Library Inventory System Design and Implementation

You are tasked with designing and implementing a Library Inventory System that manages the storage and checkout of items in a library. The system must adhere to the following specifications and constraints:

Storage System

1. Structure:
- The library storage is modeled using **shelves**, where:
  - Each shelf contains up to 15 **compartments**.
  - Each compartment can hold a single item.

2. Access:
- Implement the system so that it supports the **[] operator** for accessing items.
- For example, if the library storage is represented by a variable `libraryInventory`, you should be able to access the fifth compartment on the third shelf using: `libraryInventory[2][4]`.

Items in the Library

1. General Structure:
- All items should have the following properties:
  - **name**: Name of the item.
  - **description**: Description of the item.
  - **id**: Unique identifier for the item.

2. Specific Item Types:
- <u>Books:</u>
  - **Additional properties:** title, author, and copyrightDate.
- <u>Movies:</u>
  - **Additional properties:** title, director, and a collection of mainActors.
- <u>Magazines:</u>
  - **Additional properties:** edition and the title of the main article.

3. Operator Overloading:
- Overload the **<< operator** to display the details of each type of item.

System Operations

1. Add an Item:
- Add an item to the storage system at a specific shelf and compartment location.

2. Checkout an Item:
- Allow a person to check out an item from a specific compartment.
  - When checked out:
    - Record the name of the person who checked out the item.

- Record the due date for returning the item.

3. Checkin an Item:
- Allow a person to return an item to its original compartment.

4. Print Items in Storage:
- Print all items that are currently checked into storage, displaying their shelf and compartment locations.

5. Print Checked-Out Items:
- Print all items that are currently checked out, along with:
  - The name of the person who checked them out.
  - Their due date.

6. Swap Items:
- Swap the contents of two compartments in the library storage.
- Ensure there is an item in both compartments before performing the swap.

Requirements

1. Use classes to model the storage system, items, and operations. Organize your code in a clean and modular fashion.
2. Ensure proper handling of edge cases, such as:
- Attempting to access compartments or shelves that do not exist.
- Attempting to check out, check in, or swap items when a compartment is empty.
3. Demonstrate object-oriented programming principles, including:
- Encapsulation.
- Inheritance (for the item types).
- Polymorphism (where applicable).
4. Include appropriate input validation and error messages for invalid operations.

What You Must Submit

1. Class Definitions:
- Define all necessary classes (e.g., LibraryStorage, Item, Book, Movie, Magazine, etc.).

2. Implementation:
- Implement the functionality described above, including operator overloads.

3. Testing Code:
- Write a main function to test your library system:
  - Add items to storage.
  - Check out and check in items.
  - Print the current state of the storage and checked-out items.
  - Perform a swap operation and demonstrate error handling.

Evaluation Criteria

- Correctness:
  - Does the system correctly implement the described functionality?
- Code Quality:
  - Is the code well-organized, readable, and adherent to OOP principles?
- Testing:
  - Does the testing code cover all edge cases and demonstrate the functionality of the system?