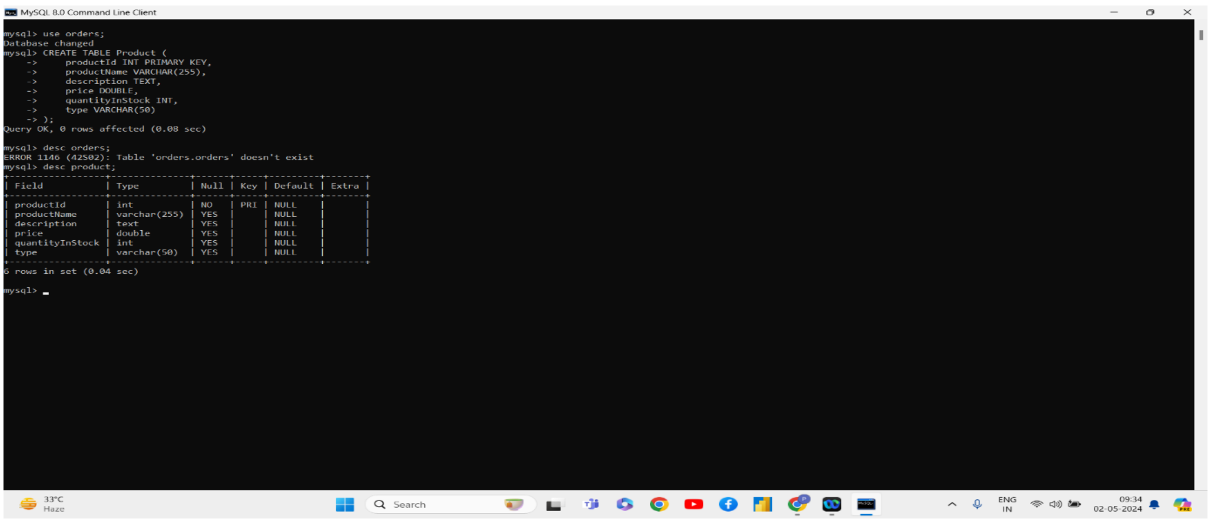
# Coding Challenge - Order Management System

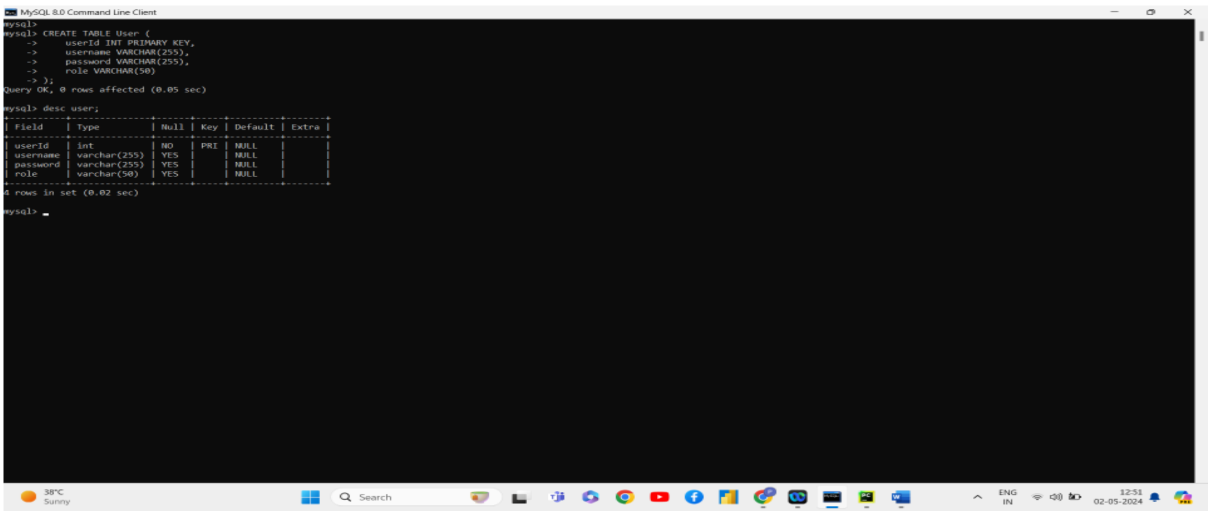
**Create SQL Schema from the product and user class, use the class attributes for table column names.**

**1. Create a base class called Product with the following attributes:**

- productId (int)
- productName (String)
- description (String)
- price (double)
- quantityInStock (int)
- type (String) [Electronics/Clothing]

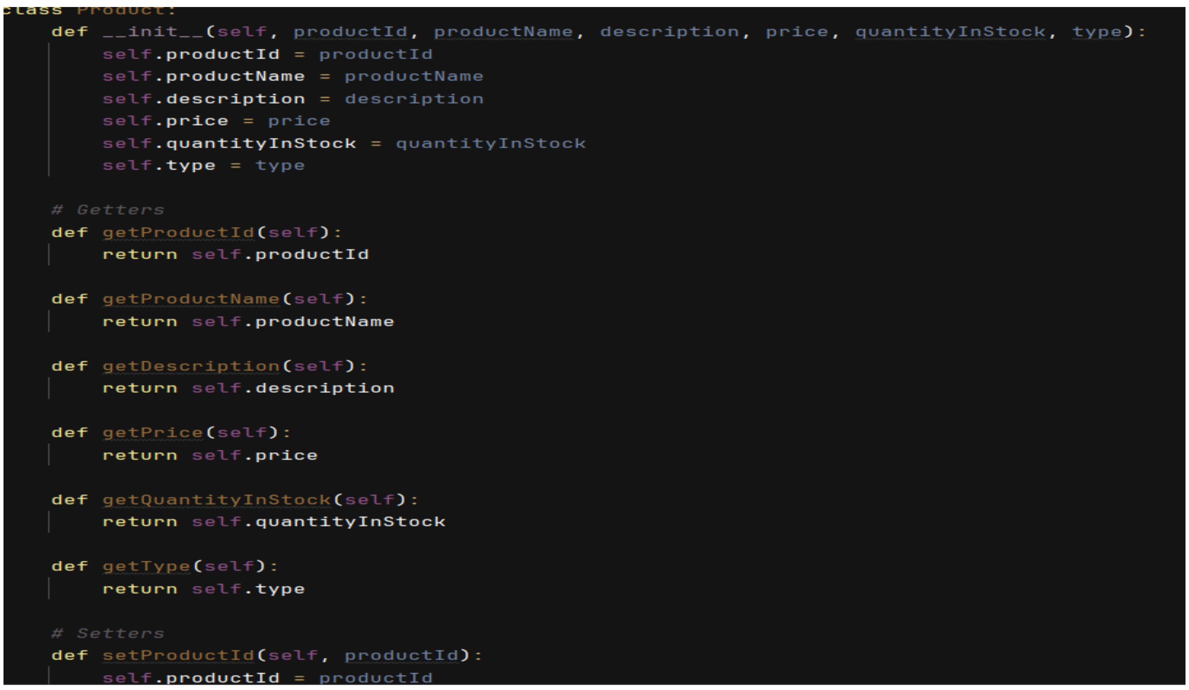


```
mysql> use orders;
Database changed
mysql> CREATE TABLE Product (
    ->     productId INT PRIMARY KEY,
    ->     productName VARCHAR(255),
    ->     description TEXT,
    ->     price DOUBLE,
    ->     quantityInStock INT,
    ->     type VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> desc orders;
ERROR 1146 (42S02): Table 'orders.orders' doesn't exist
mysql> desc product;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| productId       | int          | NO   | PRI | NULL    |       |
| productName     | varchar(255) | YES  |     | NULL    |       |
| description     | text         | YES  |     | NULL    |       |
| price           | double       | YES  |     | NULL    |       |
| quantityInStock | int          | YES  |     | NULL    |       |
| type            | varchar(50)  | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
6 rows in set (0.04 sec)

mysql> _
```

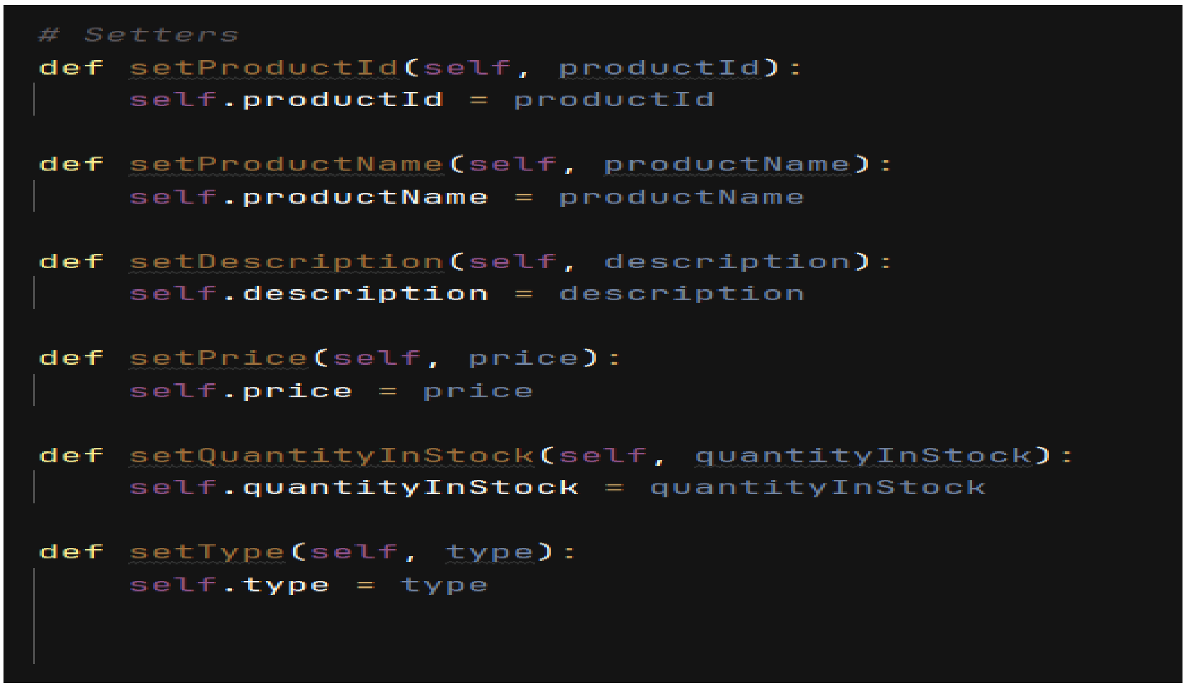


```
mysql>
mysql> CREATE TABLE User (
    ->     userId INT PRIMARY KEY,
    ->     username VARCHAR(255),
    ->     password VARCHAR(255),
    ->     role VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| userId   | int          | NO   | PRI | NULL    |       |
| username | varchar(255) | YES  |     | NULL    |       |
| password | varchar(255) | YES  |     | NULL    |       |
| role     | varchar(50)  | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.02 sec)

mysql> _
```

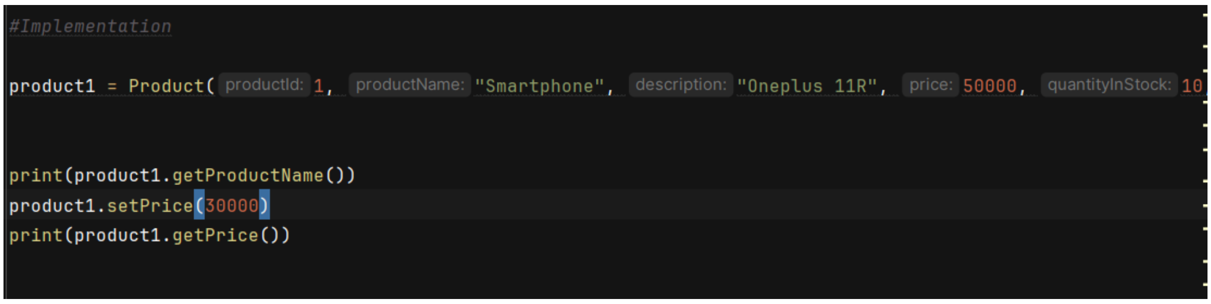
**2. Implement constructors, getters, and setters for the Product class.**



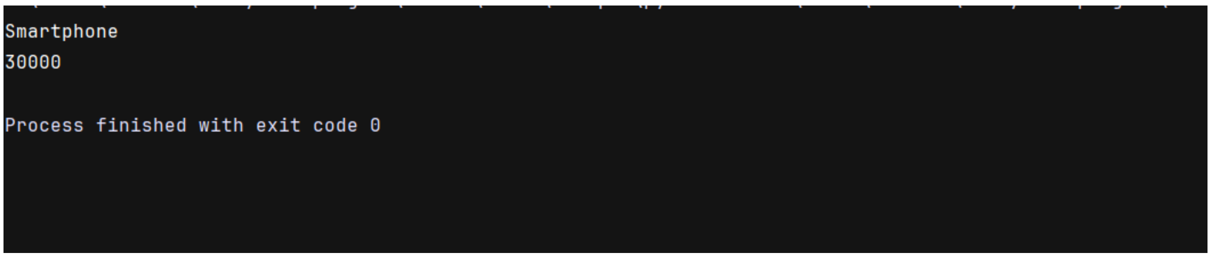
```python
class Product:
    def __init__(self, productId, productName, description, price, quantityInStock, type):
        self.productId = productId
        self.productName = productName
        self.description = description
        self.price = price
        self.quantityInStock = quantityInStock
        self.type = type

    # Getters
    def getProductId(self):
        return self.productId

    def getProductName(self):
        return self.productName

    def getDescription(self):
        return self.description

    def getPrice(self):
        return self.price

    def getQuantityInStock(self):
        return self.quantityInStock

    def getType(self):
        return self.type

    # Setters
    def setProductId(self, productId):
        self.productId = productId
```



```python
# Setters
def setProductId(self, productId):
    self.productId = productId

def setProductName(self, productName):
    self.productName = productName

def setDescription(self, description):
    self.description = description

def setPrice(self, price):
    self.price = price

def setQuantityInStock(self, quantityInStock):
    self.quantityInStock = quantityInStock

def setType(self, type):
    self.type = type
```



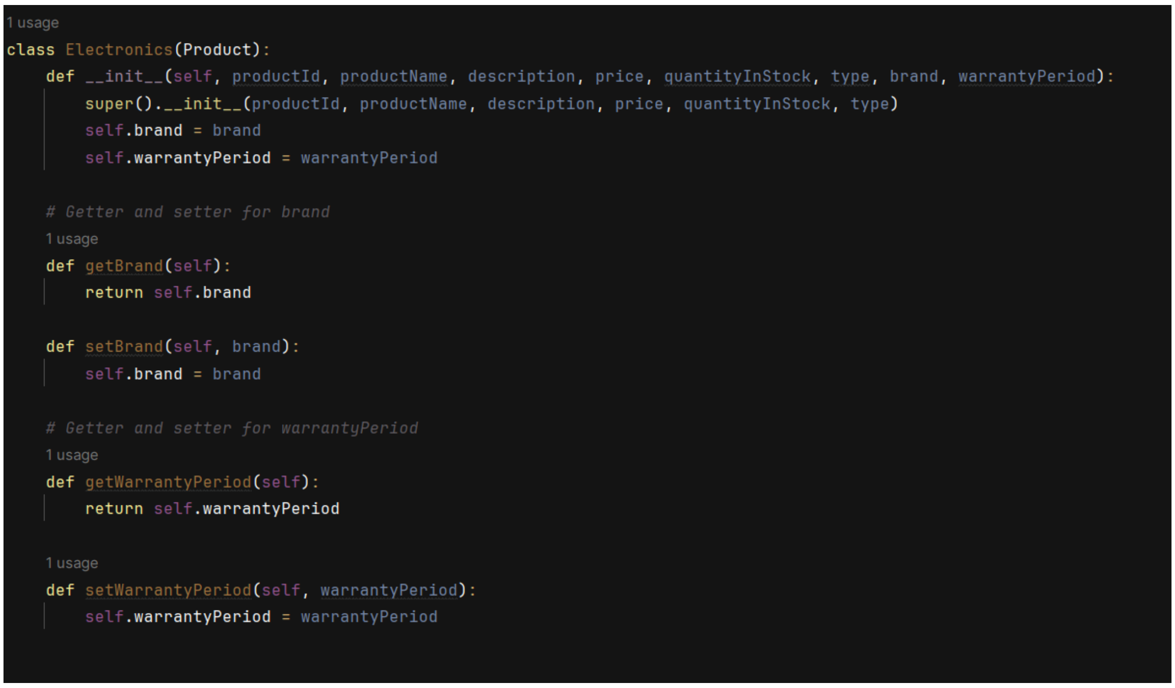
```
#Implementation

product1 = Product( productId: 1,  productName: "Smartphone",  description: "Oneplus 11R",  price: 50000,  quantityInStock: 10

print(product1.getProductName())
product1.setPrice(30000)
print(product1.getPrice())
```



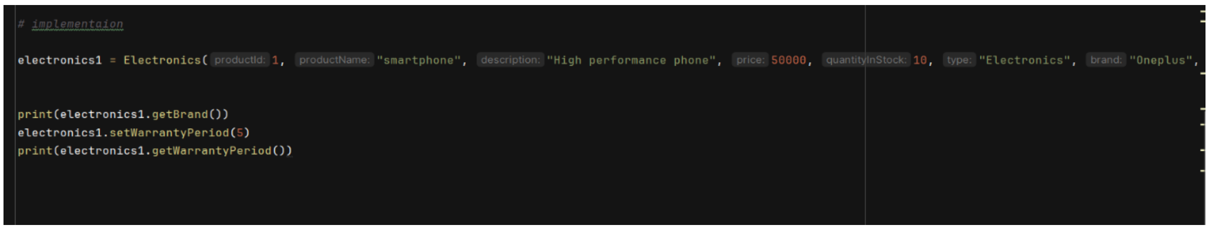
```
Smartphone
30000

Process finished with exit code 0
```

**3. Create a subclass Electronics that inherits from Product. Add attributes specific to electronics products, such as:**

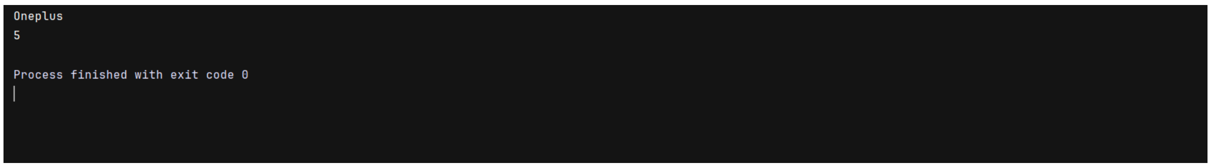
- brand (String)
- warrantyPeriod (int)



```
1 usage
class Electronics(Product):
    def __init__(self, productId, productName, description, price, quantityInStock, type, brand, warrantyPeriod):
        super().__init__(productId, productName, description, price, quantityInStock, type)
        self.brand = brand
        self.warrantyPeriod = warrantyPeriod

    # Getter and setter for brand
    1 usage
    def getBrand(self):
        return self.brand

    def setBrand(self, brand):
        self.brand = brand

    # Getter and setter for warrantyPeriod
    1 usage
    def getWarrantyPeriod(self):
        return self.warrantyPeriod

    1 usage
    def setWarrantyPeriod(self, warrantyPeriod):
        self.warrantyPeriod = warrantyPeriod
```



```
# implementaion

electronics1 = Electronics( productId: 1,  productName: "smartphone",  description: "High performance phone",  price: 50000,  quantityInStock: 10,  type: "Electronics",  brand: "Oneplus",


print(electronics1.getBrand())
electronics1.setWarrantyPeriod(5)
print(electronics1.getWarrantyPeriod())
```



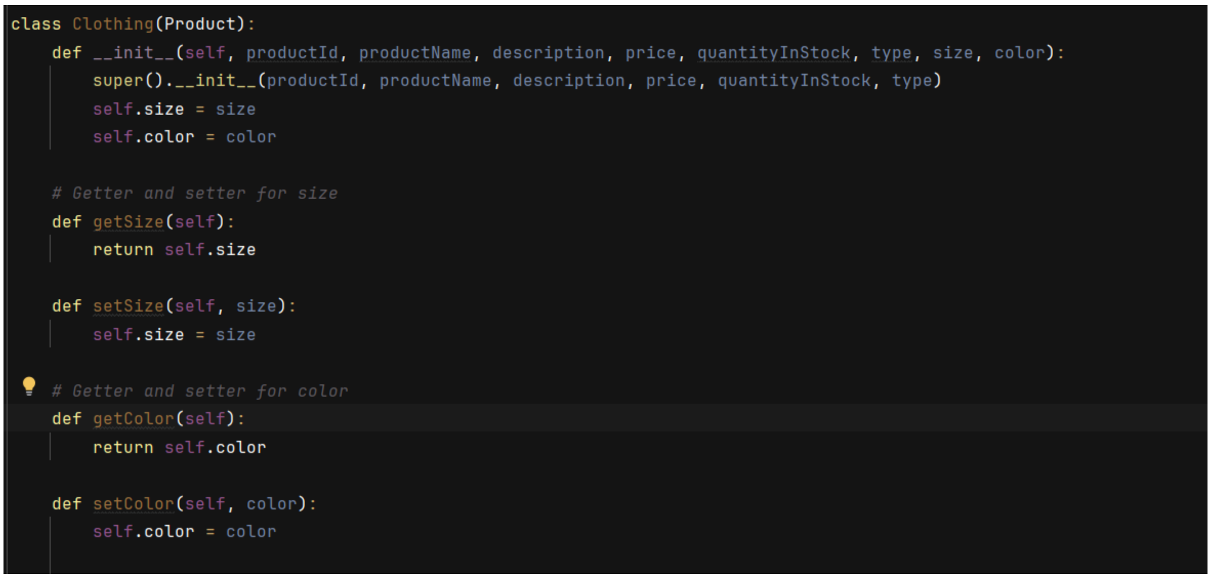
```
Oneplus
5

Process finished with exit code 0
```

**4. Create a subclass Clothing that also inherits from Product. Add attributes specific to clothing**

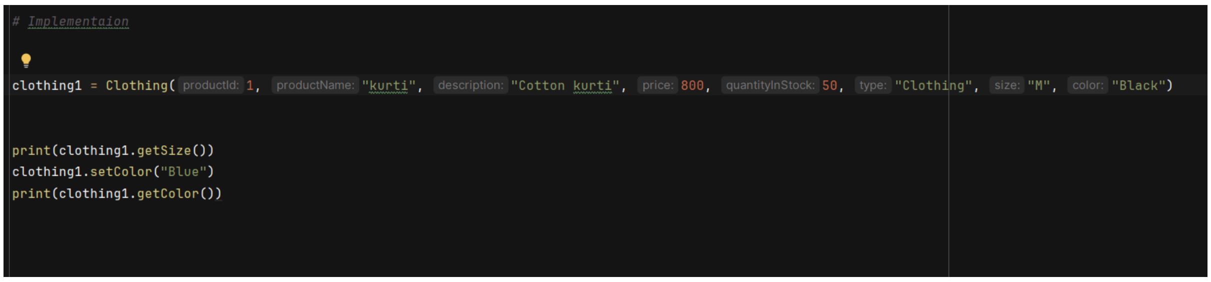
products, such as:

• size (String)

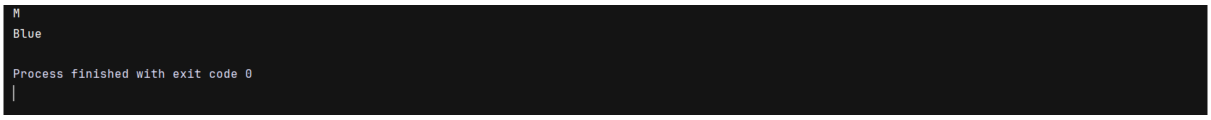
• color (String)



```
class Clothing(Product):
    def __init__(self, productId, productName, description, price, quantityInStock, type, size, color):
        super().__init__(productId, productName, description, price, quantityInStock, type)
        self.size = size
        self.color = color

    # Getter and setter for size
    def getSize(self):
        return self.size

    def setSize(self, size):
        self.size = size

    # Getter and setter for color
    def getColor(self):
        return self.color

    def setColor(self, color):
        self.color = color
```



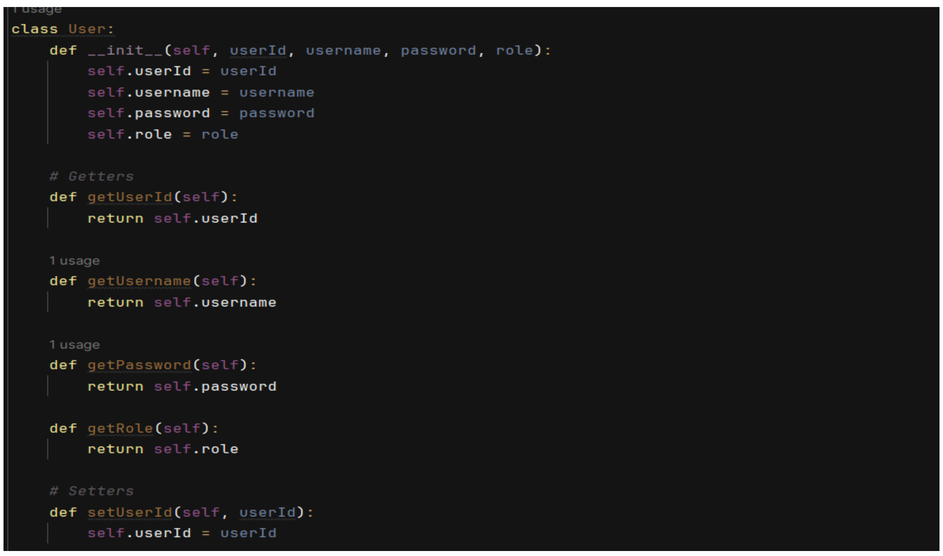
```
# Implementaion

clothing1 = Clothing( productId: 1,  productName: "kurti",  description: "Cotton kurti",  price: 800,  quantityInStock: 50,  type: "Clothing",  size: "M",  color: "Black")


print(clothing1.getSize())
clothing1.setColor("Blue")
print(clothing1.getColor())
```



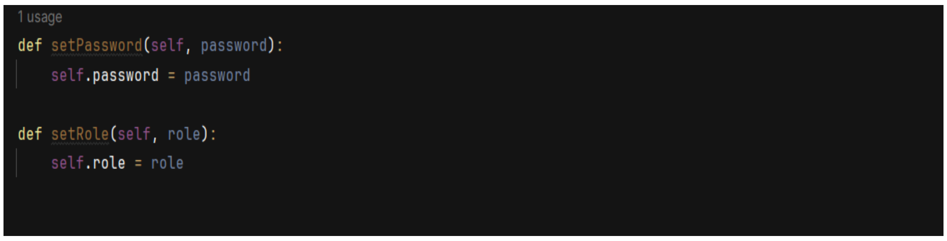
```
M
Blue

Process finished with exit code 0
```

**5. Create a User class with attributes**:

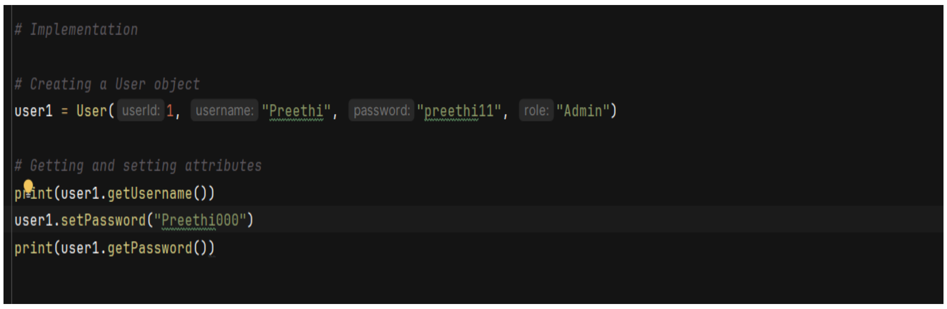
- userId (int)
- username (String)
- password (String)
- role (String) // "Admin" or "User"



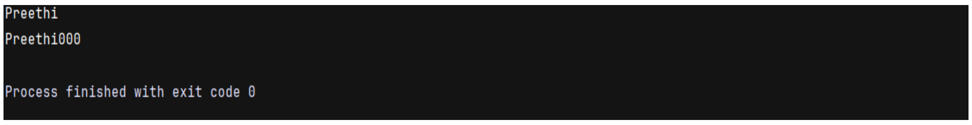
```
1 usage
class User:
    def __init__(self, userId, username, password, role):
        self.userId = userId
        self.username = username
        self.password = password
        self.role = role

    # Getters
    def getUserId(self):
        return self.userId

    1 usage
    def getUsername(self):
        return self.username

    1 usage
    def getPassword(self):
        return self.password

    def getRole(self):
        return self.role

    # Setters
    def setUserId(self, userId):
        self.userId = userId
```



```
1 usage
def setPassword(self, password):
    self.password = password

def setRole(self, role):
    self.role = role
```



```
# Implementation

# Creating a User object
user1 = User( userId: 1,  username: "Preethi",  password: "preethi11",  role: "Admin")

# Getting and setting attributes
print(user1.getUsername())
user1.setPassword("Preethi000")
print(user1.getPassword())
```

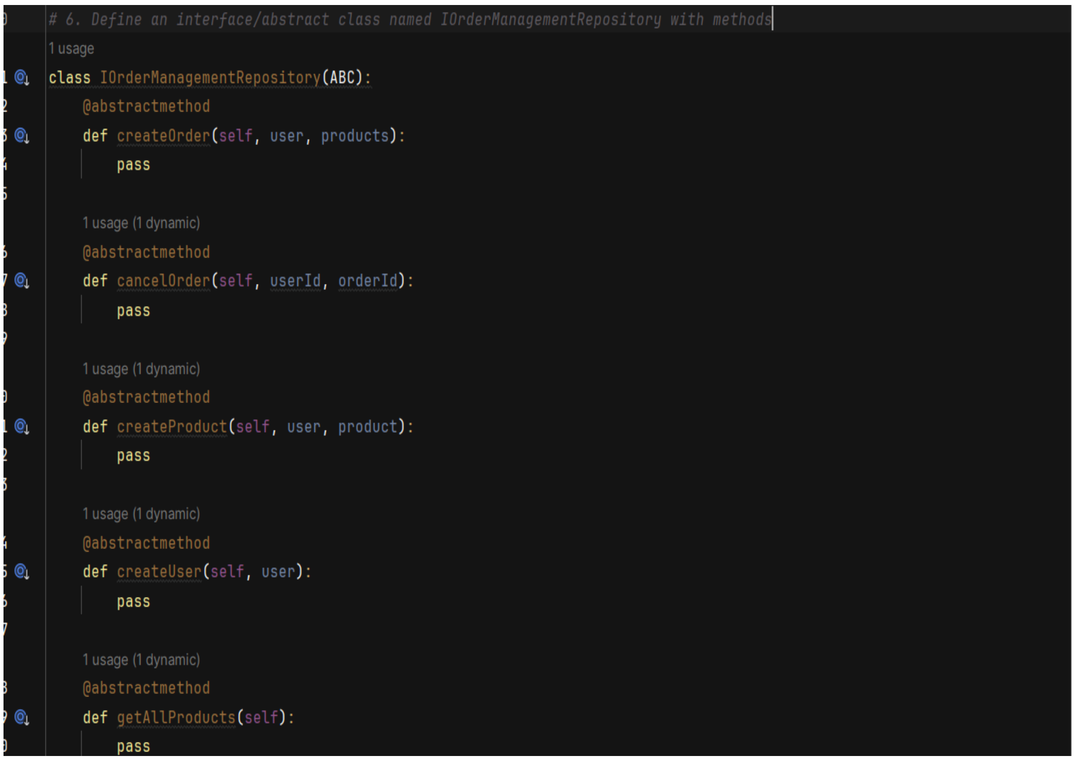


```
Preethi
Preethi000

Process finished with exit code 0
```

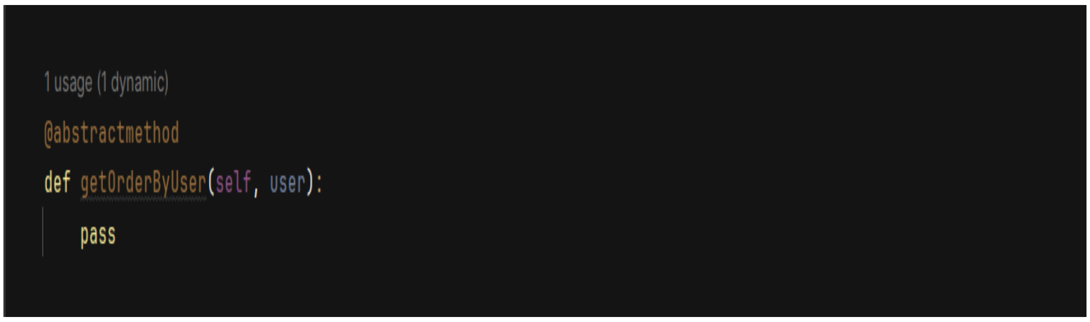
**6. Define an interface/abstract class named IOrderManagementRepository with methods for:**

• createOrder(User user, list of products): check the user as already present in database

to create order or create user (store in database) and create order.

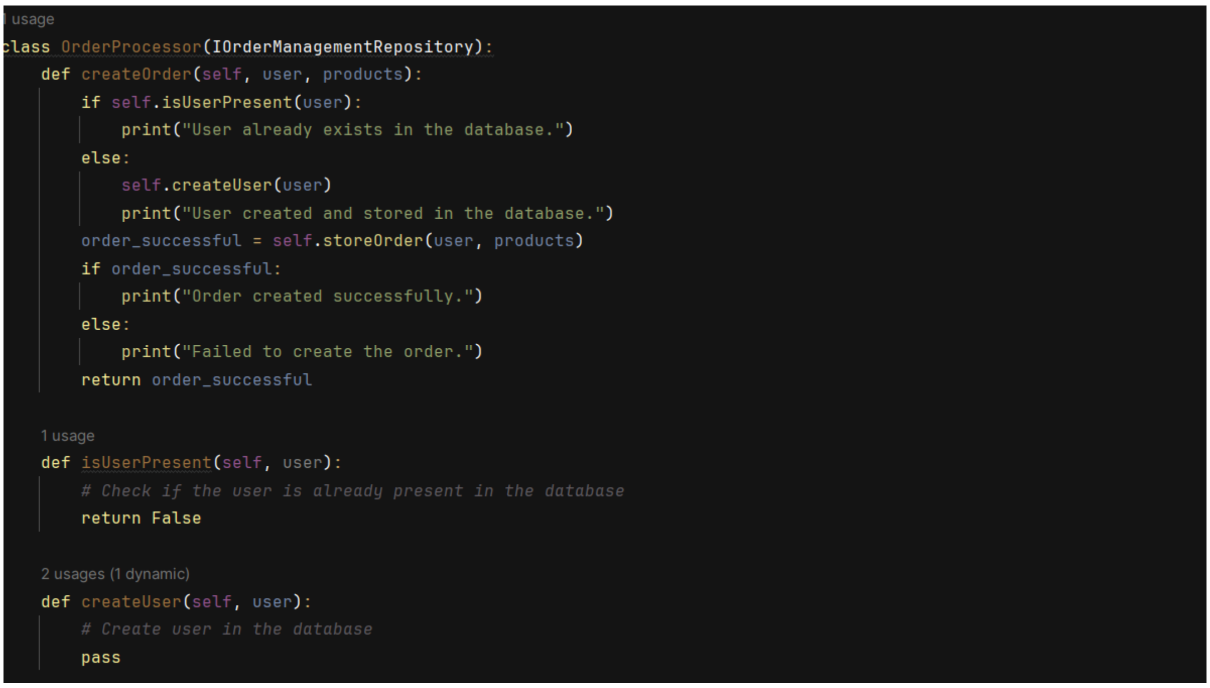


```
# 6. Define an interface/abstract class named IOrderManagementRepository with methods
1 usage
class IOrderManagementRepository(ABC):
    @abstractmethod
    def createOrder(self, user, products):
        pass

    1 usage (1 dynamic)
    @abstractmethod
    def cancelOrder(self, userId, orderId):
        pass

    1 usage (1 dynamic)
    @abstractmethod
    def createProduct(self, user, product):
        pass

    1 usage (1 dynamic)
    @abstractmethod
    def createUser(self, user):
        pass

    1 usage (1 dynamic)
    @abstractmethod
    def getAllProducts(self):
        pass
```

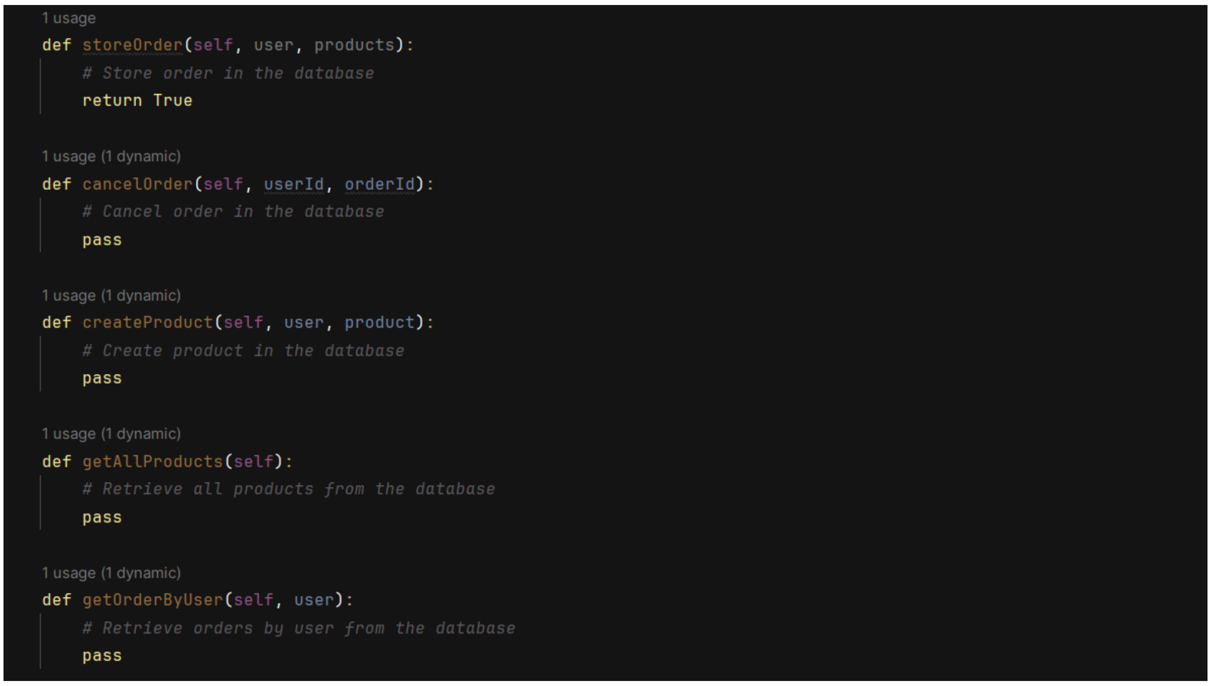


```
1 usage (1 dynamic)
@abstractmethod
def getOrderByUser(self, user):
    pass
```

**7. Implement the IOrderManagementRepository interface/abstractclass in a class called OrderProcessor. This class will be responsible for managing orders.**



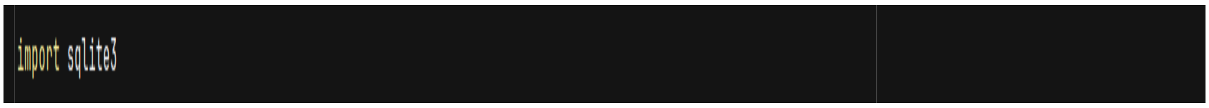
```
1 usage
class OrderProcessor(IOrderManagementRepository):
    def createOrder(self, user, products):
        if self.isUserPresent(user):
            print("User already exists in the database.")
        else:
            self.createUser(user)
            print("User created and stored in the database.")
        order_successful = self.storeOrder(user, products)
        if order_successful:
            print("Order created successfully.")
        else:
            print("Failed to create the order.")
        return order_successful

    1 usage
    def isUserPresent(self, user):
        # Check if the user is already present in the database
        return False

    2 usages (1 dynamic)
    def createUser(self, user):
        # Create user in the database
        pass
```



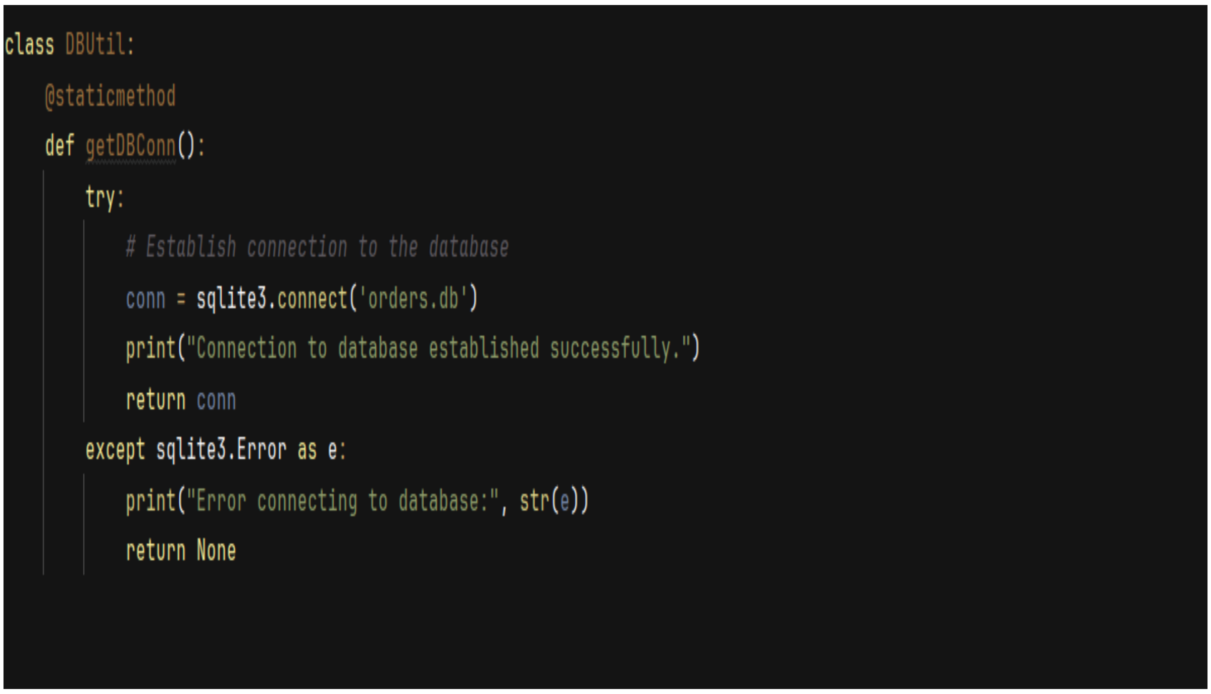
```
    1 usage
    def storeOrder(self, user, products):
        # Store order in the database
        return True

    1 usage (1 dynamic)
    def cancelOrder(self, userId, orderId):
        # Cancel order in the database
        pass

    1 usage (1 dynamic)
    def createProduct(self, user, product):
        # Create product in the database
        pass

    1 usage (1 dynamic)
    def getAllProducts(self):
        # Retrieve all products from the database
        pass

    1 usage (1 dynamic)
    def getOrderByUser(self, user):
        # Retrieve orders by user from the database
        pass
```

**8. Create DBUtil class and add the following method.**

• static getDBConn():Connection Establish a connection to the database and return database Connection



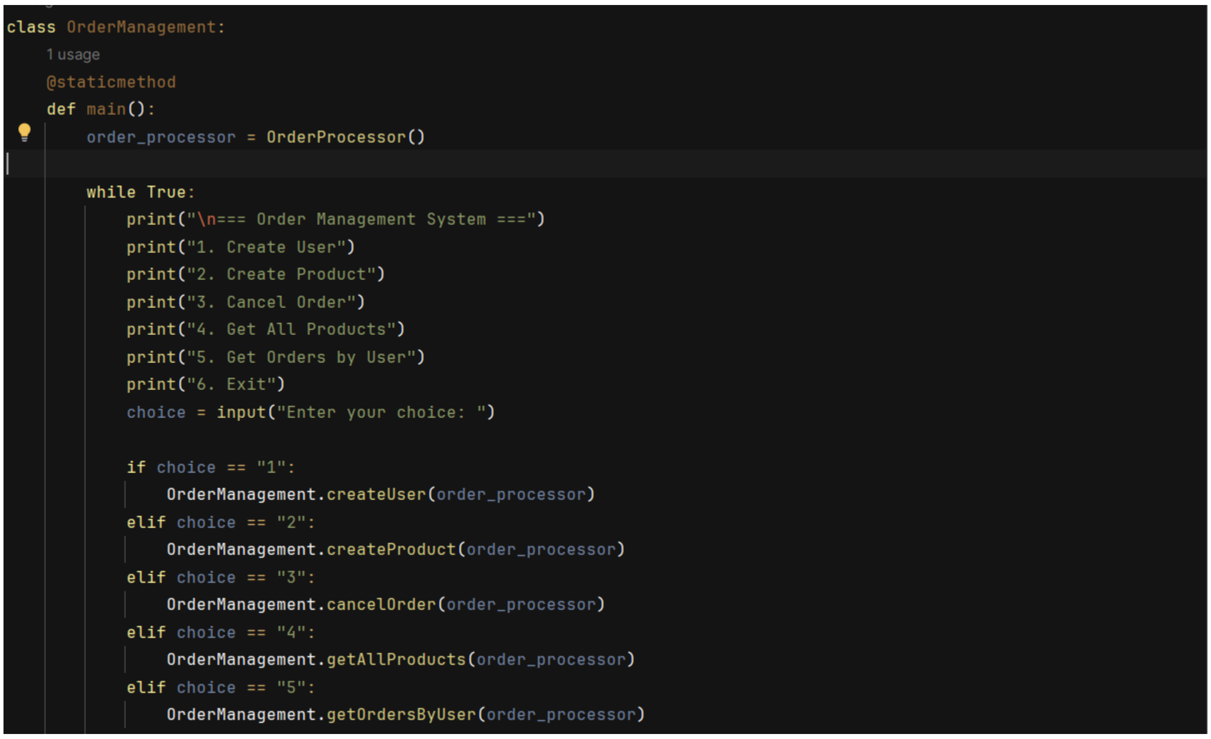
```python
import sqlite3
```



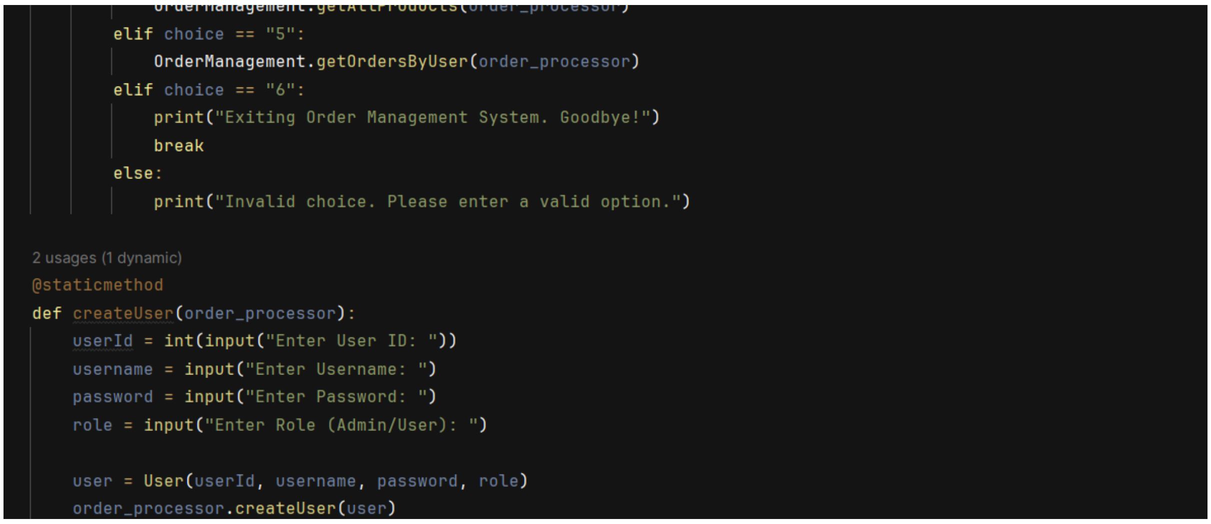
```python
class DBUtil:
    @staticmethod
    def getDBConn():
        try:
            # Establish connection to the database
            conn = sqlite3.connect('orders.db')
            print("Connection to database established successfully.")
            return conn
        except sqlite3.Error as e:
            print("Error connecting to database:", str(e))
            return None
```

**9. Create OrderManagement main class and perform following operation:**

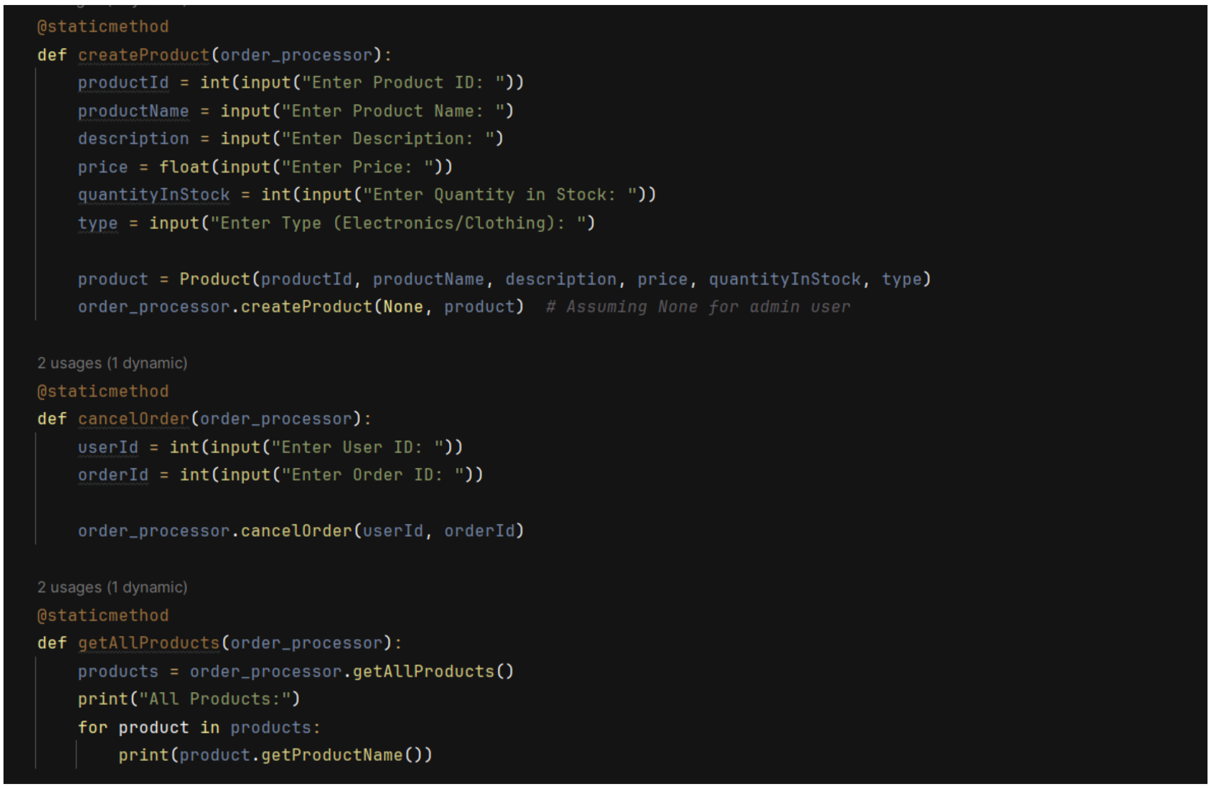
• main method to simulate the loan management system. Allow the user to interact with the system by entering choice from menu such as "createUser", "createProduct", "cancelOrder", "getAllProducts", "getOrderbyUser", "exit".



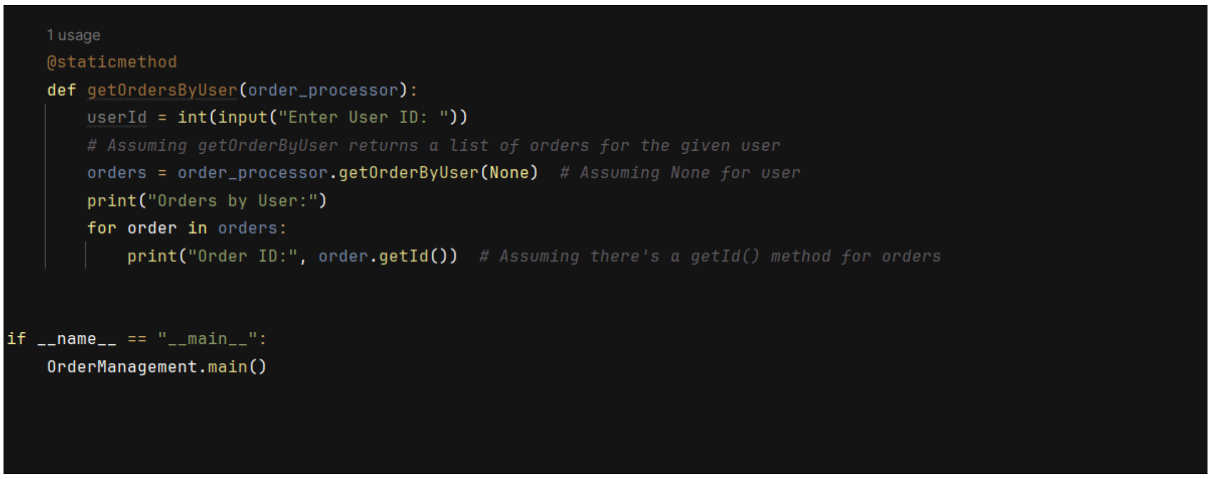
```
class OrderManagement:
    1 usage
    @staticmethod
    def main():
        order_processor = OrderProcessor()

        while True:
            print("\n=== Order Management System ===")
            print("1. Create User")
            print("2. Create Product")
            print("3. Cancel Order")
            print("4. Get All Products")
            print("5. Get Orders by User")
            print("6. Exit")
            choice = input("Enter your choice: ")

            if choice == "1":
                OrderManagement.createUser(order_processor)
            elif choice == "2":
                OrderManagement.createProduct(order_processor)
            elif choice == "3":
                OrderManagement.cancelOrder(order_processor)
            elif choice == "4":
                OrderManagement.getAllProducts(order_processor)
            elif choice == "5":
                OrderManagement.getOrdersByUser(order_processor)
```



```
                OrderManagement.getAllProducts(order_processor)
            elif choice == "5":
                OrderManagement.getOrdersByUser(order_processor)
            elif choice == "6":
                print("Exiting Order Management System. Goodbye!")
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    2 usages (1 dynamic)
    @staticmethod
    def createUser(order_processor):
        userId = int(input("Enter User ID: "))
        username = input("Enter Username: ")
        password = input("Enter Password: ")
        role = input("Enter Role (Admin/User): ")

        user = User(userId, username, password, role)
        order_processor.createUser(user)
```

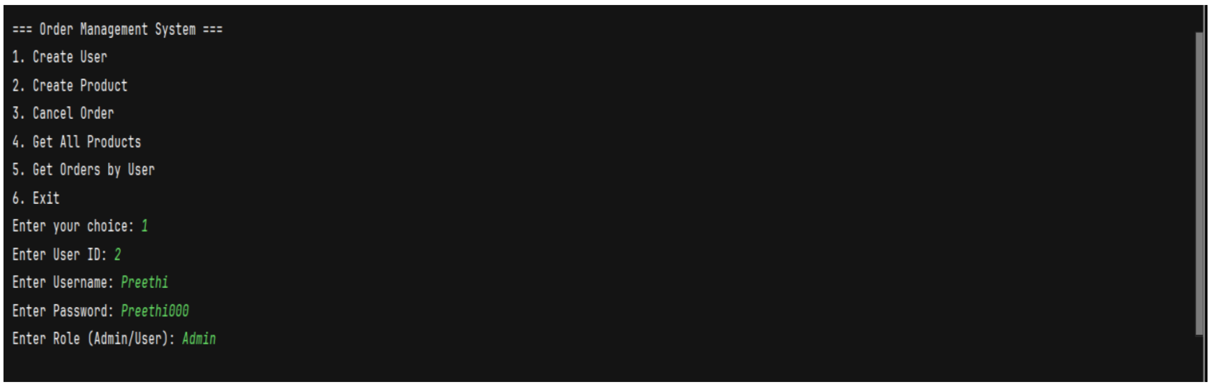


```
    @staticmethod
    def createProduct(order_processor):
        productId = int(input("Enter Product ID: "))
        productName = input("Enter Product Name: ")
        description = input("Enter Description: ")
        price = float(input("Enter Price: "))
        quantityInStock = int(input("Enter Quantity in Stock: "))
        type = input("Enter Type (Electronics/Clothing): ")

        product = Product(productId, productName, description, price, quantityInStock, type)
        order_processor.createProduct(None, product)  # Assuming None for admin user

    2 usages (1 dynamic)
    @staticmethod
    def cancelOrder(order_processor):
        userId = int(input("Enter User ID: "))
        orderId = int(input("Enter Order ID: "))

        order_processor.cancelOrder(userId, orderId)

    2 usages (1 dynamic)
    @staticmethod
    def getAllProducts(order_processor):
        products = order_processor.getAllProducts()
        print("All Products:")
        for product in products:
            print(product.getProductName())
```



```
    1 usage
    @staticmethod
    def getOrdersByUser(order_processor):
        userId = int(input("Enter User ID: "))
        # Assuming getOrderByUser returns a list of orders for the given user
        orders = order_processor.getOrderByUser(None)  # Assuming None for user
        print("Orders by User:")
        for order in orders:
            print("Order ID:", order.getId())  # Assuming there's a getId() method for orders


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

**Output :**



```
=== Order Management System ===
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Orders by User
6. Exit
Enter your choice: 1
Enter User ID: 2
Enter Username: Preethi
Enter Password: Preethi000
Enter Role (Admin/User): Admin
```