

# Assignment 1: TechShop, an electronic gadgets shop

Implement OOPs Task 1:

Classes and Their Attributes:

You are working as a software developer for TechShop, a company that sells electronic gadgets. Your task is to design and implement an application using Object-Oriented Programming (OOP) principles to manage customer information, product details, and orders. Below are the classes you need to create:

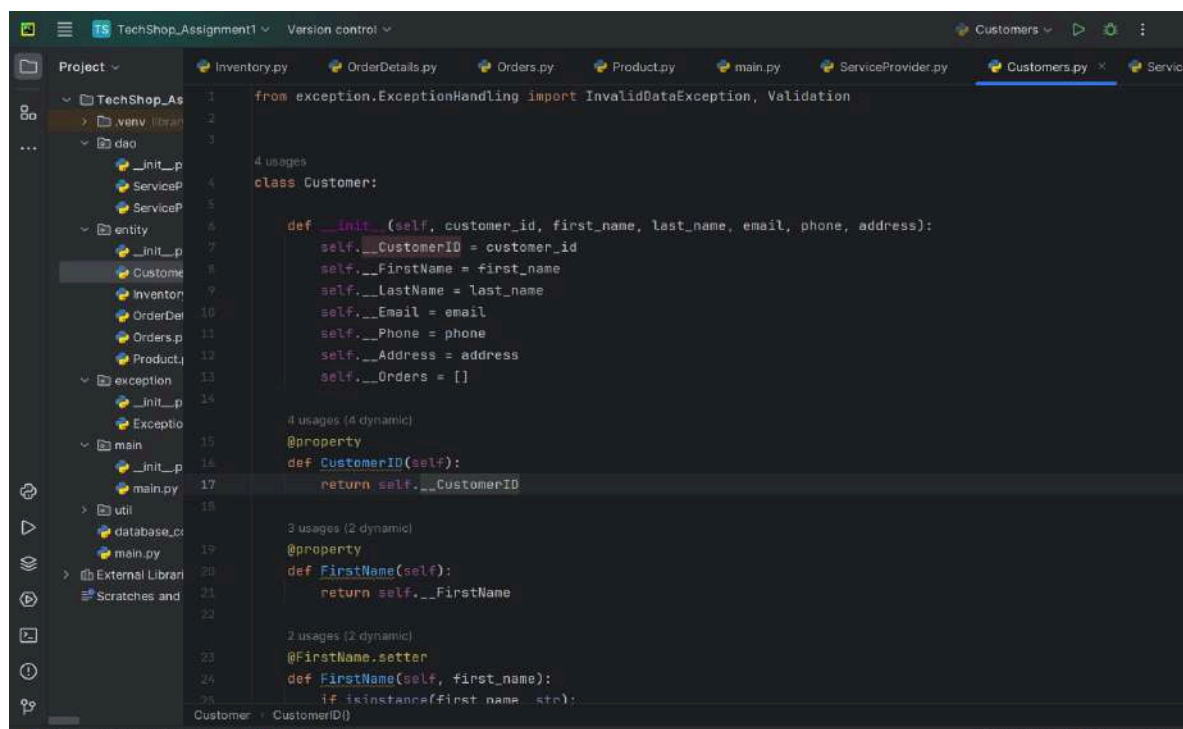
Customers Class:

Attributes:

- CustomerID (int) • FirstName (string) • LastName (string) • Email (string) • Phone (string) • Address (string)

Methods:

- CalculateTotalOrders(): Calculates the total number of orders placed by this customer.
- GetCustomerDetails(): Retrieves and displays detailed information about the customer.
- UpdateCustomerInfo(): Allows the customer to update their information (e.g., email, phone, or address).



```
from exception.ExceptionHandling import InvalidDataException, Validation

class Customer:

    def __init__(self, customer_id, first_name, last_name, email, phone, address):
        self.__CustomerID = customer_id
        self.__FirstName = first_name
        self.__LastName = last_name
        self.__Email = email
        self.__Phone = phone
        self.__Address = address
        self.__Orders = []

    @property
    def CustomerID(self):
        return self.__CustomerID

    @property
    def FirstName(self):
        return self.__FirstName

    @FirstName.setter
    def FirstName(self, first_name):
        if isinstance(first_name, str):
```

```

def __init__(self, first_name):
    if isinstance(first_name, str):
        self.__first_name = first_name
    else:
        raise Exception("First name must be a string.")

    3 usages (2 dynamic)
@property
def last_name(self):
    return self.__last_name

    2 usages (2 dynamic)
@last_name.setter
def last_name(self, last_name):
    if isinstance(last_name, str):
        self.__last_name = last_name
    else:
        raise Exception("Last name must be a string.")

    3 usages (2 dynamic)
@property
def email(self):
    return self.__email

    2 usages (2 dynamic)
@email.setter
def email(self, email):
    try:
        Validation.validate_email(email)
    except:
        raise Exception("Email must be a valid email address.")

    1 usage (1 dynamic)
def calculate_total_orders(self, customer_id):
    pass

```

```

return self.__phone

    2 usages (2 dynamic)
@phone.setter
def phone(self, phone):
    if isinstance(phone, str):
        self.__phone = phone
    else:
        raise Exception("Phone must be a string.")

@property
def orders(self):
    return self.__orders

    3 usages (2 dynamic)
@property
def address(self):
    return self.__address

    2 usages (2 dynamic)
@address.setter
def address(self, address):
    if isinstance(address, str):
        self.__address = address
    else:
        raise ValueError("Address must be a string.")

def calculate_total_orders(self, customer_id):
    pass

```

## Products Class: Attributes:

- ProductID (int) • ProductName (string) • Description (string) • Price (decimal)

## Methods:

- GetProductDetails(): Retrieves and displays detailed information about the product.
- UpdateProductInfo(): Allows updates to product details (e.g., price, description).
- IsProductInStock(): Checks if the product is currently in stock.

```

class Product():
    def __init__(self, product_id, name, description, price):
        self.__ProductID = product_id
        self.__ProductName = name
        self.__Description = description
        self.__Price = price

    4 usages (1 dynamic)
    @property
    def ProductID(self):
        return self.__ProductID

    3 usages (2 dynamic)
    @property
    def ProductName(self):
        return self.__ProductName

    2 usages (2 dynamic)
    @ProductName.setter
    def ProductName(self, product_name):
        if isinstance(product_name, str):
            self.__ProductName = product_name
        else:
            raise Exception("Product name should be string only")

    3 usages (2 dynamic)
    @property
    def Description(self):

```

```

    @Description.setter
    def Description(self, product_description):
        if isinstance(product_description, str):
            self.__Description = product_description
        else:
            raise Exception("Description should be String")

    4 usages (3 dynamic)
    @property
    def Price(self):
        return self.__Price

    3 usages (3 dynamic)
    @Price.setter
    def Price(self, product_price):
        if isinstance(product_price, int) and product_price > 0:
            self.__Price = product_price
        else:
            raise Exception("Price must be numeric and non negative")

    def get_product_by_id(self, product_id):
        pass

    def update_product_info(self, product_id):

```

Orders Class:

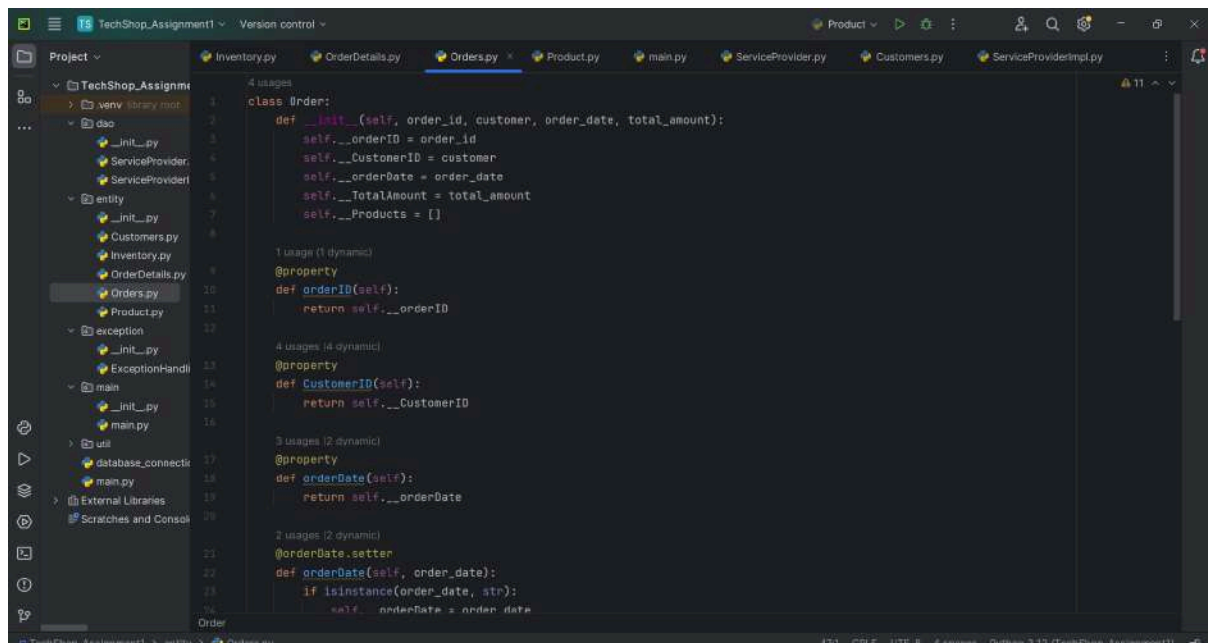
Attributes:

- OrderID (int) • Customer (Customer) - Use composition to reference the Customer who placed the order. • OrderDate (DateTime) • TotalAmount (decimal)

Methods:

- CalculateTotalAmount() - Calculate the total amount of the order. • GetOrderDetails(): Retrieves and displays the details of the order (e.g., product list and quantities). • UpdateOrderStatus(): Allows updating the

status of the order (e.g., processing, shipped). • **CancelOrder():** Cancels the order and adjusts stock levels for products.



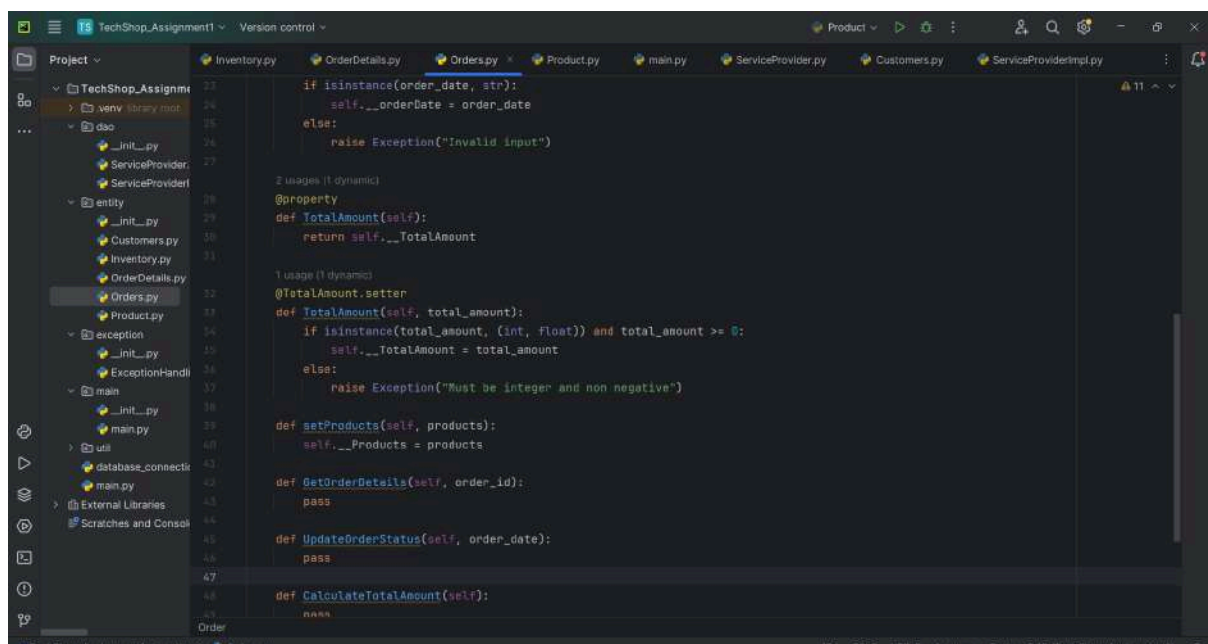
```
class Order:
    def __init__(self, order_id, customer, order_date, total_amount):
        self.__orderID = order_id
        self.__CustomerID = customer
        self.__orderDate = order_date
        self.__TotalAmount = total_amount
        self.__Products = []

    @property
    def orderID(self):
        return self.__orderID

    @property
    def CustomerID(self):
        return self.__CustomerID

    @property
    def orderDate(self):
        return self.__orderDate

    @orderDate.setter
    def orderDate(self, order_date):
        if isinstance(order_date, str):
            self.__orderDate = order_date
```



```
    @property
    def TotalAmount(self):
        return self.__TotalAmount

    @TotalAmount.setter
    def TotalAmount(self, total_amount):
        if isinstance(total_amount, (int, float)) and total_amount >= 0:
            self.__TotalAmount = total_amount
        else:
            raise Exception("Must be integer and non negative")

    def setProducts(self, products):
        self.__Products = products

    def getOrderDetails(self, order_id):
        pass

    def updateOrderStatus(self, order_date):
        pass

    def calculateTotalAmount(self):
        pass
```

## OrderDetails Class:

### Attributes:

- **OrderDetailID (int)**
- **Order (Order)** - Use composition to reference the Order to which this detail belongs.
- **Product (Product)** - Use composition to reference the Product included in the order detail.
- **Quantity (int)**

### Methods:

- **CalculateSubtotal()** - Calculate the subtotal for this order detail.
- **GetOrderDetailInfo()** - Retrieves and displays information about this order

detail. • UpdateQuantity(): Allows updating the quantity of the product in this order detail. • AddDiscount(): Applies a discount to this order detail.

```

1 class OrderDetail:
2     def __init__(self, order_details_id, order_id, product_id, quantity):
3         self.__OrderDetailID = order_details_id
4         self.__Order = order_id
5         self.__ProductID = product_id
6         self.__Quantity = quantity
7
8     @property
9     def OrderDetailID(self):
10         return self.__OrderDetailID
11
12     @property
13     def Order(self):
14         return self.__Order
15
16     @property
17     def ProductID(self):
18         return self.__ProductID
19
20     @property
21     def Quantity(self):
22         return self.__Quantity
23
24     @Quantity.setter
25     def Quantity(self, quantity):
26         pass
27
28     def GetOrderDetailInfo(self):
29         pass

```

```

18         return self.__ProductID
19
20     @property
21     def Quantity(self):
22         return self.__Quantity
23
24     @Quantity.setter
25     def Quantity(self, quantity):
26         if quantity > 0:
27             self.__Quantity = quantity
28         else:
29             raise Exception("Quantity must be 0 or greater than 0")
30
31     def CalculateSubtotal(self):
32         pass
33
34     def GetOrderDetailInfo(self):
35         pass
36
37     def UpdateQuantity(self):
38         pass
39
40     def AddDiscount(self):
41         pass
42

```

Inventory class:

Attributes:

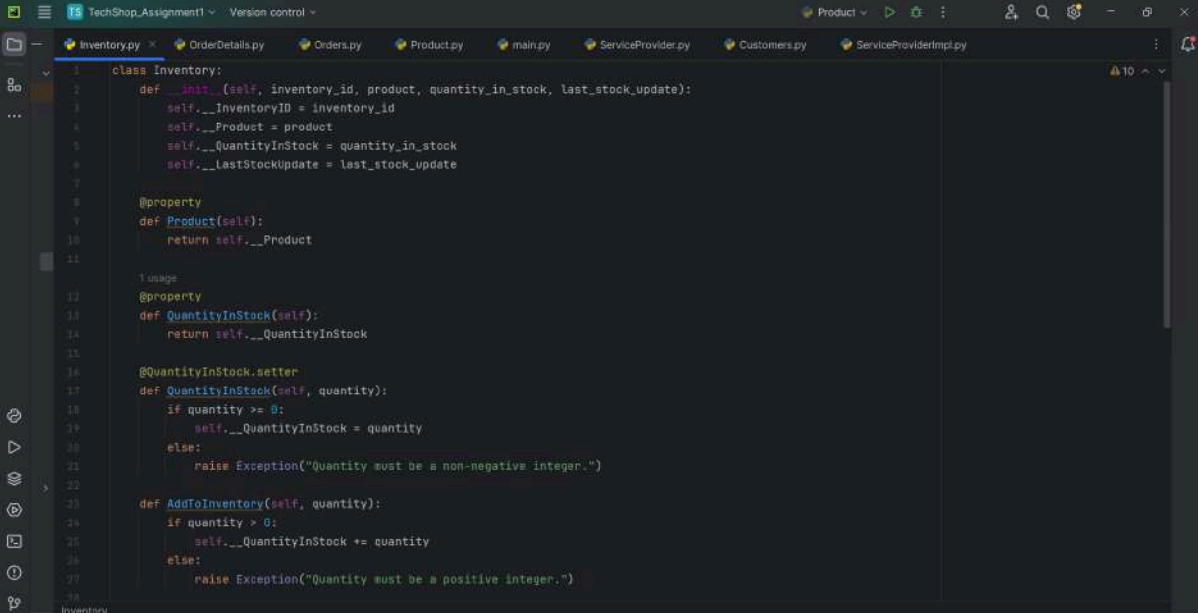
- InventoryID(int) • Product (Composition): The product associated with the inventory item.
- QuantityInStock: The quantity of the product currently in stock.
- LastStockUpdate

Methods:

- GetProduct(): A method to retrieve the product associated with this inventory item.
- GetQuantityInStock(): A method to get the current



quantity of the product in stock. • **AddToInventory(int quantity):** A method to add a specified quantity of the product to the inventory. • **RemoveFromInventory(int quantity):** A method to remove a specified quantity of the product from the inventory. • **UpdateStockQuantity(int newQuantity):** A method to update the stock quantity to a new value. • **IsProductAvailable(int quantityToCheck):** A method to check if a specified quantity of the product is available in the inventory. • **GetInventoryValue():** A method to calculate the total value of the products in the inventory based on their prices and quantities. • **ListLowStockProducts(int threshold):** A method to list products with quantities below a specified threshold, indicating low stock. • **ListOutOfStockProducts():** A method to list products that are out of stock.



```
1 class Inventory:
2     def __init__(self, inventory_id, product, quantity_in_stock, last_stock_update):
3         self.__inventory_id = inventory_id
4         self.__product = product
5         self.__quantity_in_stock = quantity_in_stock
6         self.__last_stock_update = last_stock_update
7
8     @property
9     def Product(self):
10         return self.__product
11
12     @property
13     def QuantityInStock(self):
14         return self.__quantity_in_stock
15
16     @QuantityInStock.setter
17     def QuantityInStock(self, quantity):
18         if quantity >= 0:
19             self.__quantity_in_stock = quantity
20         else:
21             raise Exception("Quantity must be a non-negative integer.")
22
23     def AddToInventory(self, quantity):
24         if quantity > 0:
25             self.__quantity_in_stock += quantity
26         else:
27             raise Exception("Quantity must be a positive integer.")
28
```

```

77         raise Exception("Quantity must be a positive integer.")
78
79     def RemoveFromInventory(self, quantity):
80         if 0 < quantity <= self.__QuantityInStock:
81             self.__QuantityInStock -= quantity
82         else:
83             raise Exception("Invalid quantity to remove from inventory.")
84
85     def UpdateStockQuantity(self, new_quantity):
86         if new_quantity >= 0:
87             self.__QuantityInStock = new_quantity
88         else:
89             raise Exception("New quantity must be a non-negative integer.")
90
91     def IsProductAvailable(self, quantity_to_check):
92         return quantity_to_check <= self.__QuantityInStock
93
94     def GetInventoryValue(self):
95         return self.__QuantityInStock * self.__Product.Price
96
97     def ListLowStockProducts(self, threshold):
98         if self.__QuantityInStock < threshold:
99             return self.__Product
100
101     def ListOutOfStockProducts(self):
102         if self.__QuantityInStock == 0:
103             return self.__Product
104
105

```

## Task 2: Class Creation:

- Create the classes (Customers, Products, Orders, OrderDetails and Inventory) with the specified attributes.
- Implement the constructor for each class to initialize its attributes.
- Implement methods as specified.

```

1  from exception.ExceptionHandling import InvalidDataException, Validation
2
3
4  4 usages
5  class Customer:
6
7      def __init__(self, customer_id, first_name, last_name, email, phone, address):
8          self.__CustomerID = customer_id
9          self.__FirstName = first_name
10         self.__LastName = last_name
11         self.__Email = email
12         self.__Phone = phone
13         self.__Address = address
14         self.__Orders = []
15
16  4 usages (4 dynamic)
17  @property
18  def CustomerID(self):
19      return self.__CustomerID
20
21  3 usages (2 dynamic)
22  @property
23  def FirstName(self):
24      return self.__FirstName
25
26  2 usages (2 dynamic)
27  @FirstName.setter
28  def FirstName(self, first_name):
29      if isinstance(first_name, str):
30
31  Customer > CustomerID()

```

TechShop\_Assignment1 - Version control

Product

Project

- TechShop\_Assignment1
  - venv
  - src
    - \_\_init\_\_.py
    - ServiceP
    - ServiceP
    - entity
      - \_\_init\_\_.py
      - Customer
      - Inventory
      - OrderDetails
      - Orders
      - Product
    - exception
      - \_\_init\_\_.py
      - Exception
    - main
      - \_\_init\_\_.py
      - main.py
    - util
      - database\_connect
      - main.py
    - External Libraries
    - Scratches and Console

Inventory.py

```
class Product():
    def __init__(self, product_id, name, description, price):
        self.__ProductID = product_id
        self.__ProductName = name
        self.__Description = description
        self.__Price = price

    4 usages (4 dynamic)
    @property
    def ProductID(self):
        return self.__ProductID

    3 usages (2 dynamic)
    @property
    def ProductName(self):
        return self.__ProductName

    2 usages (2 dynamic)
    @ProductName.setter
    def ProductName(self, product_name):
        if isinstance(product_name, str):
            self.__ProductName = product_name
        else:
            raise Exception("Product name should be string only")

    3 usages (2 dynamic)
    @property
    def Description(self):
```

Product

TechShop\_Assignment1 - Version control

Product

Project

- TechShop\_Assignment1
  - venv
  - src
    - \_\_init\_\_.py
    - ServiceP
    - ServiceP
    - entity
      - \_\_init\_\_.py
      - Customer
      - Inventory
      - OrderDetails
      - Orders
      - Product
    - exception
      - \_\_init\_\_.py
      - Exception
    - main
      - \_\_init\_\_.py
      - main.py
    - util
      - database\_connect
      - main.py
    - External Libraries
    - Scratches and Console

Inventory.py

```
class Order:
    def __init__(self, order_id, customer, order_date, total_amount):
        self.__orderID = order_id
        self.__CustomerID = customer
        self.__orderDate = order_date
        self.__TotalAmount = total_amount
        self.__Products = []

    1 usage (1 dynamic)
    @property
    def orderID(self):
        return self.__orderID

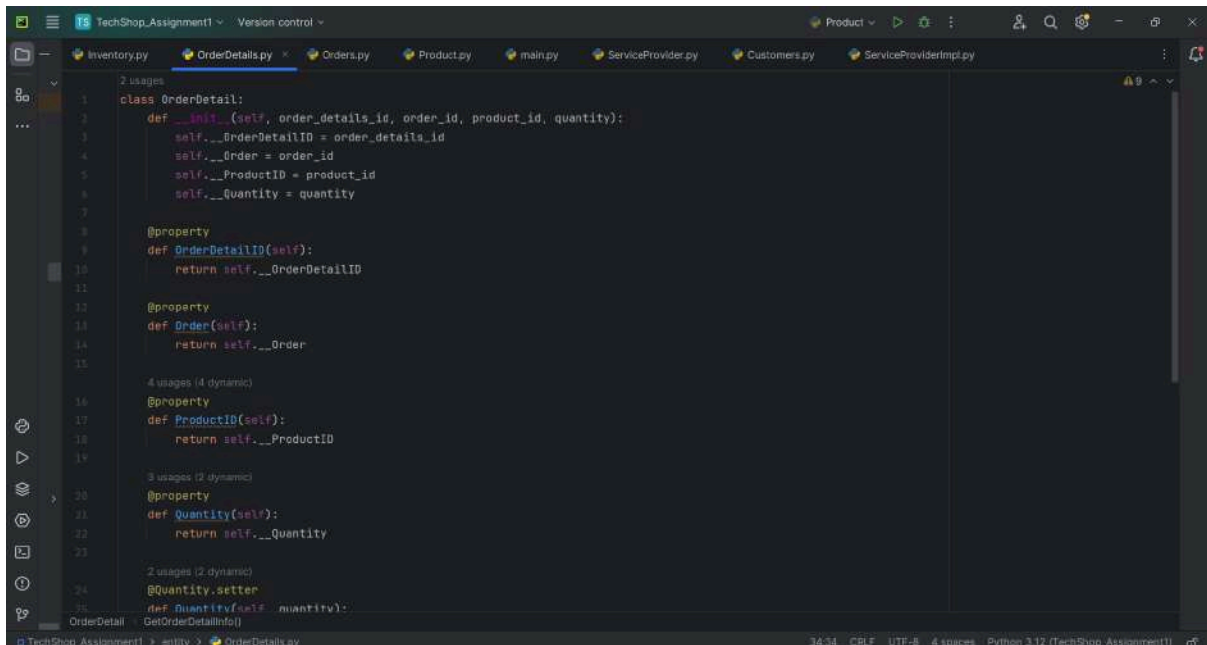
    4 usages (4 dynamic)
    @property
    def CustomerID(self):
        return self.__CustomerID

    3 usages (2 dynamic)
    @property
    def orderDate(self):
        return self.__orderDate

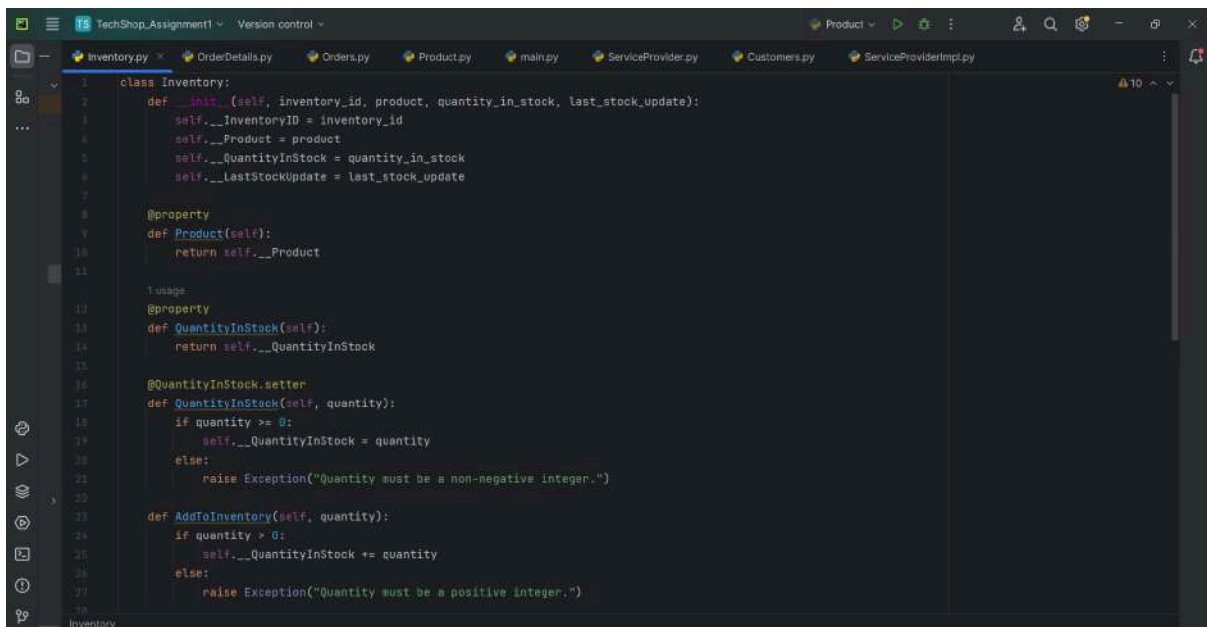
    2 usages (2 dynamic)
    @orderDate.setter
    def orderDate(self, order_date):
        if isinstance(order_date, str):
            self.__orderDate = order_date
```

Order





```
1 class OrderDetail:
2     def __init__(self, order_details_id, order_id, product_id, quantity):
3         self.__OrderDetailID = order_details_id
4         self.__Order = order_id
5         self.__ProductID = product_id
6         self.__Quantity = quantity
7
8     @property
9     def OrderDetailID(self):
10         return self.__OrderDetailID
11
12     @property
13     def Order(self):
14         return self.__Order
15
16     4 usages (4 dynamic)
17     @property
18     def ProductID(self):
19         return self.__ProductID
20
21     3 usages (2 dynamic)
22     @property
23     def Quantity(self):
24         return self.__Quantity
25
26     2 usages (2 dynamic)
27     @Quantity.setter
28     def QuantitySetter(self, quantity):
29         self.__Quantity = quantity
30
31 OrderDetail
32 GetOrderDetailInfo()
```



```
1 class Inventory:
2     def __init__(self, inventory_id, product, quantity_in_stock, last_stock_update):
3         self.__InventoryID = inventory_id
4         self.__Product = product
5         self.__QuantityInStock = quantity_in_stock
6         self.__LastStockUpdate = last_stock_update
7
8     @property
9     def Product(self):
10         return self.__Product
11
12     1 usage
13     @property
14     def QuantityInStock(self):
15         return self.__QuantityInStock
16
17     @QuantityInStock.setter
18     def QuantityInStockSetter(self, quantity):
19         if quantity >= 0:
20             self.__QuantityInStock = quantity
21         else:
22             raise Exception("Quantity must be a non-negative integer.")
23
24     def AddToInventory(self, quantity):
25         if quantity > 0:
26             self.__QuantityInStock += quantity
27         else:
28             raise Exception("Quantity must be a positive integer.")
29
30 Inventory
```

### Task 3: Encapsulation:

- Implement encapsulation by making the attributes private and providing public properties (getters and setters) for each attribute.
- Add data validation logic to setter methods (e.g., ensure that prices are non-negative, quantities are positive integers).

### Task 4:

#### Composition:

Ensure that the Order and OrderDetail classes correctly use composition to reference Customer and Product objects.

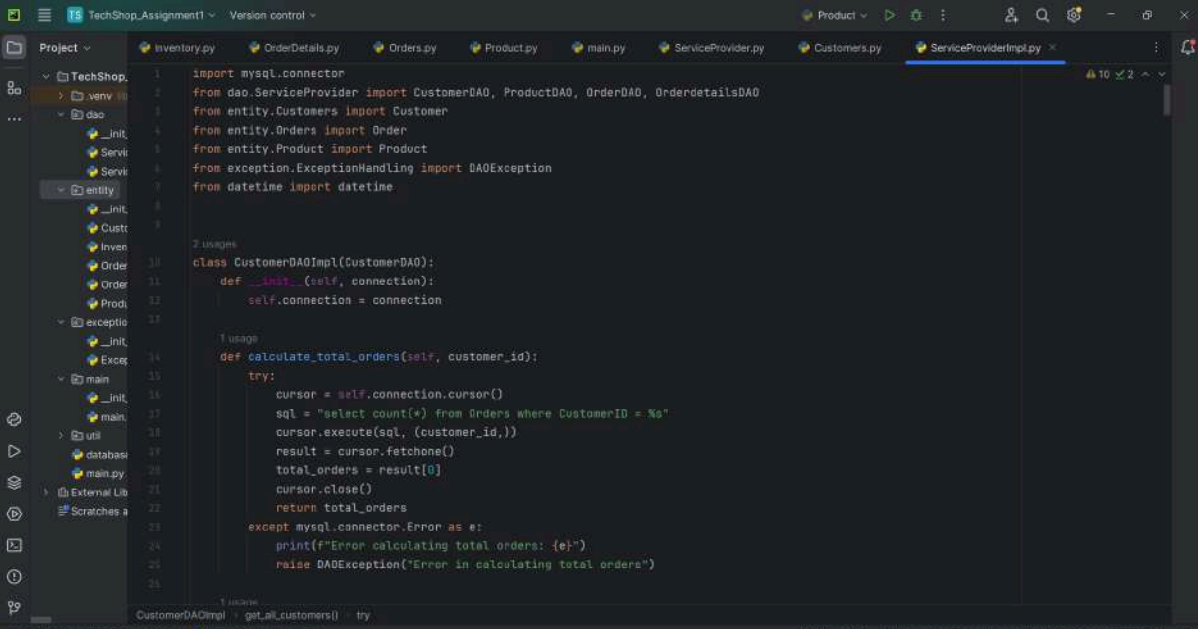
- Orders Class with Composition: o In the Orders class, we want to establish a composition relationship with the Customers class, indicating

that each order is associated with a specific customer. o In the Orders class, we've added a private attribute customer of type Customers, establishing a composition relationship. The Customer property provides access to the Customers object associated with the order.

- OrderDetails Class with Composition: o Similarly, in the OrderDetails class, we want to establish composition relationships with both the Orders and Products classes to represent the details of each order, including the product being ordered. o In the OrderDetails class, we've added two private attributes, order and product, of types Orders and Products, respectively, establishing composition relationships. The Order property provides access to the Orders object associated with the order detail, and the Product property provides access to the Products object representing the product in the order detail.

- Customers and Products Classes: o The Customers and Products classes themselves may not have direct composition relationships with other classes in this scenario. However, they serve as the basis for composition relationships in the Orders and OrderDetails classes, respectively.

- Inventory Class: o The Inventory class represents the inventory of products available for sale. It can have composition relationships with the Products class to indicate which products are in the inventory.



```
1 import mysql.connector
2 from dao.ServiceProvider import CustomerDAO, ProductDAO, OrderDAO, OrderdetailsDAO
3 from entity.Customers import Customer
4 from entity.Orders import Order
5 from entity.Product import Product
6 from exception.ExceptionHandling import DAOException
7 from datetime import datetime
8
9
10 class CustomerDAOImpl(CustomerDAO):
11     def __init__(self, connection):
12         self.connection = connection
13
14     1 usage
15     def calculate_total_orders(self, customer_id):
16         try:
17             cursor = self.connection.cursor()
18             sql = "select count(*) from Orders where CustomerID = %s"
19             cursor.execute(sql, (customer_id,))
20             result = cursor.fetchone()
21             total_orders = result[0]
22             cursor.close()
23             return total_orders
24         except mysql.connector.Error as e:
25             print(f"Error calculating total orders: {e}")
26             raise DAOException("Error in calculating total orders")
27
28
29 1 usage
30 CustomerDAOImpl().get_all_customers() try
```

This screenshot shows the `add_customer` method in `CustomerDAOImpl.py`. The method takes a `customer` object and attempts to insert it into a `Customers` table. It uses a cursor to execute an SQL `INSERT` statement with values from the customer object. If successful, it returns `True`; otherwise, it catches a `mysql.connector.Error` and raises a `DAOException`.

```
1 usage
def add_customer(self, customer):
    try:
        cursor = self.connection.cursor()
        sql = ("Insert into Customers ( CustomerID, FirstName, LastName, Email, Phone, Address) "
              "VALUES (%s, %s, %s, %s, %s, %s)")
        val = (customer.CustomerID, customer.FirstName, customer.LastName, customer.Email, customer.Phone,
              customer.Address)
        cursor.execute(sql, val)
        self.connection.commit()
        return True
    except mysql.connector.Error as e:
        print(f"Error in creating customer: {e}")
        raise DAOException("Error creating customer")

4 usages
def get_customer_by_id(self, customer_id):
    try:
        cursor = self.connection.cursor()
        sql = "select * from Customers where CustomerID = %s"
        cursor.execute(sql, (customer_id,))
        result = cursor.fetchone()
        cursor.close()
        if result:
            customer1 = Customer(result[0], result[1], result[2], result[3], result[4], result[5])
            return customer1
        else:
            return None
    except mysql.connector.Error as e:
        print(f"Error in fetching customer: {e}")
        raise DAOException("Error fetching customer")

1 usage
def update_customer_info(self, customer_id, email=None, phone=None, address=None):
    try:
        cursor = self.connection.cursor()
        sql = "UPDATE Customers SET"
        val = []

        if email is not None:
            sql += " Email=%s,"
            val.append(email)

        if phone is not None:
            sql += " Phone=%s,"
            val.append(phone)

        if address is not None:
            sql += " Address=%s,"
            val.append(address)

        # Remove the trailing comma from the SQL query
        if len(val) > 0:
            sql = sql.rstrip(',')
            sql += " WHERE CustomerID=%s"
            val.append(customer_id)

        cursor.execute(sql, val)
        self.connection.commit()
        return True
    except mysql.connector.Error as e:
        print(f"Error in updating customer info: {e}")
        raise DAOException("Error updating customer info")

1 usage
def delete_customer(self, customer_id):
    try:
        cursor = self.connection.cursor()
        sql = "DELETE FROM Customers WHERE CustomerID = %s"
        cursor.execute(sql, (customer_id,))
        self.connection.commit()
        return True
    except mysql.connector.Error as e:
        print(f"Error in deleting customer: {e}")
        raise DAOException("Error deleting customer")

1 usage
def get_all_customers(self):
    try:
        cursor = self.connection.cursor()
        sql = "select * from Customers"
        cursor.execute(sql)
        results = cursor.fetchall()
        cursor.close()
        customers = []
        for result in results:
            customer = Customer(result[0], result[1], result[2], result[3], result[4], result[5])
            customers.append(customer)
        return customers
    except mysql.connector.Error as e:
        print(f"Error in getting all customers: {e}")
        raise DAOException("Error getting all customers")

1 usage
def __init__(self, connection):
    self.connection = connection
```

This screenshot shows the `update_customer_info` method in `CustomerDAOImpl.py`. The method takes a `customer_id` and optional `email`, `phone`, and `address` parameters. It constructs an SQL `UPDATE` statement by appending the provided fields to a base `UPDATE Customers SET` query. It then executes the query and returns `True` if successful, or raises a `DAOException` if an error occurs.

```
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
25
```

This screenshot shows the implementation of the `get_all_customers` method in `ServiceProviderImpl.py`. The code is as follows:

```
sql = sql.rstrip(',')
sql += " WHERE CustomerID=%s"
val.append(customer_id)

cursor.execute(sql, val)
self.connection.commit()
cursor.close()
return True
else:
    print("No parameters provided for update")
    return False
except mysql.connector.Error as e:
    print(f"Error updating customer: {e}")
    raise DAOException("Error updating customer")

1 usage
def get_all_customers(self):
    try:
        cursor = self.connection.cursor()
        sql = "select * from Customers"
        cursor.execute(sql)
        results = cursor.fetchall()
        return results
    except mysql.connector.Error as e:
        print(f"Error in fetching all customers: {e}")
        raise DAOException("Error fetching all customers")

1 usage
def delete_customer(self, customer_id):
    CustomerDAOImpl().get_all_customers() : try
```

This screenshot shows the implementation of the `delete_customer` method in `ServiceProviderImpl.py`. The code is as follows:

```
print(f"Error in fetching all customers: {e}")
raise DAOException("Error fetching all customers")

1 usage
def delete_customer(self, customer_id):
    try:
        cursor = self.connection.cursor()
        sql = "DELETE FROM Customers WHERE CustomerID=%s"
        cursor.execute(sql, (customer_id,))
        self.connection.commit()
        cursor.close()
        return True
    except mysql.connector.Error as e:
        print(f"Error deleting customer: {e}")
        raise DAOException("Error deleting customer")

2 usages
class ProductDAOImpl(ProductDAO):
    def __init__(self, connection):
        self.connection = connection

1 usage
def add_products(self, product):
    try:
        cursor = self.connection.cursor()
        sql = "INSERT INTO Products (ProductID, ProductName, Description, Price) VALUES (%s, %s, %s, %s)"
        val = (product.ProductID, product.ProductName, product.Description, product.Price)
        cursor.execute(sql, val)
        CustomerDAOImpl().get_all_customers() : try
```

```
Project -> TechShop_Assignment1 -> Version control -> Product -> ServiceProviderImpl.py x
TechShop_Assignment1 -> Version control -> Product -> ServiceProviderImpl.py x
124 val = (product.ProductID, product.ProductName, product.Description, product.Price)
125 cursor.execute(sql, val)
126 self.connection.commit()
127 return True
128 except mysql.connector.Error as e:
129     print(f"Error creating customer: {e}")
130     raise DAOException("Error creating customer")
131
4 usages
132 def get_product_by_id(self, product_id):
133     try:
134         cursor = self.connection.cursor()
135         sql = "SELECT * FROM Products WHERE ProductID = %s"
136         cursor.execute(sql, (product_id,))
137         result = cursor.fetchone()
138         cursor.close()
139         if result:
140             product1 = Product(result[0], result[1], result[2], result[3])
141             return product1
142         else:
143             return None
144     except mysql.connector.Error as e:
145         print(f"Error fetching customer: {e}")
146         raise DAOException("Error fetching customer")
147
1 usage
148 def update_product_info(self, product_id, new_price=None):
149     try:
150         cursor = self.connection.cursor()
151         CustomerDAOImpl.get_all_customers() -> try
```

```
Project -> TechShop_Assignment1 -> Version control -> Product -> ServiceProviderImpl.py x
TechShop_Assignment1 -> Version control -> Product -> ServiceProviderImpl.py x
152 cursor = self.connection.cursor()
153 sql = "UPDATE Products SET"
154 val = []
155
156 if new_price is not None:
157     sql += " Price=%s,"
158     val.append(new_price)
159
160 if len(val) > 0:
161     sql = sql.rstrip(',')
162     sql += " WHERE ProductID=%s"
163     val.append(product_id)
164
165 cursor.execute(sql, val)
166 self.connection.commit()
167 cursor.close()
168 return True
169 else:
170     print("No parameters provided for update")
171     return False
172 except mysql.connector.Error as e:
173     print(f"Error updating customer: {e}")
174     raise DAOException("Error updating customer")
175
1 usage
176 def is_product_in_stock(self, product_id):
177     try:
178         cursor = self.connection.cursor()
179         CustomerDAOImpl.get_all_customers() -> try
```



The screenshot shows an IDE with a project named 'TechShop\_Assignment1'. The file explorer on the left shows a directory structure with files like 'inventory.py', 'OrderDetails.py', 'Orders.py', 'Product.py', 'main.py', 'ServiceProvider.py', 'Customers.py', and 'ServiceProviderImpl.py'. The main editor displays the code for 'ServiceProviderImpl.py'. The method 'is\_product\_in\_stock' is implemented as follows:

```
1 usage:
173 def is_product_in_stock(self, product_id):
174     try:
175         cursor = self.connection.cursor()
176         sql = "SELECT QuantityInStock FROM Inventory WHERE ProductID = %s"
177         cursor.execute(sql, (product_id,))
178         total_orders = cursor.fetchone()
179         cursor.close()
180         if total_orders is not None:
181             return True
182         else:
183             return False
184     except mysql.connector.Error as e:
185         print(f"Error calculating total orders: {e}")
186         raise DAOException("Error calculating total orders")
187
188 1 usage:
189 def get_all_products(self):
190     try:
191         cursor = self.connection.cursor()
192         sql = "SELECT * FROM Products"
193         cursor.execute(sql)
194         results = cursor.fetchall()
195         return results
196     except mysql.connector.Error as e:
197         print(f"Error fetching all customers: {e}")
198         raise DAOException("Error fetching all customers")
199
```

The status bar at the bottom indicates 'CustomerDAOImpl | get\_all\_customers() | try'.

The screenshot shows the same IDE with the file 'OrderDAOImpl.py' open. The method 'delete\_products' is implemented as follows:

```
199 raise DAOException("Error fetching all customers")
200
201 1 usage:
202 def delete_products(self, product_id):
203     try:
204         cursor = self.connection.cursor()
205         sql = "DELETE FROM Products WHERE ProductID = %s"
206         cursor.execute(sql, (product_id,))
207         self.connection.commit()
208         cursor.close()
209         return True
210     except mysql.connector.Error as e:
211         print(f"Error deleting product: {e}")
212         raise DAOException("Error deleting product")
213
214 2 usages:
215 class OrderDAOImpl(OrderDAO):
216     def __init__(self, connection):
217         self.connection = connection
218
219 1 usage:
220 def create_orders(self, order, order_details):
221     try:
222         cursor = self.connection.cursor()
223         total_amount = 0
224         for order_detail in order_details:
225             sql_get_price = "SELECT Price FROM Products WHERE ProductID = %s"
226             cursor.execute(sql_get_price, (order_detail.ProductID,))
227
```

The status bar at the bottom indicates 'CustomerDAOImpl | get\_all\_customers() | try'.



This screenshot shows the Visual Studio Code editor with the file `ServiceProviderImpl.py` open. The editor displays the `get_order_details` method, which is used to retrieve order information from a database. The method includes a try-except block to handle database errors and a loop to process the results. The left sidebar shows the project structure, and the bottom status bar indicates the current file and line numbers.

```
2 usages
def get_order_details(self, order_id):
    try:
        cursor = self.connection.cursor()
        sql = "SELECT * FROM Orders WHERE OrderID = %s"
        cursor.execute(sql, (order_id,))
        results = cursor.fetchall()
        cursor.close()
        orders = []
        if results:
            for result in results:
                order = Order(result[0], result[1], result[2], result[3])
                orders.append(order)
            return orders
        else:
            return None
    except mysql.connector.Error as e:
        print(f"Error fetching customer: {e}")
        raise DAOException("Error fetching customer")

1 usage
def calculate_total_amount(self):
    try:
        cursor = self.connection.cursor()
        sql = "SELECT SUM(TotalAmount) from Orders"
        cursor.execute(sql)
        total_price_info = cursor.fetchone()
        cursor.close()
        if total_price_info:
            return total_price_info
        else:
            return 0
    except mysql.connector.Error as e:
        print(f"Error calculating total amount for order: {e}")
        raise DAOException("Error calculating total amount for order")

1 usage
def update_order_status(self, order_id):
    try:
        cursor = self.connection.cursor()
        sql_get_order_date = "SELECT OrderDate FROM Orders WHERE OrderID = %s"
        cursor.execute(sql_get_order_date, (order_id,))
        result = cursor.fetchone()
        if result:
            order_date = result[0]
            current_date = datetime.now()
            order_date = datetime.combine(order_date, datetime.min.time())
            difference = current_date - order_date
            if difference.days > 3:
                return "shipped"
            else:
                return "processing"
        else:
            return "processing"
    except mysql.connector.Error as e:
        print(f"Error calculating total amount for order: {e}")
        raise DAOException("Error calculating total amount for order")
```

This screenshot shows the Visual Studio Code editor with the file `ServiceProviderImpl.py` open. The editor displays the `calculate_total_amount` and `update_order_status` methods. The `calculate_total_amount` method uses a SQL query to sum the total amount from the Orders table. The `update_order_status` method uses a SQL query to get the order date and compares it with the current date to determine the status. The left sidebar shows the project structure, and the bottom status bar indicates the current file and line numbers.

```
sql = "SELECT SUM(TotalAmount) from Orders"
cursor.execute(sql)
total_price_info = cursor.fetchone()
cursor.close()
if total_price_info:
    return total_price_info
else:
    return 0
except mysql.connector.Error as e:
    print(f"Error calculating total amount for order: {e}")
    raise DAOException("Error calculating total amount for order")

1 usage
def update_order_status(self, order_id):
    try:
        cursor = self.connection.cursor()
        sql_get_order_date = "SELECT OrderDate FROM Orders WHERE OrderID = %s"
        cursor.execute(sql_get_order_date, (order_id,))
        result = cursor.fetchone()
        if result:
            order_date = result[0]
            current_date = datetime.now()
            order_date = datetime.combine(order_date, datetime.min.time())
            difference = current_date - order_date
            if difference.days > 3:
                return "shipped"
            else:
                return "processing"
        else:
            return "processing"
    except mysql.connector.Error as e:
        print(f"Error calculating total amount for order: {e}")
        raise DAOException("Error calculating total amount for order")
```

```

    return "Processing"
    else:
        return "Order not found"
    except mysql.connector.Error as e:
        print(f"Error updating order status: {e}")
        raise DAOException("Error updating order status")

2 usages
class OrderDetailsDAOImpl(OrderDetailsDAO):
    def __init__(self, connection):
        self.connection = connection

1 usage
    def GetAllOrderDetail(self):
        try:
            cursor = self.connection.cursor()
            sql = "SELECT * FROM orderdetails"
            cursor.execute(sql)
            results = cursor.fetchall()
            return results
        except mysql.connector.Error as e:
            print(f"Error fetching all customers: {e}")
            raise DAOException("Error fetching all customers")

2 usages
    def CalculateSubtotal(self, order_detail_id):
        try:
            cursor = self.connection.cursor()
            sql = "SELECT od.ProductID, od.Quantity, p.ProductName, od.Quantity FROM OrderDetails od INNER JOIN Products p ON od.ProductID = p.ProductID WHERE od.OrderDetailID = %s"
            cursor.execute(sql, (order_detail_id,))
            result = cursor.fetchone()
            if result:
                price, quantity = result
                subtotal = price * quantity
                return subtotal
            else:
                return None
        except mysql.connector.Error as e:
            print(f"Error calculating subtotal: {e}")
            raise DAOException("Error calculating subtotal")

1 usage
    def GetOrderDetailInfo(self, order_detail_id):
        try:
            cursor = self.connection.cursor()
            sql = ("SELECT od.OrderDetailID, s.OrderID, p.ProductName, od.Quantity FROM OrderDetails od INNER JOIN Orders o ON od.OrderID = s.OrderID INNER JOIN Products p ON od.ProductID = p.ProductID "
                  "WHERE od.OrderDetailID = %s")
            cursor.execute(sql, (order_detail_id,))
            result = cursor.fetchone()
            if result:
                order_detail_id, order_id, product_name, quantity = result
                print("Order Detail ID:", order_detail_id)
                print("Order ID:", order_id)
                print("Product Name:", product_name)
                print("Quantity:", quantity)
            else:
                print("Order Detail not found")
        except mysql.connector.Error as e:
            print(f"Error getting order detail info: {e}")
            raise DAOException("Error getting order detail info")

```

```

    return "Processing"
    else:
        return "Order not found"
    except mysql.connector.Error as e:
        print(f"Error updating order status: {e}")
        raise DAOException("Error updating order status")

2 usages
class OrderDetailsDAOImpl(OrderDetailsDAO):
    def __init__(self, connection):
        self.connection = connection

1 usage
    def GetAllOrderDetail(self):
        try:
            cursor = self.connection.cursor()
            sql = "SELECT * FROM orderdetails"
            cursor.execute(sql)
            results = cursor.fetchall()
            return results
        except mysql.connector.Error as e:
            print(f"Error fetching all customers: {e}")
            raise DAOException("Error fetching all customers")

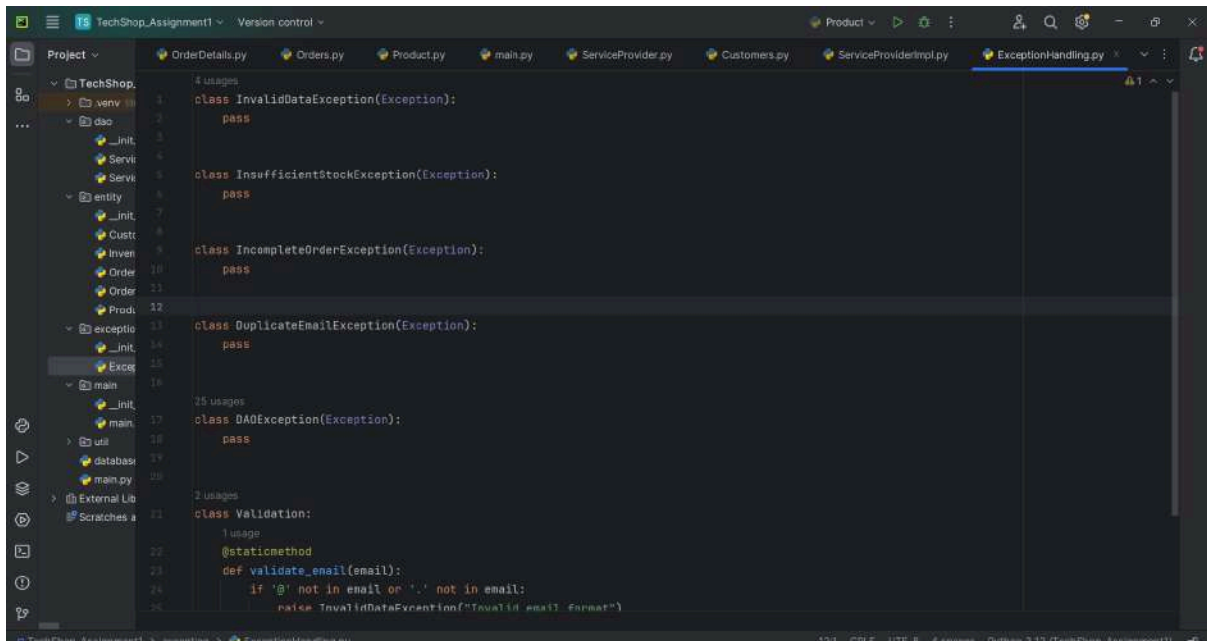
2 usages
    def CalculateSubtotal(self, order_detail_id):
        try:
            cursor = self.connection.cursor()
            sql = "SELECT od.ProductID, od.Quantity, p.ProductName, od.Quantity FROM OrderDetails od INNER JOIN Products p ON od.ProductID = p.ProductID WHERE od.OrderDetailID = %s"
            cursor.execute(sql, (order_detail_id,))
            result = cursor.fetchone()
            if result:
                price, quantity = result
                subtotal = price * quantity
                return subtotal
            else:
                return None
        except mysql.connector.Error as e:
            print(f"Error calculating subtotal: {e}")
            raise DAOException("Error calculating subtotal")

1 usage
    def GetOrderDetailInfo(self, order_detail_id):
        try:
            cursor = self.connection.cursor()
            sql = ("SELECT od.OrderDetailID, s.OrderID, p.ProductName, od.Quantity FROM OrderDetails od INNER JOIN Orders o ON od.OrderID = s.OrderID INNER JOIN Products p ON od.ProductID = p.ProductID "
                  "WHERE od.OrderDetailID = %s")
            cursor.execute(sql, (order_detail_id,))
            result = cursor.fetchone()
            if result:
                order_detail_id, order_id, product_name, quantity = result
                print("Order Detail ID:", order_detail_id)
                print("Order ID:", order_id)
                print("Product Name:", product_name)
                print("Quantity:", quantity)
            else:
                print("Order Detail not found")
        except mysql.connector.Error as e:
            print(f"Error getting order detail info: {e}")
            raise DAOException("Error getting order detail info")

```

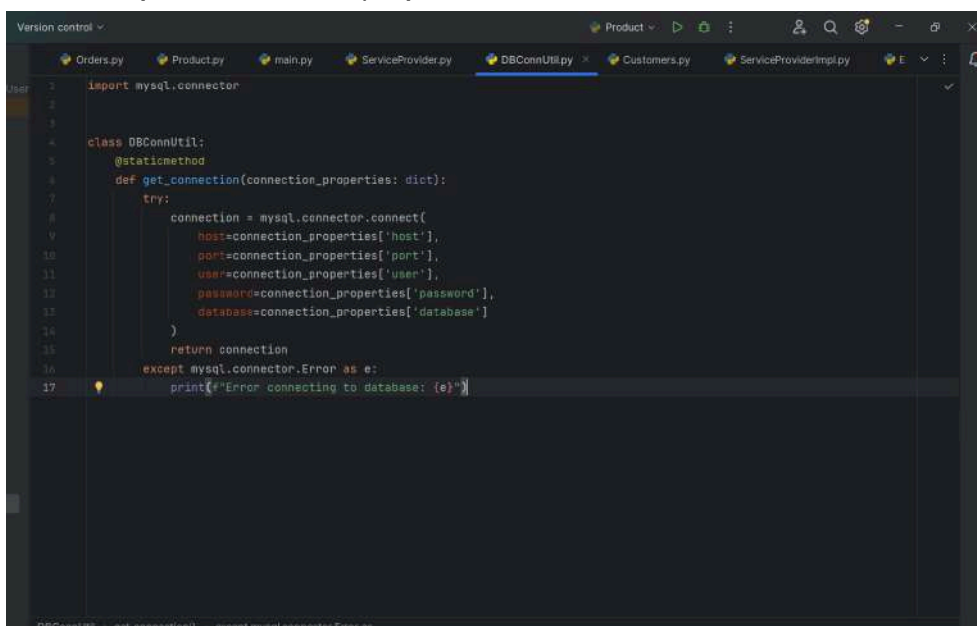
## Task 5: Exceptions handling

- Data Validation:
- Inventory Management:
- Order Processing:
- Payment Processing:
- Database Access:
- Concurrency Control:
- Security and Authentication:



## Task 7: Database Connectivity

- Implement a DatabaseConnector class responsible for establishing a connection to the "TechShopDB" database. This class should include methods for opening, closing, and managing database connections.
- Implement classes for Customers, Products, Orders, OrderDetails, Inventory with properties, constructors, and methods for CRUD (Create, Read, Update, Delete) operations.



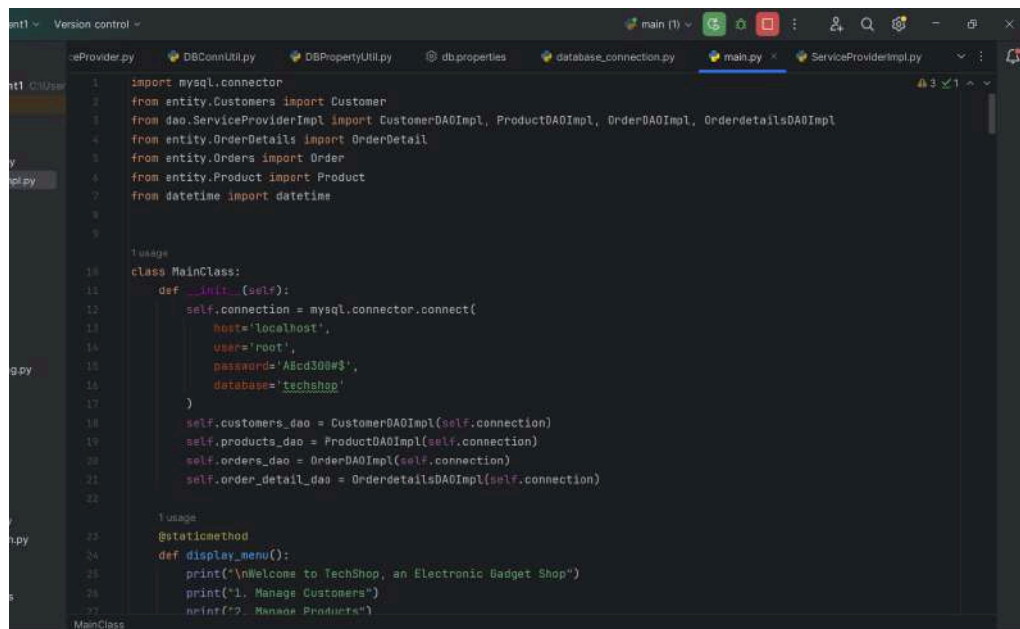


```
version control
Product
Orders.py Product.py ServiceProvider.py DBConnUtil.py DBPropertyUtil.py Customers.py ServiceProviderImpl.py
class DBPropertyUtil:
    @staticmethod
    def get_connection_properties(file_name: str) -> dict:
        connection_properties = {}
        try:
            with open(file_name, 'r') as file:
                for line in file:
                    key, value = line.strip().split('=')
                    connection_properties[key.strip()] = value.strip()
            return connection_properties
        except FileNotFoundError:
            print(f"Error: File {file_name} not found.")
        return {}
```

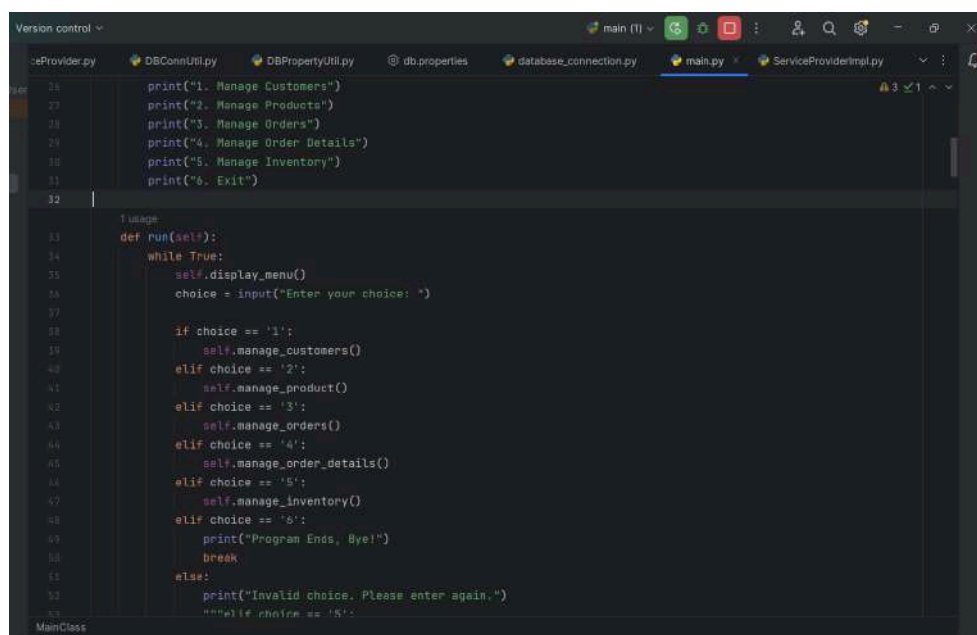
```
Assignment1 Version control
Product
Orders.py Product.py ServiceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties ServiceProviderImpl.py
1 host=localhost
2 user=root
3 password=ABcd300#$
4 database=techshop
5 port=3306
```

```
Assignment1 Version control
Product
Orders.py ServiceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py ServiceProvider
1 import mysql.connector
2
3
4 class DatabaseConnection:
5     def __init__(self, host, user, password, database):
6         self.connection = mysql.connector.connect(
7             host="localhost",
8             user="root",
9             password="ABcd300#$",
10            database="techshop"
11        )
12        self.cursor = self.connection.cursor()
13
14    def execute_query(self, query, params=None):
15        if params:
16            self.cursor.execute(query, params)
17        else:
18            self.cursor.execute(query)
19        return self.cursor
20
21    9 usages (9 dynamic)
22    def commit(self):
23        self.connection.commit()
24
25    def close_connection(self):
26        self.connection.close()
```

- 1: Customer Registration
- 2: Product Catalog Management'
- 3: Placing Customer Orders
- 4: Tracking Order Status
- 6: Sales Reporting
- 7: Customer Account Updates
- 8: Payment Processing
- 9: Product Search and Recommendations



```
1 import mysql.connector
2 from entity.Customers import Customer
3 from dao.ServiceProviderImpl import CustomerDAOImpl, ProductDAOImpl, OrderDAOImpl, OrderDetailsDAOImpl
4 from entity.OrderDetails import OrderDetail
5 from entity.Orders import Order
6 from entity.Product import Product
7 from datetime import datetime
8
9
10
11 class MainClass:
12     def __init__(self):
13         self.connection = mysql.connector.connect(
14             host='localhost',
15             user='root',
16             password='Abcd30@W$',
17             database='techshop'
18         )
19         self.customers_dao = CustomerDAOImpl(self.connection)
20         self.products_dao = ProductDAOImpl(self.connection)
21         self.orders_dao = OrderDAOImpl(self.connection)
22         self.order_detail_dao = OrderDetailsDAOImpl(self.connection)
23
24     @staticmethod
25     def display_menu():
26         print("\nWelcome to TechShop, an Electronic Gadget Shop")
27         print("1. Manage Customers")
28         print("2. Manage Products")
```



```
24 print("1. Manage Customers")
25 print("2. Manage Products")
26 print("3. Manage Orders")
27 print("4. Manage Order Details")
28 print("5. Manage Inventory")
29 print("6. Exit")
30
31
32
33
34 def run(self):
35     while True:
36         self.display_menu()
37         choice = input("Enter your choice: ")
38
39         if choice == '1':
40             self.manage_customers()
41         elif choice == '2':
42             self.manage_product()
43         elif choice == '3':
44             self.manage_orders()
45         elif choice == '4':
46             self.manage_order_details()
47         elif choice == '5':
48             self.manage_inventory()
49         elif choice == '6':
50             print("Program Ends, Bye!")
51             break
52         else:
53             print("Invalid choice. Please enter again.")
54             """elif choice == '5':
```

```
Version control
Provider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
1 usage
36 def manage_customers(self):
37     print("\n*****CUSTOMER MENU*****")
38     while True:
39         print("\nMenu:")
40         print("1. Add Customer(C)")
41         print("2. Get all the Customer Details(R)")
42         print("3. Update the Customer Information(U)")
43         print("4. Delete the Customer(D)")
44         print("5. View Specific Customer Details")
45         print("6. Calculate Total Orders")
46         print("7. Exit")
47
48         choice = input("Enter your choice (1-7): ")
49
50         if choice == "1":
51             customer_id = input("Enter ID: ")
52             first_name = input("Enter First Name: ")
53             last_name = input("Enter Last Name: ")
54             email = input("Enter Email: ")
55             phone = input("Enter Phone: ")
56             address = input("Enter Address: ")
57             customer = Customer(customer_id, first_name, last_name, email, phone, address)
58             self.customers_dao.add_customer(customer)
59             print("Customer added successfully.")
60
61         elif choice == "2":
```

```
Version control
Provider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
62         elif choice == "2":
63             customers = self.customers_dao.get_all_customers()
64             if customers:
65                 for customer in customers:
66                     print(customer)
67             else:
68                 print("Sorry! No customer found")
69
70         elif choice == "3":
71             customer_id = int(input("Enter Customer ID: "))
72             if self.customers_dao.get_customer_by_id(customer_id):
73                 new_email = input("Enter new Email (leave blank to keep unchanged): ").strip() or None
74                 new_phone = input("Enter new Phone (leave blank to keep unchanged): ").strip() or None
75                 new_address = input("Enter new Address (leave blank to keep unchanged): ").strip() or None
76                 self.customers_dao.update_customer_info(customer_id, new_email, new_phone, new_address)
77                 print("Customer information updated successfully.")
78             else:
79                 print("Not found.")
80
81         elif choice == "4":
82             customer_id = int(input("Enter Customer ID: "))
83             if self.customers_dao.get_customer_by_id(customer_id):
84                 self.customers_dao.delete_customer(customer_id)
85                 print("Customer deleted successfully.")
86             else:
87                 print("Id not found")
88
89         elif choice == "5":
```

```
Version control
ceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
110
111 elif choice == "5":
112     customer_id = int(input("Enter Customer ID: "))
113     customer = self.customers_dao.get_customer_by_id(customer_id)
114     if customer:
115         print(customer.CustomerID, customer.FirstName, customer.LastName, customer.Email, customer.Phone,
116               customer.Address)
117     else:
118         print("Customer not found.")
119
120 elif choice == "6":
121     customer_id = int(input("Enter Customer ID: "))
122     if self.customers_dao.get_customer_by_id(customer_id):
123         total_orders = self.customers_dao.calculate_total_orders(customer_id)
124         print(f"Total orders for customer {customer_id}: {total_orders}")
125     else:
126         print("Customer not found.")
127
128 elif choice == "7":
129     print("program Ends.")
130     break
131 else:
132     print("Invalid choice. Please enter a valid option.")
133
134
135 ! usage
136 def manage_product(self):
137     print("\n*****PRODUCT MENU*****")
138     while True:
139         print(f"\nMenu:")
140
MainClass
```

```
Version control
ceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
134 while True:
135     print("\nMenu:")
136     print("1. Add Product(C)")
137     print("2. Get all Product Details(R)")
138     print("3. Update Product Information(U)")
139     print("4. Delete Product(D)")
140     print("5. View Specific Product Details")
141     print("6. Is Product In Stock?")
142     print("7. Exit")
143
144     choice = input("Enter your choice (1-7): ")
145
146     if choice == "1":
147         product_id = input("Enter Product ID: ")
148         product_name = input("Enter Product Name: ")
149         description = input("Enter the Description: ")
150         price = float(input("Enter the Price: "))
151         product = Product(product_id, product_name, description, price)
152         self.products_dao.add_products(product)
153         print("Product added successfully.")
154
155     elif choice == "2":
156         products = self.products_dao.get_all_products()
157         if products:
158             for product in products:
159                 print(product)
160         else:
161             print("No Product found")
162
MainClass
```

```
version control
ServiceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
161 print("No Product found")
162
163 elif choice == "3":
164     product_id = int(input("Enter Product ID: "))
165     if self.products_dao.get_product_by_id(product_id):
166         new_price = input("Enter new Price (leave blank to keep unchanged): ").strip() or None
167         self.products_dao.update_product_info(product_id, new_price)
168         print("Product information updated successfully.")
169     else:
170         print("Not found.")
171
172 elif choice == "4":
173     product_id = int(input("Enter Product ID: "))
174     if self.products_dao.get_product_by_id(product_id):
175         self.products_dao.delete_products(product_id)
176         print("Product deleted successfully.")
177     else:
178         print("Product not found")
179
180 elif choice == "5":
181     product_id = int(input("Enter Product ID: "))
182     product = self.products_dao.get_product_by_id(product_id)
183     if product:
184         print(product.ProductID, product.ProductName, product.Description, product.Price)
185     else:
186         print("Product not found.")
187
188 elif choice == "6":
189     product_id = int(input("Enter Product ID: "))
190
MainClass
```

```
version control
ServiceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
188 elif choice == "6":
189     product_id = int(input("Enter Product ID: "))
190     if self.products_dao.get_product_by_id(product_id):
191         if self.products_dao.is_product_in_stock(product_id):
192             print("Currently in Stock")
193         else:
194             print("Out of Stock")
195     else:
196         print("Product not found.")
197
198 elif choice == "7":
199     print("Exiting program.")
200     break
201 else:
202     print("Invalid choice. Please enter a valid option.")
203
204 def manage_orders(self):
205     print("\n*****ORDER MENU*****")
206     while True:
207         print("\nMenu:")
208         print("1. Create the Order(C)")
209         print("2. Display the Orders(R)")
210         print("3. Cancel the Order(D)")
211         print("4. Get Order Details")
212         print("5. Calculate Total Amount")
213         print("6. UpdateOrderStatus (Processed/shipped)")
214         print("7. Exit")
215
MainClass
```



```
version control
ceProvider.py DBConnUtility.py DBPropertyUtility.py @ db.properties database_connection.py main.py ServiceProviderImpl.py
214 print("7. Exit")
215
216 choice = input("Enter your choice (1-8): ")
217
218 if choice == "1":
219     customer_id = input("Enter Customer id: ")
220     order_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
221     order_details = []
222     while True:
223         product_id = input("Enter Product ID: ")
224         quantity = int(input("Enter Quantity: "))
225         order_detail = OrderDetail(order_details_id=None, order_id=None, product_id=product_id, quantity=quantity)
226         order_details.append(order_detail)
227         add_more = input("Add more products? (yes/no): ").lower()
228         if add_more != 'yes':
229             break
230     order = Order(order_id=None, customer_id=customer_id, order_date=order_date, total_amount=None)
231     self.orders_dao.create_order(order, order_details)
232     print("Order added successfully.")
233
234 elif choice == "2":
235     orders = self.orders_dao.display_orders()
236     if orders:
237         for order in orders:
238             print(order)
239     else:
240         print("No Order Found")
241
242 elif choice == "3":
```

```
version control
ceProvider.py DBConnUtility.py DBPropertyUtility.py @ db.properties database_connection.py main.py ServiceProviderImpl.py
239 print(order)
240 else:
241     print("No Order found")
242
243 elif choice == "3":
244     order_id = int(input("Enter Order ID: "))
245     if self.orders_dao.getOrderDetails(order_id):
246         self.orders_dao.CancelOrder(order_id)
247         print("Order cancelled successfully.")
248     else:
249         print("Order not found")
250
251 elif choice == "4":
252     order_id = int(input("Enter Order ID: "))
253     orders = self.orders_dao.getOrderDetails(order_id)
254     if orders:
255         for order in orders:
256             print("Order ID:", order.orderID)
257             print("Customer ID:", order.CustomerID)
258             print("Order Date:", order.orderDate)
259             print("Total Amount:", order.TotalAmount)
260     else:
261         print("Order not found for this customer.")
262
263 elif choice == "5":
264     print(self.orders_dao.CalculateTotalAmount())
265 elif choice == "6":
266     order_id = int(input("Enter Order ID: "))
267     print(self.orders_dao.UpdateOrderStatus(order_id))
268
269 MainClass
```

```
Version control
ceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
265         order_id = int(input("Enter Order ID: "))
266         print(self.orders_dao.UpdateOrderStatus(order_id))
267
268
269     elif choice == "7":
270         print("Exiting program.")
271         break
272     else:
273         print("Invalid choice. Please enter a valid option.")
274
275
276     1 usage
277     def manage_order_details(self):
278         print("\n*****ORDER DETAILS MENU*****")
279         while True:
280             print("\nMenu:")
281             print("1. Calculate Subtotal")
282             print("2. Get Order Detail Info")
283             print("3. Update Quantity")
284             print("4. Add Discount")
285             print("5. Display All OrderDetails")
286             print("6. Exit")
287
288             choice = input("Enter your choice (1-6): ")
289
290             if choice == "1":
291                 order_detail_id = int(input("Enter Order Detail ID: "))
292                 subtotal = self.order_detail_dao.CalculateSubtotal(order_detail_id)
293                 if subtotal is not None:
294                     print("Subtotal:", subtotal)
295                 else:
296                     print("Order detail not found.")
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2
```

```
Project - TechShop_Assignment1 - Version control - main (1) x
Run - ExceptionHandling - main (1) x
C:\Users\astha\PycharmProjects\TechShop_Assignment1\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\TechShop_Assignment1\main\main.py

Welcome to TechShop, an Electronic Gadget Shop
1. Manage Customers
2. Manage Products
3. Manage Orders
4. Manage Order Details
5. Manage Inventory
6. Exit
Enter your choice: 1

*****CUSTOMER MENU*****

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 2
(1, 'Aman', 'Kumar', 'aman@gmail.com', '1234567897', 'Motihar', 1)
(2, 'Ansu', 'Singh', 'anshu@gmail.com', '1111111111', 'Bihar', 1)
(3, 'Amit', 'Sah', 'amit@gmail.com', '1111111112', 'Rechhi', 1)
(4, 'Asmita', 'Si', 'asmita@gmail.com', '1111111121', 'Burgapur', 1)
(5, 'Swati', 'Singh', 'swati@gmail.com', '1111111122', 'Asansol', 2)
```

```
Project - TechShop_Assignment1 - Version control - main (1) x
Run - ExceptionHandling - main (1) x
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 2
(1, 'Aman', 'Kumar', 'aman@gmail.com', '1234567897', 'Motihar', 1)
(2, 'Ansu', 'Singh', 'anshu@gmail.com', '1111111111', 'Bihar', 1)
(3, 'Amit', 'Sah', 'amit@gmail.com', '1111111112', 'Rechhi', 1)
(4, 'Asmita', 'Si', 'asmita@gmail.com', '1111111121', 'Burgapur', 1)
(5, 'Swati', 'Singh', 'swati@gmail.com', '1111111122', 'Asansol', 2)
(6, 'Prateeti', 'Maji', 'prateeti@gmail.com', '1111111133', 'Burgapur', 1)
(7, 'Anjali', 'Sneha', 'anjali@gmail.com', '1111111134', 'Raghunathpur', 1)
(8, 'Om', 'Kumar', 'om123@gmail.com', '1111111136', 'Bihar', 1)
(9, 'Roshan', 'Singh', 'roshan@gmail.com', '1111111137', 'Sandhinagar', 0)
(10, 'Arava', 'Krishnavenni', 'arava@gmail.com', '1111111139', 'Chennai', 0)
(11, 'Jasmine', 'Roy', 'jasmine123@gmail.com', '2258746984', 'Mumbai', 0)

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 1
Enter ID: |
```

```
TechShop_Assignment1 | Version control | main (1) | 315/44 CRLF UTF-8 4 spaces Python 3.12 (TechShop_Assignment1)

Project | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py

Run | ExceptionHandling | main (1) |

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 1
Enter ID: 12
Enter First Name: Ankush
Enter Last Name: Singh
Enter Email: ankush@gmail.com
Enter Phone: 4587123650
Enter Address: Delhi
Customer added successfully.

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7):
```

```
TechShop_Assignment1 | Version control | main (1) | 315/44 CRLF UTF-8 4 spaces Python 3.12 (TechShop_Assignment1)

Project | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py

Run | ExceptionHandling | main (1) |

4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 2
(1, 'Aman', 'Kumar', 'aman@gmail.com', '1234567897', 'Motihar', 1)
(2, 'Ansu', 'Singh', 'anshu@gmail.com', '1111111111', 'Bihar', 1)
(3, 'Amit', 'Sah', 'amit@gmail.com', '1111111112', 'Ranchi', 1)
(4, 'Asmita', 'Si', 'asmita@gmail.com', '1111111121', 'Durgapur', 1)
(5, 'Swati', 'Singh', 'swati@gmail.com', '1111111122', 'Asansol', 2)
(6, 'Prateeti', 'Maji', 'prateeti@gmail.com', '1111111133', 'Durgapur', 1)
(7, 'Anjali', 'Sneha', 'anjali@gmail.com', '1111111134', 'Raghunathpur', 1)
(8, 'Om', 'Kumar', 'om123@gmail.com', '1111111136', 'Bihar', 1)
(9, 'Roshan', 'Singh', 'roshan@gmail.com', '1111111137', 'Gandhinagar', 0)
(10, 'Arava', 'Krishnavenni', 'arava@gmail.com', '1111111139', 'Chennai', 0)
(11, 'Jasmine', 'Roy', 'jasmine123@gmail.com', '2258766986', 'Mumbai', 0)
(12, 'Ankush', 'Singh', 'ankush@gmail.com', '4587123650', 'Delhi', None)

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7):
```

```
TechShop_Assignment1 | Version control | main (1) |
Project | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py |
Run | ExceptionHandling | main (1) |
... |
(8, 'Om', 'Kumar', 'om123@gmail.com', '1111111136', 'Bihar', 1)
(9, 'Roshan', 'Singh', 'roshan@gmail.com', '1111111137', 'Bandhinagar', 9)
(10, 'Arava', 'Krishnavenni', 'arava@gmail.com', '1111111139', 'Chennai', 0)
(11, 'Jasmine', 'Roy', 'jasmine123@gmail.com', '2258746984', 'Mumbai', 0)
(12, 'Ankush', 'Singh', 'ankush@gmail.com', '4587123650', 'Delhi', None)

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 4
Enter Customer ID: 12
Customer deleted successfully.

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7):
```

```
TechShop_Assignment1 | Version control | main (1) |
Project | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py |
Run | ExceptionHandling | main (1) |
... |
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 4
Enter Customer ID: 12
Customer deleted successfully.

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7): 6
Enter Customer ID: 5
Total orders for customer 5: 2

Menu:
1. Add Customer(C)
2. Get all the Customer Details(R)
3. Update the Customer Information(U)
4. Delete the Customer(D)
5. View Specific Customer Details
6. Calculate Total Orders
7. Exit
Enter your choice (1-7):
```



```
TechShop_Assignment1 | Version control | main (1) | Python 3.12 (TechShop_Assignment1)
Project | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py
Run | ExceptionHandling | main (1) |
*****PRODUCT MENU*****
Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): 2
(5, 'abc', 'dede', Decimal('78.37'))
(11, 'Laptop', 'electronic gadget', Decimal('550000.15'))
(12, 'Mouse', 'electronic gadget', Decimal('550.26'))
(13, 'Keyboard', 'electronic gadget', Decimal('880.59'))
(14, 'Printer', 'electronic gadget', Decimal('8800.37'))
(15, 'Phone', 'electronic gadget', Decimal('19800.92'))
(16, 'WallClock', 'Home appliances', Decimal('5000.14'))
(17, 'Headphone', 'electronic gadget', Decimal('2200.15'))
(18, 'Camera', 'electronic gadget', Decimal('220001.09'))
(19, 'Smartwatch', 'electronic gadget', Decimal('2200.61'))
(20, 'Speaker', 'Home appliances', Decimal('20000.85'))
(21, 'Monitor', 'electronic gadget', Decimal('8000.00'))
Menu:
1. Add Product(C)
2. Get all Product Details(R)
```

```
TechShop_Assignment1 | Version control | main (1) | Python 3.12 (TechShop_Assignment1)
Project | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py
Run | ExceptionHandling | main (1) |
(20, 'Speaker', 'Home appliances', Decimal('20000.85'))
(21, 'Monitor', 'electronic gadget', Decimal('8000.00'))
Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): 1
Enter Product ID: 25
Enter Product Name: Iphone
Enter the Description: electronic gadget
Enter the Price: 25842.33
Product added successfully.
Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): |
```

```
TechShop_Assignment1 - Version control
main (1)
Project - ceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
Run ExceptionHandling main (1)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): 2
(5, 'abc', 'dede', Decimal('78.37'))
(11, 'Laptop', 'electronic gadget', Decimal('550000.15'))
(12, 'Mouse', 'electronic gadget', Decimal('550.26'))
(13, 'Keyboard', 'electronic gadget', Decimal('880.59'))
(14, 'Printer', 'electronic gadget', Decimal('8800.37'))
(15, 'Phone', 'electronic gadget', Decimal('19800.92'))
(16, 'WallClock', 'Home appliances', Decimal('5000.14'))
(17, 'Headphone', 'electronic gadget', Decimal('2200.15'))
(18, 'Camera', 'electronic gadget', Decimal('220001.09'))
(19, 'Smartwatch', 'electronic gadget', Decimal('2200.61'))
(20, 'Speaker', 'Home appliances', Decimal('20000.85'))
(21, 'Monitor', 'electronic gadget', Decimal('8000.00'))
(25, 'Iphone', 'electronic gadget', Decimal('25842.33'))

Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): |
```

```
TechShop_Assignment1 - Version control
main (1)
Project - ceProvider.py DBConnUtil.py DBPropertyUtil.py db.properties database_connection.py main.py ServiceProviderImpl.py
Run ExceptionHandling main (1)
(18, 'Camera', 'electronic gadget', Decimal('220001.09'))
(19, 'Smartwatch', 'electronic gadget', Decimal('2200.61'))
(20, 'Speaker', 'Home appliances', Decimal('20000.85'))
(21, 'Monitor', 'electronic gadget', Decimal('8000.00'))
(25, 'Iphone', 'electronic gadget', Decimal('25842.33'))

Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): 6
Enter Product ID: 14
Currently in Stock

Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7):
```

```
Project > TechShop_Assignment1 > Version control > main (1) >
Run > ExceptionHandling > main (1) >
...
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): 6
Enter Product ID: 14
Currently in Stock

Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7): 5
Enter Product ID: 25
25 Iphone electronic gadget 25842.33

Menu:
1. Add Product(C)
2. Get all Product Details(R)
3. Update Product Information(U)
4. Delete Product(D)
5. View Specific Product Details
6. Is Product In Stock?
7. Exit
Enter your choice (1-7):
TechShop_Assignment1 > main > main.py 315:44 CRLF UTF-8 4 spaces Python 3.12 (TechShop_Assignment1) 0813
```

```
Project > TechShop_Assignment1 > Version control > main (1) >
Run > ExceptionHandling > main (1) >
...
1. Create the Order(C)
2. Display the Orders(R)
3. Cancel the Order(D)
4. Get Order Details
5. Calculate Total Amount
6. UpdateOrderStatus (Processed/shipped)
7. Exit
Enter your choice (1-8): 2
(101, 1, datetime.date(2024, 1, 1), Decimal('550000.15'), 'Shipped')
(102, 2, datetime.date(2024, 1, 2), Decimal('1100.52'), 'Pending')
(103, 3, datetime.date(2024, 1, 3), Decimal('2641.77'), 'Pending')
(104, 4, datetime.date(2024, 1, 4), Decimal('35201.48'), 'Pending')
(105, 5, datetime.date(2024, 1, 5), Decimal('99004.60'), 'Pending')
(106, 6, datetime.date(2024, 1, 6), Decimal('30000.84'), 'Pending')
(107, 7, datetime.date(2024, 1, 7), Decimal('15401.05'), 'Pending')
(108, 8, datetime.date(2024, 1, 8), Decimal('1760008.72'), 'Pending')
(111, 5, datetime.date(2024, 1, 11), None, 'Pending')

Menu:
1. Create the Order(C)
2. Display the Orders(R)
3. Cancel the Order(D)
4. Get Order Details
5. Calculate Total Amount
6. UpdateOrderStatus (Processed/shipped)
7. Exit
Enter your choice (1-8):
```

```
Project - TechShop_Assignment1 - Version control
Run - ExceptionHandling - main (1)
(108, 8, datetime.date(2024, 1, 8), Decimal('1768988.72'), 'Pending')
(111, 5, datetime.date(2024, 1, 11), None, 'Pending')

Menu:
1. Create the Order(C)
2. Display the Orders(R)
3. Cancel the Order(D)
4. Get Order Details
5. Calculate Total Amount
6. UpdateOrderStatus (Processed/shipped)
7. Exit
Enter your choice (1-8): 4
Enter Order ID: 107
Order ID: 107
Customer ID: 7
Order Date: 2024-01-07
Total Amount: 15401.05

Menu:
1. Create the Order(C)
2. Display the Orders(R)
3. Cancel the Order(D)
4. Get Order Details
5. Calculate Total Amount
6. UpdateOrderStatus (Processed/shipped)
7. Exit
Enter your choice (1-8): |
```

```
Project - TechShop_Assignment1 - Version control
Run - ExceptionHandling - main (1)
Enter Order ID: 107
Order ID: 107
Customer ID: 7
Order Date: 2024-01-07
Total Amount: 15401.05

Menu:
1. Create the Order(C)
2. Display the Orders(R)
3. Cancel the Order(D)
4. Get Order Details
5. Calculate Total Amount
6. UpdateOrderStatus (Processed/shipped)
7. Exit
Enter your choice (1-8): 6
Enter Order ID: 107
shipped

Menu:
1. Create the Order(C)
2. Display the Orders(R)
3. Cancel the Order(D)
4. Get Order Details
5. Calculate Total Amount
6. UpdateOrderStatus (Processed/shipped)
7. Exit
Enter your choice (1-8): |
```

```
Project > TechShop_Assignment1 | Version control | main (1) | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py | Run | ExceptionHandling | main (1) | ... | 6. Exit | Enter your choice: 4 | *****ORDER DETAILS MENU***** | Menu: | 1. Calculate Subtotal | 2. Set Order Detail Info | 3. Update Quantity | 4. Add Discount | 5. Display All OderDetails | 6. Exit | Enter your choice (1-8): 2 | Enter Order Detail ID: 25 | Order Detail ID: 25 | Order ID: 105 | Product Name: Phone | Quantity: 5 | Menu: | 1. Calculate Subtotal | 2. Set Order Detail Info | 3. Update Quantity | 4. Add Discount | 5. Display All OderDetails | 6. Exit | Enter your choice (1-8): | TechShop_Assignment1 > main | main.py | 315:44 | CRLF | UTF-8 | 4 spaces | Python 3.12 (TechShop_Assignment1) | 08:18
```

```
Project > TechShop_Assignment1 | Version control | main (1) | ceProvider.py | DBConnUtil.py | DBPropertyUtil.py | db.properties | database_connection.py | main.py | ServiceProviderImpl.py | Run | ExceptionHandling | main (1) | ... | Quantity: 5 | Menu: | 1. Calculate Subtotal | 2. Set Order Detail Info | 3. Update Quantity | 4. Add Discount | 5. Display All OderDetails | 6. Exit | Enter your choice (1-8): 5 | (21, 101, 11, '1') | (22, 102, 12, '2') | (23, 103, 13, '3') | (24, 104, 14, '4') | (25, 105, 15, '5') | (26, 106, 16, '6') | (27, 107, 17, '7') | (28, 108, 18, '8') | Menu: | 1. Calculate Subtotal | 2. Set Order Detail Info | 3. Update Quantity | 4. Add Discount | 5. Display All OderDetails | 6. Exit | Enter your choice (1-8): | TechShop_Assignment1 > main | main.py | 315:44 | CRLF | UTF-8 | 4 spaces | Python 3.12 (TechShop_Assignment1) | 08:18
```



```
TechShop_Assignment1 | Version control | main (1) | Run | ExceptionHandling | main (1) | ...
Quantity: 5
Menu:
1. Calculate Subtotal
2. Set Order Detail Info
3. Update Quantity
4. Add Discount
5. Display All OderDetails
6. Exit
Enter your choice (1-8): 5
(21, 101, 11, '1')
(22, 102, 12, '2')
(23, 103, 13, '3')
(24, 104, 14, '4')
(25, 105, 15, '5')
(26, 106, 16, '6')
(27, 107, 17, '7')
(28, 108, 18, '8')

Menu:
1. Calculate Subtotal
2. Set Order Detail Info
3. Update Quantity
4. Add Discount
5. Display All OderDetails
6. Exit
Enter your choice (1-8): J

TechShop_Assignment1 | main | main.py | 315/44 | CRLF | UTF-8 | 4 spaces | Python 3.12 (TechShop_Assignment1) | 08:13
```