**City Class**

This class creates a city with fields name, the x coordinate and the y coordiante. The method getDistance is used to get the distance between the city and another city

by calculating the gradient using abs( delta y/ delta x ).

**Route Class**

This is used to populate the route with cities. It has two constructors , the first Route(ObservableList<City> cities) is used to populate the List of cities with the cities supplied in the argument, the second Route(Route route) is used to create an object of cities from an already existing route object.

getTotalDistance() is used to find the total distance from the first city to the last plus the distance between the first city and the last. What it actually does is it takes two cities , call the getDistance method of the City class for each pair of adjacent cities and then sums the total it then adds the result to the difference between the first city and the last and returns the result.

**MainFX**

This is the class where the gui is built. setCities() is used to populate the cities list after which the route is initialized.

findShortestRoute(route) is then called .This method tries to find the shortest route by creating an adjacent route from the current route. This is achieved by the getAdjacentRoute( currentRoute ) method which after being created takes the value of two cities (excluding the first) and swaps them. The total distance of this new route created is then calculated and then compared with the currentRoute. If the value of the adjacent route is less than the current route, then the cities in the adjacent route replaces the cities in the current route in that same order. The process then repeats itself again. If the value of adjacent route is greater than or equal to that of the current route then getAdjacentRoute( currentRoute ) is called to get a new adjacent route. This cycle continues until the Hundred iteration of the while loop. The current route at this point and its cities is the solution.

The value of the current city is used to create two line charts. The first contains the path from the first city to the last city. The second only contains a line from the first city to the last. These two graphs are then layed on each other to produce the graph.