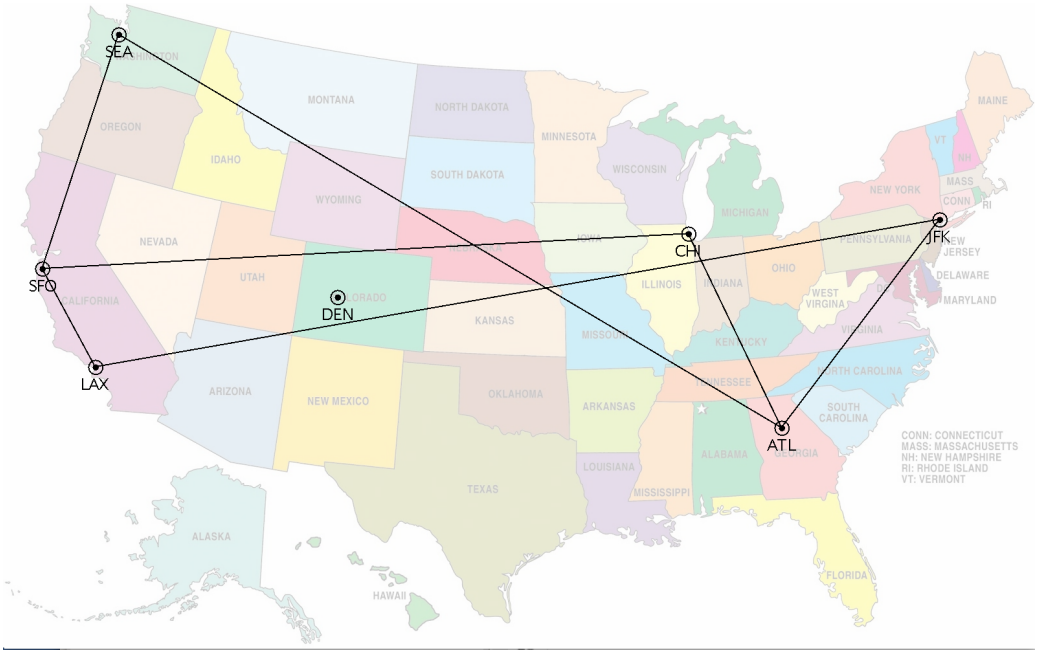
CS/SE 46B
Fall 2019
Homework 9: General Graphs
Due 11:59 PM, Sunday Dec 1



In this assignment you'll be an airline executive. You're going to need some software.

The starter code you downloaded with this assignment is an incomplete graphical tool (GUI) that lets you point and click to create and destroy airports and flight routes. The GUI code is complete. Your job is to complete the Airport class and write the FlightNet class as described below.

First create a new Java project in Eclipse. Right-click on the project name in the Package Explorer, select New->Folder, and create a Folder called "pix". Drag the downloaded file usmap.jpg into the pix folder.

Create a new Java package in your project. Call the package "airlines". Import the 3 downloaded source files (RoutesPanel.java, Airport.java, and AirGrader.java) into the package. You won't edit RoutesPanel or AirGrader. The compiler errors in those files will go away as you complete Airport.java.

Complete Airport.java by providing the following methods, as described by the comments in the starter file:

- public void connectTo(Airport that)
- public void disconnectFrom(Airport that)
- public boolean equals(Object x)
- public int compareTo(Airport that)
- public boolean isConnectedTo(Airport that)

Then write class airlines.FlightNet, which models a company's air routes. The class should aggregate instances of Airport. It should do this by extending HashSet<Airport>. It should *not* also have an instance variable of type Hashet. This class doesn't need *any* instance variables, and you'll get zero points if you have any. The class should have the following methods:

- public boolean nameIsAvailable(String name) – Returns true if the FlightNet doesn't contain an airport with the specified name. Be sure to use deep-equals, not shallow-equals, to check this.
- public void connect(Airport a1, Airport a2) – Connects a1 and a2 in both directions. You'll have to connect a1 to a2, and also a2 to a1.
- public void disconnect(Airport a1, Airport a2) – Opposite of above.
- public void removeAndDisconnect(Airport removeMe) – Removes removeMe from the FlightNet, and disconnects it from any airports that are still in the FlightNet.
- public Airport getAirportNearXY(int x, int y, int maximumDistance) – Checks all airports in the FlightNet. Returns the first airport whose (x,y) location is within maximumDistance of the x,y args of the method. Returns null if no airport is within maximumDistance. Note: Check out the hypot method of the Math class.

Run AirGrader to see how you're doing.

After you complete Airport and FlightNet, you'll be able to run RoutesPanel as an app. You'll see a map like the one on page 1. Left-click on open territory in the map to create a new airport. Double-click on any airport to delete it and any routes to it. Single-click on any airport to create or delete a route: existing routes will appear red, and you can delete one by clicking the other airport. Nonexistent routes will appear green, and you can create a route by clicking on the other airport. All these operations cause calls to a FlightNet instance that is owned by RoutesPanel.