Java Manager Level Question: Advanced OOP, Collection, and File Handling

# Question:

You are tasked with designing an enhanced library management system with advanced Object-Oriented Programming (OOP) principles, efficient collection management, and robust file handling.

1. \*\*Advanced OOP Design\*\*:  
- Design a set of classes to model a complex library system, which includes the following:  
 - `Book`: This should include details such as `ISBN`, `title`, `author`, `publishedYear`, and `genre`. The `Book` class should also support the ability to track its availability status and allow for updating.  
 - `Library`: This class should manage the collection of books and users. It should include methods to add, remove, and update books, as well as a method for checking the availability of a specific book by ISBN.  
 - `User`: This class represents the library users and should track the books that the user has checked out. Implement appropriate methods for checking out and returning books, ensuring that a user cannot check out more than a set number of books at once.  
- Use the \*\*Composite pattern\*\* to handle complex objects such as `Book` and `Library` and treat them uniformly.  
- Implement \*\*polymorphism\*\* to allow the system to handle different types of books (such as `PhysicalBook`, `Ebook`, `AudioBook`) that have unique behaviors for certain operations, such as printing or displaying information.

2. \*\*Collection Framework\*\*:  
- Use appropriate Java collections to efficiently manage books, users, and transactions. Specifically:  
 - Store the books in a \*\*`TreeMap`\*\* to maintain a sorted order by ISBN. Discuss why a `TreeMap` is appropriate for this use case and how it will benefit the search and retrieval process.  
 - Implement a method that allows users to search for books by multiple criteria (title, author, genre, etc.). Use a \*\*`HashSet`\*\* to store unique search results.  
 - Implement a \*\*`Queue`\*\* to manage the users who are waiting for a book that is currently unavailable. Ensure that the queue is processed efficiently and follows a first-come-first-serve model.

3. \*\*File Handling and Persistence\*\*:  
- The system needs to save and load its data to/from a file. Implement methods for reading and writing the library data in a \*\*serialized\*\* format (e.g., using Java’s `ObjectOutputStream` and `ObjectInputStream`).  
 - Serialize the entire `Library` class, including its collection of books and users, to a file. Discuss the trade-offs between using serialization and other file formats such as CSV or JSON.  
 - Ensure that the system can handle reading and writing the library data asynchronously. Discuss how you would implement asynchronous file operations using Java's `ExecutorService` to avoid blocking the main application thread during file I/O operations.