This module, we learned how to use the Java conditional execution statements—if, if/else and if/else if/else statements. With this knowledge, our programs became more versatile and complex that allows us to create amazing filters like the Chromakey, where we replace specific color range with another set of colors. I have learned a lot from this module especially creative yet simple ideas that seems complicated at the perspective of a program user, but at a programmer’s perspective, they really are just simple if/else statements “under the hood.”

While I was thinking of creating a unique idea for Picture object filter methods, I thought it interesting how advanced photo editors like Adobe Photoshop manages to create such dynamic filters where you can just drag over your mouse to a certain location in the picture, then voila! It almost immediately transforms the picture right under the mouse pointer. I thought, if photoshop manages to create something this dynamic, I am certain it will be a huge amount of calculations and mental resource to implement a filter like that. But after giving it a thought, I realized something that somehow put me into perspective of how would a programmer would implement something like this. Ideas keeps popping into my head as soon as I realized it. “Maybe they somehow calculate for the distance between a point and the point under the mouse pointer, then loop over the radius to apply the filter surrounding the cursor?”, or “It wouldn’t be a great idea to use an array of coordinates to loop over on just to isolate certain pixels, its inefficient and tedious!”, then I thought “Only if there’s something that I can compare pixel coordinates to like a ‘magic formula’, where any coordinates it compares with, will give me the sort of “fork in the road” scenario that determines whether to apply a filter or not. Then I came up with an idea.

I chose to implement the vignette filter—filter that gradually dims the surrounding picture around a focal point. Of course, before, I had no idea nor the inclination to implement something seemingly complex, but after pondering about how photo editor filters work, it gave me a hint. First, using the Euclidean formula from the previous textbook exercises to calculate the distance between two colors, why not use it to calculate the distance between two points. Using the Euclidean distance formula, `d = sqrt([x1 - x2]^2 + [y1 – y2]^2)` will give us the distance between two points which always returns a positive value. This piece of information can single-handedly finish the vignette filter. With the module’s conditional execution statements, I can simply compare the distance between two points with a specified radius that I set. e.g., `if distance > 100, dim color of current pixel, else do nothing`. This piece of code is almost the entire logic that is needed to complete the filter. It just blew me away how easy it is to implement something so complex at the bigger picture.