CSE21 : Lab #7 –Array of Objects

Overview

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

Reading : Chapter 7.10, 7.11 & 7.12

- Answer Activity 7.10.4
- Answer Activity 7.11.3
- Answer Activity 7.12.2

Getting started

You should have a Java project in Eclipse title Lab 21_7. This pdf is included in the project in the *doc* directory. The Java files you will use in this lab are in the *src* directory, as usual.

Copy over RunShop.java and Cheese.java from the previous lab.

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    : $1.25
```

```
We sell 2 kinds of Cheese
Dalaran Sharp: $1.25 per pound
```

```
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    : $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    : $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    : $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    : $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    : $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    : $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    : $136.57
```

(Exercise) 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() on this variable *shop*. Now we no longer have an Object or Class called Shop but instead we will be using ShopArr. Import RunShop.java from the previous lab to do this step.

Q1. What are the minimal changes required to instantiate ShopArr() and invoke run() on it?

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

(Exercise) 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 now split it into two functions in this lab: init() and intro(). The reason is we want to repeat intro() if need be but creating objects using init() has to be done only once. So all the println statements will be moved to intro(). Now take a look at ShopArr.java and init() method in it. We see the code is very similar but the variables have been 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*, now we use *max* in this code.

Q2. We can also use an <something>.length instead of *max*. What is the valid <something> to use in ShopArr.java?
Instead of using three different arrays (names, prices and amounts), we now only have an array of cheeses. So the code is changed to `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 Cheese object so we must access it using the "." operator. We instantiate Sharp using default constructor with 0 arguments and invoke two mutators after. Brie is instantiated with 1 argument constructor and invokes only one mutator after. Finally Swiss uses 2 argument constructors so no mutator calls are necessary after. Note that we will use the corresponding accessor method calls in other parts of the code to get to the value of the variables set by mutators.

Q3. How can we tell which instantiation (new Cheese) corresponds to which constructor definition inside Cheese Class?

Q4. How can we tell a mutator method call?

Q5. What would be the result if we added this line after Swiss is created:

```
cheese[2].setName("Wrong Name");
```
?

Now we need to implement the for-loop into ShopArr.java. 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;
}
```

Figure out the transformations needed from 3 arrays to cheese array as shown by the code already inside init(). You can assume the amount is already set to 0 for each cheese so doesn't need to zero it.

Q6. Why is the init() method both private and void?

Two Constructors:

```
public ShopArr() {
      init(10);
}

public ShopArr(int max) {
      // Fill-in Code
}
```

We implemented the basic version of ShopArr constructor which just invokes init method with a fixed number, 10. Now write the 1 argument constructor which will invoke init using the *max* parameter instead.

Q7. What are the distinguishing features of constructor methods? (ie. How do we tell them apart from other methods?)

Now implement intro(Scanner), calcSubTotal() and itemizedList() so it works for an array of cheese pointer.

Q8. How can we figure out the number of required iterations for each loop?

Q9. Should we pass in Cheese array pointer (cheese[]) as arguments into calcSubTotal or itemizedList? (Why or why not)

(Exercise) 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 it will always create 10 cheeses to sell. We will need to make use of the **second constructor** which takes in an argument max which 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 into ShopArr constructor so the program now starts like the following:

```
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).

Q10. What value will get printed out from RunShop for "Ran with Cheese Total"? (fixed number or a formula)

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have done the following ***before*** you press Submit:

- Include Answers to Question Set 7.10.4, 7.11.4, & 7.12.2
- Include answers to questions (Q1-Q10)
- Attach filled in ShopArr.java and RunShop.java
- List of Collaborators