

## Section Solutions 8

---

```
import acm.program.*;
import acm.util.*;
import java.util.*;
import java.io.*;

public class FlightPlanner extends ConsoleProgram {
    /* The name of the flights file. */
    private static final String FLIGHTS_FILE = "flights.txt";

    /* The separator used to delimit the start and end of a flight. */
    private static final String FLIGHT_DELIMITER = " -> ";

    /* A map from the lower-case representation of a name to its original
     * capitalization.
     */
    private HashMap<String, String> capitalizationMap
        = new HashMap<String, String>();

    /* A map from cities to cities reachable from there. The keys in the
     * map are the lower-case representations of the city names, and the
     * values are the lowercase representations.
     */
    private HashMap<String, ArrayList<String>> flights
        = new HashMap<String, ArrayList<String>>();

    public void run() {
        /* Populate the data structures. */
        loadFlights();

        /* Find the flight path. */
        ArrayList<String> flightPath = chooseFlightPath();

        /* Display the flight path. */
        printFlightPath(flightPath);
    }
}
```

```

/**
 * Populates the internal data structures using the flight information from the
 * file.
 */
private void loadFlights() {
    try {
        BufferedReader br = new BufferedReader(new FileReader(FLIGHTS_FILE));

        while (true) {
            String line = br.readLine();
            if (line == null) break;

            /* If the line is nonempty, process it as a flight entry. */
            if (!line.isEmpty()) {
                processFlight(line);
            }
        }
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

/**
 * Given a line of the file encoding a flight, extracts the flight information
 * from that line.
 *
 * @param line The line to parse.
 */
private void processFlight(String line) {
    /* Find where the -> in the string is, then get the source and
     * destination of the flight.
     */
    int splitPoint = line.indexOf(FLIGHT_DELIMITER);
    String source = line.substring(0, splitPoint);
    String destination = line.substring(splitPoint + FLIGHT_DELIMITER.length());

    /* If this is the first time we've seen the source, create an entry for it.
     * in our data structures.
     */
    if (!capitalizationMap.containsKey(source.toLowerCase())) {
        capitalizationMap.put(source.toLowerCase(), source);
        flights.put(source.toLowerCase(), new ArrayList<String>());
    }

    /* Add this flight. */
    flights.get(source.toLowerCase()).add(destination.toLowerCase());
}

```

```

/**
 * Prompts the user to enter a flight path, returning the ultimate path. The
 * returned path uses the lower-case representations of the city names.
 *
 * @return The chosen flight path.
 */
private ArrayList<String> chooseFlightPath() {
    ArrayList<String> result = new ArrayList<String>();

    /* Find out where we're starting. */
    String source = chooseStartingCity();
    result.add(source);

    /* Track which city we are currently at. */
    String currentCity = source;
    while (true) {
        String nextCity = chooseNextCity(currentCity);
        result.add(nextCity);

        /* Stop if we're back where we started. */
        if (source.equals(nextCity))
            return result;

        /* Update our position. */
        currentCity = nextCity;
    }
}

/**
 * Prompts the user to choose a starting city, returning the city that was
 * chosen.
 *
 * @return The city that was chosen.
 */
private String chooseStartingCity() {
    displayWelcome();

    while (true) {
        String choice = readLine("Enter the starting city: ").toLowerCase();

        /* If this is a valid city, return it. */
        if (flights.containsKey(choice))
            return choice;

        /* Otherwise, reprompt. */
        println("Sorry, that's not a valid choice.");
    }
}

```

```

/**
 * Displays a nice welcome message to the user.
 */
private void displayWelcome() {
    println("Welcome to Flight Planner!");
    println("Here's a list of all the cities in our database: ");

    /* List all the cities that we know of. One way to do this would be
     * to iterate across the capitalization map's keys and find the
     * associated values, but since we just want the properly-capitalized
     * cities we can just iterate over the value set.
     */
    for (String city: capitalizationMap.values()) {
        println(" " + city);
    }
}

/**
 * Prompts the user to choose the next city in the path, which must be a city
 * that's reachable from the current city.
 *
 * @param currentCity The current city.
 * @return The next city in the path.
 */
private String chooseNextCity(String currentCity) {
    printCitiesReachableFrom(currentCity);

    /* Get the properly-capitalized representation of the current city. */
    String printCity = capitalizationMap.get(currentCity);

    while (true) {
        String line =
            readLine("Where do you want to go from " + printCity + "? ").toLowerCase();

        /* If the city is reachable, go there. */
        if (flights.get(currentCity).contains(line)) {
            return line;
        }

        println("Sorry, you can't go there from " + printCity + ".");
    }
}

/**
 * Lists all the cities reachable from some given city.
 *
 * @param city The city to list reachable cities from.
 */
private void printCitiesReachableFrom(String city) {
    println("From " + capitalizationMap.get(city) + " you can go to: ");

    /* Iterate across the reachable cities. */
    for (String destination: flights.get(city)) {
        println(" " + capitalizationMap.get(destination));
    }
}

```

```

/**
 * Prints a human-readable representation of a flight path.
 *
 * @param path The flight path to display.
 */
private void printFlightPath(ArrayList<String> path) {
    println("The route you've chosen is: ");

    /* Build up the path to display incrementally. */
    String toDisplay = "";
    for (int i = 0; i < path.size(); i++) {
        toDisplay += capitalizationMap.get(path.get(i));

        /* Insert an arrow in-between all of the cities. Be sure not to append
         * an unnecessary arrow at the end!
         */
        if (i != path.size() - 1)
            toDisplay += " -> ";
    }

    println(toDisplay);
}
}

```