Task 3 Create Python Classes and Objects:

Code:

```
# Define the Order class
class Order:
    # Constructor to initialize Order object with 5 attributes
    def __init__(self, order_number, reference_number, delivery_date, status, total_weight):
        self._order_number = order_number  # Unique identifier for the order
        self._reference_number = reference_number  # Reference number for tracking
        self._delivery_date = delivery_date  # Scheduled delivery date
        self._status = status  # Current status of the order (e.g., Pending, Delivered)
        self._total_weight = total_weight  # Total weight of the package

    # Getter method for order_number
    def get_order_number(self):
        return self._order_number

    # Getter method for reference_number
    def get_reference_number(self):
        return self._reference_number

    # Getter method for delivery_date
```

```
def get_delivery_date(self):
    return self._delivery_date

# Getter method for status
def get_status(self):
    return self._status

# Getter method for total_weight
def get_total_weight(self):
    return self._total_weight

# Setter method for order_number
def set_order_number(self, order_number):
    self._order_number = order_number

# Setter method for reference_number
def set_reference_number(self, reference_number):
    self._reference_number = reference_number

# Setter method for delivery_date
def set_delivery_date(self, delivery_date):
    self._delivery_date = delivery_date
```

```
    # Setter method for status
    def set_status(self, status):
        self._status = status

    # Setter method for total_weight
    def set_total_weight(self, total_weight):
        self._total_weight = total_weight

    # Method to create a new order (currently a placeholder)
    def create_order(self):
        """Creates a new order in the system."""
        pass

    # Method to update the status of the order (currently a placeholder)
    def update_status(self):
        """Updates the status of the order."""
        pass


# Define the Recipient class
class Recipient:
    # Constructor to initialize Recipient object with 5 attributes
    def __init__(self, name, contact, address, email, phone):
```

```
    self._name = name  # Name of the recipient
    self._contact = contact  # Contact information of the recipient
    self._address = address  # Delivery address of the recipient
    self._email = email  # Email address of the recipient
    self._phone = phone  # Phone number of the recipient

# Getter method for name
def get_name(self):
    return self._name

# Getter method for contact
def get_contact(self):
    return self._contact

# Getter method for address
def get_address(self):
    return self._address

# Getter method for email
def get_email(self):
    return self._email

# Getter method for phone
```

```
def get_phone(self):
    return self._phone

# Setter method for name
def set_name(self, name):
    self._name = name

# Setter method for contact
def set_contact(self, contact):
    self._contact = contact

# Setter method for address
def set_address(self, address):
    self._address = address

# Setter method for email
def set_email(self, email):
    self._email = email

# Setter method for phone
def set_phone(self, phone):
    self._phone = phone
```

```
    # Method to update recipient's contact information (currently a placeholder)
    def update_contact(self):
        """Updates the recipient's contact information."""
        pass

    # Method to update recipient's delivery address (currently a placeholder)
    def update_address(self):
        """Updates the recipient's delivery address."""
        pass



# Define the Item class
class Item:
    # Constructor to initialize Item object with 5 attributes
    def __init__(self, item_code, description, quantity, unit_price, total_price):
        self._item_code = item_code  # Unique identifier for the item
        self._description = description  # Description of the item
        self._quantity = quantity  # Quantity of the item
        self._unit_price = unit_price  # Price of one unit of the item
        self._total_price = total_price  # Total price of the item (quantity * unit price)

    # Getter method for item_code
    def get_item_code(self):
```

```
    return self._item_code

# Getter method for description
def get_description(self):
    return self._description

# Getter method for quantity
def get_quantity(self):
    return self._quantity

# Getter method for unit_price
def get_unit_price(self):
    return self._unit_price

# Getter method for total_price
def get_total_price(self):
    return self._total_price

# Setter method for item_code
def set_item_code(self, item_code):
    self._item_code = item_code

# Setter method for description
```

```
    def set_description(self, description):
        self._description = description

    # Setter method for quantity
    def set_quantity(self, quantity):
        self._quantity = quantity

    # Setter method for unit_price
    def set_unit_price(self, unit_price):
        self._unit_price = unit_price

    # Setter method for total_price
    def set_total_price(self, total_price):
        self._total_price = total_price

    # Method to calculate total price of the item (currently a placeholder)
    def calculate_total_price(self):
        """Calculates the total price of the item (quantity * unit price)."""
        pass


# Define the Delivery class
class Delivery:
```

```
# Constructor to initialize Delivery object with 5 attributes
def __init__(self, delivery_id, order_number, status, delivery_date, delivery_personnel):
    self._delivery_id = delivery_id  # Unique identifier for the delivery
    self._order_number = order_number  # Order number associated with the delivery
    self._status = status  # Current status of the delivery (e.g., In Transit, Delivered)
    self._delivery_date = delivery_date  # Scheduled delivery date
    self._delivery_personnel = delivery_personnel  # Name of the delivery personnel

# Getter method for delivery_id
def get_delivery_id(self):
    return self._delivery_id

# Getter method for order_number
def get_order_number(self):
    return self._order_number

# Getter method for status
def get_status(self):
    return self._status

# Getter method for delivery_date
def get_delivery_date(self):
    return self._delivery_date
```

```
# Getter method for delivery_personnel
def get_delivery_personnel(self):
    return self._delivery_personnel

# Setter method for delivery_id
def set_delivery_id(self, delivery_id):
    self._delivery_id = delivery_id

# Setter method for order_number
def set_order_number(self, order_number):
    self._order_number = order_number

# Setter method for status
def set_status(self, status):
    self._status = status

# Setter method for delivery_date
def set_delivery_date(self, delivery_date):
    self._delivery_date = delivery_date

# Setter method for delivery_personnel
def set_delivery_personnel(self, delivery_personnel):
```

```
        self._delivery_personnel = delivery_personnel

    # Method to track the delivery status (currently a placeholder)
    def track_delivery(self):
        """Tracks the current status of the delivery."""
        pass


# Define the Notification class
class Notification:
    # Constructor to initialize Notification object with 5 attributes
    def __init__(self, notification_id, recipient_contact, message, timestamp, status):
        self._notification_id = notification_id  # Unique identifier for the notification
        self._recipient_contact = recipient_contact  # Contact information of the recipient
        self._message = message  # Content of the notification
        self._timestamp = timestamp  # Date and time when the notification was sent
        self._status = status  # Status of the notification (e.g., Sent, Delivered)

    # Getter method for notification_id
    def get_notification_id(self):
        return self._notification_id

    # Getter method for recipient_contact
```

```
def get_recipient_contact(self):
    return self._recipient_contact

# Getter method for message
def get_message(self):
    return self._message

# Getter method for timestamp
def get_timestamp(self):
    return self._timestamp

# Getter method for status
def get_status(self):
    return self._status

# Setter method for notification_id
def set_notification_id(self, notification_id):
    self._notification_id = notification_id

# Setter method for recipient_contact
def set_recipient_contact(self, recipient_contact):
    self._recipient_contact = recipient_contact
```

```
    # Setter method for message
    def set_message(self, message):
        self._message = message

    # Setter method for timestamp
    def set_timestamp(self, timestamp):
        self._timestamp = timestamp

    # Setter method for status
    def set_status(self, status):
        self._status = status

    # Method to send a notification (currently a placeholder)
    def send_notification(self):
        """Sends a notification to the recipient."""
        pass


# Define the Inventory class
class Inventory:
    # Constructor to initialize Inventory object with 5 attributes
    def __init__(self, item_code, quantity_in_stock, item_name, location, reorder_level):
        self._item_code = item_code  # Unique identifier for the item
```

```
    self._quantity_in_stock = quantity_in_stock  # Current quantity of the item in stock
    self._item_name = item_name  # Name of the item
    self._location = location  # Location of the item in the warehouse
    self._reorder_level = reorder_level  # Minimum stock level before reordering

# Getter method for item_code
def get_item_code(self):
    return self._item_code

# Getter method for quantity_in_stock
def get_quantity_in_stock(self):
    return self._quantity_in_stock

# Getter method for item_name
def get_item_name(self):
    return self._item_name

# Getter method for location
def get_location(self):
    return self._location

# Getter method for reorder_level
def get_reorder_level(self):
```

```
    return self._reorder_level

# Setter method for item_code
def set_item_code(self, item_code):
    self._item_code = item_code

# Setter method for quantity_in_stock
def set_quantity_in_stock(self, quantity_in_stock):
    self._quantity_in_stock = quantity_in_stock

# Setter method for item_name
def set_item_name(self, item_name):
    self._item_name = item_name

# Setter method for location
def set_location(self, location):
    self._location = location

# Setter method for reorder_level
def set_reorder_level(self, reorder_level):
    self._reorder_level = reorder_level

# Method to update the stock level (currently a placeholder)
```

```
    def update_stock(self):
        """Updates the stock level of the item."""
        pass



# Example usage of the classes
if __name__ == "__main__":
    # Create an Order object
    order = Order("ORD555", "EOC321", "2025-01-25", "Pending", 7.5)
    print("Order Number:", order.get_order_number())  # Print the order number

    # Create a Recipient object
    recipient = Recipient("khlood mohamed", "khlood.mohamed@hotmail.com", "155 first St",
"khlood.mohamed@hotmail.com", "971-345-1234")
    print("Recipient Name:", recipient.get_name())  # Print the recipient's name

    # Create an Item object
    item = Item("MNL007", "Charger", 1, 50.0, 50.0)
    print("Item Code:", item.get_item_code())  # Print the item code

    # Create a Delivery object
    delivery = Delivery("DEL001", "ORD55", "In Transit", "2025-01-25", "Khlood")
    print("Delivery Status:", delivery.get_status())  # Print the delivery status
```

```
    # Create a Notification object
    notification = Notification("NOTIF001", "khlood.mohamed@ehotmail.com", "Your order is out for delivery.", "2025-01-25 8:00 AM", "Sent")
    print("Notification Message:", notification.get_message())  # Print the notification message

    # Create an Inventory object
    inventory = Inventory("ITM674", 60, "charger", "Warehouse A", 10)
    print("Item Name:", inventory.get_item_name())  # Print the item name
```

Output:

```
Order Number: ORD555
Recipient Name: khlood mohamed
Item Code: MNL007
Delivery Status: In Transit
Notification Message: Your order is out for delivery.
Item Name: charger
```