Jake Felzien
Object Oriented
HW 5 Readme

The assignment that I chose to remap and improve upon was my first assignment with the input file reader project. Overall this assignment contained the usage of the Strategy pattern to read in different file types and then output matches within the given data. My goal for assignment five was to improve the coupling and cohesion within my first assignment while also adding some functionality to the assignment as well.

For this assignment I wished to add the decorator pattern to my assignment. See the attached UML for my design choices on this matter. My overall goal within the project was to add a decorator pattern that would allow for each strategy portion that read in different file types to be decorated with characteristics and objects.

The main motivation behind this lay in the fact that I wanted to generate and create a more general purpose data input reader and analyzer. While my additions were ultimately relatively simplistic I added several other ideas within the diagram and code, on how more elaborate ideas could really be useful within this design model. The first assignment was powerful in that it allowed for encapsulation and modularity within the input values and data that could be provided. We found that it was extremely advantageous to allow json and xml input within different strategy patterns. Now, for an overall data analysis tool my goal was to then create objects that would decorate each strategy.

The reason for this decoration is that upon the decision of an input strategy I figured that for a larger data processing tool it would be advantageous to add a decorator pattern to each strategy thus allowing for different features to be added to each strategy pattern. In the first assignment we were able to generate matches within given files, but for a more general purpose solution I was hoping to add a decorator pattern to allow for more widely used functions within data science.

These could include, and are not limited to, neural net training data sets, cosine similarity, nearest neighbor analysis, sorting or any other data science application that could be extremely modular. Therefore, with a given file, a processing query or selection could be used and selected by the user. The program is able to generate the correct strategy for the given file (as we did in assignment 1), but then moving forward the user is able to input the decorations in which he would like for the algorithm to have. Rather than finding matches, the algorithm could sort the data as well as display the nearest neighbor algorithm, or the cosine similarity between certain pairs of data. While we only implemented this whole thing for people input the beauty of this flexible system would be in the fact that different data files and data sets could be inputted and then the user could ask for the analysis or the algorithm that they wanted from the data file or data set, and then the program will decorate the strategy accordingly.

This was the goal of my assignment 5 as well as adding more robust tests and better code implementation of the second assignment through cleaner implementation choices and cleaner practices and cleaner design execution.