Chris Piech
CS 106A

Section #8
May 30, 2018

Solution to Section #8

Based on problems by Brandon Burr, Patrick Young, and Nick Troccoli

1. Flight

```
/*
 * File: FlightSolution.java
 * A fully implemented Flight class.
 */
public class FlightSolution {
   private String source;
   private String destination;
   private double duration;

   public FlightSolution(String src, String dest, double dur) {
      this.source = src;
      this.destination = dest;
      this.duration = dur;
   }

   public String getSource() {
      return this.source;
   }

   public String getDestination() {
      return this.destination;
   }

   public double getDuration() {
      return this.duration;
   }

   public String toString() {
      return this.source + "->" + this.destination + ":" +
               this.duration;
   }
}
```

2. Flight Planner Server

```
/*
 * File: FlightPlannerServer.java
 * ---------------------
 * A server program that, when run, reads in information
 * about available flights from a data file, and then listens
 * for incoming network requests.  This program can respond to
 * two types of requests:
 *
 * "getAllCities" -> we send back a list of all cities
 * "getDestinations" -> (needs parameter "city") we send back a
 *                      list of all cities reachable from the
 *                      provided city with the travel time between
 *                      the two.
 */
```

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

public class FlightPlannerServerSolution extends ConsoleProgram
    implements SimpleServerListener {

    /* The port number where we listen for requests */
    private static final int PORT = 8080;

    /* The name of the file containing our flight data */
    private static final String FLIGHT_DATA_FILE = "flights.txt";

    /* The server object that we use to listen for requests */
    private SimpleServer server;

    /* A map of cities to a list of flights starting from that city */
    private HashMap<String, ArrayList<FlightSolution>> flightMap;

    public void run() {
        flightMap = new HashMap<String, ArrayList<FlightSolution>>();
            readFlightData(FLIGHT_DATA_FILE);
        server = new SimpleServer(this, PORT);
        server.start();
        println("Starting server...");
    }


    /* Deal with a request */
    public String requestMade(Request request) {
        String cmd = request.getCommand();
            print("");
        if (cmd.equals("getAllCities")) {
            return getAllCities();
        } else if (cmd.equals("getDestinations")) {
                  return getDestinations(request);
        } else {
            return "Error, cannot process request: " + request;
        }
    }

    private void readFlightData(String filename) {
        try {
            Scanner fileScanner = new Scanner(new File(filename));
            while (fileScanner.hasNextLine()) {
                String line = fileScanner.nextLine();
                if (line.length() != 0) {
                    // make sure the line isn't blank
                    processLine(line);
                }
            }
            fileScanner.close();
        } catch (IOException ex) {
            throw new ErrorException(ex);
        }
    }
    private void processLine(String line) {
        String[] flightComponents = line.split(",");
        if (flightComponents.length != 3) {
            throw new ErrorException("Illegal entry in flights file: "
                                              + line);
        }
        // get the first city and get rid of spaces
        String fromCity = flightComponents[0].trim();
        // get the second city and get rid of spaces
        String toCity = flightComponents[1].trim();
        //get the flight time in hours as a double
        double flightTime =
                  Double.parseDouble(flightComponents[2].trim());

        addFlight(fromCity, toCity, flightTime);
    }

    /*
     * Add the fromCity -> toCity route to our map, making sure to put
     * the key in the map if it isn't already there.
     */
    private void addFlight(String fromCity, String toCity,
                               double duration) {
      FlightSolution flight
                  = new FlightSolution(fromCity, toCity, duration);
        if (!flightMap.containsKey(fromCity)) {
            flightMap.put(fromCity, new ArrayList<FlightSolution>());
        }
        flightMap.get(fromCity).add(flight);
    }

    /* Deal with a getAllCities request */
    private String getAllCities() {
        println("Received request to get all cities");
        ArrayList<String> cities = new ArrayList<String>();
        for (String city : flightMap.keySet()) {
            // iterate over the keys and add it to an arraylist
            cities.add(city);
        }
        String result = cities.toString();
        println("       => " + result);
        return result;
    }

    /* Deal with a getDestinations request */
    private String getDestinations(Request request) {
        String city = request.getParam("city");
        println("Received request to getDestinations for " + city);

        if (!flightMap.containsKey(city)) {
            return null;
        }
        ArrayList<FlightSolution> flights = flightMap.get(city);
        String result = flights.toString();
        println("       => " + result);
        return result;
    }
}
```