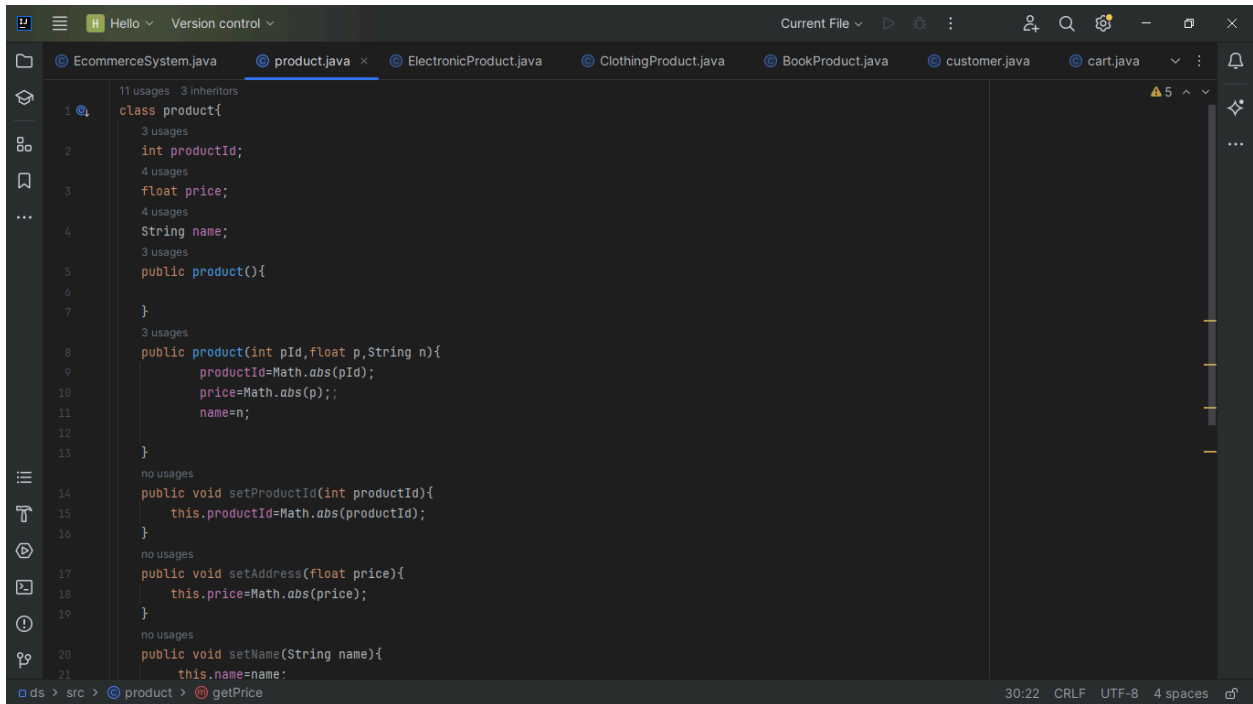


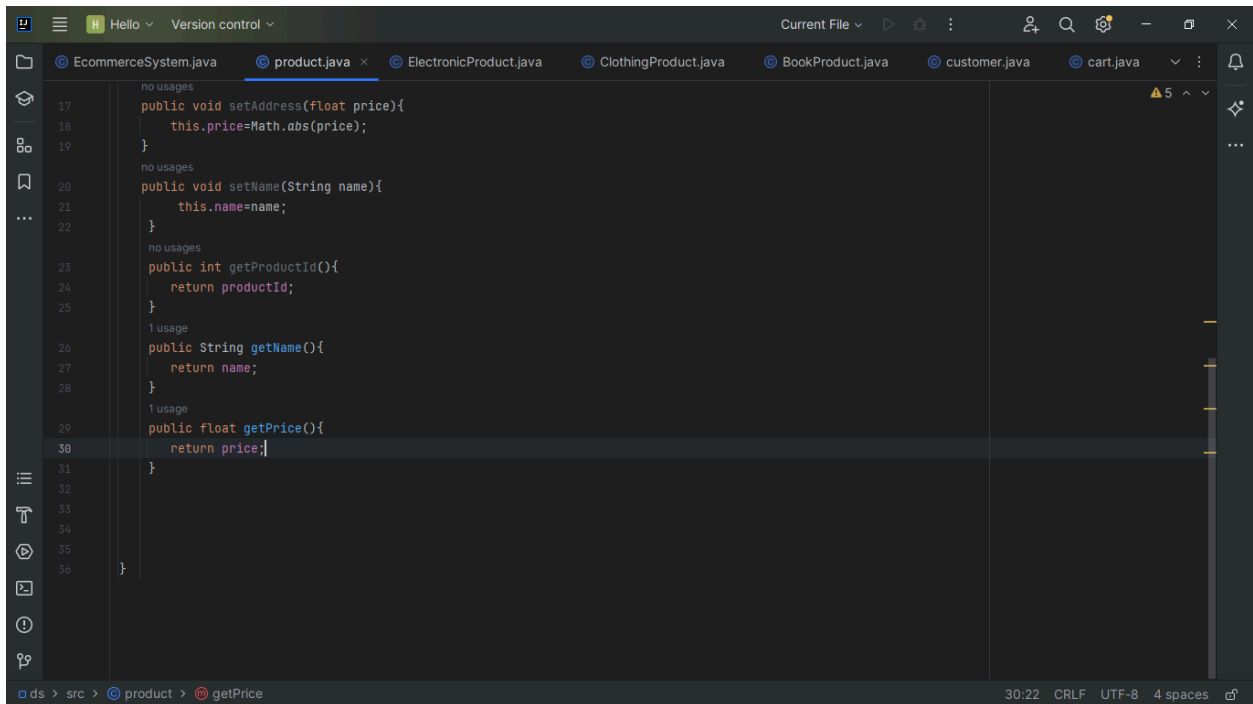
Product class:



This screenshot shows the IDE interface with the `product.java` file open. The class `product` is defined with the following methods and attributes:

```
class product{
    int productId;
    float price;
    String name;
    public product(){
    }
    public product(int pId,float p,String n){
        productId=Math.abs(pId);
        price=Math.abs(p);
        name=n;
    }
    public void setProductId(int productId){
        this.productId=Math.abs(productId);
    }
    public void setAddress(float price){
        this.price=Math.abs(price);
    }
    public void setName(String name){
        this.name=name;
    }
}
```

The IDE shows 11 usages and 3 inheritors for the `product` class. The status bar at the bottom indicates the file is at `ds > src > product > @getPrice`, with a cursor at line 30:22, CRLF, UTF-8, and 4 spaces.



This screenshot shows the IDE interface with the `product.java` file open, focusing on the getter methods. The methods shown are:

```
public void setAddress(float price){
    this.price=Math.abs(price);
}

public void setName(String name){
    this.name=name;
}

public int getProductId(){
    return productId;
}

public String getName(){
    return name;
}

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

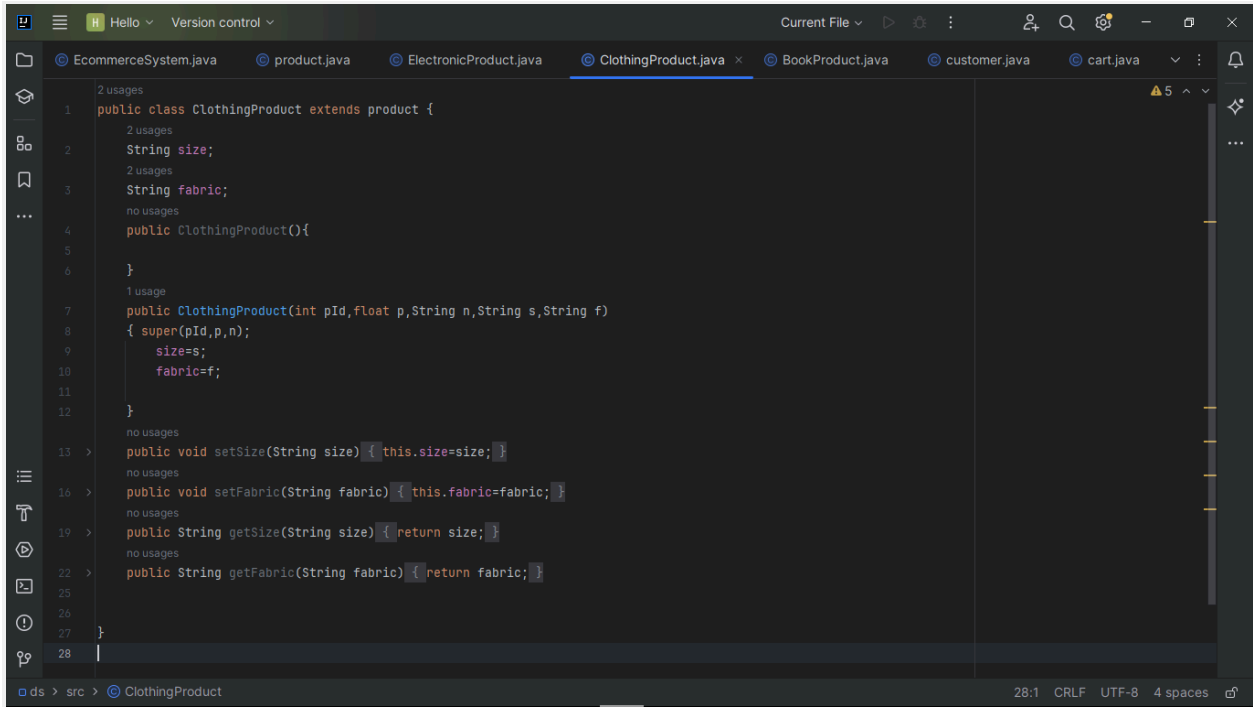
The IDE shows 1 usage for the `getName` method and 1 usage for the `getPrice` method. The status bar at the bottom indicates the file is at `ds > src > product > @getPrice`, with a cursor at line 30:22, CRLF, UTF-8, and 4 spaces.

ElectronicProduct class:

```
1 public class ElectronicProduct extends product {
2     String brand;
3     int warrantyPeriod;
4     public ElectronicProduct(){
5     }
6     public ElectronicProduct(int pId,float p,String n,String b,int w)
7     { super(pId,p,n);
8       brand=b;
9       warrantyPeriod = Math.abs(w);
10    }
11    public void setBrand(String brand){
12        this.brand=brand;
13    }
14    public void setWarrantyPeriod(int warrantyPeriod){
15        this.warrantyPeriod=Math.abs(warrantyPeriod);
16    }
17    public int getWarrantyPeriod(){
18        return warrantyPeriod;
19    }
20    public String getBrand(){
21        return brand;
22    }
23 }
```

```
5 }
6 public ElectronicProduct(int pId,float p,String n,String b,int w)
7 { super(pId,p,n);
8   brand=b;
9   warrantyPeriod = Math.abs(w);
10 }
11 public void setBrand(String brand){
12     this.brand=brand;
13 }
14 public void setWarrantyPeriod(int warrantyPeriod){
15     this.warrantyPeriod=Math.abs(warrantyPeriod);
16 }
17 public int getWarrantyPeriod(){
18     return warrantyPeriod;
19 }
20 public String getBrand(){
21     return brand;
22 }
23 }
24 }
```

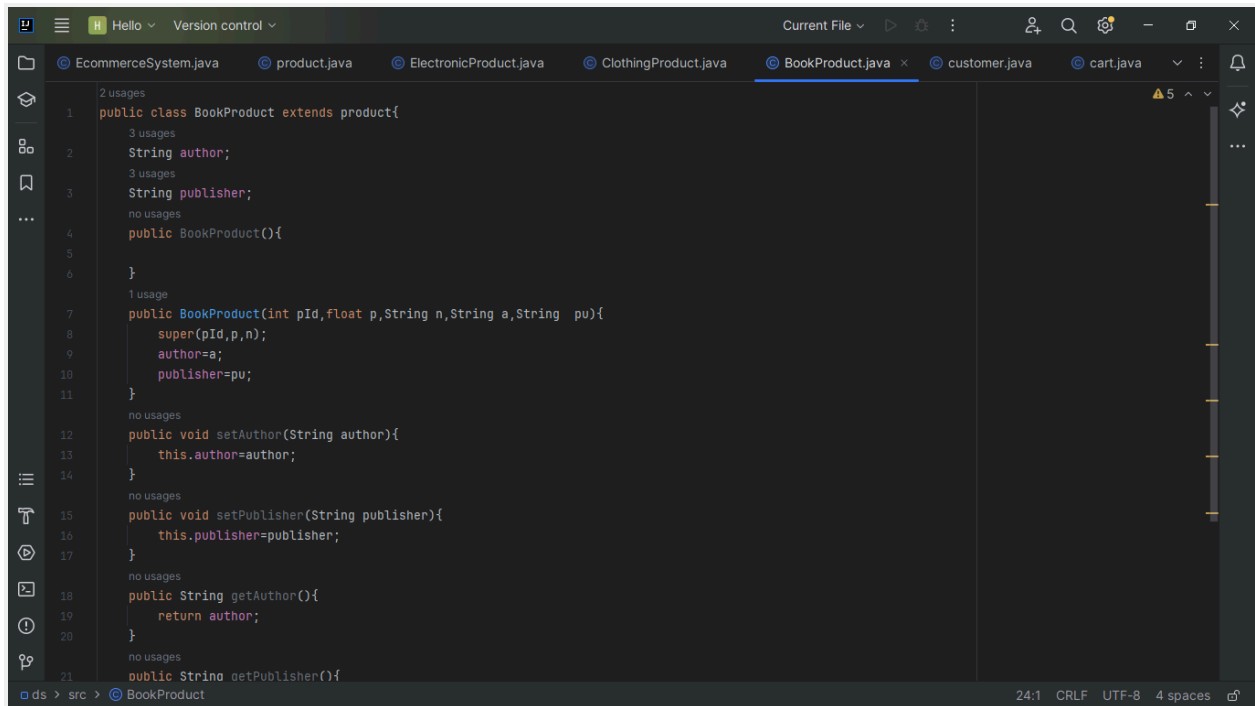
ClothingProduct class:



```
1 2 usages
2 public class ClothingProduct extends product {
3     2 usages
4     String size;
5     2 usages
6     String fabric;
7     no usages
8     public ClothingProduct(){
9
10    }
11    1 usage
12    public ClothingProduct(int pId,float p,String n,String s,String f)
13    { super(pId,p,n);
14        size=s;
15        fabric=f;
16    }
17    no usages
18    public void setSize(String size) { this.size=size; }
19    no usages
20    public void setFabric(String fabric) { this.fabric=fabric; }
21    no usages
22    public String getSize(String size) { return size; }
23    no usages
24    public String getFabric(String fabric) { return fabric; }
25
26 }
27
28
```

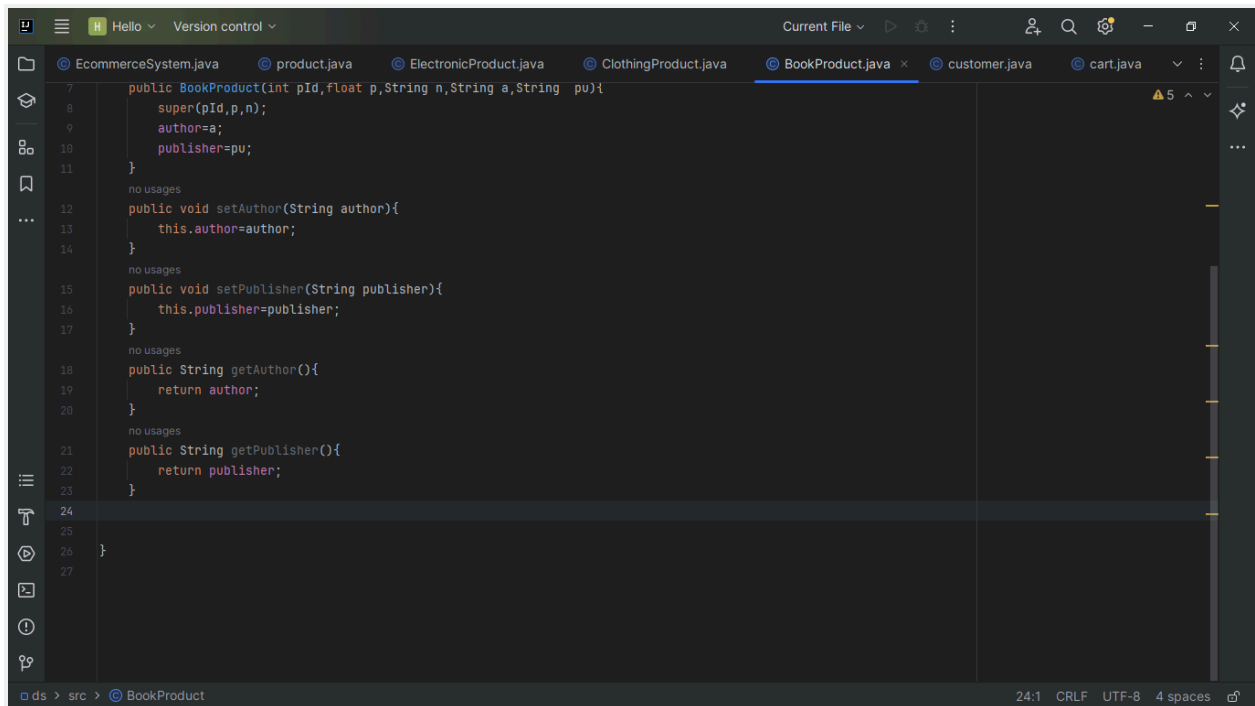
ds > src > ClothingProduct 28:1 CRLF UTF-8 4 spaces

BookProduct class:



This screenshot shows the full definition of the `BookProduct` class in an IDE. The class extends `product` and includes attributes for `author` and `publisher`, along with their respective getters and setters. The IDE interface includes a sidebar with project navigation, a top toolbar with various icons, and a status bar at the bottom indicating the file path and encoding.

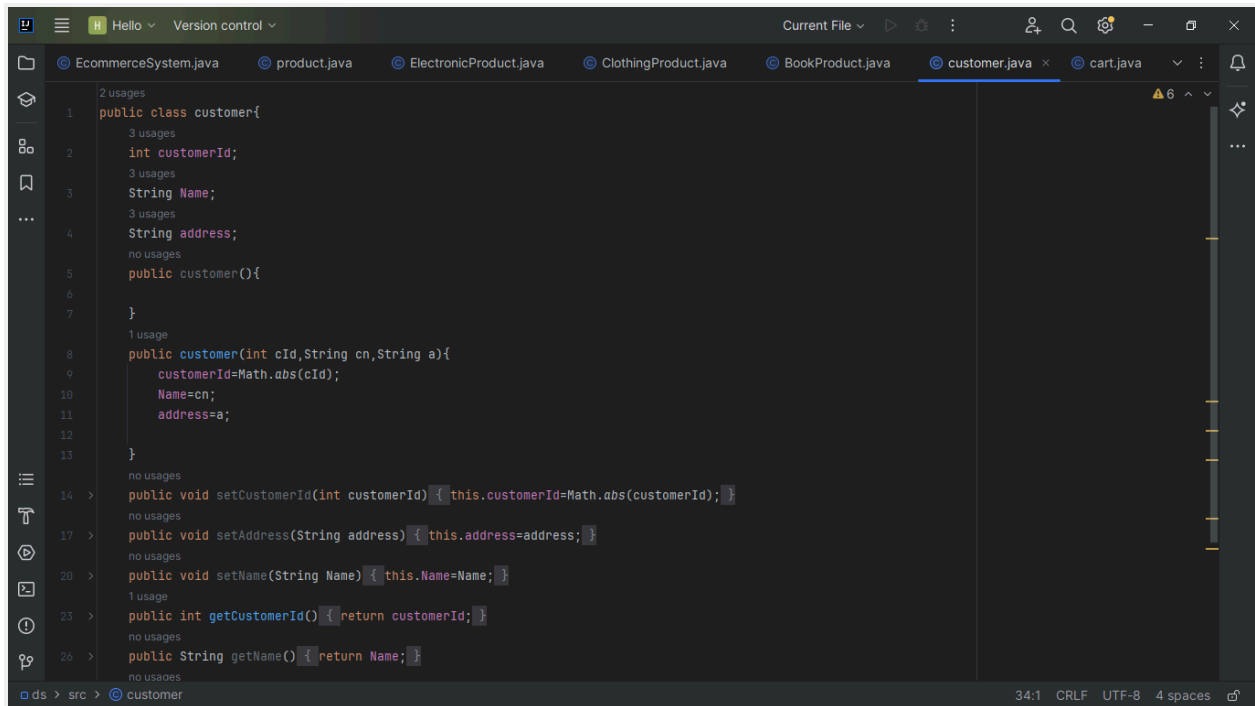
```
1 2 usages
2 public class BookProduct extends product{
3     3 usages
4     String author;
5     3 usages
6     String publisher;
7     no usages
8     public BookProduct(){
9
10 }
11
12 1 usage
13 public BookProduct(int pid,float p,String n,String a,String pu){
14     super(pid,p,n);
15     author=a;
16     publisher=pu;
17 }
18
19 no usages
20 public void setAuthor(String author){
21     this.author=author;
22 }
23
24 no usages
25 public void setPublisher(String publisher){
26     this.publisher=publisher;
27 }
28
29 no usages
30 public String getAuthor(){
31     return author;
32 }
33
34 no usages
35 public String getPublisher(){
36     return publisher;
37 }
```



This screenshot shows the same IDE interface as the first, but with the `BookProduct` class body collapsed. The code is now visible from line 7 to line 23, showing the constructor and the setter and getter methods. The status bar at the bottom remains the same.

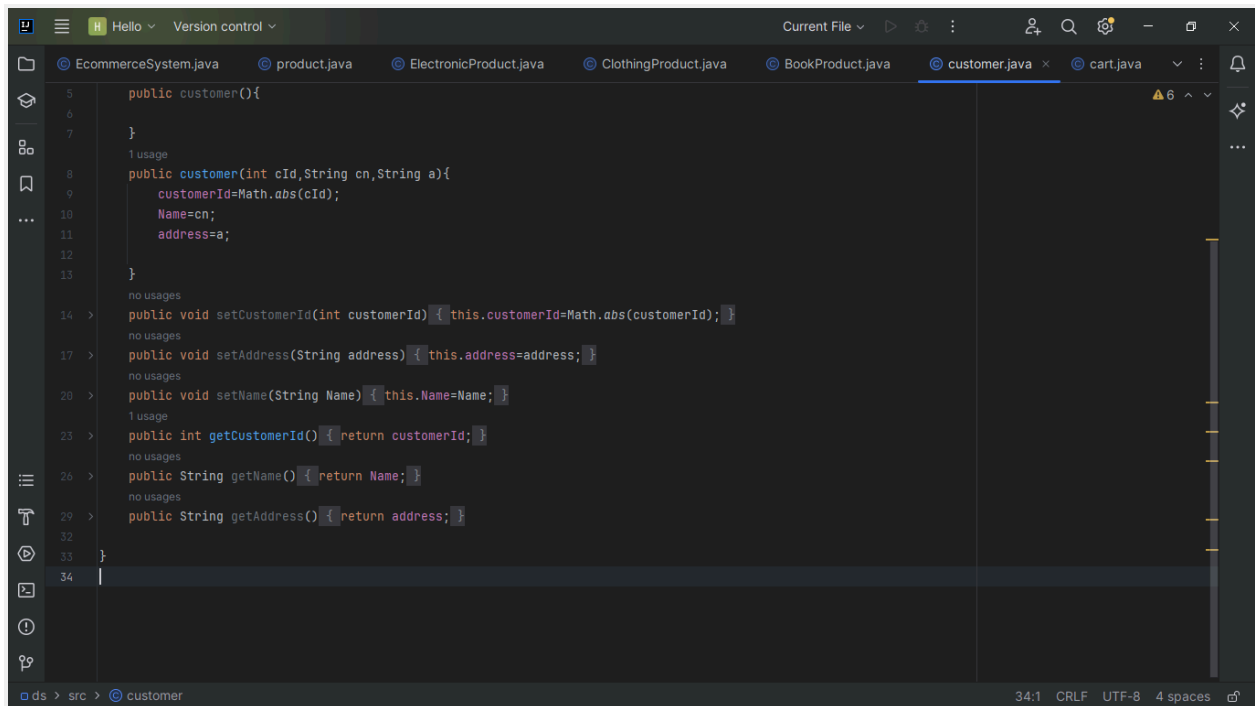
```
7 public BookProduct(int pid,float p,String n,String a,String pu){
8     super(pid,p,n);
9     author=a;
10    publisher=pu;
11 }
12
13 no usages
14 public void setAuthor(String author){
15     this.author=author;
16 }
17
18 no usages
19 public void setPublisher(String publisher){
20     this.publisher=publisher;
21 }
22
23 no usages
24 public String getAuthor(){
25     return author;
26 }
27
28 no usages
29 public String getPublisher(){
30     return publisher;
31 }
```

Customer class:



This screenshot shows the initial state of the `customer.java` file in an IDE. The code defines a `customer` class with attributes `customerId`, `Name`, and `address`, and methods for setting and getting these values. The IDE interface includes a top toolbar with icons for file operations, a sidebar with project navigation, and a status bar at the bottom indicating the file path and encoding.

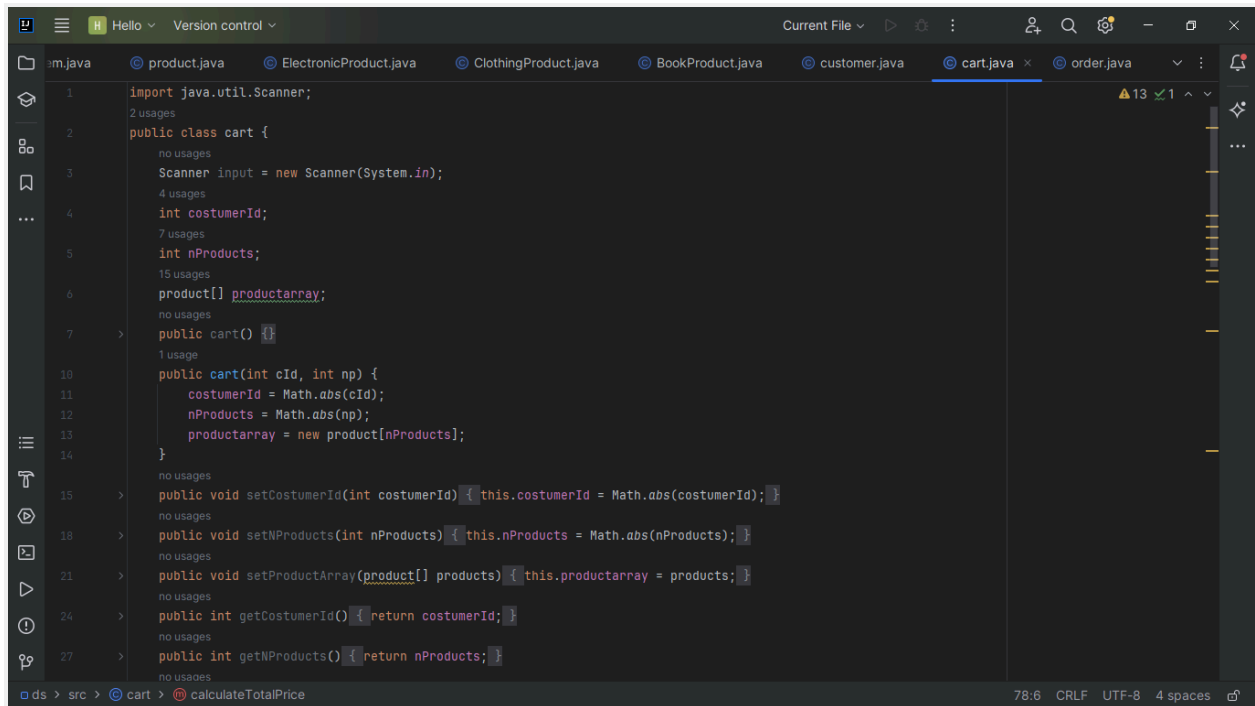
```
1 2 usages
2 public class customer{
3     3 usages
4     int customerId;
5     3 usages
6     String Name;
7     3 usages
8     String address;
9     no usages
10    public customer(){
11
12    }
13
14    1 usage
15    public customer(int cId,String cn,String a){
16        customerId=Math.abs(cId);
17        Name=cn;
18        address=a;
19    }
20
21    no usages
22    public void setCustomerId(int customerId){ this.customerId=Math.abs(customerId); }
23    no usages
24    public void setAddress(String address){ this.address=address; }
25    no usages
26    public void setName(String Name){ this.Name=Name; }
27    1 usage
28    public int getCustomerId(){ return customerId; }
29    no usages
30    public String getName(){ return Name; }
31    no usages
32
33
34
```



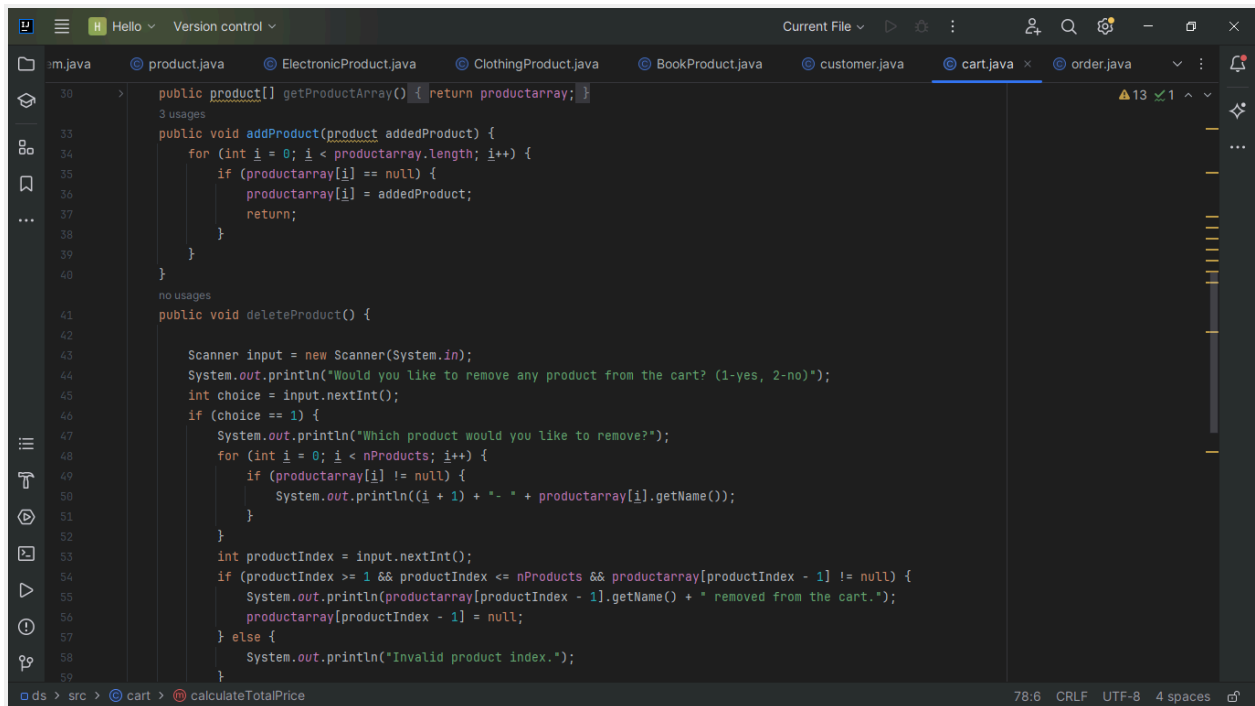
This screenshot shows the `customer.java` file after an update. A new method, `getAddress()`, has been added to the class. The IDE interface remains the same, showing the same project structure and status bar information.

```
5 public customer(){
6
7 }
8
9 1 usage
10 public customer(int cId,String cn,String a){
11     customerId=Math.abs(cId);
12     Name=cn;
13     address=a;
14 }
15
16 no usages
17 public void setCustomerId(int customerId){ this.customerId=Math.abs(customerId); }
18 no usages
19 public void setAddress(String address){ this.address=address; }
20 no usages
21 public void setName(String Name){ this.Name=Name; }
22 1 usage
23 public int getCustomerId(){ return customerId; }
24 no usages
25 public String getName(){ return Name; }
26 no usages
27 public String getAddress(){ return address; }
28
29
30
31
32
33 }
34
```

Cart class:



```
1 import java.util.Scanner;
2 public class cart {
3     Scanner input = new Scanner(System.in);
4     int costumerId;
5     int nProducts;
6     product[] productarray;
7     public cart() {}
8     public cart(int cId, int np) {
9         costumerId = Math.abs(cId);
10        nProducts = Math.abs(np);
11        productarray = new product[nProducts];
12    }
13    public void setCostumerId(int costumerId) { this.costumerId = Math.abs(costumerId); }
14    public void setNProducts(int nProducts) { this.nProducts = Math.abs(nProducts); }
15    public void setProductArray(product[] products) { this.productarray = products; }
16    public int getCostumerId() { return costumerId; }
17    public int getNProducts() { return nProducts; }
```



```
18 public product[] getProductArray() { return productarray; }
19 public void addProduct(product addedProduct) {
20     for (int i = 0; i < productarray.length; i++) {
21         if (productarray[i] == null) {
22             productarray[i] = addedProduct;
23             return;
24         }
25     }
26 }
27 public void deleteProduct() {
28     Scanner input = new Scanner(System.in);
29     System.out.println("Would you like to remove any product from the cart? (1=yes, 2=no)");
30     int choice = input.nextInt();
31     if (choice == 1) {
32         System.out.println("Which product would you like to remove?");
33         for (int i = 0; i < nProducts; i++) {
34             if (productarray[i] != null) {
35                 System.out.println((i + 1) + "- " + productarray[i].getName());
36             }
37         }
38         int productIndex = input.nextInt();
39         if (productIndex >= 1 && productIndex <= nProducts && productarray[productIndex - 1] != null) {
40             System.out.println(productarray[productIndex - 1].getName() + " removed from the cart.");
41             productarray[productIndex - 1] = null;
42         } else {
43             System.out.println("Invalid product index.");
44         }
45     }
46 }
```

```

40     }
41     no usages
42     public void deleteProduct() {
43
44         Scanner input = new Scanner(System.in);
45         System.out.println("Would you like to remove any product from the cart? (1=yes, 2=no)");
46         int choice = input.nextInt();
47         if (choice == 1) {
48             System.out.println("Which product would you like to remove?");
49             for (int i = 0; i < nProducts; i++) {
50                 if (productarray[i] != null) {
51                     System.out.println((i + 1) + "- " + productarray[i].getName());
52                 }
53             }
54             int productIndex = input.nextInt();
55             if (productIndex >= 1 && productIndex <= nProducts && productarray[productIndex - 1] != null) {
56                 System.out.println(productarray[productIndex - 1].getName() + " removed from the cart.");
57                 productarray[productIndex - 1] = null;
58             } else {
59                 System.out.println("Invalid product index.");
60             }
61         }
62         product[] remainingProducts = productarray;
63         System.out.println("Remaining products in the cart:");
64         for (int i = 0; i < remainingProducts.length; i++) {
65             product product = remainingProducts[i];
66             if (product != null) {
67                 System.out.println(product.getName() + " - $" + product.getPrice());
68             }
69         }
70     }
71 }
72
73 ds > src > cart > calculateTotalPrice
78:6 CRLF UTF-8 4 spaces

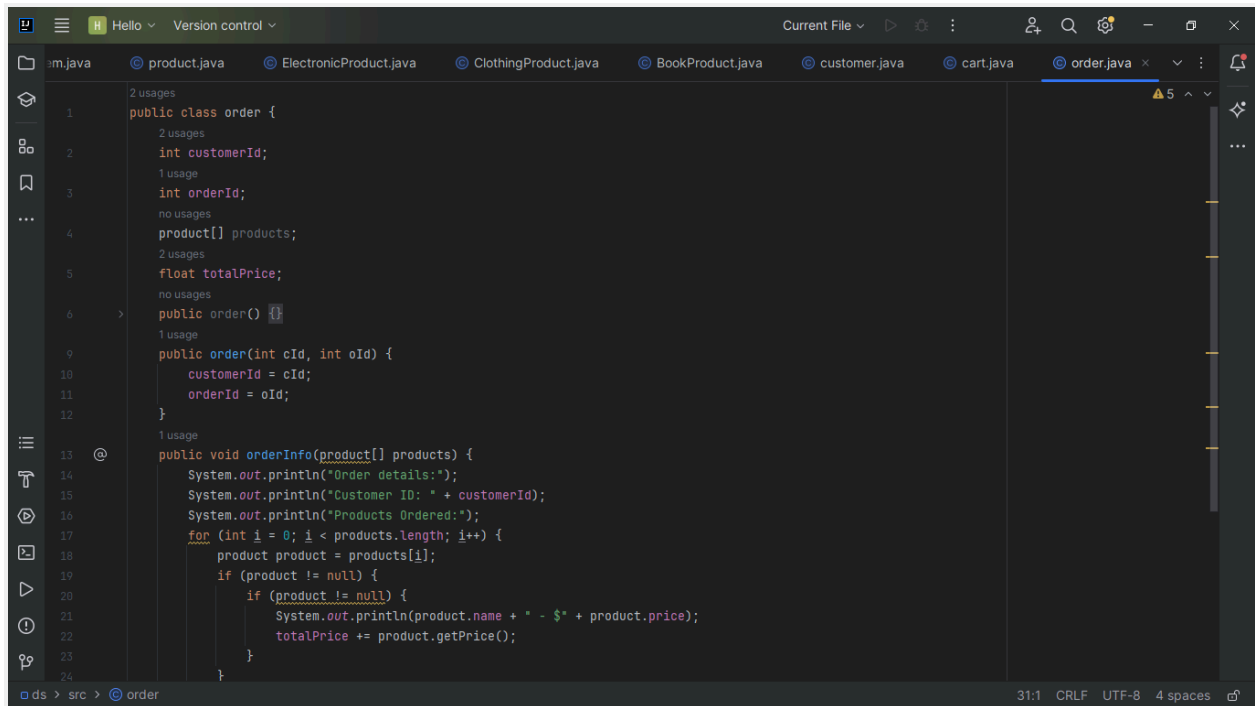
```

```

69     }
70     1 usage
71     public float calculateTotalPrice() {
72         float totalPrice = 0;
73         for (int i = 0; i < nProducts; i++) {
74             if (productarray[i] != null) {
75                 totalPrice += productarray[i].getPrice();
76             }
77         }
78         return totalPrice;
79     }
80     1 usage
81     public void placeOrder() {
82         Scanner input = new Scanner(System.in);
83         System.out.println("Your total price is: $" + calculateTotalPrice());
84         System.out.println("Would you like to place the order? (1=yes, 2=no)");
85         int choice = input.nextInt();
86         if (choice == 1) {
87             System.out.println("Order placed successfully!");
88             order order = new order(customerId, oldId);
89             order.orderInfo(productarray);
90         } else if (choice == 2) {
91             System.out.println("Order canceled.");
92         } else {
93             System.out.println("Invalid choice.");
94         }
95     }
96 }
97
98 ds > src > cart
97:1 CRLF UTF-8 4 spaces

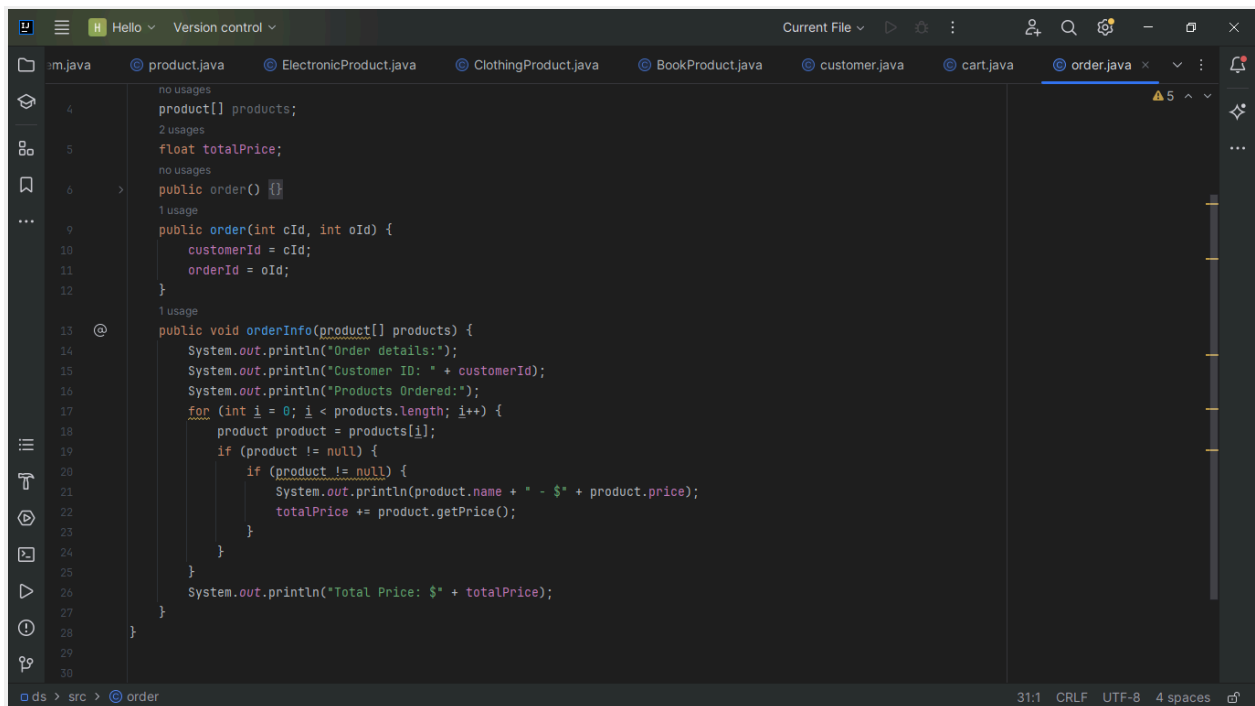
```

Order class:



This screenshot shows the 'order.java' file in an IDE. The code defines a public class 'order' with several attributes and methods. The attributes include 'customerId', 'orderId', 'products', and 'totalPrice'. There are two constructors: a no-argument constructor and a constructor that takes 'cId' and 'oId'. The 'orderInfo' method prints order details and iterates through the 'products' array to calculate the total price.

```
1 public class order {
2     int customerId;
3     int orderId;
4     product[] products;
5     float totalPrice;
6     public order() {
7     }
8     public order(int cId, int oId) {
9         customerId = cId;
10        orderId = oId;
11    }
12    public void orderInfo(product[] products) {
13        System.out.println("Order details:");
14        System.out.println("Customer ID: " + customerId);
15        System.out.println("Products Ordered:");
16        for (int i = 0; i < products.length; i++) {
17            product product = products[i];
18            if (product != null) {
19                if (product != null) {
20                    System.out.println(product.name + " - $" + product.price);
21                    totalPrice += product.getPrice();
22                }
23            }
24        }
25    }
26 }
```



This screenshot shows the same 'order.java' file, but with a different view of the code. The 'products' array is declared at the top. The 'order' constructor is shown with its parameters. The 'orderInfo' method is shown with its implementation, including the loop to calculate the total price and the final print statement.

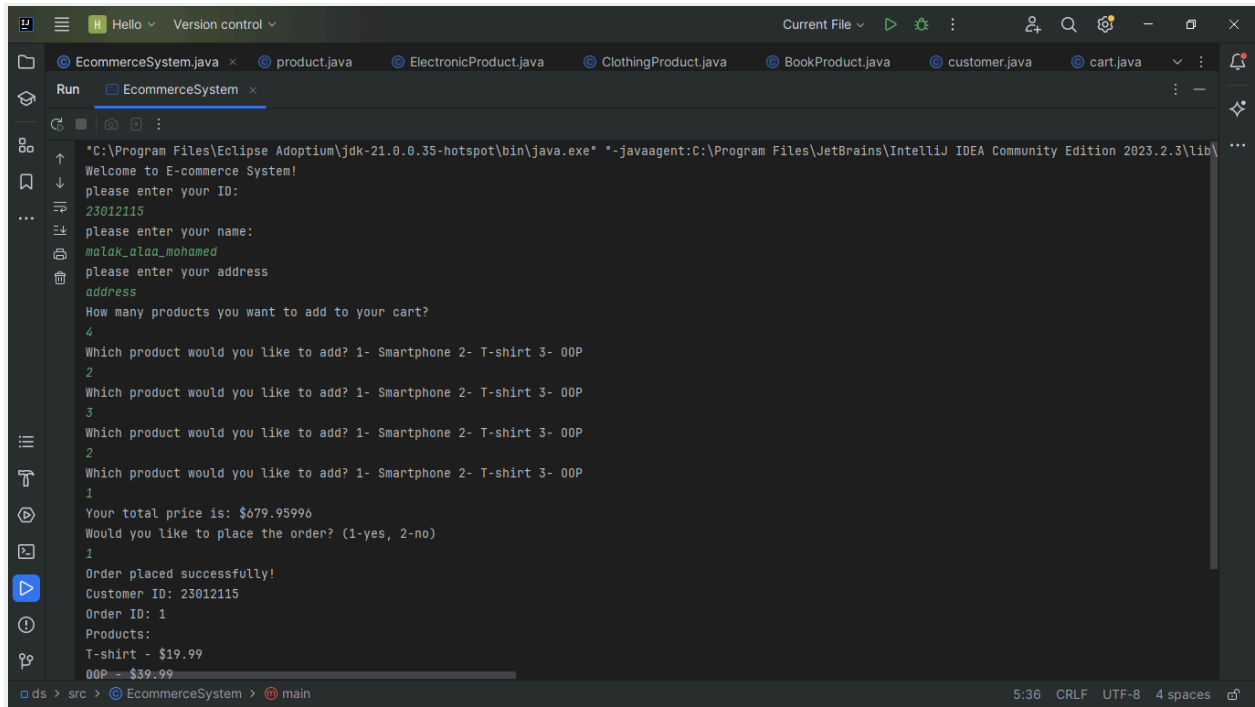
```
4 product[] products;
5 float totalPrice;
6 public order() {
7 }
8 public order(int cId, int oId) {
9     customerId = cId;
10    orderId = oId;
11 }
12 public void orderInfo(product[] products) {
13     System.out.println("Order details:");
14     System.out.println("Customer ID: " + customerId);
15     System.out.println("Products Ordered:");
16     for (int i = 0; i < products.length; i++) {
17         product product = products[i];
18         if (product != null) {
19             if (product != null) {
20                 System.out.println(product.name + " - $" + product.price);
21                 totalPrice += product.getPrice();
22             }
23         }
24     }
25     System.out.println("Total Price: $" + totalPrice);
26 }
27 }
28 }
29 }
30 }
```


EcommerceSystem class:

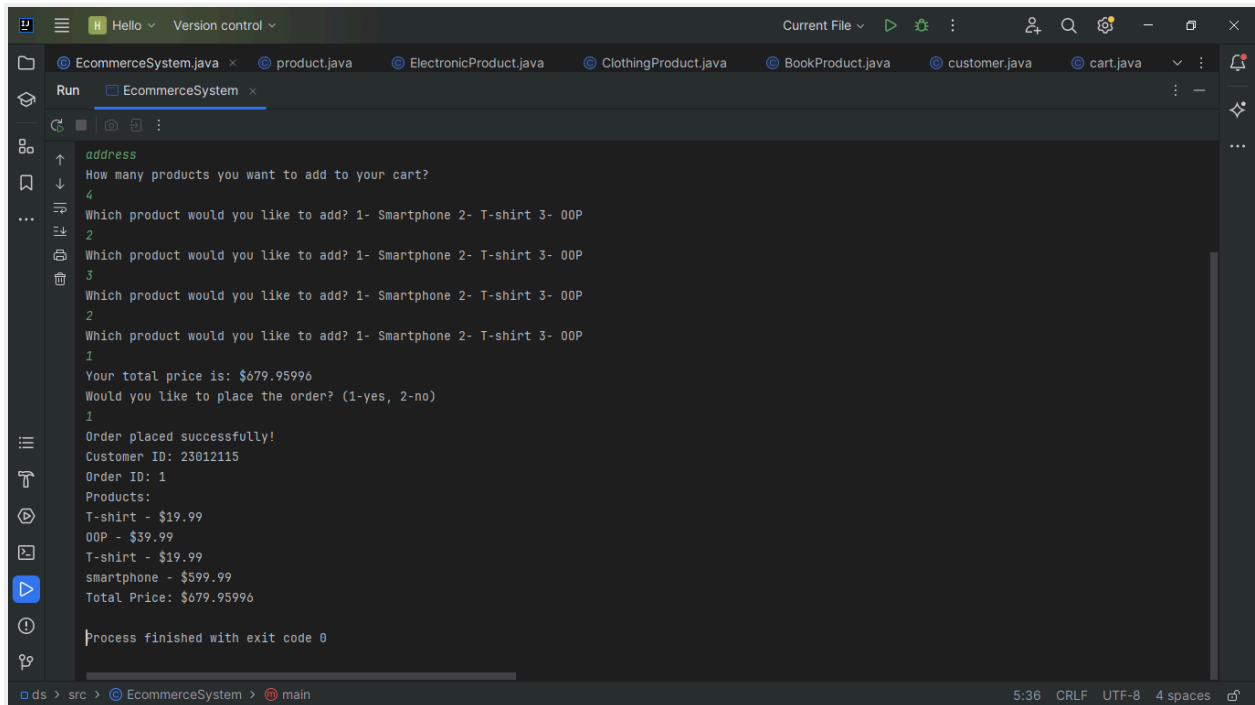
```
1  import java.util.Scanner;
2  public class EcommerceSystem {
3      public static void main(String[] args) {
4          Scanner input = new Scanner(System.in);
5          ElectronicProduct e1 = new ElectronicProduct(1, 599.99f, "smartphone", "samsung", 1);
6          ClothingProduct cp1 = new ClothingProduct(2, 19.99f, "T-shirt", "Medium", "Cotton");
7          BookProduct b1 = new BookProduct(3, 39.99f, "00P", "O'Reilly", "X Publication");
8          System.out.println("Welcome to E-commerce System!");
9          System.out.println("please enter your ID: ");
10         int customerId = input.nextInt();
11         System.out.println("please enter your name: ");
12         String customerName = input.next();
13         System.out.println("please enter your address ");
14         String customerAddress = input.next();
15         customer customer = new customer(customerId, customerName, customerAddress);
16         System.out.println("How many products you want to add to your cart?");
17         int nProduct = input.nextInt();
18         cart cart = new cart(customer.getCustomerId(), nProduct);
19         for (int i = 0; i < nProduct; i++) {
20             System.out.println("Which product would you like to add? 1- Smartphone 2- T-shirt 3- 00P ");
21             int type = input.nextInt();
22             switch (type) {
23                 case 1:
24                     cart.addProduct(e1);
25                     break;
26                 case 2:
27                     cart.addProduct(cp1);
28                     break;
29                 case 3:
30                     cart.addProduct(b1);
```

```
19         for (int i = 0; i < nProduct; i++) {
20             System.out.println("Which product would you like to add? 1- Smartphone 2- T-shirt 3- 00P ");
21             int type = input.nextInt();
22             switch (type) {
23                 case 1:
24                     cart.addProduct(e1);
25                     break;
26                 case 2:
27                     cart.addProduct(cp1);
28                     break;
29                 case 3:
30                     cart.addProduct(b1);
31                     break;
32                 default:
33                     System.out.println("invalid product type");
34                     break;
35             }
36         }
37
38         cart.placeOrder();
39     }
40 }
41 }
42
43
44
45
```

Output :



```
"C:\Program Files\Eclipse Adoptium\jdk-21.0.0-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.3\lib\idea_rt.jar=6025:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.3\bin" -Dfile.encoding=UTF-8
Welcome to E-commerce System!
please enter your ID:
23012115
please enter your name:
malak_alaa_mohamed
please enter your address
address
How many products you want to add to your cart?
4
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
2
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
3
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
2
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
1
Your total price is: $679.95996
Would you like to place the order? (1=yes, 2=no)
1
Order placed successfully!
Customer ID: 23012115
Order ID: 1
Products:
T-shirt - $19.99
OOP - $39.99
```



```
address
How many products you want to add to your cart?
4
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
2
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
3
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
2
Which product would you like to add? 1- Smartphone 2- T-shirt 3- OOP
1
Your total price is: $679.95996
Would you like to place the order? (1=yes, 2=no)
1
Order placed successfully!
Customer ID: 23012115
Order ID: 1
Products:
T-shirt - $19.99
OOP - $39.99
T-shirt - $19.99
smartphone - $599.99
Total Price: $679.95996

Process finished with exit code 0
```

