What the code does:
The code does all of the specified things in the handout. More specifically, here is an outline of the program:

- Classes have been created for Roads, Nodes, Segments and Polygons. These hold the data that is provided in the files given to us. The specific information held in each class is particular to the information given in the file.
- A class, called “Connect” connects all the other classes, as well as storing the data structures which keeps track of the collection of Nodes, Roads, Segments and Polygons. This could have been combined with the Main method, but I thought it would be too complicated to combine the GUI extension with the holding of the data structures, and so I made two seperate classes.
- The Main Class, extends the GUI class given to us. This Overrides some methods, such as the onLoad method.
  - The onLoad method creates buffered readers for each of the files which it is passed, and calls a seperate load method in the connect class, which loads individual items into the structures
- The redraw method is in the Main class, but all it does is call methods in the connect class which redraws the seperate items.
  - redrawNodes draws all of the nodes in the nodes in the nodes map. This is done by converting the location into a point on the screen, and drawing an oval (relative to the scale of the map). If the node is highlighted, it will draw it in orange.
  - redrawSegments draws all of the segments in the segments list (orange if highlighted). It does this drawing a straight line between each point along the segment.
  - drawPoly draws all of the polygons. First it checks if the polygon should be drawn at the current scale level (if scale is bigger than the end level). If it is supposed to be drawn, it converts all the points of the polygon into a point on the screen, and adds these to arrays (one for the x value, one for the y). A polygon can then be drawn using those 2 arrays.
- The user is able to navigate the map in a few different ways:
  - All are controlled using the onMove method in the Main class, which zooms in/out or shifts the origin (and the rest of the map by default) based on the input ‘m’
    - Moving the map can be done by pressing the buttons provided
    - Using a scroll wheel can zoom in or out
    - Using the arrow keys moves the map around. Please note that for this to work, the user has to press the text output area once.
      - I decided to implement arrow keys instead of mouse dragging to avoid confusion between clicking on a node and dragging the mouse to move.
  - The user is able to click on a node (or close to it) and it selects a node (turns it orange). This is done using a Quad tree, which is defined by the QuadNode class. The quad tree itself is made in the onLoad method. The onClick method creates a location from the point it is clicked on, finds the closest node to this location, and highlights this. Although this is only meant to be one node, it sometimes will select two nodes if they are very close to each other. It also prints information about the node, and roads connected to it.
  - The user is able to search for a road, based on a given prefix. Searching for a road will highlight all segments of it (make them orange). This is done using a Trie structure, which holds a single letter, and traversing through the tree spells out a certain road name (for ease of printing the road name, I have included a field of the name of the road (ie all the parents of the node plus its own letter)). The trie itself is created in the Connect class, createTrie method.

Data Structures:
There are multiple data structures used for different purposes:

- Maps to hold all the nodes (indexed by ID) and roads(x2, indexed once by Name, and once by ID)
- Lists to hold all the segments
- A set to hold all the polygons
- A Trie structure to easily search for all roads that match a certain prefix.
  - This is a structure where each Trie Node holds a Map of Trie Nodes which are connected to it (indexed by the character of that child node). By traversing down the tree structure, you are able to find all roads matching a certain prefix or name.
- A Quad Tree to find the closest node to a specific mouse click.
  - A Quad tree splits an area into 4 quadrants, each of which is able to hold one 'Nodes' object. If there is already a nodes object in a certain quadrant, it subdivides the quadrant into 4 smaller quadrants, each with their own quad node. To retrieve a Nodes object, you just have to search for the specific quadrant that the mouse click is in, and return the nodes object inside it.
- In the Nodes class, there are two lists of Segments, the inList and the outList, which hold all of the segments coming into and out of that node, this is based on the Adjacency list covered in class

Test:
I found it difficult to create tests for this project since so much of it relied on user input. Therefore I did my testing by hand. At first I spent a lot of time on making sure things that should work did, (clicking on a node should highlight it, searching a road should do the same). During the final days of the project I tried to break my program, to identify problems I had (loading polygons when there wasn't a file (small map), clicking without having loaded the map etc). Although this testing was by no means exhaustive, it was extensive enough in my opinion given the timeframe. Ideally I would have spent more time testing, but the difficulty of creating the Quad tree and loading the polygons prevented this, as this took up a large amount of time.