

## Case Study: Online Shopping Cart System(CHALLA.RAVITEJA)

### Scenario:

You are tasked with developing an online shopping cart system for an e-commerce website. The system should handle products, customers, and orders, allowing customers to add products to their cart, view the cart contents, and proceed to checkout.

### Requirements:

#### 1. Product Class:

- Attributes: `productId (String)`, `name (String)`, `price (double)`, and `stockQuantity (int)`.
- Methods: `updateStockQuantity(int quantity)` to adjust stock levels when a product is purchased.

### Program:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // Create Product instances  
  
        Product product1 = new Product("P001", "Laptop", 999.99, 10);  
  
        Product product2 = new Product("P002", "Smartphone", 699.99, 20);  
  
        Product product3 = new Product("P003", "Tablet", 299.99, 15);  
  
  
        // Display product details  
  
        System.out.println("Product details:");  
  
        System.out.println(product1);  
  
        System.out.println(product2);  
  
        System.out.println(product3);  
  
  
        // Update stock quantity  
  
        System.out.println("\nUpdating stock quantity...");  
  
        product1.updateStockQuantity(2); // Sold 2 Laptops  
  
        product2.updateStockQuantity(5); // Sold 5 Smartphones  
  
  
        // Display updated product details
```

```
        System.out.println("\nUpdated product details:");  
        System.out.println(product1);  
        System.out.println(product2);  
        System.out.println(product3);  
    }  
}
```

```
class Product {  
    private String productId;  
    private String name;  
    private double price;  
    private int stockQuantity;  
  
    public Product(String productId, String name, double price, int stockQuantity) {  
        this.productId = productId;  
        this.name = name;  
        this.price = price;  
        this.stockQuantity = stockQuantity;  
    }  
  
    public String getProductId() {  
        return productId;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPrice() {
```

```
    return price;
}
```

```
public int getStockQuantity() {
    return stockQuantity;
}
```

```
public void updateStockQuantity(int quantity) {
    if (quantity <= this.stockQuantity) {
        this.stockQuantity -= quantity;
    } else {
        System.out.println("Error: Insufficient stock to update quantity.");
    }
}
```

```
@Override
```

```
public String toString() {
    return name + " (ID: " + productId + ", Price: $" + price + ", Stock: " + stockQuantity + ")";
}
}
```

Ouput:

```
java -cp /tmp/D3dWfmyOtv/Main
```

Product details:

Laptop (ID: P001, Price: \$999.99, Stock: 10)

Smartphone (ID: P002, Price: \$699.99, Stock: 20)

Tablet (ID: P003, Price: \$299.99, Stock: 15)

Updating stock quantity...

Updated product details:

Laptop (ID: P001, Price: \$999.99, Stock: 8)

Smartphone (ID: P002, Price: \$699.99, Stock: 15)

Tablet (ID: P003, Price: \$299.99, Stock: 15)

=== Code Execution Successful ===

## 2. Customer Class:

- Attributes: `customerId (String)`, `name (String)`, `email (String)`, and `cart (List<Product>)`.
- Methods: `addToCart(Product product)`, `removeFromCart(Product product)`, `viewCart()`, and `checkout()`.

## Program:

```
import java.util.ArrayList;

import java.util.List;

class Product {

    private String productId;

    private String name;

    private double price;

    private int stockQuantity;

    public Product(String productId, String name, double price, int stockQuantity) {

        this.productId = productId;

        this.name = name;

        this.price = price;

        this.stockQuantity = stockQuantity;

    }

    public String getProductId() {

        return productId;

    }

    public String getName() {

        return name;

    }

    public double getPrice() {

        return price;

    }

}
```

```
public int getStockQuantity() {  
    return stockQuantity;  
}
```

```
public void updateStockQuantity(int quantity) {  
    if (quantity <= this.stockQuantity) {  
        this.stockQuantity -= quantity;  
    } else {  
        System.out.println("Error: Insufficient stock to update quantity.");  
    }  
}
```

```
@Override
```

```
public String toString() {  
    return name + " (ID: " + productId + ", Price: $" + price + ", Stock: " + stockQuantity + ")";  
}  
}
```

```
class Customer {  
    private String customerId;  
    private String name;  
    private String email;  
    private List<Product> cart;  
  
    public Customer(String customerId, String name, String email) {  
        this.customerId = customerId;  
        this.name = name;  
        this.email = email;
```

```
    this.cart = new ArrayList<>();  
}
```

```
public void addToCart(Product product) {  
    cart.add(product);  
}
```

```
public void removeFromCart(Product product) {  
    cart.remove(product);  
}
```

```
public void viewCart() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty.");  
    } else {  
        System.out.println("Cart contents:");  
        for (Product product : cart) {  
            System.out.println(product);  
        }  
    }  
}
```

```
public void checkout() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty. Add items to the cart before checking out.");  
    } else {  
        Order order = new Order(customerId, cart);  
        order.calculateTotalAmount();  
        System.out.println("Order placed successfully. Total amount: $" + order.getTotalAmount());  
    }  
}
```

```
        cart.clear();
    }
}
}
```

```
class Order {

    private String customerId;
    private List<Product> products;
    private double totalAmount;

    public Order(String customerId, List<Product> products) {
        this.customerId = customerId;
        this.products = new ArrayList<>(products);
        this.totalAmount = 0.0;
    }

    public void calculateTotalAmount() {
        for (Product product : products) {
            totalAmount += product.getPrice();
        }
    }

    public double getTotalAmount() {
        return totalAmount;
    }
}
```

```
public class Main {

    public static void main(String[] args) {
```



```
// Create Product instances

Product product1 = new Product("P001", "Laptop", 999.99, 10);
Product product2 = new Product("P002", "Smartphone", 699.99, 20);
Product product3 = new Product("P003", "Tablet", 299.99, 15);


// Create Customer instance

Customer customer = new Customer("C001", "John Doe", "john.doe@example.com");


// Add products to the customer's cart

customer.addToCart(product1);
customer.addToCart(product2);


// View cart contents

System.out.println("Viewing cart:");
customer.viewCart();


// Remove a product from the cart

customer.removeFromCart(product2);


// View cart contents again

System.out.println("\nViewing cart after removing an item:");
customer.viewCart();


// Checkout

System.out.println("\nChecking out:");
customer.checkout();


// View cart contents after checkout

System.out.println("\nViewing cart after checkout:");
```

```
customer.viewCart();  
}  
}
```

Output:

```
^ java -cp /tmp/GS6AVlgvNI/Main  
Viewing cart:  
Cart contents:  
Laptop (ID: P001, Price: $999.99, Stock: 10)  
Smartphone (ID: P002, Price: $699.99, Stock: 20)  
  
Viewing cart after removing an item:  
Cart contents:  
Laptop (ID: P001, Price: $999.99, Stock: 10)  
  
Checking out:  
Order placed successfully. Total amount: $999.99  
  
Viewing cart after checkout:  
Your cart is empty.  
  
=== Code Execution Successful ===
```

### 3. Order Class:

- Attributes: `orderId (String)`, `customer (Customer)`, `products (List<Product>)`, `totalAmount (double)`, and `orderDate (LocalDateTime)`.
- Methods: `calculateTotalAmount()` to compute the total cost of the order.

### program:

```
import java.time.LocalDateTime;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
class Product {
```

```
    private String productId;
```

```
    private String name;
```

```
    private double price;
```

```
    private int stockQuantity;
```

```
    public Product(String productId, String name, double price, int stockQuantity) {
```

```
        this.productId = productId;
```

```
        this.name = name;
```

```
        this.price = price;
```

```
        this.stockQuantity = stockQuantity;
```

```
    }
```

```
    public String getProductId() {
```

```
        return productId;
```

```
    }
```

```
    public String getName() {
```

```
        return name;
```

```
    }
```

```
public double getPrice() {  
    return price;  
}
```

```
public int getStockQuantity() {  
    return stockQuantity;  
}
```

```
public void updateStockQuantity(int quantity) {  
    if (quantity <= this.stockQuantity) {  
        this.stockQuantity -= quantity;  
    } else {  
        System.out.println("Error: Insufficient stock to update quantity.");  
    }  
}
```

```
@Override
```

```
public String toString() {  
    return name + " (ID: " + productId + ", Price: $" + price + ", Stock: " + stockQuantity + ")";  
}  
}
```

```
class Customer {  
    private String customerId;  
    private String name;  
    private String email;  
    private List<Product> cart;
```

```
public Customer(String customerId, String name, String email) {  
    this.customerId = customerId;  
    this.name = name;  
    this.email = email;  
    this.cart = new ArrayList<>();  
}  
  
public void addToCart(Product product) {  
    cart.add(product);  
}  
  
public void removeFromCart(Product product) {  
    cart.remove(product);  
}  
  
public void viewCart() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty.");  
    } else {  
        System.out.println("Cart contents:");  
        for (Product product : cart) {  
            System.out.println(product);  
        }  
    }  
}  
  
public void checkout() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty. Add items to the cart before checking out.");  
    }  
}
```

```

    } else {
        Order order = new Order(customerId, cart);
        order.calculateTotalAmount();
        System.out.println("Order placed successfully. Total amount: $" + order.getTotalAmount());
        cart.clear();
    }
}
}

```

```

class Order {
    private String orderId;
    private String customerId;
    private List<Product> products;
    private double totalAmount;
    private LocalDateTime orderDate;

    public Order(String customerId, List<Product> products) {
        this.orderId = "ORD" + System.currentTimeMillis();
        this.customerId = customerId;
        this.products = new ArrayList<>(products);
        this.orderDate = LocalDateTime.now();
    }

    public void calculateTotalAmount() {
        totalAmount = 0;
        for (Product product : products) {
            totalAmount += product.getPrice();
        }
    }
}

```

```
public double getTotalAmount() {  
    return totalAmount;  
}
```

```
@Override
```

```
public String toString() {  
    return "Order ID: " + orderId + "\nCustomer ID: " + customerId + "\nOrder Date: " + orderDate +  
    "\nTotal Amount: $" + totalAmount;  
}  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        // Create Product instances  
  
        Product product1 = new Product("P001", "Laptop", 999.99, 10);  
        Product product2 = new Product("P002", "Smartphone", 699.99, 20);  
        Product product3 = new Product("P003", "Tablet", 299.99, 15);  
  
        // Create Customer instance  
        Customer customer = new Customer("C001", "John Doe", "john.doe@example.com");  
  
        // Add products to the customer's cart  
        customer.addToCart(product1);  
        customer.addToCart(product2);  
  
        // View cart contents  
        System.out.println("Viewing cart:");  
        customer.viewCart();  
    }  
}
```

```
// Remove a product from the cart
customer.removeFromCart(product2);

// View cart contents again
System.out.println("\nViewing cart after removing an item:");
customer.viewCart();

// Checkout
System.out.println("\nChecking out:");
customer.checkout();

// View cart contents after checkout
System.out.println("\nViewing cart after checkout:");
customer.viewCart();
}
}
```

Output:



```
java -cp /tmp/Ryj47aGepG/Main
Viewing cart:
Cart contents:
Laptop (ID: P001, Price: $999.99, Stock: 10)
Smartphone (ID: P002, Price: $699.99, Stock: 20)

Viewing cart after removing an item:
Cart contents:
Laptop (ID: P001, Price: $999.99, Stock: 10)

Checking out:
Order placed successfully. Total amount: $999.99

Viewing cart after checkout:
Your cart is empty.

=== Code Execution Successful ===
```

#### 4. Inventory Class:

- Attributes: products (List<Product>).
- Methods: addProduct(Product product), getProductById(String productId), and updateProductStock(String productId, int quantity).

#### Program:

```
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
```

```
class Product {
    private String productId;
```

```
private String name;

private double price;

private int stockQuantity;


public Product(String productId, String name, double price, int stockQuantity) {
    this.productId = productId;
    this.name = name;
    this.price = price;
    this.stockQuantity = stockQuantity;
}


public String getProductId() {
    return productId;
}


public String getName() {
    return name;
}


public double getPrice() {
    return price;
}


public int getStockQuantity() {
    return stockQuantity;
}


public void updateStockQuantity(int quantity) {
    if (quantity <= this.stockQuantity) {
```

```
        this.stockQuantity -= quantity;
    } else {
        System.out.println("Error: Insufficient stock to update quantity.");
    }
}
```

```
@Override
public String toString() {
    return name + " (ID: " + productId + ", Price: $" + price + ", Stock: " + stockQuantity + ")";
}
}
```

```
class Inventory {
    private List<Product> products;

    public Inventory() {
        this.products = new ArrayList<>();
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public Product getProductById(String productId) {
        for (Product product : products) {
            if (product.getId().equals(productId)) {
                return product;
            }
        }
    }
}
```

```
        return null;
    }

    public void updateProductStock(String productId, int quantity) {
        Product product = getProductById(productId);
        if (product != null) {
            product.updateStockQuantity(quantity);
        }
    }
}
```

```
class Customer {
    private String customerId;
    private String name;
    private String email;
    private List<Product> cart;

    public Customer(String customerId, String name, String email) {
        this.customerId = customerId;
        this.name = name;
        this.email = email;
        this.cart = new ArrayList<>();
    }

    public void addToCart(Product product) {
        cart.add(product);
    }

    public void removeFromCart(Product product) {
```

```
    cart.remove(product);  
}
```

```
public void viewCart() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty.");  
    } else {  
        System.out.println("Cart contents:");  
        for (Product product : cart) {  
            System.out.println(product);  
        }  
    }  
}
```

```
public void checkout(Inventory inventory) {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty. Add items to the cart before checking out.");  
    } else {  
        Order order = new Order(customerId, cart);  
        order.calculateTotalAmount();  
        System.out.println("Order placed successfully. Total amount: $" + order.getTotalAmount());  
  
        for (Product product : cart) {  
            inventory.updateProductStock(product.getProductId(), 1);  
        }  
        cart.clear();  
    }  
}
```

```

class Order {

    private String orderId;

    private String customerId;

    private List<Product> products;

    private double totalAmount;

    private LocalDateTime orderDate;


    public Order(String customerId, List<Product> products) {

        this.orderId = "ORD" + System.currentTimeMillis();

        this.customerId = customerId;

        this.products = new ArrayList<>(products);

        this.orderDate = LocalDateTime.now();
    }


    public void calculateTotalAmount() {

        totalAmount = 0;

        for (Product product : products) {

            totalAmount += product.getPrice();

        }

    }


    public double getTotalAmount() {

        return totalAmount;

    }


    @Override

    public String toString() {

```

```
        return "Order ID: " + orderId + "\nCustomer ID: " + customerId + "\nOrder Date: " + orderDate +  
        "\nTotal Amount: $" + totalAmount;  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        // Create Inventory instance  
        Inventory inventory = new Inventory();  
  
        // Create Product instances and add them to the inventory  
        Product product1 = new Product("P001", "Laptop", 999.99, 10);  
        Product product2 = new Product("P002", "Smartphone", 699.99, 20);  
        Product product3 = new Product("P003", "Tablet", 299.99, 15);  
        inventory.addProduct(product1);  
        inventory.addProduct(product2);  
        inventory.addProduct(product3);  
  
        // Create Customer instance  
        Customer customer = new Customer("C001", "John Doe", "john.doe@example.com");  
  
        // Add products to the customer's cart  
        customer.addToCart(product1);  
        customer.addToCart(product2);  
  
        // View cart contents  
        System.out.println("Viewing cart:");  
        customer.viewCart();  
    }  
}
```

```
// Remove a product from the cart
customer.removeFromCart(product2);

// View cart contents again
System.out.println("\nViewing cart after removing an item:");
customer.viewCart();

// Checkout
System.out.println("\nChecking out:");
customer.checkout(inventory);

// View cart contents after checkout
System.out.println("\nViewing cart after checkout:");
customer.viewCart();

// View updated stock in inventory
System.out.println("\nUpdated inventory stock:");
System.out.println(inventory.getProductById("P001"));
System.out.println(inventory.getProductById("P002"));
System.out.println(inventory.getProductById("P003"));
}
}
```

Output:



```
^ java -cp /tmp/eLnD9EVxzt/Main
```

Viewing cart:

Cart contents:

Laptop (ID: P001, Price: \$999.99, Stock: 10)

Smartphone (ID: P002, Price: \$699.99, Stock: 20)

Viewing cart after removing an item:

Cart contents:

Laptop (ID: P001, Price: \$999.99, Stock: 10)

Checking out:

Order placed successfully. Total amount: \$999.99

Viewing cart after checkout:

Your cart is empty.

Updated inventory stock:

Laptop (ID: P001, Price: \$999.99, Stock: 9)

Smartphone (ID: P002, Price: \$699.99, Stock: 20)

Tablet (ID: P003, Price: \$299.99, Stock: 15)

=== Code Execution Successful ===

### Tasks:

1. **Implement the Product Class:**
  - Define the class with appropriate attributes and methods.
  - Implement logic to update the stock quantity when products are purchased.
2. **Implement the Customer Class:**
  - Define the class with attributes and methods to manage the shopping cart.
  - Implement methods to add products to the cart, remove products from the cart, view the cart contents, and proceed to checkout.
3. **Implement the Order Class:**
  - Define the class with attributes and methods to handle order details.
  - Implement the `calculateTotalAmount()` method to compute the total cost of the order.
4. **Implement the Inventory Class:**
  - Define the class to manage the product inventory.
  - Implement methods to add products, retrieve a product by its ID, and update stock levels.
5. **Develop a Main Class to Test the System:**
  - Create instances of `Product`, `Customer`, and `Inventory`.
  - Add products to the inventory.
  - Simulate adding products to the customer's cart, viewing the cart, and checking out.

### Main program:

```
import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;
```

```
class Product {  
    private String productId;  
    private String name;  
    private double price;  
    private int stockQuantity;  
  
    public Product(String productId, String name, double price, int stockQuantity) {  
        this.productId = productId;  
        this.name = name;  
        this.price = price;  
        this.stockQuantity = stockQuantity;  
    }  
  
    public String getProductId() {  
        return productId;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
  
    public int getStockQuantity() {  
        return stockQuantity;  
    }  
}
```

```

public void updateStockQuantity(int quantity) {
    if (quantity <= this.stockQuantity) {
        this.stockQuantity -= quantity;
    } else {
        System.out.println("Error: Insufficient stock to update quantity.");
    }
}

```

```

@Override
public String toString() {
    return name + " (ID: " + productId + ", Price: $" + price + ", Stock: " + stockQuantity + ")";
}
}

```

```

class Inventory {
    private List<Product> products;

    public Inventory() {
        this.products = new ArrayList<>();
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public Product getProductById(String productId) {
        for (Product product : products) {
            if (product.getProductId().equals(productId)) {
                return product;
            }
        }
    }
}

```

```
    }  
    }  
    return null;  
}
```

```
public void updateProductStock(String productId, int quantity) {  
    Product product = getProductById(productId);  
    if (product != null) {  
        product.updateStockQuantity(quantity);  
    }  
}  
}
```

```
class Customer {  
    private String customerId;  
    private String name;  
    private String email;  
    private List<Product> cart;  
  
    public Customer(String customerId, String name, String email) {  
        this.customerId = customerId;  
        this.name = name;  
        this.email = email;  
        this.cart = new ArrayList<>();  
    }  
  
    public void addToCart(Product product) {  
        cart.add(product);  
    }  
}
```

```
public void removeFromCart(Product product) {  
    cart.remove(product);  
}
```

```
public void viewCart() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty.");  
    } else {  
        System.out.println("Cart contents:");  
        for (Product product : cart) {  
            System.out.println(product);  
        }  
    }  
}
```

```
public void checkout() {  
    if (cart.isEmpty()) {  
        System.out.println("Your cart is empty. Add items to the cart before checking out.");  
    } else {  
        Order order = new Order(customerId, cart);  
        order.calculateTotalAmount();  
        System.out.println("Order placed successfully. Total amount: $" + order.getTotalAmount());  
        cart.clear();  
    }  
}
```

```
class Order {
```

```
private String orderId;

private String customerId;

private List<Product> products;

private double totalAmount;

private LocalDateTime orderDate;


public Order(String customerId, List<Product> products) {

    this.orderId = "ORD" + System.currentTimeMillis();

    this.customerId = customerId;

    this.products = new ArrayList<>(products);

    this.orderDate = LocalDateTime.now();

}


public void calculateTotalAmount() {

    totalAmount = 0;

    for (Product product : products) {

        totalAmount += product.getPrice();

    }

}


public double getTotalAmount() {

    return totalAmount;

}


@Override

public String toString() {

    return "Order ID: " + orderId + "\nCustomer ID: " + customerId + "\nOrder Date: " + orderDate +
"\nTotal Amount: $" + totalAmount;

}
```

```
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Inventory inventory = new Inventory();  
        Product product1 = new Product("P001", "Laptop", 1000.0, 10);  
        Product product2 = new Product("P002", "Smartphone", 500.0, 20);  
  
        inventory.addProduct(product1);  
        inventory.addProduct(product2);  
  
        Customer customer = new Customer("C001", "John Doe", "john@example.com");  
  
        customer.addToCart(product1);  
        customer.addToCart(product2);  
  
        customer.viewCart();  
  
        customer.checkout();  
  
        inventory.updateProductStock("P001", 1);  
        inventory.updateProductStock("P002", 1);  
  
        System.out.println("Updated inventory:");  
        System.out.println(inventory.getProductById("P001"));  
        System.out.println(inventory.getProductById("P002"));  
    }  
}
```

Output:



## Output

```
^ java -cp /tmp/BaV1LHrs8P/Main
Cart contents:
Laptop (ID: P001, Price: $1000.0, Stock: 10)
Smartphone (ID: P002, Price: $500.0, Stock: 20)
Order placed successfully. Total amount: $1500.0
Updated inventory:
Laptop (ID: P001, Price: $1000.0, Stock: 9)
Smartphone (ID: P002, Price: $500.0, Stock: 19)

=== Code Execution Successful ===
```