName(name);

        setContactNumber(contactNumber);

        setVehicleType(vehicleType);

        setGoodsType(goodsType);

        setAvailability(availability);

    }

    @Override

    public String toString()

    {

        return "Agent ID: " + agentID + "\n" +

               "Name: " + getName() + "\n" +

               "Contact Number: " + getContactNumber() + "\n" +

               "Vehicle Type: " + getVehicleType() + "\n" +

               "Goods Type: " + getGoodsType() + "\n" +

               "Availability: " + isAvailability();

    }

    public static DeliveryAgent fromCSV(String csvLine)

    {

        String[] fields = csvLine.split(","); // Split the line by commas

        int id = Integer.parseInt(fields[0].trim());

        String name = fields[1].trim();

        String contact = fields[2].trim();

        String goods = fields[3].trim();

        String vehicle = fields[4].trim();

        boolean availability = Boolean.parseBoolean(fields[5].trim());

        return new DeliveryAgent(id, name, contact, vehicle, goods, availability);

    }

}

**Person Class**

/\*\*

 \* Person Class

 \*

 \* This class represents a person with basic attributes: name and contact number.

 \* It provides getter and setter methods for managing the person's name and contact details.

 \*

 \* This class serves as the base class for more specialized classes, such as DeliveryAgent,

 \* in the Delivery Agent Management System.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/19/2024

 \*/

 public class Person

 {

    private String name;            // The name of the person

    private String contactNumber;   // The contact number of the person

    public Person(String name, String contactNumber)

    {

        this.name = name;

        this.contactNumber = contactNumber;

    }

    public String getName()

    {

        return name;

    }

    public void setName(String name)

    {

        this.name = name;

    }

    public String getContactNumber()

    {

        return contactNumber;

    }

    public void setContactNumber(String contactNumber)

    {

        this.contactNumber = contactNumber;

    }

}

**DeliveryAgentManager Class**

/\*\*

 \* Delivery Agent Management System

 \*

 \* This program manages delivery agents and categorizes them by the type of goods they deliver

 \* and the vehicle they use (e.g., bike, motorcycle, car, van, truck).

 \* It supports adding, updating, removing, and searching for agents.

 \* Additionally, the system can filter agents based on their vehicle type or goods type.

 \*

 \* The manager loads agent data from a CSV file during initialization and can save updated data back to the file.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/14/2024

 \*/

import java.util.ArrayList;

import java.util.List;

public class DeliveryAgentManager

{

    private List<DeliveryAgent> agentsList = new ArrayList<>();

    private String csvFilePath;

    // Constructor to initialize the CSV file path and load agents from CSV

    public DeliveryAgentManager(String csvFilePath)

    {

        this.csvFilePath = csvFilePath;

        loadAgentsFromCSV(); // Load agents from the CSV file during initialization

    }

    // Load agents from the CSV file

    private void loadAgentsFromCSV()

    {

        agentsList = CSVUtils.readFromCSV(csvFilePath);

    }

    // Save agents to the CSV file

    public void saveAgentsToCSV()

    {

        if (CSVUtils.writeToCSV(csvFilePath, agentsList))

        {

            System.out.println("Agents saved successfully to CSV.");

        } else {

            System.out.println("Error saving agents to CSV.");

        }

    }

    public void addAgent(DeliveryAgent agent)

    {

        agentsList.add(agent);

    }

    public void removeAgent(int agentID)

    {

        DeliveryAgent agent = getAgentById(agentID);

        if (agent != null)

        {

            agentsList.remove(agent);

            System.out.println("Agent removed successfully.");

        }

        else

        {

            System.out.println("Error: Agent ID not found.");

        }

    }

    public DeliveryAgent getAgentById(int agentID)

    {

        for (DeliveryAgent agent : agentsList)

        {

            if (agent.getAgentID() == agentID)

            {

                return agent;

            }

        }

        return null;

    }

    public void updateAgent(int agentID, String name, String contactNumber, String vehicleType, String goodsType, boolean availability) {

    DeliveryAgent agent = getAgentById(agentID);

    if (agent != null)

    {

        agent.updateAgentDetails(name, contactNumber, vehicleType, goodsType, availability);

        System.out.println("Agent updated successfully.");

    }

    else

    {

        System.out.println("Error: Agent ID not found.");

    }

}

    public List<DeliveryAgent> getAllAgents()

    {

        return agentsList;

    }

    public List<DeliveryAgent> filterByVehicleType(String vehicleType)

    {

        List<DeliveryAgent> filteredAgents = new ArrayList<>();

        for (DeliveryAgent agent : agentsList)

        {

            if (agent.getVehicleType().equalsIgnoreCase(vehicleType))

            {

                filteredAgents.add(agent);

            }

        }

        return filteredAgents;

    }

    public List<DeliveryAgent> searchByGoodsType(String goodsType)

    {

        List<DeliveryAgent> searchedAgents = new ArrayList<>();

        for (DeliveryAgent agent : agentsList) {

            if (agent.getGoodsType().equalsIgnoreCase(goodsType))

            {

                searchedAgents.add(agent);

            }

        }

        return searchedAgents;

    }

}

**CSVUtils Class**

/\*\*

 \* CSVUtils Class

 \*

 \* This utility class handles CSV file operations for managing delivery agent data

 \* in the Delivery Agent Management System. It includes methods for:

 \*

 \* - Reading delivery agent data from a specified CSV file to populate a list of DeliveryAgent objects.

 \* - Writing updated delivery agent information back to a CSV file, ensuring correct formatting

 \*   for Excel viewing, including header labels.

 \* - Handling I/O exceptions during read and write operations, providing feedback on errors.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/14/2024

 \*/

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class CSVUtils

{

    public static boolean writeToCSV(String filePath, List<DeliveryAgent> agents)

    {

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath)))

        {

            // Write the header line for the CSV

            writer.write("ID,Name,Contact Number,Goods Type,Vehicle Type,Availability");

            writer.newLine();

            // Write each agent's details to the CSV

            for (DeliveryAgent agent : agents)

            {

                writer.write(String.format("%d,%s,%s,%s,%s,%b",

                        agent.getAgentID(),

                        agent.getName(),

                        agent.getContactNumber(), // Now as String

                        agent.getGoodsType(),

                        agent.getVehicleType(),

                        agent.isAvailability()));

                writer.newLine();

            }

            return true; // Successful write

        }

        catch (IOException e)

        {

            System.out.println("Error writing to CSV: " + e.getMessage());

            return false; // Failed write

        }

    }

    public static List<DeliveryAgent> readFromCSV(String filePath)

    {

        List<DeliveryAgent> agents = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath)))

        {

            String line;

            // Skip the header line to focus on the data rows

            reader.readLine();

            // Read each line and convert it into a DeliveryAgent object

            while ((line = reader.readLine()) != null)

            {

                DeliveryAgent agent = DeliveryAgent.fromCSV(line);

                agents.add(agent); // Add the agent to the list

            }

        }

        catch (IOException e)

        {

            System.out.println("Error reading from CSV: " + e.getMessage());

        }

        return agents; // Return the populated list of agents

    }

}

**ExceptionHandling Class**

/\*\*

 \* ExceptionHandling Class

 \*

 \* This class provides custom exception handling methods for the Delivery Agent Management System.

 \* It handles specific exceptions such as invalid input and agent not found, and includes a method for

 \* handling general exceptions with custom error messages.

 \*

 \* Methods:

 \* - handleInvalidInputException(): Displays a message for invalid user input.

 \* - handleAgentNotFoundException(): Displays a message when an agent is not found.

 \* - handleGeneralException(String message): Displays a custom error message for general exceptions.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/22/2024

 \*/

public class ExceptionHandling

{

    // Handle InvalidInputException

    public void handleInvalidInputException()

    {

        System.out.println("Error: Invalid input. Please check your input and try again.");

    }

    // Handle AgentNotFoundException

    public void handleAgentNotFoundException()

    {

        System.out.println("Error: Agent not found. Please provide a valid agent ID.");

    }

    // Handle general exceptions with a custom message

    public void handleGeneralException(String message)

    {

        System.out.println("Error: " + message);

    }

}

**UserInterface Class**

/\*\*

 \* UserInterface Class

 \*

 \* This class provides the console-based user interface for the Delivery Management System.

 \* It handles user interactions, including registering, updating, deleting, and displaying

 \* delivery agents, filtering by vehicle type, and searching by goods type.

 \*

 \* Features:

 \* - Displays a menu with various options for managing delivery agents.

 \* - Captures and validates user inputs, such as agent details, vehicle type, and goods type.

 \* - Performs operations like adding, updating, removing, and listing delivery agents.

 \* - Saves delivery agent information to a CSV file.

 \* - Ensures user inputs are valid with custom validation for vehicle type, goods type, and availability.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/22/2024

 \*/

import java.util.Scanner;

import java.util.List;

public class UserInterface

{

    private Scanner scanner = new Scanner(System.in);

    private GoodsType goodsTypeObj = new GoodsType(); // Correct instance for goods type

    private VehicleType vehicle = new VehicleType();

    public void displayMenu()

    {

        System.out.println("========== Delivery Management System ===========");

        System.out.println("1. Add Agent");

        System.out.println("2. Update Agent");

        System.out.println("3. Delete Agent");

        System.out.println("4. List All Agents");

        System.out.println("5. Filter Agents by Vehicle Type");

        System.out.println("6. Search Agents by Goods");

        System.out.println("7. Save Agents to CSV");

        System.out.println("8. Exit");

        System.out.print("Enter your choice (1-8): ");

    }

    public String getUserInput()

    {

        return scanner.nextLine().trim();

    }

    public void handleUserChoice(DeliveryAgentManager agentManager)

    {

        while (true)

        {

            displayMenu();

            String choice = getUserInput();

            switch (choice)

            {

                case "1":

                    handleRegistration(agentManager);

                    break;

                case "2":

                    System.out.print("Enter the agent ID to update: ");

                    int agentIDToUpdate = Integer.parseInt(getUserInput());

                    handleUpdate(agentManager, agentIDToUpdate);

                    break;

                case "3":

                    System.out.print("Enter the agent ID to delete: ");

                    int agentIDToDelete = Integer.parseInt(getUserInput());

                    agentManager.removeAgent(agentIDToDelete);

                    break;

                case "4":

                    List<DeliveryAgent> agents = agentManager.getAllAgents();

                    displayAgentsList(agents);

                    break;

                case "5":

                    filterByVehicleType(agentManager);

                    break;

                case "6":

                    searchByGoodsType(agentManager);

                    break;

                case "7":

                    agentManager.saveAgentsToCSV();

                    System.out.println("Agents saved to CSV successfully.");

                    break;

                case "8":

                    System.out.println("Exiting...");

                    scanner.close(); // Close scanner here before exiting

                    return;

                default:

                    System.out.println("Invalid choice. Please enter a number between 1 and 8.");

            }

        }

    }

    // Handle Registration Method

    public void handleRegistration(DeliveryAgentManager agentManager)

    {

        try

        {

            System.out.print("Enter agent ID: ");

            int agentID = Integer.parseInt(getUserInput());

            System.out.print("Enter agent name: ");

            String name = capitalizeString(getUserInput());

            System.out.print("Enter contact number: ");

            String contactNumber = getValidContactNumber();

            System.out.print("Enter vehicle type [Bike, Motorcycle, Car, Van, Truck]: ");

            String vehicleType = getUserInput();

            while (!vehicle.isValidVehicleType(vehicleType))

            {

                System.out.println("Invalid vehicle type. Please enter a valid vehicle type.");

                vehicleType = getUserInput();

            }

            System.out.print("Enter goods type [Food, Medicine, Furniture, Livestock, Clothing, Jewelry, Tech Products, Games, Customized Products, Toys]: ");

            String goodsType = getUserInput();

            while (!goodsTypeObj.isValidGoodsType(goodsType, true))

            {

                System.out.println("Invalid goods type. Please enter a valid goods type.");

                goodsType = getUserInput();

            }

            boolean availability = getAgentAvailability();

            DeliveryAgent agent = new DeliveryAgent(agentID, name, contactNumber, vehicleType, goodsType, availability);

            agentManager.addAgent(agent);

            System.out.println("Agent registered successfully.");

        }

        catch (NumberFormatException e)

        {

            System.out.println("Invalid input. Please enter numeric values where required.");

        }

        catch (Exception e)

        {

            System.out.println("Error: " + e.getMessage());

        }

    }

    // Method to get and validate agent availability

    private boolean getAgentAvailability()

    {

        while (true) {

            System.out.print("Is the agent available? (true/false): ");

            String input = getUserInput();

            if (input.equalsIgnoreCase("true"))

            {

                return true;

            }

            else if (input.equalsIgnoreCase("false"))

            {

                return false;

            }

            else

            {

                System.out.println("Invalid input. Please enter 'true' or 'false'.");

            }

        }

    }

    // Capitalize String Method

    public String capitalizeString(String str)

    {

        if (str == null || str.isEmpty()) return str; // Handle empty input

        String[] words = str.split(" ");

        for (int i = 0; i < words.length; i++) {

            if (!words[i].isEmpty())

            { // Check for empty words

                words[i] = words[i].substring(0, 1).toUpperCase() + words[i].substring(1).toLowerCase();

            }

        }

        return String.join(" ", words);

    }

    // Handle Update Method

    public void handleUpdate(DeliveryAgentManager agentManager, int agentID)

    {

        try

        {

            System.out.print("Enter new name: ");

            String name = capitalizeString(getUserInput());

            System.out.print("Enter new contact number (Minimum of 10 Number like 9876543210): ");

            String contactNumber = getUserInput();

            System.out.print("Enter new vehicle type [Bike, Motorcycle, Car, Van, Truck]: ");

            String vehicleType = getUserInput();

            while (!vehicle.isValidVehicleType(vehicleType))

            {

                System.out.println("Invalid vehicle type. Please enter a valid vehicle type.");

                vehicleType = getUserInput();

            }

            System.out.print("Enter new goods type [Food, Medicine, Furniture, Livestock, Clothing, Jewelry, Tech Products, Games, Customized Products, Toys]: ");

            String goodsType = getUserInput();

            while (!goodsTypeObj.isValidGoodsType(goodsType, true))

            {

                System.out.println("Invalid goods type. Please enter a valid goods type.");

                goodsType = getUserInput();

            }

            // Handle new availability status

            boolean availability = getAgentAvailability();

            // Update agent details

            agentManager.updateAgent(agentID, name, contactNumber, vehicleType, goodsType, availability);

            System.out.println("Agent details updated successfully.");

        }

        catch (Exception e)

        {

            System.out.println("Error: " + e.getMessage());

        }

    }

    // Display Agents List Method

    public void displayAgentsList(List<DeliveryAgent> agents)

    {

        if (agents.isEmpty())

        {

            System.out.println("No agents found.");

            return;

        }

        System.out.printf("%-10s %-20s %-15s %-15s %-15s %-10s\n", "Agent ID", "Name", "Contact Number", "Vehicle Type", "Goods Type", "Availability");

        System.out.println("----------------------------------------------------------------------------------------------------------------");

        for (DeliveryAgent agent : agents)

        {

            System.out.printf("%-10d %-20s %-15s %-15s %-15s %-10s\n",

                agent.getAgentID(),

                agent.getName(),

                agent.getContactNumber(),

                agent.getVehicleType(),

                agent.getGoodsType(),

                agent.isAvailability() ? "True" : "False"

            );

        System.out.println("----------------------------------------------------------------------------------------------------------------");

        }

    }

    // Filter by Vehicle Type Method

    public void filterByVehicleType(DeliveryAgentManager agentManager)

    {

        System.out.print("Enter vehicle type to filter: ");

        String vehicleType = getUserInput();

        List<DeliveryAgent> filteredAgents = agentManager.filterByVehicleType(vehicleType);

        displayAgentsList(filteredAgents);

    }

    // Search by Goods Type Method

    public void searchByGoodsType(DeliveryAgentManager agentManager)

    {

        System.out.print("Enter goods type to search: ");

        String goodsType = getUserInput();

        List<DeliveryAgent> searchedAgents = agentManager.searchByGoodsType(goodsType);

        displayAgentsList(searchedAgents);

    }

    public void displayErrorMessage(String message)

    {

        System.out.println("Error: " + message);

    }

    // Method to get valid contact number

    private String getValidContactNumber()

    {

        while (true)

        {

            String contactNumber = getUserInput();

            if (isValidContactNumber(contactNumber))

            {

                return contactNumber;

            } else {

                System.out.println("Invalid contact number. Please enter a valid number.");

            }

        }

    }

    private boolean isValidContactNumber(String contactNumber)

    {

        return contactNumber.matches("\\d{10}"); // Example for a 10-digit number

    }

}

**GoodsType Class**

/\*\*

 \* GoodsType Class

 \*

 \* This class defines and manages the list of goods types that delivery agents

 \* in the Delivery Agent Management System can handle. It provides methods to:

 \* - Validate whether a given goods type is valid, with an option for case-insensitive matching.

 \* - Retrieve the list of all available goods types.

 \*

 \* The goods types included are: Food, Medicine, Furniture, Livestock, Clothing, Jewelry,

 \* Tech Products, Games, Customized Products, and Toys.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/19/2024

 \*/

import java.util.Arrays;

import java.util.List;

public class GoodsType

{

     private final List<String> goodsTypes = Arrays.asList("Food", "Medicine", "Furniture", "Livestock", "Clothing",

                                                                "Jewelry", "Tech Products", "Games", "Customized Products", "Toys");

    public boolean isValidGoodsType(String goodsType, boolean ignoreCase)

    {

        if (ignoreCase)

        {

            return goodsTypes.stream().map(String::toLowerCase).anyMatch(type -> type.equals(goodsType.toLowerCase()));

        }

        else

        {

            return goodsTypes.contains(goodsType);

        }

    }

    public List<String> getAllGoodsTypes()

    {

        return goodsTypes;

    }

}

**VehicleType Class**

/\*\*

 \* VehicleType Class

 \*

 \* This class defines and manages the list of vehicle types that can be used

 \* by delivery agents in the Delivery Agent Management System. It provides

 \* methods to:

 \* - Validate whether a given vehicle type is valid, using case-insensitive matching.

 \* - Retrieve the list of all available vehicle types.

 \*

 \* The vehicle types included are: Bike, Motorcycle, Car, Van, and Truck.

 \*

 \* Author: OOP GROUP 7

 \* Date: 10/19/2024

 \*/

 import java.util.Arrays;

 import java.util.List;

 public class VehicleType

 {

    private final List<String> vehicleTypes = Arrays.asList("Bike", "Motorcycle", "Car", "Van", "Truck");

    public boolean isValidVehicleType(String vehicleType)

    {

        return vehicleTypes.stream().anyMatch(v -> v.equalsIgnoreCase(vehicleType));

    }

    public List<String> getAllVehicleTypes()

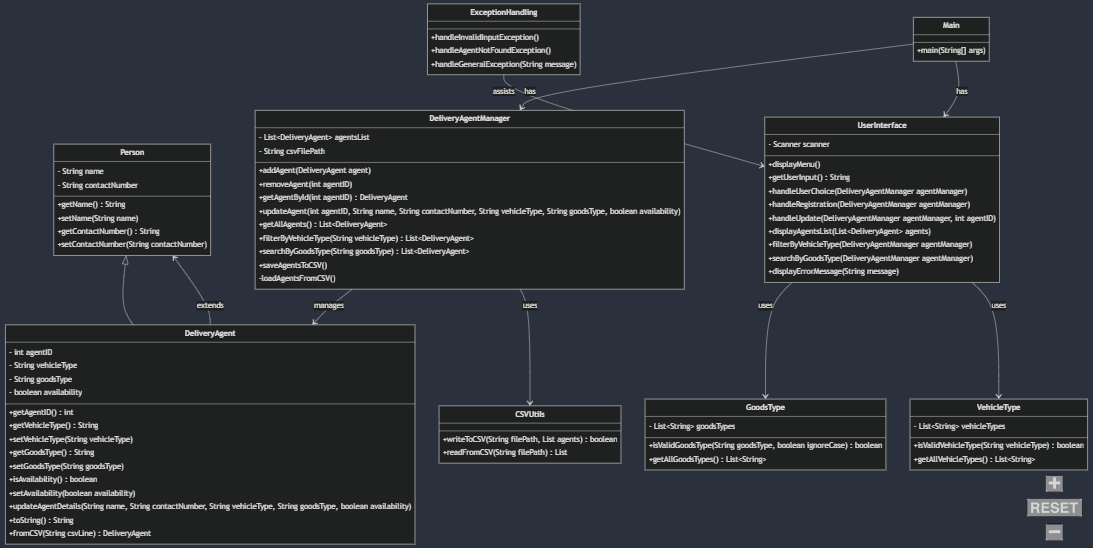
    {

        return vehicleTypes;

    }

}

**UML Diagram:**



**Appendices**

**Main Menu**

A screenshot of a computer

Description automatically generated

**Add Agent:**

A screen shot of a computer

Description automatically generated

**Update Agent:**

**A screenshot of a computer

Description automatically generated**

**Delete Agent:**

**A screenshot of a computer

Description automatically generated**

**List Agents:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Filter Agents by vehicle type:**

**A screenshot of a computer

Description automatically generated**

**Filter Agents by Goods:**

**A screenshot of a computer screen

Description automatically generated**

**Save Agents to CSV:**

**A screen shot of a computer

Description automatically generated**

**Exit:**

**A black screen with white text

Description automatically generated**

CSV file

A screenshot of a computer

Description automatically generated