A library management system using **stack** and **queue** can be implemented with the following structure:

Features:

1. **Stack**: Used to track borrowed books (Last-In-First-Out order).
2. **Queue**: Used to manage the list of books available in the library (First-In-First-Out order).

```
class Book:
    """A class to represent a book in the library."""
    def __init__(self, book_id, title, author):
        self.book_id = book_id
        self.title = title
        self.author = author


class Library:
    """A library management system using stack and queue."""
    def __init__(self):
        self.available_books = []  # Queue to manage available books
        self.borrowed_books = []   # Stack to track borrowed books

    def add_book(self, book_id, title, author):
        """Adds a book to the library (enqueue)."""
        new_book = Book(book_id, title, author)
        self.available_books.append(new_book)  # Add to the end of the queue
        print(f"Book '{title}' added to the library.")

    def remove_book(self, book_id):
        """Removes a book from the library."""
        for i, book in enumerate(self.available_books):
            if book.book_id == book_id:
```

```
            removed_book = self.available_books.pop(i)  # Remove from the queue
            print(f"Book '{removed_book.title}' removed from the library.")
            return
    print("Book not found in the library.")

def display_books(self):
    """Displays all available books in the library."""
    if not self.available_books:
        print("No books available in the library.")
        return
    print("Available books:")
    for book in self.available_books:
        print(f"ID: {book.book_id}, Title: {book.title}, Author: {book.author}")

def borrow_book(self, book_id):
    """Borrows a book from the library."""
    for i, book in enumerate(self.available_books):
        if book.book_id == book_id:
            borrowed_book = self.available_books.pop(i)  # Dequeue
            self.borrowed_books.append(borrowed_book)    # Push onto the stack
            print(f"Book '{borrowed_book.title}' borrowed.")
            return
    print("Book not available.")

def return_book(self):
    """Returns the most recently borrowed book (LIFO)."""
    if not self.borrowed_books:
        print("No borrowed books to return.")
```

```
            return
        returned_book = self.borrowed_books.pop()  # Pop from the stack
        self.available_books.append(returned_book)  # Enqueue back
        print(f"Book '{returned_book.title}' returned to the library.")

# Example usage:
library = Library()
library.add_book(1, "The Great Gatsby", "F. Scott Fitzgerald")
library.add_book(2, "1984", "George Orwell")
library.add_book(3, "To Kill a Mockingbird", "Harper Lee")

library.display_books()

library.borrow_book(2)  # Borrow "1984"
library.borrow_book(3)  # Borrow "To Kill a Mockingbird"

library.display_books()

library.return_book()  # Return the most recently borrowed book
library.display_books()

library.remove_book(1)  # Remove "The Great Gatsby"
library.display_books()
```

Explanation:

1. **Queue for Available Books**:
    - Managed using a list where books are appended at the end (enqueue) and removed from the front or a specific index.
2. **Stack for Borrowed Books**:
    - Managed using a list where books are appended at the end (push) and removed from the end (pop).
3. **Methods**:
    - add_book: Enqueues a book to the available books.
    - remove_book: Removes a book from the available books queue.
    - borrow_book: Moves a book from the queue to the stack (borrow).
    - return_book: Moves a book from the stack back to the queue (return).