For project 3, our team chose an aspect-oriented design so that all four data structures would be easily implemented into the software. This design paradigm allows for the project to be modularized into four components for each of the data structures, allowing them to be implemented separately. Separate implementations of the distinct features helps greatly with testing and design since everything can be made essentially without affecting the rest of the program. The stack, for example, can be developed completely separately from the queue, linked list, and binary search tree. All simply funnel into the main interface, from which you can individually branch off into the different data structure visualizations.

In terms of the creation of the individual data structure functionalities, each structure has its own distinct set of methods that can be divided further into modules that describe the software as a whole. For example, the stack data structure has separate components for push, pop, and peek. These are modularized in the prototype and can be restructure to fit into the design of the other data structures further down the line. This cross-cutting of concerns is what led to us choosing an aspect-oriented design paradigm. It makes the overall design of the software easier because of the structure of the requirements we have for our project. Overall, the organization of our different components in our prototype will make it much simpler to implement the features we have planned for our final result because of the design paradigm we chose and the intersecting nature of the data structures we are visualizing.