Lab 08: Array of Objects
CSE-021-01 **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 to 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 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 **Lab21_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 runs given below.

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] = " Humboldt Fog";
      prices[0] = 25.00;
   }
   if (names.length > 1) {
      names[1] = "Red Hawk ";
      prices[1] = 40.50;
   }
   if (names.length > 2) {
      names[2] = "Teleme";
      prices[2] = 17.25;
   }
```

```
        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 Humboldt Fog cheese:

```
if (max > 0) {
    cheese[0] = new Cheese();
    cheese[0].setName("Humboldt Fog");
    cheese[0].setPrice(25.00);
}
```

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 Humboldt Fog using default constructor with 0 arguments followed by invoking two mutators. Red Hawk is instantiated with a 1-argument constructor followed by invoking only one mutator. And finally, Teleme 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)`, `itemizedList()`, `calcSubTotal()` and `discountSpecials()` so they work for an array of cheese pointer. The methods `printSubTotals()` and `printFinalTotal()` should be identical to Lab 07.

[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` (from Part 1) 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 Cheeses for shop setup: 12
We sell 12 types of Cheese (in 0.5 lb packages)
```

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

Create a Word document or text file named `Part4` that contains answers to the following:

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 `Teleme` is created:
   `cheese[2].setName("Wrong Name");` ?
6) Why is the `init()` method both private and void?
7) What is the difference between `ShopArr()` and `ShopArr(int max)` constructors.
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)

Sample Runs (user input shown in green, with each run separated by a dashed-line):

```
--------------------------------------------------------------------------------SAMPLE RUN 1
 Enter the number of Cheeses for shop setup: 0
 We sell 0 kinds of Cheese (in 0.5 lb packages)


 Display the itemized list? (1 for yes): 1
 No items were purchased.

 Original Sub Total:               $0.00
 Specials...
 None                             -$0.0
 New Sub Total:                    $0.00
 Additional 0% Discount:          -$0.0
 Final Total:                      $0.00

 Do you wish to redo your whole order? (1 for yes): 0

 Thanks for coming!
 Ran with Cheese Total: 0
--------------------------------------------------------------------------------SAMPLE RUN 2
 Enter the number of Cheeses for shop setup: 1
 We sell 1 kinds of Cheese (in 0.5 lb packages)
 Humboldt Fog: $25.0 per pound

 Enter the amount of Humboldt Fog in lb: 1

 Display the itemized list? (1 for yes): 1
 1.0 lb of Humboldt Fog @ $25.00 = $25.00

 Original Sub Total:               $25.00
 Specials...
 Humboldt Fog (Buy 1 Get 1 Free): -$12.50
 New Sub Total:                    $12.50
 Additional 0% Discount:          -$0.0
 Final Total:                      $12.50

 Do you wish to redo your whole order? (1 for yes): 1

 We sell 1 kinds of Cheese (in 0.5 lb packages)
 Humboldt Fog: $25.0 per pound

 Enter the amount of Humboldt Fog in lb: 0

 Display the itemized list? (1 for yes): 1
 No items were purchased.

 Original Sub Total:               $0.00
 Specials...
 None                             -$0.0
 New Sub Total:                    $0.00
 Additional 0% Discount:          -$0.0
 Final Total:                      $0.00

 Do you wish to redo your whole order? (1 for yes): 0

 Thanks for coming!
 Ran with Cheese Total: 1
```

```
--------------------------------------------------------------------------------SAMPLE RUN 3
 Enter the number of Cheeses for shop setup: 4
 We sell 4 kinds of Cheese (in 0.5 lb packages)
 Humboldt Fog: $25.0 per pound
 Red Hawk: $40.5 per pound
 Teleme: $17.25 per pound
 Cheese Type D: $9.15 per pound

 Enter the amount of Humboldt Fog in lb: 4.5
 Enter the amount of Red Hawk in lb: 4.5
Enter the amount of Teleme in lb: -3
Invalid input. Enter a value >= 0: 1.7
Invalid input. Enter a value that's multiple of 0.5: -4
Invalid input. Enter a value >= 0: 1.9
Invalid input. Enter a value that's multiple of 0.5: 1.5
Enter the amount of Cheese Type D in lb: 10

Display the itemized list? (1 for yes): 1
4.5 lb of Humboldt Fog @ $25.00 = $112.50
4.5 lb of Red Hawk @ $40.50 = $182.25
1.5 lb of Teleme @ $17.25 = $25.88
10.0 lb of Cheese Type D @ $9.15 = $91.50

Original Sub Total:            $412.13
Specials...
Humboldt Fog (Buy 1 Get 1 Free): -$50.00
Red Hawk (Buy 2 Get 1 Free):  -$60.75
New Sub Total:                  $301.38
Additional 15% Discount:      -$75.34
Final Total:                    $226.03

Do you wish to redo your whole order? (1 for yes): 0

Thanks for coming!
Ran with Cheese Total: 4
--------------------------------------------------------------------------------SAMPLE RUN 4
 Enter the number of Cheeses for shop setup: 12
 We sell 12 kinds of Cheese (in 0.5 lb packages)
 Humboldt Fog: $25.0 per pound
 Red Hawk: $40.5 per pound
 Teleme: $17.25 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
Cheese Type K: $2.88 per pound
Cheese Type L: $7.23 per pound

Enter the amount of Humboldt Fog in lb: 1.5
Enter the amount of Red Hawk in lb: 2.2
Invalid input. Enter a value that's multiple of 0.5: 2.5
Enter the amount of Teleme in lb: 0
Enter the amount of Cheese Type D in lb: 0
Enter the amount of Cheese Type E in lb: 0
Enter the amount of Cheese Type F in lb: 0
Enter the amount of Cheese Type G in lb: 8
Enter the amount of Cheese Type H in lb: 10
Enter the amount of Cheese Type I in lb: 0
```

```
Enter the amount of Cheese Type J in lb: 0
Enter the amount of Cheese Type K in lb: 0
Enter the amount of Cheese Type L in lb: 0

Display the itemized list? (1 for yes): 1
1.5 lb of Humboldt Fog @ $25.00 = $37.50
2.5 lb of Red Hawk @ $40.50 = $101.25
8.0 lb of Cheese Type G @ $9.88 = $79.04
10.0 lb of Cheese Type H @ $2.91 = $29.10

Original Sub Total:            $246.89
Specials...
Humboldt Fog (Buy 1 Get 1 Free): -$12.50
Red Hawk (Buy 2 Get 1 Free):  -$20.25
New Sub Total:                  $214.14
Additional 10% Discount:      -$21.41
Final Total:                    $192.73

Do you wish to redo your whole order? (1 for yes): 1

We sell 12 kinds of Cheese (in 0.5 lb packages)
Humboldt Fog: $25.0 per pound
Red Hawk: $40.5 per pound
Teleme: $17.25 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
Cheese Type K: $2.88 per pound
Cheese Type L: $7.23 per pound

Enter the amount of Humboldt Fog in lb: 0
Enter the amount of Red Hawk in lb: 0
Enter the amount of Teleme in lb: 12
Enter the amount of Cheese Type D in lb: 0
Enter the amount of Cheese Type E in lb: 1
Enter the amount of Cheese Type F in lb: 2
Enter the amount of Cheese Type G in lb: 3
Enter the amount of Cheese Type H in lb: 0
Enter the amount of Cheese Type I in lb: 0
Enter the amount of Cheese Type J in lb: 5
Enter the amount of Cheese Type K in lb: 0
Enter the amount of Cheese Type L in lb: -3
Invalid input. Enter a value >= 0: 1.1
Invalid input. Enter a value that's multiple of 0.5: 4

Display the itemized list? (1 for yes): 0

Original Sub Total:            $287.34
Specials...
None                          -$0.0
New Sub Total:                  $287.34
Additional 15% Discount:      -$71.84
Final Total:                    $215.51

Do you wish to redo your whole order? (1 for yes): 0

Thanks for coming!
```

```
86 Ran with Cheese Total: 12
```

Specification Compliance

The following are some additional instructions to make sure your project complies with specifications:

1. Your program must produce an output that **exactly resembles the Sample Output shown above, including identical wording of prompts, spacing, input locations, etc**.
2. Your program must ensure the following:
   a. Your program should not display items that are not bought (amount is 0) when listing them out [cf. Sample Run 4, Lines 31-34]. If no items are purchased, then a special message is displayed [cf. Sample Run 2, Line 35].
   b. If no discounts are applied, a special message is displayed [cf. Sample Run 4, Line 79].
3. Your program must check for invalid inputs when entering the amounts [cf. Sample Run 3, Lines 8-15].
4. Before you submit, in Eclipse, type CTRL-A (to select everything) followed by a CTRL-I (to fix indentation) on each of your java programs. In MacOS the corresponding keystrokes are Cmd-A followed by Cmd-I.

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:

- Attached the file named `Part4` containing answers to Assessment questions (1 – 10).
- Attached the `ShopArr.java` and `RunShop.java` files.
- Filled in your collaborator's name (if any) in the "Comments…" text-box at the submission page.

Also, remember to demonstrate your code to the TA or instructor before the end of the grace period.