M10 Classwork Mon

Objective: In this **pair programming activity**, you will create a Python module called `library.py` that simulates a library management system. The module will contain classes for books, library members, and library operations. You will then use this module in a separate Python script called `main.py` to perform various library operations.

Module Setup:

1. **Create a Directory:** Begin by creating a new directory or folder for this activity on your computer.
2. **Create Module and Script:** Inside the directory, create two Python files:
   - `library.py`: This file will serve as your module and will contain the classes and functions related to the library.
   - `main.py`: This file will be your main script where you will import and use the `library` module.

Task 1: Create Library Classes (in `library.py`):

In the `library.py` file, define the following classes:

a. `Book` Class:

- Create a class named `Book` with the following attributes:
  - `title` (str): The title of the book.
  - `author` (str): The author of the book.
  - `isbn` (str): The ISBN number of the book.
  - `checked_out` (bool): A flag indicating if the book is checked out (initially set to `False`).
- Implement a constructor (`__init__`) method that initializes the book's attributes.
- Implement methods to:

  - Check out the book (`check_out`): Set the `checked_out` flag to `True`.
  - Check in the book (`check_in`): Set the `checked_out` flag to `False`.

Example Run for `Book` Class:

```
# Create a book instance
book1 = Book("Python Programming", "John Smith", "978-1234567890")

# Check out the book
book1.check_out()

# Check the book's status
print(book1.checked_out)  # Expected output: True

# Check in the book
book1.check_in()

# Check the book's status
print(book1.checked_out)  # Expected output: False
```

b. `Member` Class:

- Create a class named `Member` with the following attributes:
  - `member_id` (str): The unique ID of the library member.
  - `name` (str): The name of the library member.
  - `checked_out_books` (list): A list to store books checked out by the member (initially empty).
- Implement a constructor (`__init__`) method that initializes the member's attributes.
- Implement methods to:
  - Check out a book (`check_out_book`): Check out a book and add it to the member's list of checked-out books.
  - Return a book (`return_book`): Return a book and remove it from the member's list of checked-out books.

Example Run for `Member` Class:

```
# Create a member instance
member1 = Member("M001", "Alice")
```

```python
# Check out a book
member1.check_out_book(book1)

# Check the member's checked-out books
print(member1.checked_out_books)  # Expected output: [book1]

# Return a book
member1.return_book(book1)

# Check the member's checked-out books
print(member1.checked_out_books)  # Expected output: []
```

c. `Library` Class:

- Create a class named `Library` to represent the library itself.
- Implement a constructor (`__init__`) method that initializes an empty list to store books (`self.books`).
- Implement methods to:
  - Add a book to the library (`add_book`): Add a book object to the `self.books` list.
  - Search for books based on a search query (e.g., title) (`search_book`): Return a list of books that match the query.
  - Display all books in the library (`display_books`): Print information about all books in the library.

Example Run for `Library` Class:

```python
# Create a second book instance
book2 = Book("Data Science Handbook", "Alice Johnson", "978-9876543210")

# Create a library instance
library = Library()

# Add books to the library
library.add_book(book1)
library.add_book(book2)

# Search for books by title
found_books = library.search_book("Python")
print(found_books)  # Expected output: [book1]

# Display all books in the library
library.display_books()
# Expected output:
```

```
# Python Programming by John Smith (ISBN: 978-1234567890)
# Data Science Handbook by Alice Johnson (ISBN: 978-9876543210)
```

d. `Transaction` Class:

- Create a class named `Transaction`. This class will be used to record library transactions and does not have a constructor.
- Implement a **class-level** variable `transactions` (a list) to store transaction records.
- Implement a **class method** `record_transaction` that records a transaction (use @classmethod). The method should take parameters for the member, book, and action (e.g., "checked out" or "returned") and add a transaction record (a string in the form of member name action book detail e.g. *Alice checked out 'Python Programming' (ISBN: 978-1234567890)*) to the `transactions` list.

Example Run for `Transaction` Class:

```
member2 = Member("M002", "Bob")

# Record transactions
Transaction.record_transaction(member1, book1, "checked out")
Transaction.record_transaction(member2, book2, "checked out")
Transaction.record_transaction(member1, book1, "returned")

# Display transaction history
for transaction in Transaction.transactions:
    print(transaction)
# Expected output:
# Alice checked out 'Python Programming' (ISBN: 978-1234567890)
# Bob checked out 'Data Science Handbook' (ISBN: 978-9876543210)
# Alice returned 'Python Programming' (ISBN: 978-1234567890)
```

In the `library.py` file, write functions to perform the following tasks:

- Function to add books to the library.
- Function to search for books based on a search query (e.g., title).
- Function to display all books in the library.

Task 3: Use the Module (in `main.py`):

In the `main.py` file, follow these steps:

- Import the necessary classes and functions from the `library` module.
- Create instances of the classes and perform library operations:
  - Create instances of books, library members
  - Simulate library operations like checking out and returning books.
  - Search for books based on a query and display results.
  - Record transactions using the `Transaction` class method.
- You can test your code using the following example:

```
# Create library instance
library = Library()

# Create books
book1 = Book("Python Programming", "John Smith", "978-1234567890")
book2 = Book("Data Science Handbook", "Alice Johnson", "978-9876543210")
book3 = Book("Web Development Basics", "Bob Brown", "978-1111111111")

# Add books to the library
library.add_book(book1)
library.add_book(book2)
library.add_book(book3)

# Create library members
member1 = Member("M001", "Alice")
member2 = Member("M002", "Bob")

# Member checks out books
member1.check_out_book(book1)
member2.check_out_book(book2)

# Member returns books
member1.return_book(book1)
member2.return_book(book2)

# Search for books
found_books = library.search_book("Python")
print("Books matching 'Python':")
library.display_books()
```

```
# Record transactions
Transaction.record_transaction(member1, book1, "checked out")
Transaction.record_transaction(member2, book2, "checked out")
Transaction.record_transaction(member1, book1, "returned")

# Display transactions
print("\nTransaction History:")
for transaction in Transaction.transactions:
    print(transaction)
```