7 Assignment

Portfolios: Circular Data; The World Library

1. Convert your *OceanWorld* project to work in *Eclipse*. You may consult with your partner, but ultimately be responsible for your own solution. Customize it to distinguish your solution from your partner (e.g. draw the fish differently, add a sign on the top describing the game and the author, change the fish behavior slightly, etc.)

2. Do the Problem 23.6 and the Problem 24.12 in the text. Combine the solutions to both problems into one project.

Main Assignment: Mapping the World, Circular Data

7.1 Problem

Finish the circular data problem from the lab 7 that deals with buddy lists.

7.2 Graphs - Networks

Our ultimate plan is to design classes that help us draw a map of locations and the roads between them, and help the user navigate a path from one location to another. We start with a simple representation of a graph as a list of nodes, where each node contains some data, identifies its location, and has a list of its neighbors.

1. Start with the data definition for a graph as a list of nodes.

   The information about each node should be encapsulated in a class *Data*. For now, the only value stored in *Data* will be a short *String* that serves as a label for the node. Later it may contain the name of a city, the state it is in, its zip code, and more.

   The location should also be encapsulated in its own class. For now, a *Posn* or two integer valued coordinates provide all the information we need to draw the graph. Later, the location of a city may be represented by its geographic coordinates given as latitude and longitude.

   The list of neighbors should be a list of nodes, not just the names of the nodes.

2. Add to the examples you have already made (remember the *Design Recips for Data Definitions* includes that as its last step) the examples you have used in an earlier assignment when your graph represented a line drawing.

3. Design the method *drawGraph* that will display the graph nodes and edges in a Canvas. Think of a way to represent the direction of the arrows.

4. Design the method *findPath* that produces a path from a start node to a destination node as a list of nodes. If there is no way to get to the destination node from the start node, the resulting path is the empty list. *Follow the Design Recipe.*

5. Design the method *drawPath* that draws the path produced by the method *findPath*.