**Documentation for the code changes**

**Foo.java:**

1) Changed the name of the map from o to orderMap for proper naming convention.

2) Removed unused variable grandTotal.

3)removed c.clear() and initialized order object c = new Order();

**Explanation:** c.clear() will empty the order object so it removes all the list elements. When we add new list items to the object it will refer to the same object. So finally all the map values are referred to the last created order.

To avoid this and to put proper objects in map values we need to initialize order object every time.

4) Changed calculator class name to Calculator for naming convention.

**OrderLine.java:**

5) Removed assert quantity > 0; and implemented if(quantity <= 0) to throw exception.

assert will kill the application instead we can throw error and continue with the application.

6)Assigned item and quantity to this. Item and this.quantity.

**Order.java:**

7) Initialized the orderLines list with

private List<OrderLine> orderLines = new ArrayList<OrderLine>();

Otherwise it will give NullPointerException.

**Calculator.java:**

8)Changed the rounding method and used math.BigDecimal and math.RoundingMode to change the float value to double and calculate as per the given output.

9)Changed grandtotal variable to grandTotal for the proper naming convention.

10)Checked the map for not null and not empty before accesing it. Otherwise there is a chance to throw null pointer Exception

11)Checked the order for null before accessing as there is a chance of null pointer access.

12) Changed the loop iteration only to i<r.size() instead i<=r.size(). if the size of the list is 4 then elements will be only 0 to 3. If we do <= we will access r[4] which will give ArrayOutOfBoundException.

13)Changed the item description to lowercase and compared it with lowercase

"imported". Otherwise contains method will only compare for the exact string but our input has mixed case "imported" in the item description.

14)Used the rounding method with proper rounding mode to calculate tax and total price.

15)Added Item Quantity in print statement to match the given output.

16)Total should be calculated without adding the tax as per the requirement.

17)Total is calculated with proper rounding mode.

18)We can use logger instead of System.out.print for better performance.

Test Cases are implemented using junit.

OrderLineTest.java

OrderTest.java

CalulatorTest.java

**output of the application:**

\*\*\*\*\*\*\*Order 1\*\*\*\*\*\*\*

1 book: 13.74

1 music CD: 16.49

1 chocolate bar: 0.94

Sales Tax: 2.84

Total: 28.33

\*\*\*\*\*\*\*Order 2\*\*\*\*\*\*\*

1 imported box of chocolate: 11.5

1 imported bottle of perfume: 54.62

Sales Tax: 8.62

Total: 57.5

\*\*\*\*\*\*\*Order 3\*\*\*\*\*\*\*

1 Imported bottle of perfume: 32.19

1 bottle of perfume: 20.89

1 packet of headache pills: 10.73

1 box of imported chocolates: 12.94

Sales Tax: 8.77

Total: 67.98

Sum of orders: 153.81