Java Coding Final Exam Question

Problem Statement:

You are tasked with creating a mini-library management system. The system needs to manage a collection of books, categorized by genre. The following requirements must be implemented:

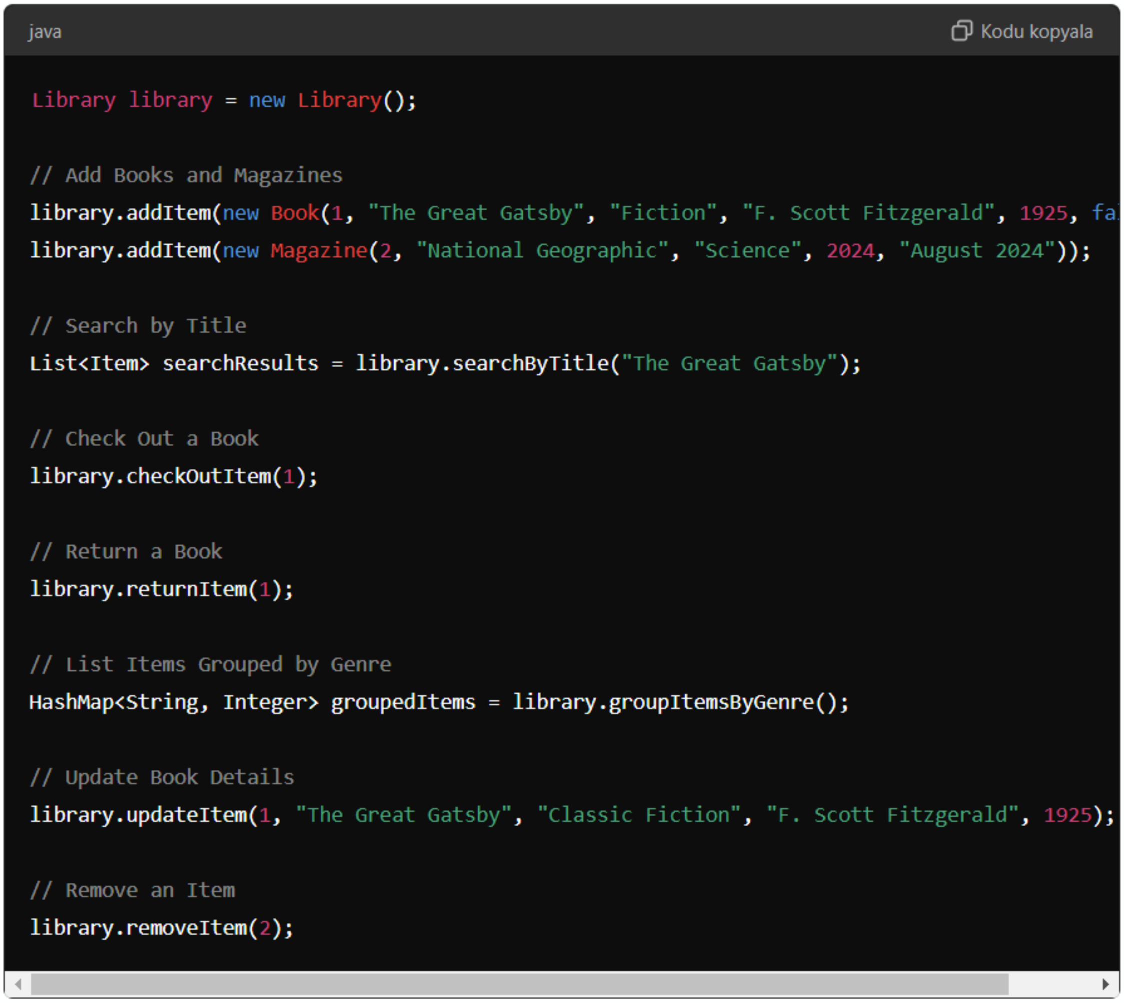
1. **Abstract Class and Inheritance**:
    - Create an abstract class `` `Item` `` with fields `` `id` ``, `` `title` ``, and `` `genre` ``. Include appropriate constructors, getters, and setters.
    - Create a class `` `Book` `` that inherits from `` `Item` ``. Add fields `` `author` ``, `` `publishedYear` ``, and `` `isCheckedOut` ``. Include appropriate constructors, getters, and setters.
    - Create a class `` `Magazine` `` that also inherits from `` `Item` ``. Add fields `` `issueNumber` `` and `` `releaseDate` ``. Include appropriate constructors, getters, and setters.
2. **Encapsulation**:
    - Ensure all fields in your classes are private and provide public methods to access and modify them as needed.
3. **Polymorphism**:
    - Create a method `` `displayDetails` `` in the abstract `` `Item` `` class, which is overridden in both `` `Book` `` and `` `Magazine` `` to display their respective details.
    - Create a class `` `Library` `` that manages a collection of `` `Item` `` objects using an `` `ArrayList` ``. This class should have methods to add and remove `` `Item` `` objects from the collection, using polymorphism to handle both `` `Book` `` and `` `Magazine` ``.
4. **HashMap and Nested Data Structures**:
    - The `` `Library` `` class should also maintain a `` `HashMap<String, ArrayList<Item>>` `` where the key is the genre and the value is a list of `` `Item` `` objects belonging to that genre. Implement methods to:
        - **Add** a new `` `Item` `` to the correct genre in the `` `HashMap` ``.
        - **Remove** an `` `Item` `` from a genre.
        - **Group** items by genre and return a `` `HashMap<String, Integer>` `` where the key is the genre and the value is the number of items in that genre.
5. **CRUD Operations**:
    - Implement methods in the `` `Library` `` class to:
        - **Update** the details of an existing `` `Item` `` (title, author, etc.).
        - **Search** for `` `Item` ``s by title, returning all matching items.
        - **Check Out** a `` `Book` `` using its `` `id` ``. This should set `` `isCheckedOut` `` to `` `true` ``.
        - **Return** a `` `Book` `` using its `` `id` ``, setting `` `isCheckedOut` `` to `` `false` ``.
        - **List** all `` `Item` ``s in the library, grouped by genre.
        - **Search by Genre**: Return all items within a specific genre.

6. **Custom Exception Handling**:
    - Create a custom exception `ItemNotFoundException` that is thrown when attempting to check out, return, update, or remove an `Item` that does not exist.
    - Handle this exception appropriately in your `Library` class, ensuring that meaningful error messages are displayed.

Bonus Task:

- Implement a method in the `Library` class to **retrieve all books published by a specific author**, grouped by the year they were published. This should return a `HashMap<Integer, ArrayList<Book>>` where the key is the year and the value is a list of `Book`s published that year by the given author.

Example Usage:



```java
Library library = new Library();

// Add Books and Magazines
library.addItem(new Book(1, "The Great Gatsby", "Fiction", "F. Scott Fitzgerald", 1925, fa
library.addItem(new Magazine(2, "National Geographic", "Science", 2024, "August 2024"));

// Search by Title
List<Item> searchResults = library.searchByTitle("The Great Gatsby");

// Check Out a Book
library.checkOutItem(1);

// Return a Book
library.returnItem(1);

// List Items Grouped by Genre
HashMap<String, Integer> groupedItems = library.groupItemsByGenre();

// Update Book Details
library.updateItem(1, "The Great Gatsby", "Classic Fiction", "F. Scott Fitzgerald", 1925);

// Remove an Item
library.removeItem(2);
```