Nathan Reitinger
UCSD: OOP



# Week 2
# Supporting Documentation

**Class**: MapGraph
**Modifications made to MapGraph (what and why):**
- Added a HashMap for the nodes (intersections)
- Added a List of type mapEdge for my edges

**Class name:** myNode
**Purpose and description of class:**
- Added a LinkedList of type mapEdge where I will store all of my relevant information about edges

- Added an “addEdge” method so I add edges during the MapGraph’s call to addEdge

- Added getters and setters

**Class name:** mapEdge
**Purpose and description of class:**
- Added getters and setters for each of the features passed in (i.e., roadName, length, and roadType); these were not heavily used in Week 2, but they might be used in the future

**Overall Design Justification (4-6 sentences):**
- I followed the support video closely when it came to creating my classes, but branched out in regard to the breadth-first search algorithm. The major change here (though we’ll see how different it really was post-peer review) was using a HashMap for the parentMap built with a geographic point and a node (see below).

```
==>HashMap<GeographicPoint, myNode> parentMap = new HashMap<GeographicPoint,
myNode>();
```

- I used nodes for the second value in the HashMap because I found that it was easier, conceptually, to think about grabbing edges and putting them into the queue. This was a little more difficult in terms of getting geopoints out of the nodes for their use in the visited HashSet, but I used a for loop to solve that challenge.