Python and OOPs coding problems

1. You are a software developer at a streaming platform like Netflix or Amazon Prime. The company wants to implement a simple **Movie Recommendation System** using Python and Object-Oriented Programming (OOPs). Your task is to build a system that stores movie details, allows users to rate movies, and provides recommendations based on user preferences.

Requirements:

1. **Movie Class (`Movie`)**
    - Attributes: `movie_id`, `title`, `genre`, `rating`, `release_year`, `cast`
    - Methods:
        - `get_movie_info()`: Returns the details of the movie.
        - `update_rating(new_rating)`: Updates the movie's rating.
2. **User Class (`User`)**
    - Attributes: `user_id`, `username`, `watched_movies`, `preferred_genres`
    - Methods:
        - `watch_movie(movie)`: Adds a movie to the user's watched list.
        - `get_recommendations(movie_list)`: Recommends movies based on genre preferences.
3. **MovieManager Class (`MovieManager`)**
    - Attributes: A list of all available movies.
    - Methods:
        - `add_movie(movie)`: Adds a new movie to the collection.
        - `search_movie(title)`: Searches for a movie by title.
        - `get_top_rated_movies(n)`: Returns the top `n` highest-rated movies.
4. **Usage of Python Libraries:**
    - **pandas**: To store and manage the list of movies.
    - **random/numpy**: To generate random ratings or sample movies.
5. **I/O Operations:**
    - Every time a new entry for a movie or a user is received, the entry should be written to 2 separate json files one each for movies and users. Proper structure should be followed in the json file (nested structure with movie_id and user_id as keys)

2. You are a software developer working for an e-commerce company that wants to implement a **Smart Inventory Management System**. The system should keep track of products, manage stock levels, and generate alerts when stock is low. Additionally, it should allow for sales transactions and automatically update inventory.

Requirements:

1. **Product Class (`Product`)**
    - Attributes: `product_id`, `name`, `category`, `price`, `stock_quantity`
    - Methods:
        - `update_stock(quantity)`: Updates the stock level when products are sold or restocked.
        - `get_product_info()`: Returns product details.
2. **Inventory Class (`Inventory`)**
    - Attributes: A dictionary to store products (`product_id` as key and `Product` object as value).
    - Methods:
        - `add_product(product)`: Adds a new product to inventory.
        - `remove_product(product_id)`: Removes a product from inventory.
        - `get_low_stock_products(threshold)`: Returns a list of products with stock below the given threshold.
3. **Order Class (`Order`)**
    - Attributes: `order_id`, `customer_name`, `items` (a dictionary with `Product` and quantity), `total_price`
    - Methods:
        - `calculate_total()`: Calculates the total price of the order.
        - `process_order(inventory)`: Updates the inventory stock after an order is placed.
4. **Usage of Python Libraries:**
    - **pandas**: To store and manage inventory data.
    - **datetime**: To track order dates.
    - **logging**: To log stock updates and order processing.
5. **I/O Operations:**
    - Every time a new entry for a product, inventory or an order is received, the entry should be written to 3 separate json files one each for products, inventory and orders. Proper structure should be followed in the json file (nested structure with product_id, inventory and order_id as keys)