Technical Report

Applied Data Types

We will be using two ADTs in our program.

1. A complete graph called Map that extends Graph from javafoundations package. This will be used to organize the points on the map.
2. LinkedLists, which will be used for two primary functions. The first is to keep track of each node on the map. The second is to keep track of clues associated with each node.

Classes

1. The first file we have defined is a public class, GeoScavenger.java. This file provides most of the functionality for the overall game.
    a. readItems creates a LinkedList of items from a randomly selected file (there are 6 possible text files to draw from, at this point)
    b. foundAllItems determines if all the items have been found in the game
    c. edgeContainsItem determines if there is an item between two nodes
    d. getMap returns the map of the game
    e. setCurrentNode allows the user to update the current node
    f. getCurrentNode returns the current node of the user
    g. getStepsTaken returns the amount of steps a user has taken in the game
    h. increaseSteps increases the number of steps taken by 1
    i. takeNextStep simulates taking one step in the game
    j. getItems returns the list of items in the game
2. Map.java is the second class we have defined, and this class will be used to create the structure of how a user will be able to move.
3. Nodes.java is the last class that we have defined so far. The Map object will be made of Nodes, and each Node will have coordinates and clues associated with it.
4. Building.java is a class that constructs Building objects, constructed of 3 parameters: An x-coordinate (integer), a y-coordinate (integer), and the name of the building (String). In addition to accessor and mutator methods, Building.java also allows the game to set the location (x and y coordinates) of the Building object.

5. GraphBuilder.java is an abstract class that extends java.lang.Object, and takes an element of type T, as follows: GraphBuilder<T>. When we want to create a graph of some specific object type, we need to extend this abstract class. The child class, BuildingGraphBuilder.java, will have to provide implementation for the "createOneThing()" method.
    a. The method createOneThing() is to be overridden in the extension of this class. It will create and return an object of the specific type the graph will contain, from the input String S.
    b. The method build reads from the input .tgf file, line by line. It creates the vertex objects, and adds them to the graph. Then, it adds the connections between the vertices. PRECONDITION: the input file must be in .tgf format.
6. BuildingGraphBuilder.java is a public class that extends GraphBuilder<Building>, meaning the GraphBuilder of Building types.
    a. Contains method createOneThing() that takes in a String s, and returns a Building object. The method creates and returns an object of the specific type the graph will contain (Building object), from the input String s. Method is overriding method createOneThing() from GraphBuilder class.
7. Item.java is a public class to represent an item to be used in the Scavenger Game. The class contains several accessor methods, along with a toString() and a method to mark when a method has been found.
    a. getName returns the name of the object
    b. getOnEdge returns the edge that the item is on
    c. getIsFound returns if the item has been found or not
    d. hasBeenFound marks the item as found
    e. toString returns a nicely formatted String version of the object
8. The public class MakeWellesleyMap.java is simply a driver class to test BuildingGraphBuilder
9. The public class MakeWellesleyTGF.java:
    a. First, the class creates the Building objects that will represent buildings on the Wellesley College Campus; each building is given an (x,y) coordinate, as well as a String to represent their name.

   b. Next, the class uses the AdjListGraph<Building> to create a map out of the Building objects that were constructed in the previous step. Each building becomes a vertex that is added to the graph, g, of Building objects.
   c. Next, edges are added between each building object on the map (edges between the nodes on the graph).
   d. Finally, the entire graph, g, is saved into a .tgf file titled “wellesley”
10. CoordinateTest.java
   a. A file used to test how to get the proper coordinates of each item in the game

GUI Files

1. AboutPanel.java
   a. Displays the information for the game
2. GameOutputPanel.java
   a. Creates the panel that represents the game output on the GUI
3. GamePanel.java
   a. Creates the panel that represents the game display on the GUI
4. NextChoicesPanel.java
   a. Displays the user's options for their next location
   b. Would have liked to include a “RESET” button in this file, alongside the “OK” and “QUIT” buttons.
5. GeocacheGUI.java
   a. Sets up a frame containing a tabbed pane

Stuff we would have liked to include/would include in future versions:

- RESET button
- Counter for steps taken so far
- Timer that counts down seconds as they pass while user is playing the game
- Alternative game boards to show different college campuses
- Changing levels of difficulty: While we already accounted for the changing of locations of items to find (using the different text files in our GeoScavenger.java), we could have additionally created graphs with more location Nodes.