# LW: Object Oriented Programming

## Objectives

* Familiarize yourself with the principles of Object-Oriented Programming.
* Develop classes and objects to model real-world entities.

## Completion

* Attend lab or have an excused absence
* Successfully implement and test all the methods in the given classes.
* Ensure your code compiles without any errors or warnings.

## Submission

Submit these files to the autograder

* main.cpp
* library.cpp
* library.h
* book.cpp
* book.h

## Overview

In this lab, you will implement the concepts of Object-Oriented Programming (OOP), a programming paradigm centered on the concept of "objects" which can contain data and code to manipulate that data. Through this paradigm, you'll model real-world entities and their interactions, offering a structured approach to complex problems. The entities in focus for this lab are books and a library system. For the purposes of this lab, each book has the attributes of title, author, and ISBN. The Library, on the other hand, serves as a collection of books, allowing operations like adding a book to its inventory, borrowing a book based on its ISBN, returning a borrowed book, and searching for a book by its title. By implementing this system, you'll gain experience in designing classes, creating objects, and defining their interactions—core concepts of OOP. This lab will challenge you to think in terms of objects and their behaviors, reinforcing the principles of encapsulation, abstraction, and interaction.

## What to do

1. Get the starter code.
2. Design and implement the classes
   1. Book Class
      1. Attributes: `title`, `author`, `ISBN`, `isBorrowed`.
      2. Methods:
         1. borrowBook(): Change the status of a book to borrowed.
         2. returnBook(): Change the status of a book to available.
         3. Necessary getters and setters.
   2. Library Class
      1. Attributes: `books` (a collection of `Book` objects).
      2. Methods:
         1. addBook(Book book): Integrate a book into the library's collection.
         2. borrowBook(string ISBN): Facilitates the borrowing of a book using its ISBN.
         3. returnBook(string ISBN): Manage the return of a book using its ISBN.
         4. searchBook(string title): Locate a book by its title within the library.
3. Test your program and fix any errors you may encounter

## Optional Tasks

* Design a User class to monitor borrowed books for each user.
* Enhance the search function to include filters like author, publication year, etc.
* Incorporate error management for scenarios such as attempting to borrow a book that's already out.

## Additional Guidelines

* Understand the Problem
  + Before diving into the code, make sure you understand the requirements and the roles of the `Book` and `Library` classes.
  + Draw a diagram or outline the relationships between classes if necessary.
* Start Simple
  + Begin by implementing the basic methods of the `Book` class, as it is the foundational unit.
  + Test each method after implementation to ensure correctness.
* Avoid Hardcoding
  + Ensure that methods are flexible and not hardcoded for specific test cases.
  + Your methods should work for various inputs, not just the ones provided in the test cases.
* Consistent Error Handling
  + When implementing methods like `borrowBook` or `returnBook`, decide on a consistent way to handle errors
    - e.g. when trying to borrow a book that's already borrowed
  + Whether you choose to return a specific value, print an error message, or handle it some other way, be consistent throughout your code.

## Test Cases

* Book Class:
  + Constructor and Getters:
    - Input: Create a book with title "1984", author "George Orwell", and ISBN "9780451524935".
    - Expected Output: The getters for title, author, and ISBN should return "1984", "George Orwell", and "9780451524935" respectively.
  + borrowBook() and returnBook():
    - Input: Borrow a newly created book.
    - Expected Output: getIsBorrowed() should return true.
    - Input: Return the borrowed book.
    - Expected Output: getIsBorrowed() should return false.
* Library Class:
  + addBook():
    - Input: Add a book to the library.
    - Expected Output: searchBook() with that book's title should return a valid book object.
  + borrowBook() and returnBook():
    - Input: Borrow a book using its ISBN.
    - Expected Output: The book's getIsBorrowed() method should return true.
    - Input: Return the borrowed book using its ISBN.
    - Expected Output: The book's getIsBorrowed() method should return false.
  + searchBook():
    - Input: Search for a book that exists in the library.
    - Expected Output: Should return a valid book object.
    - Input: Search for a book that does not exist.
    - Expected Output: Should return nullptr or a similar indication that the book isn't present.
* Edge Cases:
  + Adding Duplicate Books:
    - Input: Add a book with the same title and ISBN more than once.
    - Expected Output: Depending on the design, the library could prevent adding duplicates or allow multiple copies. The test will vary based on this design decision.
  + Borrowing a Non-existent Book:
    - Input: Try to borrow a book using an ISBN not present in the library.
    - Expected Output: The method should handle this gracefully, possibly returning a specific value (like false) or printing an error message.
  + Returning a Book That Wasn't Borrowed:
    - Input: Try to return a book that wasn't borrowed.
    - Expected Output: The method should recognize this and return a specific value or message indicating the book wasn't borrowed.
  + Library with No Books:
    - Input: Try to search or borrow from an empty library.
    - Expected Output: The methods should handle an empty library state gracefully, possibly indicating no books are available.