Cyclomatic Complexity

We tried to explore different tools to calculate cyclomatic complexity, but were unable to integrate it easily. We decided to calculate the cyclomatic complexities of the different methods in the modern way:

> Assign one point to account for the start of the method
> Add one point for each conditional construct, such as an "if" condition
> Add one point for each iterative structure
> Add one point for each case or default block in a switch statement
> Add one point for any additional boolean condition, such as the use of && or ||
> With exceptions, you can add each throws, throw, catch or finally block as a single point when calculating the McCabe cyclomatic complexity metric.

The ideal McCabe complexity is 4 or 5, but 1-10 is considered well-structured and highly testable. For the sake of brevity, we will include the calculations for the Database file, as well as the number of methods with complexity over 10, and the percentage of that file's methods that are low complexity.

Overall, there were 5 complex and 1 very complex methods. This was due to list management and different cases that we had to explore. We could have split the method up further to reduce complexity, but we found the increased complexity to still be very readable and logical, so we let those methods have their higher complexity. The project has 83.3% low complexity, with an average cyclomatic complexity of 6.36 per method (when divided by methods and not by non-method sections as well, otherwise it would be 6.32)., which is not ideal, but it isn't too far off of well-structured code and still has high testability.

Java Files

Database: 3 complex, 1 very complex. 71% low complexity.

checkUser(String): Boolean	3 (start of the method, if statement, catch)
getLists(int userID, String listname): ArrayList<Info>	14 (start, 8 if/elseif, 2 while, 2 for, catch)
addToList(int userID, Boolean isRecipe, String listname, Info i): Boolean	21 (start, 11 if/elseif, 8 ||, catch)
removeFromList(int userID, Boolean isRecipe, String listname, Info i): Boolean	15 (start, 13 if/elseif, catch)
updateLists(int userID, Boolean add, String	3 (start, 2 if, 2 &&)

listname, Info i): Boolean	
getPrevSearch(int userID): ArrayList<Searches>	3 (start, while, catch)
addPrevSearch(int userID, String testSearch, int radius, int results): Boolean	3 (start, if, catch
changeOrder(int userID, String listname, Boolean isUp, int position): void	2 (start, if)
getrestID(int position, String listname, int userID): int	6 (start, 3 if/elseif, while, catch)
updatePos(int position, String listname, int userID, int rID, Boolean isRecipe): int	14 (start, 6 if/elseif, 6 &&, catch)
move(int userID, String listname, int posit, Boolean moveUp): void	4 (start, 3 if/elseif)
getrecipeID(int position, String listname, int userID): int	6 (start, 3 if/elseif, while, catch)
changeToDatabaseFormat(String listname): String	4 (start, 3 if/elseif)

DatabaseHelper: 0 complex. 100% low complexity.

findHighestPos(String listname, int userID): int	5 (start, 2 while, 2 if)
updateIndicesAfterRemove(String listname, int pos, int userID): void	4 (start, for, 2 if)
changeToDatabaseFormat(String listname): String	4 (start, 3 if/elseif)

ListServlet: 1 complex. 0% low complexity.

doPost(HttpServletRequest request, HttpServletResponse response): void	13 (start, throws, 3 if/ifelse, 3 &&, 4 cases, catch)

SearchServlet: 0 complex. 100% low complexity.

doGet(HttpServletRequest request, HttpServletResponse response): void	7 (start, 4 if, ||, throws)
getJSONResponse(String url): String	3 (start, while, catch)
restaurantSearch(String query, int numResults, int radius, List<Info> doNotShowList, List<Info> favoritesList): ArrayList<RestaurantInfo>	9 (start, 3 for, 3 if, while, catch)

SortLists: 0 complex. 100% low complexity

moveItemUp(ArrayList<Info> list, String itemToMove): ArrayList<Info>	3 (start, 2 if/elseif)
moveItemDown(ArrayList<Info> list, String itemToMove): ArrayList<Info>	3 (start, f2 if/elseif)

JS Files

ListClient: 0 complex. 100% low complexity

reorderResults(order, listName)	1 (start)

listPage: 1 complex. 0% low complexity

(not inside a function)	15 (10 if/elseif, 4 for, ||)

resultPage: 2 complex. 82% low complexity

(not inside a function)	3 (3 if, for)
load()	1 (start)
makeList()	3 (start, 4 if/elseif)
loadRestList()	19 (start, 11 if/elseif, 2 while, for, 3 &&, ||)
loadRecList()	18(start, 2 while, for, 10 if/elseif, 3 &&, ||)
createNumberedRestButton(start, end)	2 (start, for)
createNumberedRecButton(start, end)	2 (start, for)
createRestButton(value)	3 (start, 2 if/elseif)
createRecButton(value)	3 (start, 2 if/elseif)
drawRestList(begin,end)	5 (start, 2 for, 2 if)
drawRecList(begin,end)	5 (start, 2 for, 2 if)

All files not included have no methods or functionality that would fall under complexity.