# PART #1:

Question 4

a) To begin, the technical debt included in the starter code mainly has to do with the difficulty of drawing the grid with an unordered data structure. The starter code provided us with a set of vertices which was contained within a hash set. The biggest issue with a hash set is that the elements are in no particular order. This presents a challenge as it is very difficult to connect two neighboring vertices when their places in the data structure holding them are totally unrelated. This resulted in lines being drawn totally randomly as our first instinct was to create a segment from the ‘ith’ index to the ‘i+1th’ index. This obviously did not work because, as was mentioned before, the hash set stores the vertices in totally random order. Once potential solution to this was to create a hash function to essentially decrypt the random order being given to the vertices. Since no one in our group has experience with that, we decided to change it to an array list since the data structure will be altered later on anyways. The other main source of technical debt in the starter is the fact that it is primarily set up to accommodate a square grid as the vertices are drawn specifically in a square and the variable which generates the x and y values (square size) is tailor made to create squares which will not be viable as we move to irregular meshes.

b) If we were to stay at the immutable data structure level, it would stop us from implementing a lot of functions in the future that would be much easier if we were to change it up. For example, we found that the dots stored as hash tables were not too effective when drawing the segments because it all became very random. We changed the data structure to be an array list as we found having more order in our segments would help us in finishing our step one business logic. If we were to have an immutable data structure, it would be incredibly difficult to try and work around it instead of just changing our data structure. Even more problems would arise later on if we did not alter the data structure at all, specifically if we were stuck working with unordered vertices and could not store our data as polygons. Once we move on to irregular meshes, if we were still stuck using a set of unordered vertices, it would essentially be impossible to efficiently draw segments between the consecutive vertices of each polygon. By storing the vertices in a polygon, we are able to keep track of which vertices need to be connected (and in what order) which greatly simplifies a process which would not be viable if we had to manually identify each vertex in our polygons which will be randomly generated in future iterations.

# PART #2

Question 5: