

TEST CASE

The image shows two screenshots of a web application interface. The top screenshot is a login page with a green background and a white header bar. The header bar contains the text "WELCOME TO THE E-COMMERCE SYSTEM!". Below the header, there are three input fields: "ID:" with the value "23012107", "Name:" with the value "Noureldin Ashraf Ahmed", and "Address:" with the value "Milkyway, Solar System, Earth". Below these fields is a button labeled "ENTER". The bottom screenshot is a product selection page with a green background and a white header bar. The header bar contains the text "WELCOME, Noureldin Ashraf Ahmed". Below the header, there is a section titled "Available products:" with a link "Click to choose". Below this link are three buttons: "OOP | \$39.99", "Smartphone | \$599.9", and "T-Shirt | \$19.99". Below these buttons is a section titled "CHOSEN PRODUCTS:". Below this section is a table with four rows: "Smartphone", "T-Shirt", "Smartphone", and "OOP". At the bottom of the page, there are three buttons: "CANCEL ORDER", "TOTAL: \$1259.78", and "CREATE ORDER".

WELCOME TO THE E-COMMERCE SYSTEM!

ID: 23012107

Name: Noureldin Ashraf Ahmed

Address: Milkyway, Solar System, Earth

ENTER

WELCOME, Noureldin Ashraf Ahmed

Available products:
[Click to choose](#)

OOP | \$39.99 Smartphone | \$599.9 T-Shirt | \$19.99

CHOSEN PRODUCTS:

Smartphone
T-Shirt
Smartphone
OOP

CANCEL ORDER TOTAL: \$1259.78 CREATE ORDER

—□×

THANK YOU FOR SHOPPING HERE, COME BACK SOON!

Your order details:

Products::

Smartphone

T-Shirt

Smartphone

OOP

Order ID: 1
Customer ID: 23012107
Total Price: \$1259.78

PRODUCT CLASS

```
3 public class Product {
4     private int productId;
5     private String name;
6     private double price;
7
8     public Product() {}
9     public Product(int productId, String name, double price) {
10         if (productId < 0)
11             this.productId = (-1 * productId);
12         else
13             this.productId = productId;
14
15         this.name = name;
16
17         if (price < 0)
18             this.price = (-1 * price);
19         else
20             this.price = price;
21     }
22
23     > public int getProductId() { return productId; }
26     public void setProductId(int productId) {
27         if (productId < 0)
28             this.productId = (-1 * productId);
29         else
30             this.productId = productId;
31     }
32
33     > public String getName() { return name; }
36     > public void setName(String name) { this.name = name; }
39
40     > public double getPrice() { return price; }
43     public void setPrice(double price) {
44         if (price < 0)
45             this.price = (-1 * price);
46         else
47             this.price = price;
48     }
49 }
```

ELECTRONIC PRODUCT CLASS

```
3 public class ElectronicProduct extends Product{
4     private String brand;
5     private int warrantyPeriod;
6
7     public ElectronicProduct() {}
8     public ElectronicProduct(int productId, String name, double price, String brand, int warrantyPeriod) {
9         super(productId, name, price);
10        this.brand = brand;
11
12        if (warrantyPeriod < 0)
13            this.warrantyPeriod = (-1 * warrantyPeriod);
14        else
15            this.warrantyPeriod = warrantyPeriod;
16    }
17
18    > public String getBrand() { return brand; }
21    > public void setBrand(String brand) { this.brand = brand; }
24
25    > public int getWarrantyPeriod() { return warrantyPeriod; }
28    public void setWarrantyPeriod(int warrantyPeriod) {
29        if (warrantyPeriod < 0)
30            this.warrantyPeriod = (-1 * warrantyPeriod);
31        else
32            this.warrantyPeriod = warrantyPeriod;
33    }
34 }
```

CLOTHING PRODUCT CLASS

```
3 public class ClothingProduct extends Product{
4     private String size;
5     private String fabric;
6
7     public ClothingProduct() {}
8     public ClothingProduct(int productId, String name, double price, String size, String fabric) {
9         super(productId, name, price);
10        this.size = size;
11        this.fabric = fabric;
12    }
13
14    > public String getSize() { return size; }
17    > public void setSize(String size) { this.size = size; }
20
21    > public String getFabric() { return fabric; }
24    > public void setFabric(String fabric) { this.fabric = fabric; }
27 }
```

BOOK PRODUCT CLASS

```
2
3 public class BookProduct extends Product{
4     private String author;
5     private String publisher;
6
7     public BookProduct() {}
8     public BookProduct(int productId, String name, double price, String author, String publisher) {
9         super(productId, name, price);
10        this.author = author;
11        this.publisher = publisher;
12    }
13
14    > public String getAuthor() { return author; }
17    > public void setAuthor(String author) { this.author = author; }
20
21    > public String getPublisher() { return publisher; }
24    > public void setPublisher(String publisher) { this.publisher = publisher; }
27 }
```

CUSTOMER CLASS

```
3 public class Customer {
4     private int customerId;
5     private String name;
6     private String address;
7
8     public Customer(){}
9     public Customer(int customerId, String name, String address) {
10         if (customerId < 0)
11             this.customerId = (-1 * customerId);
12         else
13             this.customerId = customerId;
14
15         this.name = name;
16         this.address = address;
17     }
18
19 > public int getCustomerId() { return customerId; }
22 public void setCustomerId(int customerId) {
23     if (customerId < 0)
24         this.customerId = (-1 * customerId);
25     else
26         this.customerId = customerId;
27 }
28
29 > public String getName() { return name; }
32 > public void setName(String name) { this.name = name; }
35
36 > public String getAddress() { return address; }
39 > public void setAddress(String address) { this.address = address; }
42 }
```

CART CLASS

```
7 public class Cart {
8     private int customerId;
9     private ArrayList<Product> products;
10
11     public Cart(int customerId) {
12         if (customerId < 0)
13             this.customerId = (-1 * customerId);
14         else
15             this.customerId = customerId;
16
17         products = new ArrayList<>();
18     }
19
20     public void addProduct(Product product) { products.add(product); }
21
22     public void removeProduct(Product product) { products.remove(product); }
23
24     public void removeAllProducts() { products.clear(); }
25
26     public int getProductsAmount() { return products.size(); }
27
28
29     public double calculatePrice(){
30         double totalPrice = 0;
31         for (Product i : products){
32             totalPrice+=i.getPrice();
33         }
34         return totalPrice;
35     }
36
37
38     public Order placeOrder(){
39         Order order = new Order(customerId, orderId: 1, products, calculatePrice());
40         return order;
41     }
42
43
44     public int getCustomerId() { return customerId; }
45
46     public void setCustomerId(int customerId) {
47         if (customerId < 0)
48             this.customerId = (-1 * customerId);
49         else
50             this.customerId = customerId;
51     }
52 }
53 }
```


ORDER CLASS

```
7 public class Order {
8     private int customerId;
9     private int orderId;
10    private ArrayList<Product> products;
11    private double totalPrice;
12
13    public Order(int customerId, int orderId, ArrayList<Product> products, double totalPrice){
14        if (customerId < 0)
15            this.customerId = (-1 * customerId);
16        else
17            this.customerId = customerId;
18
19        if (orderId < 0)
20            this.orderId = (-1 * orderId);
21        else
22            this.orderId = orderId;
23
24        this.products = products;
25
26        if (totalPrice < 0)
27            this.totalPrice = (-1 * totalPrice);
28        else
29            this.totalPrice = totalPrice;
30    }
31
32    public String printOrderInfo(){
33        String s = "Order ID: " + orderId+
34            "\nCustomer ID: " + customerId+
35            "\nTotal Price: $" + totalPrice;
36        return s;
37    }
38    public ArrayList<String> getProducts(){
39        ArrayList<String> productNames = new ArrayList<>();
40        for (Product i : products){
41            productNames.add(i.getName());
42        }
43        return productNames;
44    }
45 }
```

ECOMMERCE SYSTEM CLASS

```
16 public class EcommerceSystem extends Application {
17     public static ElectronicProduct smartphone;
18     public static ClothingProduct tShirt;
19     public static BookProduct 00P;
20     public static Customer customer;
21     public static Cart cart;
22     public static Order order;
23     private static Scene scene;
24
25     public static void main(String[] args) { launch(); }
26
27
28
29
30     @Override
31     public void start(Stage stage) throws IOException {
32         scene = new Scene(loadFXML("primary"));
33         stage.setScene(scene);
34         stage.show();
35     }
36
37     public static void setRoot(String fxml) throws IOException {
38         scene.setRoot(loadFXML(fxml));
39     }
40
41     private static Parent loadFXML(String fxml) throws IOException {
42         FXMLLoader fxmlLoader = new FXMLLoader(EcommerceSystem.class.getResource("name: fxml + ".fxml"));
43         return fxmlLoader.load();
44     }
45
46 }
```

CLASSES REQUIRED FOR JAVAFX:

PRIMARYCONTROLLER CLASS

```
27 public class PrimaryController {
28     @FXML
29     </> private Label errorMessage;
30     @FXML
31     private Button enterButton;
32     @FXML
33     </> private TextField addressTextfield;
34     @FXML
35     </> private TextField idTextfield;
36     @FXML
37     </> private TextField nameTextfield;
38     @FXML
39     public void initialize() throws IOException {
40         smartphone = new ElectronicProduct( productId: 1, name: "Smartphone", price: 599.9, brand: "Samsung", warrantyPeriod: 1);
41         tShirt = new ClothingProduct( productId: 2, name: "T-Shirt", price: 19.99, size: "Medium", fabric: "Cotton");
42         OOP = new BookProduct( productId: 3, name: "OOP", price: 39.99, author: "O'Reilly", publisher: "X Publications");
43     }
44     @FXML
45     void onClickEnter(ActionEvent event) throws IOException {
46
47         if (!hasErrors()){
48             int id = Integer.parseInt(idTextfield.getText());
49             String name = nameTextfield.getText();
50             String address = addressTextfield.getText();
51             customer = new Customer(id, name, address);
52             cart = new Cart(id);
53             switchToSecondary();
54         }
55     }
```

CONTINUE THIS CLASS IN NEXT PAGE...

```

59     private boolean hasErrors(){
60         //CHECK IF EMPTY
61         String id = idTextfield.getText();
62         String address = addressTextfield.getText();
63         String name = nameTextfield.getText();
64         if (name.isEmpty() || address.isEmpty() || id.isEmpty()){
65             errorMessage.setText("ERROR: FILL IN YOUR INFORMATION!");
66             return true;
67         }
68         if(!isNumeric(id)){
69             errorMessage.setText("ERROR: ENTER AN APPROPRIATE NUMERIC ID!");
70             return true;
71         }
72         return false;
73     }
74     private boolean isNumeric(String str) {
75         if (str == null) {
76             return false;
77         }
78         try {
79             double d = Integer.parseInt(str);
80         } catch (NumberFormatException nfe) {
81             return false;
82         }
83         return true;
84     }
85     private void switchToSecondary() throws IOException {
86         EcommerceSystem.setRoot("secondary");
87     }
88 }

```

SECONDARYCONTROLLER CLASS

```
27 public class SecondaryController {
28     @FXML
29     </> private ListView<String> productListView;
30     @FXML
31     </> private Label welcomeMessage;
32     @FXML
33     </> private Label totalMessage;
34
35     @FXML
36     public void initialize() throws IOException {
37         String welcomeMsg = welcomeMessage.getText().replaceAll(regex: "replace", customer.getName());
38         welcomeMessage.setText(welcomeMsg);
39     }
40     @FXML
41     void onClickOOP(ActionEvent event) {
42         cart.addProduct(OOP);
43         addProductToViewList(OOP.getName());
44         totalMessage.setText("TOTAL: $" + cart.calculatePrice());
45     }
46     @FXML
47     void onClickSmartphone(ActionEvent event) {
48         cart.addProduct(smartphone);
49         addProductToViewList(smartphone.getName());
50         totalMessage.setText("TOTAL: $" + cart.calculatePrice());
51     }
52     @FXML
53     void onClickTshirt(ActionEvent event) {
54         cart.addProduct(tShirt);
55         addProductToViewList(tShirt.getName());
56         totalMessage.setText("TOTAL: $" + cart.calculatePrice());
57     }
58     @FXML
59     void onClickCancelOrder(ActionEvent event) {
60         cart.removeAllProducts();
61         productListView.getItems().clear();
62         totalMessage.setText("TOTAL: $" + cart.calculatePrice());
63     }
64
65     @FXML
66     void onClickCreateOrder(ActionEvent event) throws IOException {
67         order = cart.placeOrder();
68         switchToShow();
69     }
70     private void addProductToViewList(String productName){
71         productListView.getItems().add(productName);
72     }
73     private void switchToShow() throws IOException {
74         EcommerceSystem.setRoot("show");
75     }
76 }
```

SHOWCONTROLLER CLASS

```
15 public class ShowController {
16     @FXML
17     </> private ListView<String> productListView;
18     @FXML
19     </> private Label detailsMessage;
20
21     @FXML
22     public void initialize() throws IOException {
23         productListView.getItems().addAll(order.getProducts());
24         detailsMessage.setText(order.printOrderInfo());
25     }
26 }
```