ITSC 1212 Module 13/14 Lab – Arrays

In this lab you'll build up a small food ordering application that uses arrays to better understand and get practice with declaring, creating/instantiating, and using arrays in a realistic context. This example will use a food order as a way of tracking some user choices.

Concepts covered in this lab:

- Array declaration and creation/instantiation
- Array traversal for assigning values
- Array traversal for retrieving values
- Using different iteration/loops with arrays
- Combining flow control (sequence, decision, iteration) in using arrays

Part A: Setup your Business – Establish

1. Start by deciding the following:
    a. The name of your food service that includes your name and add something to the end like "Deli" or "Pizza Shack" or "Food Cart". For example, your business might be named "Bob's Deli". Note that it should be the kind of service that aligns with the food you want to serve.
    b. Ten products that you'll have on your menu, for example, Large drink, Medium drink, Meat taco, Veggie taco, Chips, etc. You can go over ten if you want, but don't go nuts.
2. Set up your business:
    a. Create a new class named Business.java.
    b. Create a main method.
    c. Create a static String field and assign it your company name*.
    d. Create a static String field and assign it to your name*.
    e. Create a static String array field containing all your menu items*. Populate this array with the names of the items you serve. Don't get too elaborate; singular nouns ("Taco" or "Hamburger") are sufficient for now.
    f. Create a static int field that contains the number of items in your menu.

3. Set up your application (no:
    a. Define a String called **decorativeLine** and assign to it about 40 or so special characters (like ################ or ++++++++++ or **********).
    b. Print decorativeLine as the first thing and last thing in your main method. Put a line break after the first print, and before the last so it will surround the output of your program without being too close.
    c. Print out a welcome that uses your name and the business name with good punctuation/formatting.
    d. Ask the user for their name using your name in the prompt message, get the user's name, then greet them using their name. Test and fix as necessary.

e. Add some Thread.sleep() statements (test with 1000 and adjust as necessary) so your output doesn't just appear suddenly on the screen. Slow it down (but not too slow!) so it's more like a conversation with a real person would be. Review the previous lab if you don't remember how to use Thread.sleep() and make your program pause.
f. One example of this interaction is shown below but feel free to change the verbiage, characters, etc., so long as your output is clean and properly punctuated.

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Welcome to Andy's Pizza Joint!
I'm Andy. What's your name? Icabad
Hey there, Icabad!

<rest of program input/output here>

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

* You must use these fields when you print or display this information. You may not hard-code this information into print statements.

4. When you are satisfied with your code, show an instructor or a TA to receive credit for this section

Part B: Create an Array to Display Menu - Basic array traversal

1. Add a message that addresses the user by name and asks them to have a look at your menu. For example, "Here's what we serve, Felecia. Have a look:"
2. Using an enhanced for loop (i.e., a for-each loop), display all the menu items numbered 1 to 10 (or however many you included in your menu) not 0 to 9. Zero-based indexing might make sense to someone who understands programming but from a user perspective humans are more familiar with numbering items starting with 1.
3. Feel free to add Thread.sleep() statements, new lines, and tabs to space your displays vertically and horizontally so they're easier to read.
4. Your application should now welcome the user, introduce you, get the user's name, greet them by name, then display the menu. Test the program.
5. When you are satisfied with your code, show an instructor or a TA to receive credit for this section

ITSC 1212 Module 13/14 Lab – Arrays

Part C: Create an Array to Hold Order - Populate order array

1. Now you need to get the customer's order:
    a. Using an appropriate prompt, find out how many items they want to order and store that value.
    b. Create an order array that will hold that many items. (A limitation of arrays is that you have to know how many items are in an array in order to create it.)
2. Using a for loop, ask the user to specify each item they want by entering the item number from your original menu display.
    a. Prompt them with a display such as "Please enter the number of your first item:"
    b. Accept the integer item number from the user on the same line.
    c. Store that item String in your order array. Remember to fill your order array starting at index 0.
3. Once you've collected the order, thank the user by name then list the items they're ordering. Use a for-each loop to confirm their order by looping thru the order array but print the items on the same line separated by commas, for example: "Taco, Nachos, Soft Drink". Make sure the last item isn't followed by a comma.
4. Thank the user by name, tell them something about how their order is coming right up, and to have a great day.
5. When you are satisfied with your code, show an instructor or a TA to receive credit for this section

Part D: Allow for incomplete order with decision/conditionals

1. Add validation to the order process. Use a while loop to control the user entering item numbers for their order. Don't exit the loop until a valid item number has been entered. If the user enters an invalid item number, display a message reminding them of the item numbers (something like "You must enter an item number from 1 to 10."). If you are looking for inspiration on how to accomplish this you can look back at our Blackjack game. We did the exact thing when we asked the player if they wanted to hit or stand.
2. When you are satisfied with your code, show an instructor or a TA to receive credit for this section

Part E: Convert order items from Strings to Item objects

In this part we want to practice some more applying object-oriented programming concepts and declare our own datatype (class) and utilize encapsulation. Remember that encapsulation is where the data (instance variables) and the code acting on the data (methods) are wrapped together into a single unit and the implementation details are hidden.

ITSC 1212 Module 13/14 Lab – Arrays

When we declare our own datatype (class) to represent an item we can it to create specific instances (objects) that keep track of the information (state) of each menu item our application wants to offer. Encapsulation also gives us the ability to easily track multiple properties (fields/attributes) about a menu item as apposed to a single property using an array as we have done for the name of the items.

1. Create an item class with:
    a. Instance variables for:
        - Item id
        - Item name
        - Item price
    b. A constructor that takes these three values as formal parameters and assigns them to the instance variables
    i.e. **`public ClassName(parameterType1 _parameterVariableName1, parameterType2 _parameterVariableName2...)`**
    c. Getter/accessor methods for each of those fields
    i.e. **`public double getVariableName() { return variableName; }`**
    d. A toString method that creates and returns a String that look something like:
    Item ID: # >> Mini Cheese Pizza >> $4.95
    Where # is the item number, Mini Cheese… is the name, and 4.95 is the price.
2. Back in your Business class construct 10 + items within array of items:

```
Item[] menuItemsList = new Item[numberOfItemsYouWant];
menuItemsList[0] = new Item("A1", "Mac 'n Cheese", 4.95);
menuItemsList[1] = new Item("A2", "Large Drink", 1.99);
```

   … repeat for each item, updating the index until you do all of them. DO NOT COPY THIS CODE. Type it yourself. Why, you ask? For one, it's the best way to learn. For two, google docs and word often convert simple quotes into fancy quotes (“ ”) which will cause an error every time quotes appear, because fancy quotes are not actually quotation marks to the java compiler.

   Note that you could combine the declaration and the instantiation of the Items into one line, but it's more complicated and prone to errors. By putting them each on their own line it's easier to follow the formatting and see what's happening.
3. As with the previous sections, you want to now display the menu. Because you made a handy-dandy toString method to show the key information in a nice string representation, inside your loop you can just put

```
System.out.println(menuItemsList[counter]);
```

   And the toString will do the nice formatting for you.

4. Now, as before, prompt the user for the number of items they'd like to order, setup the order list (Item[] orderedItemsList = new Items[numberOfItemsDesired]) and then loop thru that array, asking the user what they'd like.
    a. This becomes trickier, because we're no longer just picking an item from an array (i.e. item 3, or 7) but rather we need to figure out which of the 10+ items their input matches. This means that WITHIN the loop we already have where we ask the user for their order, we need to loop thru the menu and find a matching item, and if there is none, tell them. We'll break this into those two steps:
    b. Take the user input, and immediately after make another loop that goes (iterates) thru the menuItemsList from start to end and using the String equalsIgnoreCase method compares their input to the id for each item….

```
String input = aScanner.nextLine();
// after asking for their item choice
input = input.trim(); // don't want spaces on either side
// get the id from this one
String thisItemId = menuItemsList[innerCounter].getId();

if(input.equalsIgnoreCase(thisItemId) // then it's a match!
{
   // the one in this iteration
   // assign this to their order
   orderedItemsList[counter] = menuItemsList[innerCounter];
}
```

    c. Now iterate thru their order using a for each loop, and simply calling the toString (i.e. **`System.out.println(item)`**)
5. When you are satisfied with your code, show an instructor or a TA to receive credit for this section.

Part F: Making it Real - do the math!

1. We have the price of each item as an item field (instance variable) but we're not using it. In a more realistic scenario we could have more details about and order such as taxes, a boolean for preferred status, etc… but for now we'll keep it simple. We just need the total of their order.
2. At this point we have, after getting the users order, a smaller array of the items they ordered. Each item has a price, and we need to get the total of all the items they ordered.
3. Create a variable to hold the total bill just before you summarize their order.
4. Within the loop where you are doing that summary, increment the total by adding the cost of teach item (**`i.e. total += item.getPrice();`**)

ITSC 1212 Module 13/14 Lab – Arrays

5. After the summary/loop include a nice bottom line by printing a bunch of underscores, then TOTAL with some tabs (/t) and then put the accumulated total.
6. When you are satisfied with your code, show an instructor or a TA to receive credit for this section

Bonuses:

- There are a number of things you could do to expand on or make this more realistic. A few include:
    - Allow the user to cancel their order by entering x or a blank
    - Figure out tax, with a subtotal and grand total
    - Have a taxable item boolean field for the items, and then use that to calculate the tax (since some items will be taxable and others will not)

Make it impossible to crash (i.e. what happens when users put in something other than a number and you only want a number) - Prompt the user if their entry is invalid for a valid entry or allow them to cancel. You'll deal more with this sort of thing in the next