Omar Malik
QuadTree

1. I made City extends a Point class so that I could access its x and y coordinates to do all of the calculation for the QuadTree.

   The Quad class parses through input text and calls the correct methods of Quad Tree. QuadTree<T> has a generic T that extends point. It contains an empty flyweight, a root node, and an int representing the boundaries of the region. It calls the methods on the root node.

   QNode<T> is an abstract class I created so I could follow a node-centric approach. It's generic type T extends point so I can access the x and y coordinates.

   InternalNode, EmptyNode, and LeafNode are all internal classes inside of QuadTree. InternalNode extends QNode and contains 4 QNode elements that act as its children. LeafNode extends QNode and contains data type T which extends Point. EmptyNode extends QNode and is used as a flyweight. Only one object of the EmptyNode class is created, and all empty nodes in the tree point to it.

2. rfind is only called on an InternalNode if the internal node falls within the boundaries of the rfind region. When rfind is called on a LeafNode, it makes sure the LeafNode's data is within in the region and it adds to an arraylist. rfind then prints the contents of the arraylist.
   find is called on an InternalNode if the coordinate falls within the bounds of that internal node. When find is called on a LeafNode, the data is only returned if the x and y values of the data match the find's search coordinates.