Part-One

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.IOException;

import java.io.ByteArrayInputStream;

import java.util.LinkedList;

public class MyMain {

    // Inner Person class

    public static class Person {

        private String firstName;

        private String lastName;

        private String id;

        // Constructor with three parameters

        public Person(String firstName, String lastName, String id) {

            this.firstName = firstName;

            this.lastName = lastName;

            this.id = id;

        }

        // Accessor methods

        public String getFirstName() {

            return firstName;

        }

        public String getLastName() {

            return lastName;

        }

        public String getId() {

            return id;

        }

        // toString() method

        @Override

        public String toString() {

            return "{" +

                    "firstName='" + firstName + '\'' +

                    ", lastName='" + lastName + '\'' +

                    ", id='" + id + '\'' +

                    '}';

        }

    }

    // store method: Reads data from input stream and stores in linked list

    public static void store(InputStream inputStream, LinkedList<Person> list) throws IOException {

        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));

        String line;

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

            String[] data = line.split(",");

            if (data.length == 3) {

                list.add(new Person(data[0], data[1], data[2]));

            }

        }

    }

    // display method: Writes data of all persons in the linked list to the output stream

    public static void display(OutputStream outputStream, LinkedList<Person> list) throws IOException {

        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outputStream));

        for (Person person : list) {

            writer.write(person.toString());

            writer.newLine();

        }

        writer.flush();

    }

    // find method: Returns the index of the person with the given id, or -1 if not found

    public static int find(String sid, LinkedList<Person> list) {

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

            if (list.get(i).getId().equals(sid)) {

                return i;

            }

        }

        return -1;

    }

    // Main method

    public static void main(String[] args) {

        LinkedList<Person> personList = new LinkedList<>();

        // Create a data file with data for a few person objects (Simulating file input)

        String data = "John,Doe,123\nJane,Smith,456\nAlice,Johnson,789\n";

        InputStream inputStream = new ByteArrayInputStream(data.getBytes());

        try {

            // Store data into the linked list

            store(inputStream, personList);

            // Display data from the linked list

            System.out.println("Displaying all persons:");

            display(System.out, personList);

            // Find and display the index of a person by id

            String searchId = "456";

            int index = find(searchId, personList);

            System.out.println("\nPerson with ID " + searchId + " is at index: " + index);

            // Test find method with a non-existing ID

            searchId = "999";

            index = find(searchId, personList);

            System.out.println("Person with ID " + searchId + " is at index: " + index);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

Part-two

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.IOException;

import java.io.ByteArrayInputStream;

import java.util.LinkedList;

public class PersonList {

    // Person class definition

    public static class Person {

        private String firstName;

        private String lastName;

        private String id;

        // Constructor with three parameters

        public Person(String firstName, String lastName, String id) {

            this.firstName = firstName;

            this.lastName = lastName;

            this.id = id;

        }

        // Accessor methods

        public String getFirstName() {

            return firstName;

        }

        public String getLastName() {

            return lastName;

        }

        public String getId() {

            return id;

        }

        // toString() method

        @Override

        public String toString() {

            return "{" +

                    "firstName='" + firstName + '\'' +

                    ", lastName='" + lastName + '\'' +

                    ", id='" + id + '\'' +

                    '}';

        }

    }

    // Internal LinkedList of Person objects

    private LinkedList<Person> list = new LinkedList<>();

    // store method: Reads data from input stream and stores in the linked list

    public void store(InputStream inputStream) throws IOException {

        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));

        String line;

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

            String[] data = line.split(",");

            if (data.length == 3) {

                list.add(new Person(data[0], data[1], data[2]));

            }

        }

    }

    // display method: Writes data of all persons in the linked list to the output stream

    public void display(OutputStream outputStream) throws IOException {

        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outputStream));

        for (Person person : list) {

            writer.write(person.toString());

            writer.newLine();

        }

        writer.flush();

    }

    // find method: Returns the index of the person with the given id, or -1 if not found

    public int find(String sid) {

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

            if (list.get(i).getId().equals(sid)) {

                return i;

            }

        }

        return -1;

    }

    // Get the entire list (if needed for further processing)

    public LinkedList<Person> getList() {

        return list;

    }

    // Main method

    public static void main(String[] args) {

        // Create a PersonList object

        PersonList personList = new PersonList();

        // Create a data file with data for a few person objects (Simulating file input)

        String data = "John,Doe,123\nJane,Smith,456\nAlice,Johnson,789\n";

        InputStream inputStream = new ByteArrayInputStream(data.getBytes());

        try {

            // Store data into the PersonList object

            personList.store(inputStream);

            // Display data from the PersonList object

            System.out.println("Displaying all persons:");

            personList.display(System.out);

            // Find and display the index of a person by id

            String searchId = "456";

            int index = personList.find(searchId);

            System.out.println("\nPerson with ID " + searchId + " is at index: " + index);

            // Test find method with a non-existing ID

            searchId = "999";

            index = personList.find(searchId);

            System.out.println("Person with ID " + searchId + " is at index: " + index);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

**Question 2 (Original Version)**: This version is more process centered. The main program directly handles the operations (store, display, find) by invoking static methods. The emphasis is on the sequence of operations and how they interact with the data structure.

**Question 3 (Modified Version with PersonList)**: This version is more data centered. The operations are encapsulated within the PersonList class, focusing on the data structure itself. The main program interacts with the data through methods that belong to the PersonList object, making the data structure the central focus.