The code creates only one instance of Order and reuses that instance for each new order. This design pattern is suitable if there is only one user who places different orders and gets the cumulative amount for all the orders placed.

It is possible to create different instance for each order, but in this case we can’t implement reusability of the cart and in this approach we create a new memory space for each order which is not recommended . To enable this feature of cart reusability and to maintain the data of each order we clear the Orderline record after each record is fetched.

The design and logical flow of the code indicates that one user places 3 different orders and at the end gets the Sum of all orders placed, also the documentation of the given code makes it obligatory to reuse the cart, so we use only one instance per user. If the code is scaled to multiple users, it is highly recommended to use separate instance for each user.

We choose to invoke calculate method after each Orderset. In this method the items in the Order are iterated and the tax of each item is calculated and Sales tax on the complete order is generated .After getting all the Orders we calculate Sum of Orders by returning the grandtotal to main method.

Maintained naming Convention and applied toString() method to make it readable and for testing purpose. To maintain the insertion Order to the cart LinkedHashMap is implemented.Tax is calculated differently for imported items, to make the selection flexible and case insensitive added toLowercase() method. The code also included unit Test cases using JUnit.