**Members:**

Opiniano, Ezkylle

San Pedro, Adrian Elijah

Manio, Marco Ruiz

Week 2: Core Functionality (Java Code)

1. DeliveryAgent Class

This class will represent a delivery agent with attributes like name, contact number, vehicle type, and availability.

/\*\*

 \* The DeliveryAgent class represents a delivery agent,

 \* encapsulating key details such as name, contact number,

 \* vehicle type, and availability.

 \*

 \* Attributes:

 \* - name: The full name of the delivery agent.

 \* - contactNumber: A string representing the agent's contact number.

 \*   This could be expanded to include validation for format in future improvements.

 \* - vehicleType: Describes the type of vehicle the agent uses

 \*   (e.g., bike, motorcycle, car, van, truck).

 \* - availability: A boolean indicating whether the agent is currently available for deliveries.

 \* - id: The unique identifier for the delivery agent, starting from 1.

 \* - goods: A string representing the type of goods the agent delivers.

 \*

 \* Methods:

 \* - Getters and setters are used to ensure encapsulation, allowing for controlled

 \*   access and modification of agent details.

 \* - toCSV: Converts the delivery agent's information to CSV format for easy storage.

 \* - fromCSV: A static method that allows for the creation of a DeliveryAgent object

 \*   from a CSV line, facilitating CSV imports.

 \*/

public class DeliveryAgent

{

    private int id; // ID will be assigned during the CSV write process

    private String name;

    private String contactNumber;

    private String vehicleType;

    private boolean availability;

    private String goods; // New attribute for the type of goods delivered

    // Constructor

    public DeliveryAgent(String name, String contactNumber, String vehicleType, boolean availability, String goods)

    {

        this.name = name;

        this.contactNumber = contactNumber;

        this.vehicleType = vehicleType;

        this.availability = availability;

        this.goods = goods; // Initialize goods

    }

    // Getters and setters for encapsulation

    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;

    }

    public String getVehicleType()

    {

        return vehicleType;

    }

    public void setVehicleType(String vehicleType)

    {

        this.vehicleType = vehicleType;

    }

    public boolean isAvailable()

    {

        return availability;

    }

    public void setAvailability(boolean availability)

    {

        this.availability = availability;

    }

    public int getId()

    {

        return id;

    }

    public void setId(int id)

    {

        this.id = id;

    }

    public String getGoods() // New getter for goods

    {

        return goods;

    }

    public void setGoods(String goods) // New setter for goods

    {

        this.goods = goods;

    }

    // Method to convert to CSV format

    public String toCSV()

    {

        return id + "," + name + "," + contactNumber + "," + vehicleType + "," + availability + "," + goods;

    }

    // Static method to create an agent from CSV

    public static DeliveryAgent fromCSV(String csvLine)

    {

        String[] fields = csvLine.split(",");

        DeliveryAgent agent = new DeliveryAgent(fields[1], fields[2], fields[3], Boolean.parseBoolean(fields[4]), fields[5]); // Include goods

        agent.setId(Integer.parseInt(fields[0])); // Set the ID from the CSV

        return agent;

    }

}

2. DeliveryAgentManager Class

This class will manage the collection of delivery agents, allowing for registration, updating, deletion, and saving/loading from a CSV file.

/\*\*

 \* The DeliveryAgentManager class manages a collection of delivery agents,

 \* providing CRUD (Create, Read, Update, Delete) operations.

 \*

 \* It is responsible for:

 \* - Registering new agents

 \* - Updating existing agents based on their index in the list

 \* - Deleting agents by index

 \* - Saving agent data to a CSV file for persistence

 \* - Loading agent data from a CSV file on application startup

 \*

 \* The class is designed with extensibility in mind, allowing additional

 \* operations to be added in the future (e.g., searching agents by name or vehicle type).

 \*/

import java.util.ArrayList;

import java.util.List;

public class DeliveryAgentManager

{

    private List<DeliveryAgent> agents;

    public DeliveryAgentManager()

    {

        this.agents = new ArrayList<>();

    }

    // Register a new delivery agent

    public void addAgent(DeliveryAgent agent)

    {

        agent.setId(agents.size() + 1); // Assign ID starting from 1

        agents.add(agent);

    }

    // Update an existing delivery agent

    public void updateAgent(int index, DeliveryAgent updatedAgent)

    {

        updatedAgent.setId(index + 1); // Update ID to match the agent's index

        agents.set(index, updatedAgent);

    }

    // Delete an existing delivery agent

    public void deleteAgent(int index)

    {

        if (index >= 0 && index < agents.size())

        {

            agents.remove(index);

            // Reassign IDs after deletion to keep them sequential

            for (int i = index; i < agents.size(); i++)

            {

                agents.get(i).setId(i + 1);

            }

        }

    }

    // Get the list of agents

    public List<DeliveryAgent> getAgents()

    {

        return agents;

    }

    // Display agents with index starting from 1

    public void displayAgents()

    {

        System.out.println("Current Agents:");

        for (int i = 0; i < agents.size(); i++)

        {

            DeliveryAgent agent = agents.get(i);

            // Display index starting from 1

            System.out.println((i + 1) + ": " + agent.getName() + " - " + agent.getVehicleType() + " - " + agent.getGoods());

        }

    }

    // Save agents to CSV

    public void saveToCSV(String filePath)

    {

        CSVUtils.writeToCSV(filePath, agents);

    }

    // Load agents from CSV

    public void loadFromCSV(String filePath)

    {

        agents = CSVUtils.readFromCSV(filePath);

    }

}

3. CSVUtils Class

This class will handle the reading from and writing to CSV files for storing agent information.

/\*\*

 \* The CSVUtils class provides utility methods for reading and writing

 \* delivery agent data to and from a CSV file.

 \*

 \* Key functionality:

 \* - writeToCSV: Writes a list of DeliveryAgent objects to a CSV file,

 \*   including a header line for better organization when viewed in Excel.

 \* - readFromCSV: Reads a CSV file and converts each line back into

 \*   DeliveryAgent objects for use in the application.

 \*

 \* This separation of concerns allows for cleaner code and easier maintenance.

 \*/

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class CSVUtils

{

    // Method to write a list of agents to a CSV file

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

    {

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

        {

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

            writer.newLine();

            for (DeliveryAgent agent : agents)

            {

                writer.write(agent.toCSV());

                writer.newLine();

            }

        }

        catch (IOException e)

        {

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

        }

    }

    // Method to read agents from a CSV file

    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

            reader.readLine();

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

            {

                DeliveryAgent agent = DeliveryAgent.fromCSV(line);

                agents.add(agent);

            }

        }

        catch (IOException e)

        {

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

        }

        return agents;

    }

}

4. Main Class

This class will manage the input/output and act as the entry point of the application.

/\*\*

 \* The Main class serves as the entry point for the Delivery Agent Management System.

 \* It provides a user interface to manage delivery agents through various operations

 \* such as adding, updating, deleting, listing, and searching agents.

 \* The application loads agent data from a CSV file on startup and saves it before exiting.

 \*/

import java.util.List;

import java.util.Scanner;

import java.util.InputMismatchException;

public class Main

{

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        DeliveryAgentManager manager = new DeliveryAgentManager();

        // Use the correct path to the data folder

        String filePath = "C:\\Users\\Ezkylle Opiniano\\Desktop\\SchoolStuff\\1st Semester\\oop\\finals\\Final OOP Group 7\\DeliveryAgentManagementSystem\\Data\\agents.csv";

        // Load agents from CSV (if the file exists)

        try {

            manager.loadFromCSV(filePath);

        }

        catch (Exception e)

        {

            System.out.println("Failed to load data from CSV. Please check the file path.");

        }

        boolean exit = false;

        while (!exit)

        {

            System.out.println("\nDelivery Agent 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. Save Agents to CSV");

            System.out.println("6. Exit");

            int choice = -1;

            // Loop to ensure valid integer input for menu selection

            while (choice == -1)

            {

                System.out.print("Choose an option: ");

                try

                {

                    choice = scanner.nextInt();

                    scanner.nextLine();  // Consume newline after nextInt

                }

                catch (InputMismatchException e)

                {

                    System.out.println("Invalid input. Please enter a number between 1 and 6.");

                    scanner.nextLine();  // Consume the invalid input

                }

            }

            switch (choice)

            {

                case 1:

                    // Add a new agent

                    DeliveryAgent newAgent = inputAgent(scanner);

                    manager.addAgent(newAgent);

                    break;

                case 2:

                    // Update an agent

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

                    int updateIndex = getValidIndex(scanner, manager.getAgents().size());

                    DeliveryAgent updatedAgent = inputAgent(scanner);

                    manager.updateAgent(updateIndex, updatedAgent);

                    break;

                case 3:

                    // Delete an agent

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

                    int deleteIndex = getValidIndex(scanner, manager.getAgents().size());

                    manager.deleteAgent(deleteIndex);

                    break;

                case 4:

                    // List all agents

                    List<DeliveryAgent> agents = manager.getAgents();

                    if (agents.isEmpty())

                    {

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

                    }

                    else

                    {

                        System.out.println("ID,Name,Contact Number,Vehicle Type,Availability"); // Header for display

                        for (int i = 0; i < agents.size(); i++)

                        {

                            System.out.println((i + 1) + ": " + agents.get(i).toCSV());  // Start displaying agents from 1

                        }

                    }

                    break;

                case 5:

                    // Save agents to CSV

                    manager.saveToCSV(filePath);

                    break;

                case 6:

                    // Exit the program

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

                    manager.saveToCSV(filePath);

                    scanner.close();

                    exit = true;

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        }

    }

    // Method to input agent details from the user

    private static DeliveryAgent inputAgent(Scanner scanner)

    {

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

        String name = scanner.nextLine();

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

        String contactNumber = scanner.nextLine();

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

        String vehicleType = scanner.nextLine();

        boolean availability;

        while (true)

        {

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

            String availabilityInput = scanner.nextLine();

            if (availabilityInput.equalsIgnoreCase("true") || availabilityInput.equalsIgnoreCase("false"))

            {

                availability = Boolean.parseBoolean(availabilityInput);

                break;

            }

            else

            {

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

            }

        }

        System.out.print("Enter goods delivered by the agent: "); // Prompt for goods

        String goods = scanner.nextLine(); // Get the goods input

        // Pass the goods parameter to the DeliveryAgent constructor

        return new DeliveryAgent(name, contactNumber, vehicleType, availability, goods);

    }

    // Method to validate and get a valid index for update/delete operations

    private static int getValidIndex(Scanner scanner, int size)

    {

        int index = -1;

        while (index < 0 || index >= size)

        {

            System.out.print("Enter a valid index: ");

            try

            {

                index = scanner.nextInt();

                scanner.nextLine();  // Consume newline after nextInt

                if (index < 0 || index >= size)

                {

                    System.out.println("Index out of range. Please try again.");

                }

            }

            catch (InputMismatchException e)

            {

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

                scanner.nextLine();  // Consume the invalid input

            }

        }

        return index;

    }

}

Output  
  
