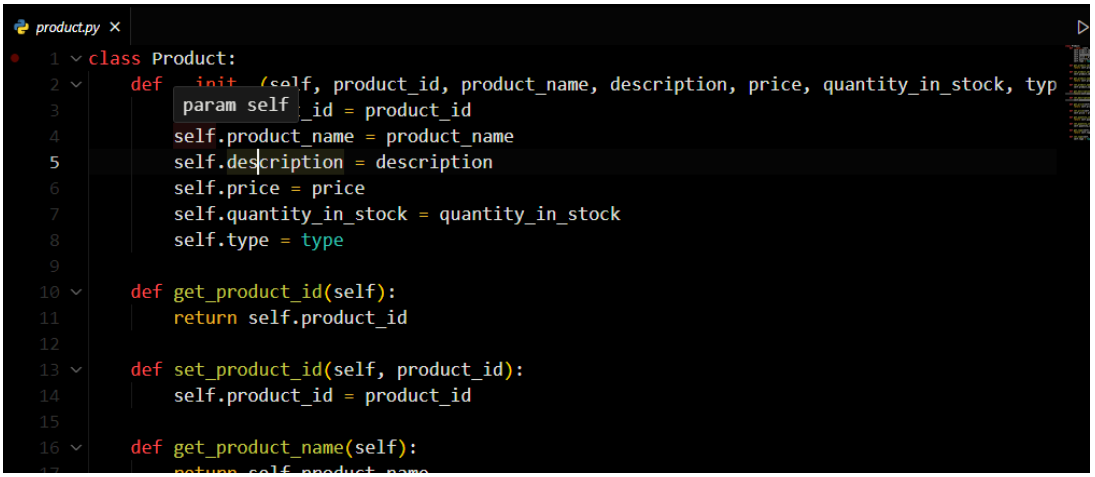
1. Create a base class called Product with the following attributes:
    - productId (int)
    - productName (String)
    - description (String)
    - price (double)
    - quantityInStock (int)
    - type (String) [Electronics/Clothing]




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

```
def get_product_id(self):
    return self.product_id

def set_product_id(self, product_id):
    self.product_id = product_id

def get_product_name(self):
    return self.product_name

def set_product_name(self, product_name):
    self.product_name = product_name

def get_description(self):
    return self.description

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

def get_price(self):
    return self.price

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

def get_quantity_in_stock(self):
    return self.quantity_in_stock

def set_quantity_in_stock(self, quantity_in_stock):
    self.quantity_in_stock = quantity_in_stock

def get_type(self):
    return self.type

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

3. Create a subclass Electronics that inherits from Product. Add attributes specific to electronics products, such as:
    - brand (String)
    - warrantyPeriod (int)

electronics.py

```python
from entity.product import Product

class Electronics(Product):
    def __init__(self, product_id, product_name, description, price, quantity_in_stock, typ
        super().__init__(product_id, product_name, description, price, quantity_in_stock, t
        self.brand = brand
        self.warranty_period = warranty_period

    def get_brand(self):
        return self.brand

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

    def get_warranty_period(self):
        return self.warranty_period

    def set_warranty_period(self, warranty_period):
        self.warranty_period = warranty_period
```

4. Create a subclass Clothing that also inherits from Product. Add attributes specific to clothing products, such as:
    - size (String)
    - color (String)

clothing.py

```python
from entity.product import Product

class Clothing(Product):
    def __init__(self, product_id, product_name, description, price, quantity_in_stock, typ
        super().__init__(product_id, product_name, description, price, quantity_in_stock, t
        self.size = size
        self.color = color

    def get_size(self):
        return self.size

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

    def get_color(self):
        return self.color

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

5. Create a User class with attributes:
    - userId (int)
    - username (String)
    - password (String)
    - role (String) // "Admin" or "User"

user.py

```python
class User:
    def __init__(self, user_id, username, password, role):
        self.user_id = user_id
        self.username = username
        self.password = password
        self.role = role

    def get_user_id(self):
        return self.user_id

    def set_user_id(self, user_id):
        self.user_id = user_id

    def get_username(self):
        return self.username

    def set_username(self, username):
        self.username = username

    def get_password(self):
        return self.password

    def set_password(self, password):
        self.password = password

    def get_role(self):
        return self.role

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

6. Define an interface/abstract class named IOrderManagementRepository with methods for:
    - createOrder(User user, list of products): check the user as already present in the database to create an order or create a user (store in the database) and create an order.
    - cancelOrder(int userId, int orderId): check the userid and orderId already present in the database and cancel the order. If any userId or orderId is not present in the database, throw an exception corresponding UserNotFound or OrderNotFound exception.
    - createProduct(User user, Product product): check the admin user as already present in the database and create a product and store it in the database.
    - createUser(User user): create a user and store it in the database for further development.
    - getAllProducts(): return all product lists from the database.
    - getOrderByUser(User user): return all products ordered by a specific user from the database.

order_management_repository.py

```python
from abc import ABC, abstractmethod

class IOrderManagementRepository(ABC):
    @abstractmethod
    def create_order(self, user, products):
        pass

    @abstractmethod
    def cancel_order(self, user_id, order_id):
        pass

    @abstractmethod
    def create_product(self, user, product):
        pass

    @abstractmethod
    def create_user(self, user):
        pass

    @abstractmethod
    def get_all_products(self):
        pass

    @abstractmethod
    def get_order_by_user(self, user):
        pass
```

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

```python
from dao.order_management_repository import IOrderManagementRepository
from exception.order_not_found_exception import OrderNotFoundException
from exception.user_not_found_exception import UserNotFoundException
import mysql.connector
from util.db_conn_util import get_db_conn

class OrderProcessor(IOrderManagementRepository):
    def __init__(self):
        self.connection = get_db_conn()

    def create_order(self, user_id, product_ids): ...

    def cancel_order(self, user_id, order_id): ...

    def create_product(self, product): ...

    def create_user(self, user): ...

    def get_all_products(self): ...

    def get_order_by_user(self, user_id): ...
```

8. Create DBUtil class and add the following method:
    - static getDBConn():Connection Establish a connection to the database and return the database Connection.

```python
import configparser

def get_db_connection_string(file_name):
    config = configparser.ConfigParser()
    config.read(file_name)

    connection_dict = {
        'user': config.get('mysql', 'user'),
        'password': config.get('mysql', 'password'),
        'host': config.get('mysql', 'host'),
        'port': config.getint('mysql', 'port'),
        'database': config.get('mysql', 'database')
    }

    return connection_dict
```

9. Create OrderManagement main class and perform the following operation:
    - main method to simulate the loan management system. Allow the user to interact with the system by entering choices from the menu such as "createUser", "createProduct", "cancelOrder", "getAllProducts", "getOrderByUser", "exit".

order_management_main.py

```python
from dao.order_processor import OrderProcessor
from entity.user import User
from entity.product import Product
from entity.electronics import Electronics
from entity.clothing import Clothing

def main():
    order_processor = OrderProcessor()

    while True:
        print("Choose an option:")
        print("1. Create User")
        print("2. Create Product")
        print("3. Create Order")
        print("4. Cancel Order")
        print("5. Get All Products")
        print("6. Get Order by User")
        print("7. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            user_id = int(input("Enter user ID: "))
            username = input("Enter username: ")
            password = input("Enter password: ")
            role = input("Enter role (Admin/User): ")
            user = User(user_id, username, password, role)
            order_processor.create_user(user)
        elif choice == "2":
            product_id = int(input("Enter product ID: "))
            product_name = input("Enter product name: ")
            description = input("Enter description: ")
            price = float(input("Enter price: "))
            quantity_in_stock = int(input("Enter quantity in stock: "))
            type = input("Enter type (Electronics/Clothing): ")
            if type == "Electronics":
                brand = input("Enter brand: ")
                warranty_period = int(input("Enter warranty period: "))
                product = Electronics(product_id, product_name, description, price, quantity_in_stock, type, brand, warranty_period)
            else:
                size = input("Enter size: ")
                color = input("Enter color: ")
                product = Clothing(product_id, product_name, description, price, quantity_in_stock, type, size, color)
            order_processor.create_product(product)
        elif choice == "3":
            user_id = int(input("Enter user ID: "))
            num_products = int(input("Enter number of products: "))
            products = []
            for i in range(num_products):
                product_id = int(input(f"Enter product ID {i + 1}: "))
                products.append(product_id)
            order_processor.create_order(user_id, products)
        elif choice == "4":
            user_id = int(input("Enter user ID: "))
            order_id = int(input("Enter order ID: "))
            order_processor.cancel_order(user_id, order_id)
        elif choice == "5":
            products = order_processor.get_all_products()
```

**CONSOLE OUTPUT:**

```
PS D:\User Files\Videos\order management system> python order_management_main.py
Choose an option:
1. Create User
2. Create Product
3. Create Order
4. Cancel Order
5. Get All Products
6. Get Order by User
7. Exit
Enter your choice: 1
Enter user ID: 102
Enter username: Praveen
Enter password: pravee123
Enter role (Admin/User): user
User 'Praveen' created successfully
```