Lab 08: Array of Objects
CSE-021-01, Spr 2018 **Points: 20**

Overview

We will see the power of Object-Oriented Programming (OOP) when we want to create a program that behaves similar to what we have done before (code reuse). This lab will not be limited to just 3 specialty cheeses but any amount set by the user. Luckily, this does not change anything for the `Cheese` class, and we just have to create a shop that contains an array of items to sell, which we will call `ShopArr`.

Before you get started, read chapters 7.10, 7.11, and 7.12. Answer activities 7.10.4, 7.11.3 and 7.12.2 in a Word or text file. Also, answer the Assessment questions as you encounter them in the next section. The prompts for answering assessment questions are placed immediately following the material to which the listed questions relate.

Getting Started

After following the import instructions in the assignment page, you should have a Java project in Eclipse titled Lab 21_8. This PDF document is included in the project in the **doc** directory. The Java files you will use in this lab are in the **src** directory.

Copy over `RunShop.java` and `Cheese.java` from the previous lab. Your program must produce an output matching the sample output shown below.

Sample Output:

```
We sell 0 kinds of Cheese
Sub Total: $0.0
-Discount: $0.0
Total:     $0.0
----------------------------------------------------------------------------------------
We sell 1 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Enter the amount of Dalaran Sharp: 1
Display the itemized list? (1 for yes) 1
1 lbs of Dalaran Sharp @ 1.25 = $1.25

Sub Total:   $1.25
- Discount:  $0.0
Total Price: $1.25
----------------------------------------------------------------------------------------
We sell 2 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Reading : Chapter 7.10, 7.11 & 7.12
Stormwind Brie: $10.0 per pound
Enter the amount of Dalaran Sharp: 1
Enter the amount of Stormwind Brie: 1
Display the itemized list? (1 for yes) 1
1 lbs of Dalaran Sharp @ 1.25 = $1.25
1 lbs of Stormwind Brie @ 10.0 = $10.0

Sub Total:   $11.25
- Discount:  $0.0
Total Price: $11.25
----------------------------------------------------------------------------------------
We sell 3 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Stormwind Brie: $10.0 per pound
```

```
Alterac Swiss: $40.0 per pound
Enter the amount of Dalaran Sharp: 1
Enter the amount of Stormwind Brie: 1
Enter the amount of Alterac Swiss:  1
Display the itemized list? (1 for yes) 1
1 lbs of Dalaran Sharp @ 1.25 = $1.25
1 lbs of Stormwind Brie @ 10.0 = $10.0
1 lbs of Alterac Swiss @ 40.0 = $40.0

Sub Total:   $51.25
- Discount:  $10.0
Total Price: $41.25
---------------------------------------------------------------------------------------------
We sell 4 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Stormwind Brie: $10.0 per pound
Alterac Swiss: $40.0 per pound
Cheese Type D: $9.15 per pound
Enter the amount of Dalaran Sharp: 1
Enter the amount of Stormwind Brie: 1
Enter the amount of Alterac Swiss: 1
Enter the amount of Cheese Type D: 1
Display the itemized list? (1 for yes) 1
1 lbs of Dalaran Sharp @ 1.25 = $1.25
1 lbs of Stormwind Brie @ 10.0 = $10.0
1 lbs of Alterac Swiss @ 40.0 = $40.0
1 lbs of Cheese Type D @ 9.15 = $9.15

Sub Total:   $60.4
- Discount:  $10.0
Total Price: $50.4
---------------------------------------------------------------------------------------------
We sell 4 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Stormwind Brie: $10.0 per pound
Alterac Swiss: $40.0 per pound
Cheese Type D: $9.15 per pound
Enter the amount of Dalaran Sharp: 0
Enter the amount of Stormwind Brie: 0
Enter the amount of Alterac Swiss: 0
Enter the amount of Cheese Type D: 0
Display the itemized list? (1 for yes) 1

Sub Total:   $0.0
- Discount:  $0.0
Total Price: $0.0
---------------------------------------------------------------------------------------------
We sell 10 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Stormwind Brie: $10.0 per pound
Alterac Swiss: $40.0 per pound
Cheese Type D: $9.15 per pound
Cheese Type E: $2.5 per pound
Cheese Type F: $8.74 per pound
Cheese Type G: $9.88 per pound
Cheese Type H: $2.91 per pound
Cheese Type I: $6.66 per pound
Cheese Type J: $0.36 per pound
Enter the amount of Dalaran Sharp: 1
Enter the amount of Stormwind Brie: 1
Enter the amount of Alterac Swiss: 1
```

```
Enter the amount of Cheese Type D: 1
Enter the amount of Cheese Type E: 1
Enter the amount of Cheese Type F: 1
Enter the amount of Cheese Type G: 1
Enter the amount of Cheese Type H: 1
Enter the amount of Cheese Type I: 1
Enter the amount of Cheese Type J: 1
Display the itemized list? (1 for yes) 1
1 lbs of Dalaran Sharp @ 1.25 = $1.25
1 lbs of Stormwind Brie @ 10.0 = $10.0
1 lbs of Alterac Swiss @ 40.0 = $40.0
1 lbs of Cheese Type D @ 9.15 = $9.15
1 lbs of Cheese Type E @ 2.5 = $2.5
1 lbs of Cheese Type F @ 8.74 = $8.74
1 lbs of Cheese Type G @ 9.88 = $9.88
1 lbs of Cheese Type H @ 2.91 = $2.91
1 lbs of Cheese Type I @ 6.66 = $6.66
1 lbs of Cheese Type J @ 0.36 = $0.36

Sub Total:   $91.44999999999999
- Discount:  $10.0
Total Price: $81.44999999999999
--------------------------------------------------------------------------------
We sell 10 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Stormwind Brie: $10.0 per pound
Alterac Swiss: $40.0 per pound
Cheese Type D: $9.15 per pound
Cheese Type E: $2.5 per pound
Cheese Type F: $8.74 per pound
Cheese Type G: $9.88 per pound
Cheese Type H: $2.91 per pound
Cheese Type I: $6.66 per pound
Cheese Type J: $0.36 per pound
Enter the amount of Dalaran Sharp: 1
Enter the amount of Stormwind Brie: 1
Enter the amount of Alterac Swiss: 1
Enter the amount of Cheese Type D: 1
Enter the amount of Cheese Type E: 1
Enter the amount of Cheese Type F: 1
Enter the amount of Cheese Type G: 1
Enter the amount of Cheese Type H: 1
Enter the amount of Cheese Type I: 1
Enter the amount of Cheese Type J: 1
Display the itemized list? (1 for yes) 0

Sub Total:   $91.44999999999999
- Discount:  $10.0
Total Price: $81.44999999999999
--------------------------------------------------------------------------------
We sell 10 kinds of Cheese
Dalaran Sharp: $1.25 per pound
Stormwind Brie: $10.0 per pound
Alterac Swiss: $40.0 per pound
Cheese Type D: $9.15 per pound
Cheese Type E: $2.5 per pound
Cheese Type F: $8.74 per pound
Cheese Type G: $9.88 per pound
Cheese Type H: $2.91 per pound
Cheese Type I: $6.66 per pound
Cheese Type J: $0.36 per pound
```

```
Enter the amount of Dalaran Sharp : 1
Enter the amount of Stormwind Brie : 0
Enter the amount of Alterac Swiss : 2
Enter the amount of Cheese Type D : 0
Enter the amount of Cheese Type E : 3
Enter the amount of Cheese Type F : 0
Enter the amount of Cheese Type G : 4
Enter the amount of Cheese Type H : 0
Enter the amount of Cheese Type I : 5
Enter the amount of Cheese Type J : 0
Display the itemized list? (1 for yes) 1
1 lbs of Dalaran Sharp @ 1.25 = $1.25
2 lbs of Alterac Swiss @ 40.0 = $80.0
3 lbs of Cheese Type E @ 2.5 = $7.5
4 lbs of Cheese Type G @ 9.88 = $39.52
5 lbs of Cheese Type I @ 6.66 = $33.3
Sub Total:   $161.57
- Discount:  $25.0
Total Price: $136.57
```

Part 1: Modify `RunShop.java` (version 2)

In the original `RunShop.java` (from previous lab), we are creating an instance of `Shop` class named `shop`. Then we call the method `run()` of this `shop` object. Now we no longer have an object or class called `Shop` and instead, we will be using `ShopArr`. Import `RunShop.java` from the previous lab to do this step.

[Answer assessment question 1]

Make the change in `RunShop.java` file which will allow you to run the newest cheese shop. (same as answer to Q1)

Part 2: Fill-in `ShopArr.java`

Everywhere you see comment to "Fill in Code" is where you need to add code to make this program behave correctly. If it says "Fix Code" you need to change existing code. In most places a sample is provided to help you get started. The program currently runs but the behavior is obviously incorrect.

Here is a simpler version of `intro()` as reference

```
public static void intro(String[] names, double[]prices, int[] amounts) {
   // Special 3 Cheeses
   if (names.length > 0) {
      names[0] = "Dalaran Sharp";
      prices[0] = 1.25;
   }
   if (names.length > 1) {
      names[1] = "Stormwind Brie";
      prices[1] = 10.00;
   }
   if (names.length > 2) {
      names[2] = "Alterac Swiss";
      prices[2] = 40.00;
   }

   Random ranGen = new Random(100);
   System.out.println("We sell " + names.length + " kinds of cheese");
   if (names.length > 0)
      System.out.println(names[0] + ": $" + prices[0] + " per pound");
```

```
        if (names.length > 1)
            System.out.println(names[1] + ": $" + prices[1] + " per pound");
        if (names.length > 2)
            System.out.println(names[2] + ": $" + prices[2] + " per pound");

        for (int i = 3; i < names.length; i++) {
            names[i] = "Cheese Type " + (char)('A'+i);
            prices[i] = ranGen.nextInt(1000)/100.0;
            amounts[i] = 0;
            System.out.println(names[i] + ": $" + prices[i] + " per pound");
        }
    }
```

We must now split this method into two methods for this lab: `init()` and `intro()`. The reason is because we want to repeat `intro()` if needed, but creating objects using `init()` needs to be done only once. So all the `println` statements will be moved to `intro()`. Now take a look at `ShopArr.java` and the `init()` method in it. We see the code is very similar but the variables have changed. The very first thing we do is create the array of `Cheese` pointers since we are given the argument `max`.

```
// Create max number of Cheese pointers
cheese = new Cheese[max];
```

Then you will see the code to handle Sharp cheese:

```
if (max > 0) {
    cheese[0] = new Cheese();
    cheese[0].setName("Dalaran Sharp");
    cheese[0].setPrice(1.25);
}
```

Instead of `names.length`, we now use `max` in this code.

[Answer assessment question 2]

Also, instead of using three different arrays (`names`, `prices` and `amounts`), we now have only 1 array of cheeses. So the code is changed to use `cheese[0]` which points to the first `Cheese` object in the array. If we want to change the name then we use a mutator `setName`, which exists inside the Cheese object (`cheese[0]`), that we access using the "`.`" operator. We instantiate Sharp using default constructor with 0 arguments followed by invoking two mutators. Brie is instantiated with a 1-argument constructor followed by invoking only one mutator. And finally, Swiss uses a 2-argument constructor, so no mutator calls are necessary. Note that you must use the corresponding accessor method calls in other parts of the code to get to the value of the variables set by the mutators.

[Answer assessment questions 3, 4 and 5]

Now you will need to implement the for-loop in `init()`. The original loop is as follows:

```
for (int i = 3; i < names.length; i++) {
    names[i] = "Cheese Type " + (char)('A'+ i);
    prices[i] = ranGen.nextInt(1000)/100.0;
    amounts[i] = 0;
}
```

You must figure out the transformations needed in the for-loop to work with a single `cheese` array instead of 3 arrays, as shown by the code already inside `init()`. You can assume the `amount` is already set to `0` for each cheese so the loop doesn't need to set it again to `0`.

[Answer assessment question 6]

We have also implemented the basic version of `ShopArr` constructor which just invokes `init` method with a fixed number, `10`. You must fill-in the 1 argument constructor which will invoke init using the `max` parameter instead.

[Answer assessment question 7]

Now implement `intro(Scanner)`, `calcSubTotal()` and `itemizedList()` so they work for an array of cheese pointer.

[Answer assessment questions 8 and 9]

Part 3: Modify `RunShop.java` (version 3)

Now notice that there are two constructors for `ShopArr` available and version 2 of `RunShop` is only calling the default constructor with no arguments. So, the program will always create 10 cheeses to sell. We will need to make use of the second constructor which takes an argument `max` and sets the amount of cheeses to sell.

Modify `RunShop` so it asks the following question to the user and then pass the number user enters to the `ShopArr` constructor. So the program now starts as follows:

```
Enter the number of Cheese :12
We sell 12 types of Cheese
```

Everything should work as it did before, now you can just change the number of cheese from 10 to any amount you want (including `0` but not negatives).

[Answer assessment question 10]

Part 4: (Assessment) Logic Check and Level of Understanding

1) What are the minimal changes required to instantiate `ShopArr` and invoke `run()` on it?
2) We can also use a `<something>.length` instead of `max`. What is the valid `<something>` to use in `ShopArr.java`?
3) How can we tell which instantiation (`new Cheese`) corresponds to which constructor definition inside the `Cheese` class?
4) How can we identify a mutator method call?
5) What would be the result if we added this line right after `Swiss` is created:
   `cheese[2].setName("Wrong Name");` ?
6) Why is the `init()` method both private and void?
7) What are the distinguishing features of constructor methods? (i.e., How do we tell them apart from other methods?)
8) How can we figure out the number of required iterations for each loop?
9) Should we pass in `Cheese` array pointer (`cheese[]`) as arguments into `calcSubTotal` or `itemizedList`? (Why or why not)
10) What value will be printed by `RunShop` for "`Ran with Cheese Total`"? (fixed number or a formula)

What to hand in

When you are done with this lab assignment, submit all your work through CatCourses.

Before you submit, make sure you have done the following:

- Verified your solution with your TA or instructor
- Included answers to activities 7.10.4, 7.11.3, and 7.12.2, and Assessment questions (1 – 10) in a Word document or text file named `Part4`
- Attached the `ShopArr.java`, `RunShop.java` and `Part4` files
- Filled in your collaborator's name in the "Comments…" text-box (if any) at the submission page.