**Assignment-2: Create a Business System with Customers, Orders, and Products**

**Objective:**  
You will design a basic business system where customers can place orders for products. This exercise will help you practice creating multiple related classes, inheritance, encapsulation, and working with relationships between objects.

**Instructions:**

**1. Class Definition:**

* **Customer Class:**
  + Properties:
    - name (string): The name of the customer.
    - email (string): The customer's email address.
    - phone (string): The customer's phone number.
  + Methods:
    - \_\_init\_\_(self, name, email, phone): Constructor to initialize the customer details.
    - \_\_str\_\_(self): Return a string representing the customer in the format:

less

Copy

"Customer Name: [name], Email: [email], Phone: [phone]"

* + - place\_order(self, order): Method to place an order (create an Order object).
* **Product Class:**
  + Properties:
    - product\_name (string): The name of the product.
    - price (float): The price of the product.
    - stock\_quantity (integer): The quantity of this product in stock.
  + Methods:
    - \_\_init\_\_(self, product\_name, price, stock\_quantity): Constructor to initialize the product details.
    - \_\_str\_\_(self): Return a string representing the product in the format:

arduino

Copy

"Product: [product\_name], Price: $[price], Stock Quantity: [stock\_quantity]"

* + - update\_stock(self, quantity): Method to update the stock quantity by adding or subtracting from the current stock.
* **Order Class:**
  + Properties:
    - order\_id (string): A unique identifier for the order.
    - customer (Customer object): The customer who placed the order.
    - products (list of tuples): A list of products and their quantities in the order (e.g., [(product1, 2), (product2, 1)]).
    - total\_price (float): The total price of the order.
  + Methods:
    - \_\_init\_\_(self, order\_id, customer): Constructor to initialize the order details, including setting the customer.
    - add\_product(self, product, quantity): Method to add a product to the order (update the order's product list).
    - calculate\_total(self): Method to calculate the total price of the order based on the products and quantities.
    - \_\_str\_\_(self): Return a string representing the order in the format:

less

Copy

"Order ID: [order\_id], Customer: [customer.name], Total Price: $[total\_price]"

* **Transaction Class:**
  + Properties:
    - transaction\_id (string): A unique identifier for the transaction.
    - order (Order object): The order associated with the transaction.
    - payment\_method (string): The method of payment (e.g., "Credit Card", "PayPal").
    - payment\_status (string): The payment status ("Pending", "Completed").
  + Methods:
    - \_\_init\_\_(self, transaction\_id, order, payment\_method): Constructor to initialize the transaction.
    - process\_payment(self): Method to process the payment, updating the payment status to "Completed".
    - \_\_str\_\_(self): Return a string representing the transaction in the format:

less

Copy

"Transaction ID: [transaction\_id], Order ID: [order\_id], Payment Method: [payment\_method], Status: [payment\_status]"

**2. Business Logic:**

* The Customer can place multiple orders.
* The Order can have multiple products.
* When a Transaction is created for an order, the payment status starts as "Pending" and should be updated to "Completed" after processing the payment.
* The Product stock should be updated whenever a product is added to an order.

**Example Output:**

# Create products

product1 = Product("Laptop", 1200.00, 10)

product2 = Product("Mouse", 25.00, 50)

# Create a customer

customer1 = Customer("John Doe", "john.doe@email.com", "123-456-7890")

# Customer places an order

order1 = Order("ORD12345", customer1)

order1.add\_product(product1, 1) # 1 Laptop

order1.add\_product(product2, 2) # 2 Mouses

order1.calculate\_total()

# Print order details

print(order1)

# Create a transaction for the order

transaction1 = Transaction("TRANS001", order1, "Credit Card")

transaction1.process\_payment()

# Print transaction details

print(transaction1)

# Update product stock

product1.update\_stock(-1) # 1 Laptop sold

product2.update\_stock(-2) # 2 Mouses sold

# Print product details after stock update

print(product1)

print(product2)

# Print customer details

print(customer1)

Courtesy: ChatGpt