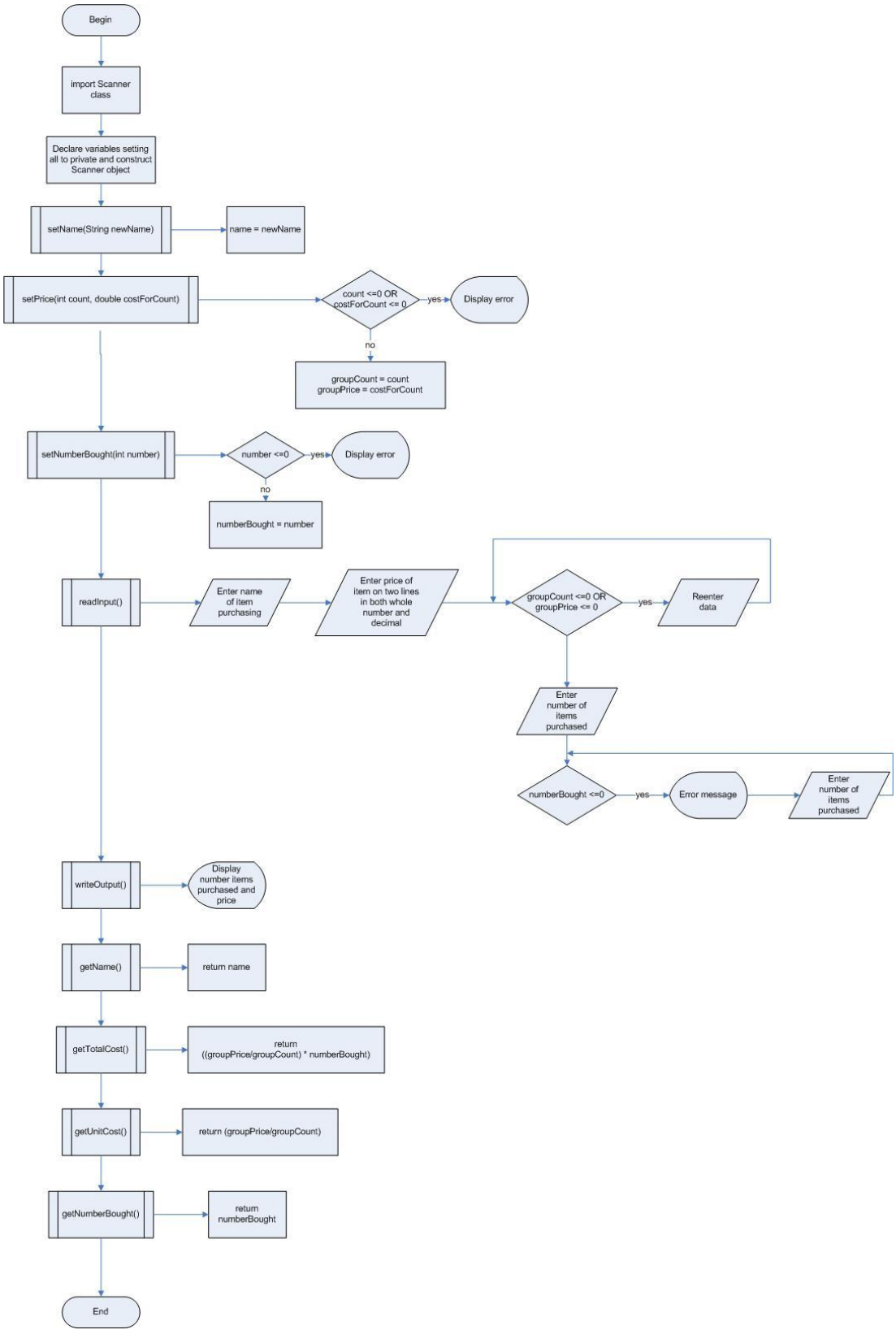
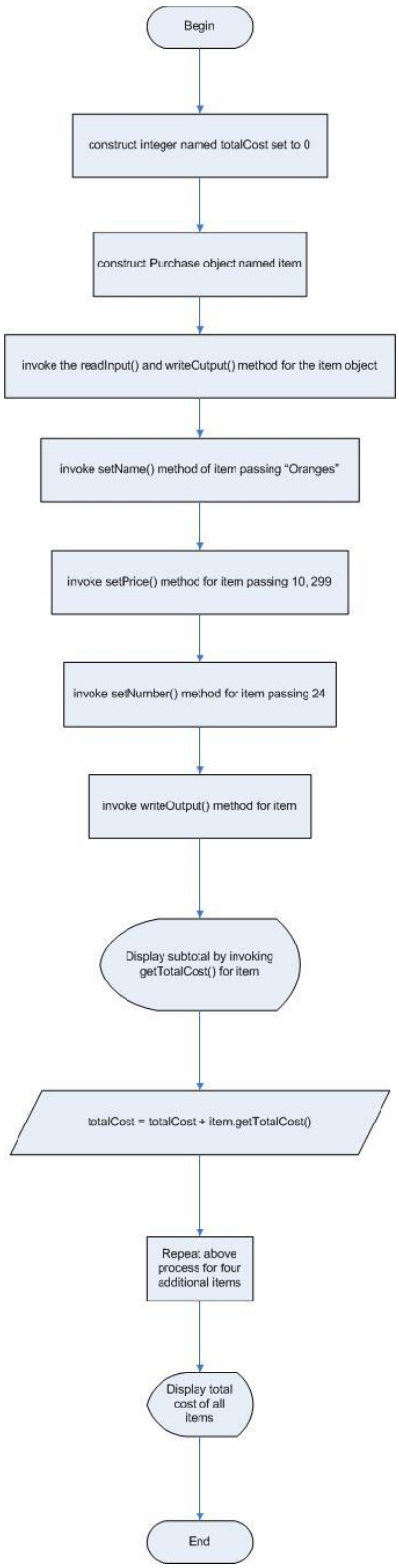
CIT 149: Java I
Chapter 5 Lab 2

This lab will consist of two separate classes. The first class will contain usable methods that will be invoked by the second class. This lab will complete programming project # 6 on page 363. The first flowchart for the external (helper) class is:


Begin
import Scanner class
Declare variables setting all to private and construct Scanner object
setName(String newName)
name = newName
setPrice(int count, double costForCount)
count <=0 OR costForCount <= 0
yes
Display error
no
groupCount = count
groupPrice = costForCount
setNumberBought(int number)
number <=0
yes
Display error
no
numberBought = number
readInput()
Enter name of item purchasing
Enter price of item on two lines in both whole number and decimal
groupCount <=0 OR groupPrice <= 0
yes
Reenter data
Enter number of items purchased
numberBought <=0
yes
Error message
Enter number of items purchased
writeOutput()
Display number items purchased and price
getName()
return name
getTotalCost()
return ((groupPrice/groupCount) * numberBought)
getUnitCost()
return (groupPrice/groupCount)
getNumberBought()
return numberBought
End

The second flowchart is for the main class:



1. Open a new document window in TextPad and save the program as Purchase.java.

2. Enter the block comments that will include the programmer's name, the program name, the date, and the purpose of the program.

3. Type the import statement to import the Scanner class.

4. Type the class header, and opening brace.

5. Type the following code that will declare several variables:

```
private String name;
private int groupCount;
private double groupPrice;
private int numberBought;//Total number being purchased.

Scanner keyboard = new Scanner(System.in);
```

6. Our first method will simply set the value of the variable name. These types of methods are called mutator methods. They are also often called setter method because they set values. Type:

```
public void setName(String newName)
{
    name = newName;
}
```

- mutator methods should always have an argument. The method receives its value from the point where the method is invoked.

7. The setprice() method is also a mutator method. It will do more than the previous method in that it will check the value of the variable passed to it for accuracy. If the value is an accurate number it will change the values of two variables. Type the following code:

```
public void setPrice(int count, double costForCount)
{
    if ((count <= 0) || (costForCount <= 0))
    {
        System.out.println("Error: Bad parameter in setPrice.");
        System.exit(0);
    }
    else
    {
        groupCount = count;
        groupPrice = costForCount;
    }
}
```

Notice that the if statement will display an error message if either variable is less than or equal to zero and will stop the program.

8. The setNumberBought() method will receive the value of an integer. If the value passed to it is equal or less than zero it will display an error message and stop the program. Type:

```
public void setNumberBought(int number)
{
    if (number <= 0)
    {
        System.out.println("Error: Bad parameter in setNumberBought.");
        System.exit(0);
    }
    else
        numberBought = number;
}
```

9. The readInput() method is void which means it will not return a value. It will instead request input from the user, setting initial values for variables. Type:

```
public void readInput()
{

    System.out.println("Enter name of item you are purchasing:");
    name = keyboard.next();

    System.out.println("Enter price of item on two lines.");
    System.out.println("For example, 3 for $2.99 is entered as");
    System.out.println("3");
    System.out.println("2.99");
    System.out.println("Enter price of item on two lines, now:");
    groupCount = keyboard.nextInt();
    groupPrice = keyboard.nextDouble();

    while ((groupCount <= 0) || (groupPrice <= 0))
    {
        //Try again:
        System.out.println("Both numbers must be positive. Try again.");
        System.out.println("Enter price of item on two lines.");

        System.out.println("For example, 3 for $2.99 is entered as");
        System.out.println("3");
        System.out.println("2.99");

        System.out.println("Enter price of item on two lines, now:");
        groupCount = keyboard.nextInt();
        groupPrice = keyboard.nextDouble();
    }

    System.out.println("Enter number of items purchased:");
    numberBought = keyboard.nextInt();

    while (numberBought <= 0)
    {
        //Try again:
        System.out.println("Number must be positive. Try again.");
        System.out.println("Enter number of items purchased:");
        numberBought = keyboard.nextInt();
    }
```

```
}
```

- The while loops in this method are used to check the input for validity. If the input is invalid the question will be repeated until the user enters a valid number.

10. The writeOutput() method is used to display values. Type:

```
public void writeOutput()
{
   System.out.println(numberBought + " " + name);
   System.out.println("at " + groupCount + " for $" + groupPrice);
}
```

11. The last few methods are accessor methods used to return the current values of variables. Type:

```
public String getName()
{
    return name;
}

 public double getTotalCost()
{
    return ((groupPrice/groupCount)*numberBought);
}

 public double getUnitCost()
{
    return (groupPrice/groupCount);
}

 public int getNumberBought()
{
    return numberBought;
}
```

12. Close the class and compile the program. Fix any errors you may have. DO NOT try to run this program. This is a helper class. Since it does not contain a main method it will not run.

13. Next we need to create our driver class.

14. Open a new document in Textpad and save the program as GroceryBill.java.

15. First we will import the DecimalFormat class. This class is used when formatting numbers in either currency format or in a specific number of decimal places. Type:

    ```
    import java.text.DecimalFormat;
    ```

16. Type the class header, opening brace, main method header and opening brace for the main method.

17. Our first step is to create a DecimalFormat object and set it to the format we want. Type:

    ```
    DecimalFormat currencyFormat = new DecimalFormat("$##,##0.00");
    ```

    - Here we are formatting it to display a dollar sign and limiting the decimal places to two. The # symbols are placeholders. These are used to prevent leading zeroes. Only if there is a value in these positions will a number display. The commas is included as well so that numbers in the 10,000 - 99,999 range will display with the comma included

18. Type the following code that will construct a double and a Purchase object.

    ```
    double totalCost = 0;
    Purchase item = new Purchase();
    ```

19. First we will request input from the user by invoking the readInput() method in the Purchase class. Type:

    ```
    item.readInput();
    ```

20. Next we will print out the information by invoking the writeOutput() method by typing:

    item.writeOutput();

21. Instead of asking the user input we'll change the current values by invoking different methods. Type:

    ```
    item.setName("Oranges");
    item.setPrice(10, 2.99);
    item.setNumberBought(24);
    item.writeOutput();
    System.out.println("Subtotal: $" + item.getTotalCost());
    totalCost = totalCost + item.getTotalCost();
    ```

    - Here we pass "Oranges" to the setName() method, so the item purchased is "Oranges"
    - The price of "Oranges" is 10 for 2.99
    - 24 were purchased
    - The result is printed out by invoking the writeOutput() method
    - The subtotal is printed out by invoking the getTotalCost() method
    - The totalCost variable has its value changed by adding the current value to the value returned by the getTotalCost() method

22. The remainder of the code in this program is repeats of the previous code, changes the items purchased and adding up the total. Type:

    ```
    item.setName("Eggs");
    item.setPrice(12, 1.69);
    item.setNumberBought(36);
    item.writeOutput();
    System.out.println("Subtotal: $" + item.getTotalCost());
    totalCost = totalCost + item.getTotalCost();

    item.setName("Apples");
    item.setPrice(3, 1.00);
    ```

```
item.setNumberBought(20);
item.writeOutput();
System.out.println("Subtotal: $" + item.getTotalCost());
totalCost = totalCost + item.getTotalCost();

item.setName("Watermelons");
item.setPrice(1, 4.39);
item.setNumberBought(2);
item.writeOutput();
System.out.println("Subtotal: $" + item.getTotalCost());
totalCost = totalCost + item.getTotalCost();

item.setName("Bagels");
item.setPrice(6, 3.50);
item.setNumberBought(12);
item.writeOutput();
System.out.println("Subtotal: $" + item.getTotalCost());
totalCost = totalCost + item.getTotalCost();
```

23. Finally the total cost is displayed by typing:

```
System.out.println();
System.out.println("Total: " + currencyFormat.format(totalCost));
```

- Here we use our DecimalFormat object to format the display of the total cost. Using the DecimalFormat format() method we apply the formatting to the value.

24. Close the main method and class.

25. Compile the program. Fix any errors if necessary.

26. Compress the following files into a single zip or rar file.

    Purchase.java
    Purchase.class
    GroceryBill.java
    GroceryBill.class

27. Submit to the appropriate drop box.