Mehran Sahami
CS 106A

Handout #43A
November 28, 2018

Solution to Section #8

Parts of this handout by Brandon Burr and Patrick Young

```
/*
 * FlightPlanner.java
 * ------------------
 * Reads in a file of cities and their corresponding flights,
 * and allows the user to plan a flight route.
 */

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

public class FlightPlanner extends ConsoleProgram {

   public void run() {
      println("Welcome to Flight Planner!");
      readFlightData("flights.txt");

      println("Here's a list of all the cities in our database:");
      printCityList(cities);

      println("Let's plan a round-trip route!");
      String startCity = readLine("Enter the starting city: ");
      ArrayList<String> route = new ArrayList<String>();
      route.add(startCity);
      String currentCity = startCity;

      while (true) {
         String nextCity = getNextCity(currentCity);
         route.add(nextCity);
         if (nextCity.equals(startCity)) break;
         currentCity = nextCity;
      }

      printRoute(route);
   }

   /** Ask user for the name of the next city in the route */
   private String getNextCity(String city) {
      ArrayList<String> destinations = getDestinations(city);
      String nextCity = null;
      while (true) {
         println("From " + city + " you can fly directly to:");
         printCityList(destinations);
         String prompt = "Where do you want to go from " + city + "? ";
         nextCity = readLine(prompt);
         if (destinations.contains(nextCity)) break;
         println("You can't get to that city by a direct flight.");
      }
      return nextCity;
   }
```

```
/**
 * Given a starting city, looks up the destinations from that
 * city in the HashMap of flights, and return an array list of
 * the destinations that are available.
 */
private ArrayList<String> getDestinations(String fromCity) {
   return flights.get(fromCity);
}


/**
 * Prints a list of cities from the array list. Each city name is
 * indented by a space.
 */
private void printCityList(ArrayList<String> cityList) {
   for(int i = 0; i < cityList.size(); i++) {
      String city = cityList.get(i);
      println(" " + city);
   }
}


/**
 * Given a list of city names, prints out the flight
 * route, with a " -> " between each pair of cities
 */
private void printRoute(ArrayList<String> route) {
   println("The route you've chosen is: ");
   for (int i = 0; i < route.size(); i++) {
      if (i > 0) print(" -> ");
      print(route.get(i));
   }
   println();
}


/**
 * Reads in the city information from the given file storing the
 * information in both the ArrayList of cities and the HashMap of
 * flights.
 */
private void readFlightData(String filename) {
   flights = new HashMap<String, ArrayList<String>>();
   cities = new ArrayList<String>();
   try {
      BufferedReader rd =
                  new BufferedReader(new FileReader(filename));
      while (true) {
         String line = rd.readLine();
         if (line == null) break;
         if (line.length() != 0) {
            readFlightEntry(line);
         }
      }
      rd.close();
   } catch (IOException ex) {
      throw new ErrorException(ex);
   }
}
```

```
   /**
    * Reads a single flight entry from the line passed as an argument,
    * which should be in the form
    *
    * fromCity -> toCity
    *
    * Each new city is added to the ArrayList cities, and each new
    * flight is recorded by adding a new destination city to the
    * ArrayList stored in the HashMap flights under the key for the
    * starting city.
    */
   private void readFlightEntry(String line) {
      int arrow = line.indexOf("->");
      if (arrow == -1) {
         throw new ErrorException("Illegal flight entry " + line);
      }

      // Note: trim() removes leading/ending spaces from a string
      String fromCity = line.substring(0, arrow).trim();
      String toCity = line.substring(arrow + 2).trim();
      defineCity(fromCity);
      defineCity(toCity);
      getDestinations(fromCity).add(toCity);
   }


   /**
    * Defines a city if it has not already been defined. Defining
    * a city consists of entering it in the cities array and
    * entering an empty ArrayList in the flights table to show
    * that it has no destinations yet.
    */
   private void defineCity(String cityName) {
      if (!cities.contains(cityName)) {
         cities.add(cityName);
         flights.put(cityName, new ArrayList<String>());
      }
   }


   /* Private instance variables */
   private Map<String, ArrayList<String>> flights;
   private ArrayList<String> cities;
}
```