

E-commerce Class

The image shows a Java IDE interface with two code editors side-by-side. Both editors have a dark theme and show Java code related to an e-commerce system.

Top Editor (Lines 1-29):

```
1 package project;
2 import java.util.Scanner;
3 public class EcommerceSystem {
4     public static void main(String[] args) {
5         EcommerceGui x = new EcommerceGui();
6         Scanner input = new Scanner(System.in);
7         System.out.println("Welcome to the E-Commerce System!");
8         System.out.print("Please enter your Id: ");
9         int customerId = input.nextInt();
10        System.out.print("Please enter your name: ");
11        String name = input.next();
12        System.out.print("Please enter your address: ");
13        String address = input.next();
14        System.out.print("How many products you want to add to your cart? ");
15        int ncart = input.nextInt();
16        Customer myCustomer = new Customer(customerId, name, address);
17        Cart myCart = new Cart(myCustomer.get_customerId(), ncart);
18        for (int i = 0; i < ncart; i++) {
19            System.out.println("Enter product type (1 for electronic, 2 for clothing, 3 for book):");
20            int productType = input.nextInt();
21            input.nextLine();
22            switch (productType) {
23                case 1 -> {
24                    ElectronicProduct newElectronicProduct = new ElectronicProduct(1, "smartphone", 599.99f, "samsung", 1);
25                    myCart.addproduct(newElectronicProduct, i);
26                }
27                case 2 -> {
28                    ClothingProduct newClothingProduct = new ClothingProduct(2, "T-shirt", 19.99f, "Medium", "cotton");
29                    myCart.addproduct(newClothingProduct, i);
30                }
31                case 3 -> {
32                    BookProduct newBookProduct = new BookProduct(3, "oop", 39.99f, "O'Reilly", "X Publications");
33                    myCart.addproduct(newBookProduct, i);
34                }
35                default -> {
36                    System.out.println("Invalid product type.");
37                }
38            }
39        }
40        System.out.println("Your total is "+ myCart.calculatePrice()+" $");
41        System.out.print("\nDo you want to place an order for the products in your cart? (1-yes/2-no)");
42        int response = input.nextInt();
43        if (response==1) {
44            System.out.println("Here's your order's summary:");
45            Order myOrder = new Order(myCustomer.get_customerId(), 1, myCart.getProducts());
46            myOrder.printOrderInfo();
47
48        } else {
49            System.out.println("Order not placed.");
50        }
51        System.exit(0);
52    }
53}
```

Bottom Editor (Lines 25-54):

```
25        myCart.addproduct(newElectronicProduct, i);
26    }
27    case 2 -> {
28        ClothingProduct newClothingProduct = new ClothingProduct(2, "T-shirt", 19.99f, "Medium", "cotton");
29        myCart.addproduct(newClothingProduct, i);
30    }
31    case 3 -> {
32        BookProduct newBookProduct = new BookProduct(3, "oop", 39.99f, "O'Reilly", "X Publications");
33        myCart.addproduct(newBookProduct, i);
34    }
35    default -> {
36        System.out.println("Invalid product type.");
37    }
38}
39
40 System.out.println("Your total is "+ myCart.calculatePrice()+" $");
41 System.out.print("\nDo you want to place an order for the products in your cart? (1-yes/2-no)");
42 int response = input.nextInt();
43 if (response==1) {
44     System.out.println("Here's your order's summary:");
45 Order myOrder = new Order(myCustomer.get_customerId(), 1, myCart.getProducts());
46 myOrder.printOrderInfo();
47
48 } else {
49     System.out.println("Order not placed.");
50 }
51 System.exit(0);
52 }
```

Product Class

A screenshot of a Java code editor window titled "product.java X". The window has a dark theme with light-colored syntax highlighting. The code defines a class named "product" with private fields for productId, name, and price. It includes a constructor that takes productId, name, and price as parameters and sets them using Math.abs(). It also includes methods to get and set each of these properties.

```
1 package project;
2 public class product {
3     int productId;
4     String name ;
5     float price ;
6     public product (int productId, String name , float price2){
7         this.productId = productId;
8         this.name = name;
9         this.price = Math.abs(price2);
10    }
11    public int get_productId (){
12        return Math.abs(productId) ;
13    }
14    public String get_name(){
15        return name;
16    }
17    public float get_price (){
18        return (float) Math.abs(price);
19    }
20    public void set_productId (int productId){
21        this.productId=Math.abs(productId);
22    }
23    public void set_name ( String name ){
24        this.name=name;
25    }
26    public void set_price ( float price){
27        this.price=price;
28    }
29 }
```

ElectronicProduct Class

A screenshot of a Java code editor window titled "ElectronicProduct.java X". The window has a dark theme with light-colored syntax highlighting. The code defines a class named "ElectronicProduct" that extends the "product" class. It adds two new private fields: "brand" and "warrantyPeriod". The constructor takes productId, name, price, brand, and warrantyPeriod as parameters, calling the super constructor and setting the new fields. It also includes methods to get and set both the brand and warranty period.

```
1 package project;
2 public class ElectronicProduct extends product {
3     String brand;
4     int warrantyPeriod;
5     public ElectronicProduct(int productId, String name, float price, String brand, int warrantyPeriod) {
6         super (productId , name,price);
7         this.brand=brand;
8         this.warrantyPeriod=warrantyPeriod;
9     }
10    public String get_brand(){
11        return brand;
12    }
13    public int get_warrantyPeriod (){
14        return Math.abs(warrantyPeriod);
15    }
16    public void set_brand (String brand){
17        this.brand=brand;
18    }
19    public void set_warrantyPeriod ( int warrantyPeriod){
20        this.warrantyPeriod=Math.abs(warrantyPeriod);
21    }
22 }
23
```

Clothingproduct Class

The screenshot shows a Java code editor window titled "ClothingProduct.java". The code defines a class named "ClothingProduct" which extends a base class "product". The class has two private instance variables: "size" and "fabric". It includes a constructor that takes productId, name, price, size, and fabric as parameters, calling the base class's constructor and setting the instance variables. It also includes four methods: "get_size()", "get_fabric()", "set_size(String size)", and "set_fabric(String fabric)". The code is color-coded for syntax highlighting.

```
1 package project;
2 public class ClothingProduct extends product {
3     String size;
4     String fabric;
5     public ClothingProduct(int productId, String name, float price, String size, String fabric) {
6         super(productId, name, price);
7         this.size = size;
8         this.fabric = fabric;
9     }
10    public String get_size() {
11        return size;
12    }
13    public String get_fabric() {
14        return fabric;
15    }
16    public void set_size(String size) {
17        this.size = size;
18    }
19    public void set_fabric(String fabric) {
20        this.fabric = fabric;
21    }
}
```

Bookproduct Class

The screenshot shows a Java code editor window titled "BookProduct.java". The code defines a class named "BookProduct" which extends a base class "product". The class has two private instance variables: "author" and "publisher". It includes a constructor that takes productId, name, price, author, and publisher as parameters, calling the base class's constructor and setting the instance variables. It also includes three methods: "get_author()", "get_publisher()", and two setters ("set_author(String author)" and "set_publisher(String publisher)"). The code is color-coded for syntax highlighting.

```
1 package project;
2 public class BookProduct extends product {
3     String author;
4     String publisher;
5     public BookProduct(int productId, String name, float price, String author, String publisher) {
6         super(productId, name, price);
7         this.author = author;
8         this.publisher = publisher;
9     }
10    public String get_author() {
11        return author;
12    }
13    public String get_publisher() {
14        return publisher;
15    }
16    public void set_author(String author) {
17        this.author = author;
18    }
19    public void set_publisher(String publisher) {
20        this.publisher = publisher;
21    }
22 }
```

Customer Class

The screenshot shows a Java code editor window titled "Customer.java x". The menu bar includes "Source" and "History". The toolbar contains various icons for file operations like new, open, save, and search. The code itself is as follows:

```
1 package project;
2 public class Customer {
3     int customerId;
4     String name;
5     String address;
6     public Customer(int customerId, String name, String address) {
7         this.customerId=Math.abs(customerId);
8         this.name=name;
9         this.address=address;
10    }
11    public int get_customerId(){
12        return Math.abs(customerId);
13    }
14    public String get_name(){
15        return name;
16    }
17    public String get_address(){
18        return address;
19    }
20    public void set_customerId(int customerId){
21        this.customerId=Math.abs(customerId);
22    }
23    public void set_name (String name){
24        this.name=name;
25    }
26    public void set_address( String address){
27        this.address=address;
28    }
29 }
```

Order Class

The screenshot shows a Java code editor window titled "order.java x". The menu bar includes "Source" and "History". The toolbar contains various icons for file operations like new, open, save, and search. The code itself is as follows:

```
1 package project;
2 public class order {
3     int customerId;
4     int orderId;
5     product[] products;
6     float totalPrice;
7     public order (int customerId, int orderId, product[] products){
8         this.customerId=Math.abs(customerId);
9         this.orderId=Math.abs(orderId);
10        this.products= products;
11        this.totalPrice=calculateTotalPrice();
12    }
13    order() {
14    }
15    public float calculateTotalPrice() {
16        float totalPrice = 0;
17        for (product product : products) {
18            if (product != null) {
19                totalPrice += product.get_price();
20            }
21        }
22        return totalPrice;
23    }
24    public void printOrderInfo() {
25        System.out.println("Order ID: " + orderId);
26        System.out.println("Customer ID: " + customerId);
27        System.out.println("Products:");
28        for (product product : products) {
29            if (product != null) {
```

The screenshot shows a Java code editor window titled "order.java X". The code is as follows:

```
23     }
24     public void printOrderInfo() {
25         System.out.println("Order ID: " + orderId);
26         System.out.println("Customer ID: " + customerId);
27         System.out.println("Products:");
28         for (product product : products) {
29             if (product != null) {
30                 System.out.println("- " + product.get_name() + ": $" + product.get_price());
31             }
32         }
33         System.out.println("Total Price: $" + totalPrice);
34     }
35     public float getTotalPrice() {
36         return totalPrice;
37     }
38 }
39
40
41
```

Cart Class

The screenshot shows a Java code editor window titled "Cart.java X". The code is as follows:

```
1 package project;
2 public class Cart {
3     int customerId;
4     int nProducts ;
5     product[] products;
6     public Cart (int customerId , int nproducts){
7         this.customerId=Math.abs(customerId);
8         this.nProducts= Math.abs(nproducts);
9         this.products=new product[nproducts];
10    }
11    public int get_customerId(){
12        return Math.abs(customerId);
13    }
14    public int get_nproducts(){
15        return Math.abs(nProducts);
16    }
17    public void set_customerId(int customerId){
18        this.customerId = customerId;
19    }
20    public void set_nproducts(int nProducts){
21        this.nProducts = nProducts;
22    }
23    public void addproduct(product product, int index){
24        if (index >= 0 && index < nProducts) {
25            products[index] = product;
26        } else {
27            System.out.println("Invalid index.");
28        }
29    }
}
```

The screenshot shows a Java code editor window titled "Cartjava X". The code is a class named "Cart" with several methods: removeproduct, calculatePrice, getProducts, getcustomerId, and placeOrder. The code uses System.out.println statements for error handling and printing the total price.

```
27     System.out.println("Invalid index.");
28   }
29 }
30 public void removeproduct(int index){
31   if (index >= 0 && index < nProducts) {
32     products[index] = null;
33   } else {
34     System.out.println("Invalid index.");
35   }
36 }
37 public float calculatePrice(){
38   float totalPrice = 0;
39   for (product product : products) {
40     if (product != null) {
41       totalPrice += product.get_price();
42     }
43   }
44   return totalPrice;
45 }
46 public product [] getProducts(){
47   return products;
48 }
49 public int getcustomerId(){
50   return customerId;
51 }
52 public void placeOrder () {
53
54 }
55 }
```

Output

The screenshot shows a terminal window with the title "Output - project (run) X". The output displays the interaction with the E-commerce system, starting with a welcome message and prompts for customer ID, name, address, and the number of products to add to the cart. It then asks for product types and lists the selected items. Finally, it asks if the user wants to place an order and provides a summary of the order details.

```
run:
Welcome to the E-Commerce System!
Please enter your Id:
20231
Please enter your name:
Noureen
Please enter your address:
address
How many products you want to add to your cart?
4
Enter product type (1 for electronic, 2 for clothing, 3 for book):
2
Enter product type (1 for electronic, 2 for clothing, 3 for book):
3
Enter product type (1 for electronic, 2 for clothing, 3 for book):
2
Enter product type (1 for electronic, 2 for clothing, 3 for book):
1
Your total is 679.95996 $

Do you want to place an order for the products in your cart? (1-yes/2-no)
1
Here's your order's summary:
Order ID: 1
Customer ID: 20231
Products:
- T-shirt: $19.99
- oop: $39.99
- T-shirt: $19.99
- smartphone: $599.99
Total Price: $679.95996
BUILD SUCCESSFUL (total time: 49 seconds)
```

Gui Classes

1-E-commerceGui class

EcommerceGui.java

```
1 package project;
2 import javax.swing.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 public class EcommerceGui extends JFrame implements ActionListener {
6     JPanel p1;
7     JLabel l1, l2, l3, l4, l5;
8     JTextField t1, t2, t3, t4;
9     JButton b1, b2;
10    private cartGui cartGui;
11    private String customerId;private String orderid;
12    EcommerceGui() {
13        this.setTitle("E-commerce");
14        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15        this.setResizable(false);
16        this.setSize(500, 400);
17        this.setLocation(200, 50);
18        l1 = new JLabel("Please enter your Id: ");
19        l2 = new JLabel("Please enter your name: ");
20        l3 = new JLabel("Please enter your address: ");
21        l4 = new JLabel("How many products you want to add to cart?");
22        l5 = new JLabel("Welcome to the E-commerce system: ");
23        p1 = new JPanel();
24        p1.setLayout(null);
25        t1 = new JTextField();
26        t2 = new JTextField();
27        t3 = new JTextField();
28        t4 = new JTextField();
29        l1.setBounds(10, 50, 300, 25);
```

EcommerceGui.java

```
30        t2.setBounds(280, 50, 200, 25);
31        l2.setBounds(10, 90, 300, 25);
32        t2.setBounds(280, 90, 200, 25);
33        t3.setBounds(280, 130, 200, 25);
34        l3.setBounds(10, 130, 300, 25);
35        l4.setBounds(10, 170, 300, 25);
36        t4.setBounds(280, 170, 200, 25);
37        l5.setBounds(10, 10, 300, 25);
38        this.add(p1);
39        p1.add(l1);p1.add(l2);p1.add(l3);p1.add(t4);p1.add(l4);
40        p1.add(t1);p1.add(t2);p1.add(t3);p1.add(l5);
41        this.setVisible(true);
42        t1.addActionListener(this);
43        t4.addActionListener(this);
44        b1 = new JButton("create");
45        b1.setBounds(50, 230, 150, 25);
46        p1.add(b1);
47        b1.addActionListener(this);
48        b2 = new JButton("Exist");
49        b2.setBounds(250, 230, 150, 25);
50        p1.add(b2);
51        b2.addActionListener(this);
52    }
53    EcommerceGui(String customerId, String orderid){
54        this.customerId=customerId;
55        this.orderid=orderid;
```

The screenshot shows the Java code for the `EcommerceGui` class. The code handles button actions to add products to a cart and dispose of the window. It also provides a method to get the `cartGui` frame.

```
50     p1.add(b1);
51     b1.addActionListener(this);
52 }
53 EcommerceGui(String customerId, String orderid) {
54     this.customerId=customerId;
55     this.orderid=orderid;
56 }
57 @Override
58 public void actionPerformed(ActionEvent e) {
59     if (e.getSource() == b1) {
60         String nproduct = t4.getText().toString();
61         int x = Integer.parseInt(nproduct);
62         String orderId = "1";
63         String customerID = t1.getText().toString();
64         cartGui a = new cartGui(customerID, orderId, x);
65         dispose();
66     }
67     if (e.getSource() == b2) {
68
69         this.dispose();
70     }
71 }
72 public cartGui getMyFrame() {
73     return cartGui;
74 }
75 }
```

2-OrderGui class

The screenshot shows the Java code for the `orderGui` class, which extends `JFrame` and implements `ActionListener`. It creates five labels and one button, sets their positions, and adds them to the frame's content pane.

```
1 package project;
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 public class orderGui extends JFrame implements ActionListener {
7     private JLabel label1,label2,label3,label4,label5;
8     private JButton button1;
9     orderGui(String customerid, String orderid, String selectedItemText, float totalPrice) {
10         setTitle("Order place : ");
11         setSize(700,400);
12         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         setBackground(Color.gray);
14         this.setResizable(false);
15         setLayout(new BoxLayout(getContentPane(),BoxLayout.Y_AXIS));
16         label1 = new JLabel("Here's your summary : ");
17         label2 = new JLabel("CustomerID : " + customerid);
18         label3 = new JLabel("OrderID : " + orderid);
19         label4 = new JLabel("<html>" + "Selected items : "+selectedItemText+"</html>");
20         label5 = new JLabel("Total price : " + totalPrice + " $");
21         button1 = new JButton("OK");
22         label1.setBounds(5,50,300,25);
23         label2.setBounds(5,100,300,25);
24         label3.setBounds(5,150,300,25);
25         label4.setBounds(5,200,300,10);
26         label5.setBounds(5,250,400,10);
27         button1.setBounds(50,350,10,25);
28         add(label1);
29         add(Box.createVerticalStrut(10));
```

The screenshot shows a Java code editor window with the file 'orderGui.java' open. The code is a Java Swing application for managing orders. It includes setting up five labels with specific bounds and adding them to a panel, creating a button, and adding an action listener to the button. The button's actionPerformed method calls dispose() and exits the application.

```
22     label1.setBounds(5,50,300,25);
23     label2.setBounds(5,100,300,25);
24     label3.setBounds(5,150,300,25);
25     label4.setPreferredSize(new Dimension(label4.getPreferredSize().width,label4.getPreferredSize().height));
26     label5.setBounds(5,250,400,10);
27     button.setBounds(50,350,10,25);
28     add(label1);
29     add(Box.createVerticalStrut(10));
30     add(label2);
31     add(Box.createVerticalStrut(10));
32     add(label3);
33     add(Box.createVerticalStrut(10));
34     add(label4);
35     add(Box.createVerticalStrut(10));
36     add(label5);
37     add(Box.createVerticalStrut(30));
38     add(button);
39     button.addActionListener(this);
40     setVisible(true);
41 }
42 
43     @Override
44     public void actionPerformed(ActionEvent e) {
45         if (e.getSource() == button) {
46             this.dispose();
47             System.exit(0);
48         }
49     }
50 }
```

3-cartGui class

The screenshot shows a Java code editor window with the file 'cartGui.java' open. The code defines a class 'cartGui' that extends JFrame and implements ActionListener. It uses a ButtonGroup for two radio buttons, initializes frame properties like title, size, and location, and sets the default close operation to EXIT_ON_CLOSE.

```
1 package project;
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 public class cartGui extends JFrame implements ActionListener {
7     ButtonGroup g1 = new ButtonGroup();
8     private JRadioButton radioButton1, radioButton2;
9     private JLabel l11;
10    private int count = 0;
11    private int maxClicks;
12    private float totalprice = 0;
13    private JLabel totalPriceLabel;
14    private String selectedItemsText = "";
15    private String customerid;
16    private String orderid;
17    cartGui(String x, String y, int maxClicks) {
18        customerid = x;
19        orderid = y;
20        this.maxClicks = maxClicks;
21        initializeFrame();
22    }
23    private void initializeFrame() {
24        this.setTitle("E-commerce products ");
25        this.setSize(500, 600);
26        this.setLocationRelativeTo(null);
27        this.setResizable(false);
28        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29        this.setLayout(null);
```

The screenshot shows a Java development environment with two code editors open. The top editor contains the code for `cartGui.java`, and the bottom editor contains the code for `EcommerceSystem.java`. Both editors have tabs for `Source` and `History`, and a toolbar with various icons.

```
cartGui.java
26     this.setLocationRelativeTo(null);
27     this.setResizable(false);
28     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29     this.setLayout(null);
30
31     JLabel l1 = new JLabel("Enter product type ");
32     JLabel l2 = new JLabel("electronic");
33     JLabel l3 = new JLabel("clothing");
34     JLabel l4 = new JLabel("book");
35
36     JButton addbutton1 = new JButton("Add");
37     JButton addbutton2 = new JButton("Add");
38     JButton addbutton3 = new JButton("Add");
39
40     JButton removebutton1 = new JButton("Remove");
41     JButton removebutton2 = new JButton("Remove");
42     JButton removebutton3 = new JButton("Remove");
43
44     JButton orderplace = new JButton("Place order");
45
46     l1.setBounds(10, 20, 160, 40);
47     l2.setBounds(10, 60, 160, 40);
48     l3.setBounds(10, 100, 160, 40);
49     l4.setBounds(10, 140, 160, 40);
50
51     addbutton1.setBounds(210, 60, 60, 30);
52     addbutton2.setBounds(210, 100, 60, 30);
53     addbutton3.setBounds(210, 140, 60, 30);
54
55     removebutton1.setBounds(310, 60, 160, 30);
56     removebutton2.setBounds(310, 100, 160, 30);
57     removebutton3.setBounds(310, 140, 160, 30);
58
59     addbutton1.addActionListener(this);
60     addbutton2.addActionListener(this);
61     addbutton3.addActionListener(this);
62     removebutton1.addActionListener(this);
63     removebutton2.addActionListener(this);
64     removebutton3.addActionListener(this);
65
66     this.add(l1);
67     this.add(l2);
68     this.add(l3);
69     this.add(l4);
70
71     this.setVisible(true);
72
73     totalPriceLabel = new JLabel("Total Price: $0.00");
74     totalPriceLabel.setBounds(70, 190, 200, 30);
75     this.add(totalPriceLabel);
76
77     l11 = new JLabel("Do you want to place an order for the products in your cart?");
78
79     radioButton1 = new JRadioButton("yes");
80     radioButton2 = new JRadioButton("No");
81
82     l11.setBounds(10, 170, 350, 150);
83     radioButton1.setBounds(100, 300, 50, 30);
84     radioButton2.setBounds(250, 300, 50, 30);
85
86     g1.add(radioButton1);
87     g1.add(radioButton2);
88
89     radioButton1.setVisible(true);
90     radioButton2.setVisible(true);
91     l11.setVisible(true);
92
93     this.add(l11);
```

The image shows a Java development environment with two code editors open. The top editor displays the `cartGui.java` file, which contains Java code for adding items to a shopping cart. The bottom editor displays the `EcommerceSystem.java` file, which contains Java code for removing items from the shopping cart.

```
cartGui.java (Top Editor):
83     this.add(l1);
84     this.add(radioButton1);
85     this.add(radioButton2);
86     radioButton1.addActionListener(this);
87     radioButton2.addActionListener(this);
88     addbutton1.addActionListener(new ActionListener() {
89         @Override
90         public void actionPerformed(ActionEvent e) {
91             totalprice += 599.99;
92             updateTotalPriceLabel();
93             recordItem("SmartPhone : price: 599.99$ ", totalprice);
94         }
95     });
96     addbutton2.addActionListener(new ActionListener() {
97         @Override
98         public void actionPerformed(ActionEvent e) {
99             totalprice += 19.99;
100            updateTotalPriceLabel();
101            recordItem("T-shirt : price: 19.99$ ", totalprice);
102        }
103    });
104    addbutton3.addActionListener(new ActionListener() {
105        @Override
106        public void actionPerformed(ActionEvent e) {
107            totalprice += 39.99;
108            updateTotalPriceLabel();
109            recordItem("OOP : price: 39.99$ ", totalprice);
110        }
111    });
}

EcommerceSystem.java (Bottom Editor):
109 }
110 });
111 removebutton1.addActionListener(new ActionListener() {
112     @Override
113     public void actionPerformed(ActionEvent e) {
114         totalprice -= 599.99;
115         updateTotalPriceLabel();
116         recordItem("", totalprice);
117         count -= 1;
118         maxClicks += 1;
119     }
120 });
121 removebutton2.addActionListener(new ActionListener() {
122     @Override
123     public void actionPerformed(ActionEvent e) {
124         totalprice -= 19.99;
125         updateTotalPriceLabel();
126         recordItem("", totalprice);
127         count -= 1;
128         maxClicks += 1;
129     }
130 });
131 removebutton3.addActionListener(new ActionListener() {
132     @Override
133     public void actionPerformed(ActionEvent e) {
134         totalprice -= 39.99;
135         updateTotalPriceLabel();
136         recordItem("", totalprice);
137         count -= 1;
138         maxClicks += 1;
139     }
140 });


```

The image shows two Java code editors side-by-side in a development environment. Both editors have tabs at the top for EcommerceSystem.java, orderGui.java, and EcommerceGui.java, with cartGui.java currently active.

cartGui.java:

```
130 }});
131     removebutton3.addActionListener(new ActionListener() {
132         @Override
133         public void actionPerformed(ActionEvent e) {
134             totalprice -= 39.99;
135             updateTotalPriceLabel();
136             recordItem("", totalprice);
137             count -= 1;
138             maxClicks += 1;
139         }
140     });
141 }
142 private void updateTotalPriceLabel() {
143     totalPriceLabel.setText("Total Price: $" + String.format("%.2f", totalprice));
144 }
145 private void recordItem(String itemName, float totalprice) {
146     selectedItemsText += itemName + " , ";
147 }
148 private void disableButtons() {
149     Component[] components = this.getContentPane().getComponents();
150     for (Component component : components) {
151         if (component instanceof JButton) {
152             JButton button = (JButton) component;
153             button.setEnabled(false);
154         }
155     }
156 }
157 @Override
158 public void actionPerformed(ActionEvent e) {
159     if (e.getSource() instanceof JButton) {
```

orderGui.java:

```
149     Component[] components = this.getContentPane().getComponents();
150     for (Component component : components) {
151         if (component instanceof JButton) {
152             JButton button = (JButton) component;
153             button.setEnabled(false);
154         }
155     }
156 }
157 @Override
158 public void actionPerformed(ActionEvent e) {
159     if (e.getSource() instanceof JButton) {
160         JButton button = (JButton) e.getSource();
161         count++;
162         if (count >= maxClicks) {
163             disableButtons();
164         }
165         if (e.getSource() == radioButton1) {
166             String customerId = customerId;
167             String orderedItem = orderedItem;
168             Guiorder = new orderGui(customerId, orderedItem, selectedItemsText, totalprice);
169             dispose();
170         } else if (e.getSource() == radioButton2) {
171             JOptionPane.showMessageDialog(null, "order not placed");
172             this.dispose();
173         }
174     }
175 }
176 }
```

GUI output

