# PART #1:

Question 4

a) Technical debt in our project can be defined in an iterative process as the debt associated with cutting corners to produce an iterative step. The technical debt that we have is when we create our squares, the dots are drawn over each other. This impacts the generator class and code that we have created. This is ok right now as we only need to draw the dots, but with later steps we must begin to average out the dot’s color to create the color for the segments. Having multiple dots means that the color will be completely wrong and our code will need some major fixing. Having the wrong color means that our visualizer code needs to be fixed to get the right colour for our dots.

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 to do so 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 and there was no order to it. We changed the data structure to be a list as we found having more order in our segments would help us in finishing our first step with 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 chancing our data structure.

# PART 2

WE NEED TO DRAW A CLASS DIAGRAM

1. To fix some of our flaws with the starter code, we first changed our data structure that would store each dot. It was a hash table. With has hash tables there is no order to them. It only has two corresponding values and the tables keys can be in any order. This made drawing the segments impossible and forced us to change everything to an array list. Having an array list allowed for a much better data structure to store our dots and segments. Likewise, we fixed up the problem with having multiple dots being drawn on each other by completely altering the code that was provided and adding if statements for some error that was occurring when drawing the segments.
2. To support requests from the user we made everything a variable that can be easily changed in the command line argument. Likewise, utilized object orient programming to make it sure that if a class was not needed in our function, we were able to just not utilize the class and our code would not be impacted because of the nature of our composition in our code. Since only the classes that can be affected by invariants use composition, this means that our code can be manipulated and changed accordingly.
3. The testing activity that we used helped us a lot. Mainly we used a lot of our testing to check where and how we could have problems arise in our code. However, we didn’t utilize it enough when trying out our code, but this means that we can learn from our error and hopefully utilize it more in the final step of our code. Mainly, we used it to narrow down where the error in our code was and used it to narrow down the area that we needed to work on. Overall, we came to learn the importance of testing and we are now changing our mistakes and testing cases that can occur and other problems that may occur.

# PART 3

DRAW A CLASS DIAGRAM