PIYUSH KUMAR MISHRA

230957212

ROLL N0:70

WEEK 7:

Exercise 1:

Inventory Management System using Python Inheritance

Scenario: Designing an Inventory Management System

You are tasked with designing an Inventory Management System that handles various types of products. Each product shares common attributes but has specific attributes based on the category of the product. The system must support operations such as adding products, calculating inventory value, applying discounts, and checking stock levels.

Problem Definition:

We need to manage three types of products:

1. Electronics (e.g., phones, laptops).

2. Clothing (e.g., shirts, pants).

3. Groceries (e.g., fruits, vegetables).

All product types share basic attributes like name, price, quantity, and SKU (Stock Keeping Unit). However, they also have specific attributes:

- **Electronics may have warranty period and brand.**
- **Clothing has attributes like size and material.**
- **Groceries include expiration date and organic status.**

Additionally, the system needs to:

1. Add new products.

2. Update stock.

3. Calculate the total value of inventory.

4. Apply discounts based on product type.

5. Check if products are low in stock.

```
class Product:
    def __init__(self, name, price, quantity, sku):
        self.name = name
        self.price = price
        self.quantity = quantity
        self.sku = sku

    def calculate_value(self):
        return self.price * self.quantity

    def is_low_stock(self, threshold=5):
        return self.quantity < threshold


class Electronics(Product):
    def __init__(self, name, price, quantity, sku, warranty_period, brand):
        super().__init__(name, price, quantity, sku)
        self.warranty_period = warranty_period
        self.brand = brand
```

```
class Clothing(Product):
    def __init__(self, name, price, quantity, sku, size, material):
        super().__init__(name, price, quantity, sku)
        self.size = size
        self.material = material


class Groceries(Product):
    def __init__(self, name, price, quantity, sku, expiration_date, organic_status):
        super().__init__(name, price, quantity, sku)
        self.expiration_date = expiration_date
        self.organic_status = organic_status


class Inventory:
    def __init__(self):
        self.products = {}

    def add_product(self, product):
        if product.sku in self.products:
            print("Product with this SKU already exists.")
        else:
            self.products[product.sku] = product
            print(f"Added {product.name} to inventory.")

    def update_stock(self, sku, quantity):
        if sku in self.products:
```

```
            self.products[sku].quantity += quantity
            print(f"Updated stock for {sku}. New quantity: {self.products[sku].quantity}.")
        else:
            print("Product not found.")

    def calculate_total_value(self):
        total_value = sum(product.calculate_value() for product in self.products.values())
        return total_value

    def apply_discount(self, sku, discount_percentage):
        if sku in self.products:
            product = self.products[sku]
            discount_amount = product.price * (discount_percentage / 100)
            product.price -= discount_amount
            print(f"Applied discount to {product.name}. New price: {product.price:.2f}.")
        else:
            print("Product not found.")

    def check_low_stock(self):
        low_stock_products = [product for product in self.products.values() if product.is_low_stock()]
        return low_stock_products


# Example Usage with user-defined input
if __name__ == "__main__":
    inventory = Inventory()
```

```
while True:
    print("\n1. Add Product")
    print("2. Update Stock")
    print("3. Calculate Total Inventory Value")
    print("4. Apply Discount")
    print("5. Check Low Stock Items")
    print("6. Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        print("\nSelect Product Type:")
        print("1. Electronics")
        print("2. Clothing")
        print("3. Groceries")
        product_type = input("Enter choice (1/2/3): ")

        name = input("Enter product name: ")
        price = float(input("Enter product price: "))
        quantity = int(input("Enter product quantity: "))
        sku = input("Enter SKU: ")

        if product_type == "1":
            warranty_period = input("Enter warranty period: ")
            brand = input("Enter brand: ")
            product = Electronics(name, price, quantity, sku, warranty_period, brand)
            inventory.add_product(product)
        elif product_type == "2":
```

```
            size = input("Enter size: ")
            material = input("Enter material: ")
            product = Clothing(name, price, quantity, sku, size, material)
            inventory.add_product(product)
        elif product_type == "3":
            expiration_date = input("Enter expiration date (YYYY-MM-DD): ")
            organic_status = input("Is it organic? (yes/no): ").lower() == "yes"
            product = Groceries(name, price, quantity, sku, expiration_date, organic_status)
            inventory.add_product(product)
        else:
            print("Invalid product type.")

    elif choice == "2":
        sku = input("Enter SKU of the product to update stock: ")
        quantity = int(input("Enter quantity to add: "))
        inventory.update_stock(sku, quantity)

    elif choice == "3":
        total_value = inventory.calculate_total_value()
        print(f"Total inventory value: ${total_value:.2f}")

    elif choice == "4":
        sku = input("Enter SKU of the product to apply discount: ")
        discount_percentage = float(input("Enter discount percentage: "))
        inventory.apply_discount(sku, discount_percentage)

    elif choice == "5":
```

```
            low_stock_items = inventory.check_low_stock()
            if low_stock_items:
                print("Low stock items:")
                for item in low_stock_items:
                    print(f"{item.name} (Quantity: {item.quantity})")
            else:
                print("No low stock items.")

        elif choice == "6":
            break

        else:
            print("Invalid choice, please try again.")
```

OUTPUT:

```
1. Add Product
2. Update Stock
3. Calculate Total Inventory Value
4. Apply Discount
5. Check Low Stock Items
6. Exit
Enter your choice:  1

Select Product Type:
1. Electronics
2. Clothing
3. Groceries
Enter choice (1/2/3):  1
Enter product name:  fRUITS
Enter product price:  500
Enter product quantity:  5
Enter SKU:  9
Enter warranty period:  1
Enter brand:  NIOI
Added fRUITS to inventory.

1. Add Product
2. Update Stock
3. Calculate Total Inventory Value
4. Apply Discount
```

5. Check Low Stock Items
6. Exit

[ ]:

Exercise 2

Building a Payment Processing System

You are tasked with designing a Payment Processing System that handles multiple payment methods (Credit Card, PayPal, Bank Transfer). Each payment method has unique steps involved in processing payments, but they all share the common interface of processing a payment and issuing a refund.

Problem Definition:

The system must support the following payment methods:

1. Credit Card Payment: Requires card number, expiry date, and CVV to process payments.

2. PayPal Payment: Uses a PayPal account email and password.

3. Bank Transfer Payment: Processes payments using a bank account number and a sort code.

Each payment method has:

- **A method to process payments.**
- **A method to issue refunds.**
- **Error handling for failed payments.**

```
class PaymentMethod:
    def process_payment(self, amount):
        raise NotImplementedError("This method should be overridden in subclasses")

    def issue_refund(self, amount):
```

```
        raise NotImplementedError("This method should be overridden in subclasses")


class CreditCardPayment(PaymentMethod):
    def __init__(self, card_number, expiry_date, cvv):
        self.card_number = card_number
        self.expiry_date = expiry_date
        self.cvv = cvv

    def process_payment(self, amount):
        if self.validate_card():
            print(f"Processing credit card payment of ${amount}")
        else:
            print("Credit card validation failed.")

    def issue_refund(self, amount):
        print(f"Issuing credit card refund of ${amount}")

    def validate_card(self):
        return True

class PayPalPayment(PaymentMethod):
    def __init__(self, email, password):
        self.email = email
        self.password = password
```

```
    def process_payment(self, amount):
        if self.authenticate():
            print(f"Processing PayPal payment of ${amount}")
        else:
            print("PayPal authentication failed.")

    def issue_refund(self, amount):
        print(f"Issuing PayPal refund of ${amount}")

    def authenticate(self):
        return True

class BankTransferPayment(PaymentMethod):
    def __init__(self, account_number, sort_code):
        self.account_number = account_number
        self.sort_code = sort_code

    def process_payment(self, amount):
        print(f"Processing bank transfer of ${amount}")

    def issue_refund(self, amount):
        print(f"Issuing bank transfer refund of ${amount}")

# User-defined input usage
def main():
```

```
print("Select Payment Method:")
print("1. Credit Card")
print("2. PayPal")
print("3. Bank Transfer")
choice = input("Enter choice (1/2/3): ")

if choice == "1":
    card_number = input("Enter Credit Card Number: ")
    expiry_date = input("Enter Expiry Date (MM/YY): ")
    cvv = input("Enter CVV: ")
    amount = float(input("Enter Payment Amount: "))
    credit_card_payment = CreditCardPayment(card_number, expiry_date, cvv)
    credit_card_payment.process_payment(amount)
    refund_choice = input("Would you like to issue a refund? (y/n): ")
    if refund_choice.lower() == 'y':
        refund_amount = float(input("Enter Refund Amount: "))
        credit_card_payment.issue_refund(refund_amount)

elif choice == "2":
    email = input("Enter PayPal Email: ")
    password = input("Enter PayPal Password: ")
    amount = float(input("Enter Payment Amount: "))
    paypal_payment = PayPalPayment(email, password)
    paypal_payment.process_payment(amount)
    refund_choice = input("Would you like to issue a refund? (y/n): ")
```

```
        if refund_choice.lower() == 'y':
            refund_amount = float(input("Enter Refund Amount: "))
            paypal_payment.issue_refund(refund_amount)

    elif choice == "3":
        account_number = input("Enter Bank Account Number: ")
        sort_code = input("Enter Sort Code: ")
        amount = float(input("Enter Payment Amount: "))
        bank_transfer_payment = BankTransferPayment(account_number, sort_code)
        bank_transfer_payment.process_payment(amount)
        refund_choice = input("Would you like to issue a refund? (y/n): ")
        if refund_choice.lower() == 'y':
            refund_amount = float(input("Enter Refund Amount: "))
            bank_transfer_payment.issue_refund(refund_amount)

    else:
        print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Select Payment Method:
1. Credit Card
2. PayPal
3. Bank Transfer
Enter choice (1/2/3):  1
Enter Credit Card Number:  122
Enter Expiry Date (MM/YY):  10/27
Enter CVV:  805
Enter Payment Amount:  5000
```

Processing credit card payment of $5000.0
Would you like to issue a refund? (y/n): n