3. WRITTEN RESPONSES (Created Independently)

Submit your responses to prompts 3a – 3d, which are described below. Your response to all prompts combined must not exceed 750 words (program code is not included in the word count). Collaboration is **not** allowed on the written responses.

3 a. Provide a written response that does all three of the following:

Approx. 150 words (for all subparts of 3a combined)

i. Describes the overall purpose of the program.

49 words

The overall purpose of the program is to simulate the process of building a sandwich from scratch consisting of a number of ingredients and layers, which is then rated on a scale of one through ten. The user is also encouraged to add a drink with their sandwich order.

ii. Describes what functionality of the program is demonstrated in the video.

71 words

The program first outputs ascii art in banner form, and the user types the help command in order to list the available options. Then the list command is utilized to display the ingredients included in the sandwich making process. The build command is then inputted and the user begins building their sandwich, and the sandwich is rated at the end. Lastly, the user selects a drink from the drink selection menu.

iii. Describes the input and output of the program demonstrated in the video.

33 words

The user inputs several commands such as help, build, list, and drink. This outputs prompts which the user responds to in regards to command options, the sandwich builder, ingredient choices, and drink choices.

3 b. Capture and paste two program code segments you developed during the administration of this task that contain a list (or other collection type) being used to manage complexity in your program.

Approx. 200 words (for all subparts of 3b combined, exclusive of program code)

i. The first program code segment must show how data have been stored in the list.

Drop Images Here

```
//Build and Price Sandwich Function
//String dictionary of ingredients and their values
public static Double build_sandwich(String ingredients, Scanner scanobject){
    Double price = 0.00;
    Map<String, Double> pricemap = Map.ofEntries(
        entry( k: "white bread", v: 2.00),
        entry( k: "wheat bread", v: 1.50),
        entry(k: "rosemary bread", v: 2.50),
        entry(k: "ham", v: 4.00),
        entry( k: "chicken", v: 3.50),
        entry( k: "roast beef", v: 5.00),
        entry(k: "cheddar", v: 1.00),
        entry(k: "colby jack", v: 1.50),
        entry(k: "mozzarella", v: 2.00),
        entry( k: "mayo", v: 1.00),
        entry( k: "ranch dressing", v: 1.50),
        entry( k: "guacamole", v: 2.50)
    );
```

ii. The second program code segment must show the data in the same list being used, such as creating new data from the existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

Drop Images Here

```
//mathematical operation to determine the overal price/rating of the sandwich built
try{
    price = pricemap.get(layer1) + pricemap.get(layer2) + pricemap.get(layer3) + pricemap.get(layer4) + pricemap.get(layer5);

}
catch (Exception e){
    System.out.println(ConsoleColors.YELLOW + "\nHmm.. you might want to double check if your ingredients are correct!" + ConsoleColors.RESET);
    System.out.println("Type " + ConsoleColors.BLUE + "\"build\"" + ConsoleColors.RESET + " to try again.");
}

return price;
```

Then, provide a written response that does all three of the following:

iii. Identifies the name of the list being used in this response.

8 words

The name of the dictionary is called *pricemap*.

iv. Describes what the data contained in the list represent in your program.

45 words

The dictionary, *pricemap*, contains numerical data in the form of doubles, which are assigned to certain sandwich ingredient values in order to calculate the overall rating of the sandwich on a scale of 1-10 based on the added value of the ingredients used in tandem.

v. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list.

50 words

If I did not use the dictionary dataset, *pricemap*, in my program code, I would not be able to check whether the user inputted ingredient existed, which is vital in order to determine whether the user is inputting a valid ingredient for the final rating calculation of their completed sandwich.

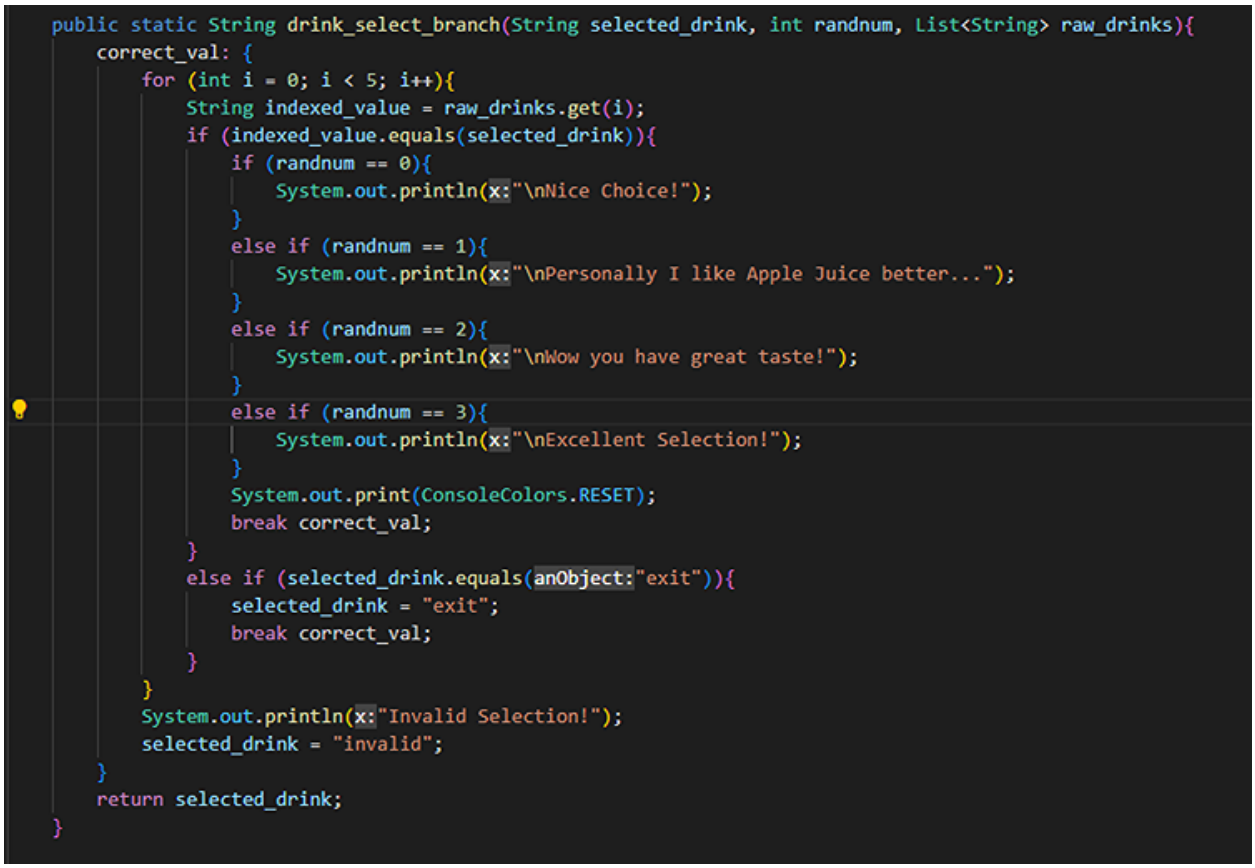
3 c. Capture and paste two program code segments you developed during the administration of this task that contain a student-developed procedure that implements an algorithm used in your program and a call to that procedure.

Approx. 200 words (for all subparts of 3c combined, exclusive of program code)

i. The first program code segment must be a student-developed procedure that:

- Defines the procedure's name and return type (if necessary)
- Contains and uses one or more parameters that have an effect on the functionality of the procedure
- Implements an algorithm that includes sequencing, selection and iteration

Drop Images Here



```
public static String drink_select_branch(String selected_drink, int randnum, List<String> raw_drinks){
    correct_val: {
        for (int i = 0; i < 5; i++){
            String indexed_value = raw_drinks.get(i);
            if (indexed_value.equals(selected_drink)){
                if (randnum == 0){
                    System.out.println(x:"\nNice Choice!");
                }
                else if (randnum == 1){
                    System.out.println(x:"\nPersonally I like Apple Juice better...");
                }
                else if (randnum == 2){
                    System.out.println(x:"\nWow you have great taste!");
                }
                else if (randnum == 3){
                    System.out.println(x:"\nExcellent Selection!");
                }
                System.out.print(ConsoleColors.RESET);
                break correct_val;
            }
            else if (selected_drink.equals(anObject:"exit")){
                selected_drink = "exit";
                break correct_val;
            }
        }
        System.out.println(x:"Invalid Selection!");
        selected_drink = "invalid";
    }
    return selected_drink;
}
```

ii. The second program code segment must show where your student-developed procedure is being called in your program.

Drop Images Here

```
//Drink Selection Function
public static String drink_select(String drinks, List<String> raw_drinks, Scanner scanobject){
    int randnum = ThreadLocalRandom.current().nextInt(origin:0, 3 + 1);
    String selected_drink = "invalid";
    System.out.println(drinks);
    System.out.println(ConsoleColors.YELLOW + "Choose a drink from the list to enjoy!\n" + ConsoleColors.RESET);
    while (selected_drink.equals(anObject:"invalid")){
        System.out.print(ConsoleColors.RED + "Select Drink: " + ConsoleColors.RESET);
        selected_drink = scanobject.nextLine();
        //check using iteration if selected_drink value exists in the raw_drinks list
        selected_drink = drink_select_branch(selected_drink, randnum, raw_drinks);
    }
    return selected_drink;
}
```

Then, provide a written response that does both of the following:

iii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program.

63 words

The procedure essentially checks whether or not the selected drink exists in the list, then provides randomly selected feedback out of four different feedback options as a fun comment to complement their drink choice. The procedure contributes to the overall functionality of the program by allowing users to pick a drink to complement their sandwich of choice to complete a full virtual meal.

iv. Explains in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

84 words

alignment, normal,

The algorithm first works by iterating through a string array containing five different drink choices, which it checks user inputted drink selections against. If the user inputted *selected_drink* value matches an item from the string array, the program will then use Java's *ThreadLocalRandom* function to generate a random integer value from zero to three. After this, the program will then check against the random number from zero through three to select a prewritten statement to print, complementing the drink choice made by the user.

3 d. Provides a written response that does all three of the following:

Approx. 200 words (for all subparts of 3d combined)

i. Describes two calls to the procedure identified in written response 3c. Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.

First call:

39 words

alignment, normal,

The first call passes a valid drink parameter such as "Milk", as a string called *selected_drink* into the *drink_select_branch* function, along with the static *randnum* and *raw_drinks* parameters. This in turn executes the first portion of the if/else clause.

Second call:

39 words

alignment, normal,

The second call passes the exit parameter in the form of the string, “exit”, called *selected_drink* into the *drink_select_branch* function, along with the static *randnum* and *raw_drinks* parameters. This in turn executes the second portion of the if/else clause.

ii. Describes what condition(s) is being tested by each call to the procedure.

Condition(s) tested by first call:

42 words

alignment, normal,

The first call tests whether or not the current indexed value in the string array, *raw_drinks* matches the value passed into the *selected_drink* parameter, such as “Milk”. If so, continue with the first portion of the if/else clause regarding valid drink selection.

Condition(s) tested by second call:

64 words

alignment, normal,

The second call first tests whether or not the current indexed value in the string array, *raw_drinks* matches the value passed into the *selected_drink* parameter, such as “exit”. If not, continue to the second portion of the if/else clause regarding the exit procedure. This portion then tests if the *selected_drink* parameter is equal to the string “exit”. If so, continue with the exit procedure.

iii. Identifies the result of each call.

Result of the first call:

89 words

alignment, normal,

The first call results in the continuation of the first portion of the if/else clause where another if/else clause tests which integer value the *randnum* variable is equal to. Depending on the value from zero to three, the algorithm will print a prewritten statement on the terminal to display to the user. The for-loop is then broken to halt further iteration of the *raw_drinks* array. After, the program will then return the *selected_drink* value as it was passed originally into the *drink_select_branch* function to use elsewhere in the program.

Result of the second call:

74 words

alignment, normal,

The second call results in the execution of the second portion of the if/else clause where the *selected_drink* variable is assigned the string, “exit”. The for-loop is then broken to halt further iteration of the *raw_drinks* array. After, the program will then return the *selected_drink* value, which in turn is used to break out of the drink selection loop, and back into the main while-loop where the main options for user interaction are displayed.

Total words: 750