UNIVERSITI TEKNOLOGI MALAYSIA

PROBLEM SOLVING 2 (2.5%)

SEMESTER II 2021/2022

CODE OF SUBJECT **: SCSJ4383**

NAME OF SUBJECT **:SOFTWARE CONSTRUCTION**

YEAR / COURSE **: 4/SCSJ**

GROUP: 9

TEAM MEMBERS:

Name	Roles
MOHAMED ZAYAN (A18CS4026)	Software Engineer
Mohamed Eldeeb (A18CS4025)	Senior developer
Mohamed Ahmed Labib (A18CS4028)	Project Manager
Md Shakil Chowdhury (A18CS4047)	Project Manager

Design by contract planning

i. Identify 3 FUNCTIONS or more and describe in detail the PRE, PROCESS and POST condition for each function:

1. **setSeller:**

   class name: Order.

   precondition: The seller exists and the customer has more balance than the product's price

   process: The seller gets added to the order.

   Postcondition : The seller is attached to the order product.

2. **updateQuantity:**

   class name: Seller

   precondition: The specified product exists in the Seller products list.

   process: The Seller updates that specific product from their list.

   postcondition: Product quantity is updated from the Seller products list.

3. **setCart:**

   class name: Order

   precondition: order items are not empty.

   process: Customer adds the order items to the cart

   postcondition: Customer has access to the cart and can proceed to checkout.

ii. Apply ASSERT and INVARIANT in the code and shows the output

The system applies the concept of assertion and invariants in three classes: Product, Seller, and Customer.

Class Invariants

Product: The Product class gets asserted and invariant applied so that its cost can never be less than 0 regardless of what else is done in the code. As a result of these concepts, the quantity of the product can never be negative. Figure 1 illustrates how the concept is used in the code.

```
1
2
3   public class Product {
4       private int id;
5       private double cost;
6       private String name;
7       private int quantity;
8
9       public Product(int id,String name,double cost)
10      {
11          // INVARIANT
12          assert this.quantity >= 0 :"Quantity can not be less than 0";
13          assert this.cost >0 :"Product cost must be positive";
14
15          this.id=id;
16          this.name=name;
17          this.cost=cost;
18
19      }
20
```

Figure 1: Product Class Constructor invariant

Seller: This class includes an invariant that ensures the seller name can never be null. Figure 2 illustrates how this invariant is applied to the Seller class.

```
1
2   import java.util.Vector;
3
4   public class Seller extends User {
5       private int productsNum;
6       Vector<Product> myProducts;
7
8       public Seller(){
9           // INVARIANT
10          this.name = " ";
11      }
12
```

Figure 2: Seller Class Constructor invariant

Customer: The customer class uses assertions and invariants to ensure that the account balance of the customer will not be less than 0 regardless of any other transactions or functionalities in other parts of the code. Figure 3 illustrates how the concept is used.

```
public Customer(){
    // INVARIANT
    assert this.balance >= 0 : "Balance Can't be less than 0";
}


public Customer(int userId, String name, double balance){

    super(userId,name);

    assert this.balance > 0 : "Balance Can't be less than 0";

    this.balance=balance;
    super.accountType="Customer";

    customerCart= new Vector<Product>();
}
```

Figure 3: Customer Class Constructor invariant

Function Assertions

setSeller: The function uses assert to the order class to ensure that the seller exists and the customer has more balance than the product's price. figure 4 shows the application of assert in this function.

```
public void setSeller(Seller seller) {
    assert this.seller !=null : "no seller";
    this.seller = seller;
}
```

Figure 4: Order Set Seller assert

updateQuantity: Assert is used to make sure that the precondition specified is that the product exists in Seller's products list. figure 5 shows the application of assert in this function.

```
public void updateQuantity(Product p, int q){
    assert myProducts.contains(p) : "The product is in not your List";
    p.setQuantity(q);
}
```

Figure 5: Seller updates product quantity assert

SetCart: By asserting that the precondition of the order items is not empty, the setCart function ensures that the order items can be saved. Figure 6 illustrates how this function utilizes assert.

```
public void setCart(Vector<Product> orderItems) {
    assert orderItems.size()>0 : "no items";
    this.orderItems = orderItems;
}
```

Figure 6: order items not empty assert

iii. Git Repository:

https://github.com/MohamedZayan-dev/software-construction-problems-solvings