In Partial Fulfillment of the Requirements for the

CS 223 - Object-Oriented Programming

**Library Management System**

Presented to:

Dr. Unife O. Cagas

**Professor**

Prepared by:

Casera, Jericho C.   
**Student**

BSCS-2A: Computer Science

May. 2024

**PROJECT DESCRIPTION**

Many people like books or borrow books from a library due to these circumstances I created an online library management system to easily management the library and also the see if the books they want to borrow is available or not.

**Objectives:**

* To show the principles of object-oriented programming.
* To show if the books are available or not.
* To demonstrate how the principles of OOP are used in the system.
* To create a user-friendly interface for library management system.

**Importance and Contribution:**

This project is intended for readers or library patrons who enjoy reading. Because of how easily managed it is, the library is greatly impacted. By creating this library management system, librarians and users can adapt to the technological age and find their jobs easier.

**Four Principles of Object-Oriented PROGRAMMING**

* **Abstract** - A class that is not directly instantiable is called an abstract class. It acts as a model for other classes. Providing a common interface and some common functionality that may be shared by several derived classes is the main goal of an abstract class. Concrete methods (those with implementation) and abstract methods (those without implementation) can both be found in abstract classes.
* **Encapsulation** - Encapsulation is a core concept in OOP that promotes the bundling of data and methods while restricting direct access to some components. This leads to improved security, controlled access, better modularity, easier maintenance, and greater reusability of code. Encapsulation is achieved using access modifiers and by providing public methods to access and modify the private data.
* **Inheritance** - Inheritance is another fundamental concept in object-oriented programming (OOP). It allows a new class, called a subclass (or derived class), to inherit attributes and methods from an existing class, called a superclass (or base class). This promotes code reusability and establishes a natural hierarchy between classes.
* **Polymorphism** - Polymorphism is one of the core principles of object-oriented programming (OOP) that allows objects of different classes to be treated as objects of a common superclass. It enables a single interface to be used for a general class of actions, with the specific action determined by the exact nature of the situation.

**Hardware & Software Used:**

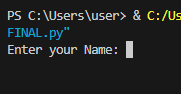
**Hardware:**

* Laptop

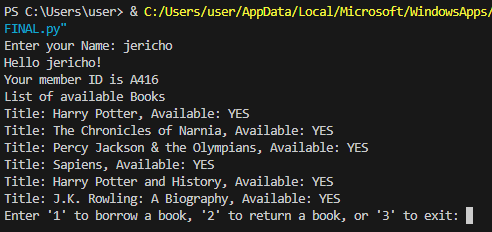
**Software:**

* Python 3.11
* Vscode

**Output with Description:**

****

First the user needs to enter their name



After the user enters their name the system automatically generates the member ID and shows all books that the library has then the user needs to enter 1 if they, would like to borrow a book, 2 if the user has already borrowed books they to return the book and finally 3 if you want to exit the program.

**Code Documentation:**

import random

from abc import ABC, abstractmethod

class Book(ABC): #base class for books

def \_\_init\_\_(self, title, author, publication\_year, genre):

self.\_\_title = title

self.\_\_author = author

self.\_\_publication\_year = publication\_year

self.\_\_genre = genre

self.\_\_available = True

@abstractmethod

def borrow(self): #abstract menthod for borrow

if self.\_\_available:

self.\_\_available = False

return True

else:

print("This book is not available.")

return False

@abstractmethod

def return\_book(self): #abstract menthod for return

self.\_\_available = True

def get\_title(self):#method for get a book

return self.\_\_title

def is\_available(self):#method if the books is available

return self.\_\_available

class FictionBook(Book):#representing fiction books and inherets from Book class

def \_\_init\_\_(self, title, author, publication\_year, genre):# Constructor method

super().\_\_init\_\_(title, author, publication\_year, genre)# Calling parent class constructor

def borrow(self):# Method to borrow a fiction book

return super().borrow()

def return\_book(self):# Method to return a fiction book

super().return\_book()

class NonFictionBook(Book):#representing non-fiction books and inherets from Book class

def \_\_init\_\_(self, title, author, publication\_year, genre):# Constructor method

super().\_\_init\_\_(title, author, publication\_year, genre)# Calling parent class constructor

def borrow(self):# Method to borrow a non-fiction book

return super().borrow()

def return\_book(self):# Method to return a non-fiction book

super().return\_book()

class LibraryMember: #class for library member

def \_\_init\_\_(self, name, member\_id):

self.\_\_name = name

self.\_\_member\_id = member\_id

self.\_\_borrowed\_books = []

def borrow(self, book):# Method to borrow a book by a member

if book.borrow():

self.\_\_borrowed\_books.append(book)

print(f"{self.\_\_name} has borrowed '{book.get\_title()}'")

else:

print(f"Sorry, '{book.get\_title()}' is not available.")

def return\_book(self, book\_title):#method to return a book by a member

for book in self.\_\_borrowed\_books:

if book.get\_title().lower() == book\_title.lower():

book.return\_book()

self.\_\_borrowed\_books.remove(book)

print(f"{self.\_\_name} has successfully returned '{book.get\_title()}'.")

return

print(f"Sorry, '{self.\_\_name}' has not borrowed '{book\_title}' or it is not a valid book title.")

class Library:

def \_\_init\_\_(self):

self.\_\_books = []#list of books in library

def add\_book(self, book):#method for adding books in a library

self.\_\_books.append(book)

def display\_books(self):

for book in self.\_\_books:#method if the book is available or not

if book.is\_available():

print(f"Title: {book.get\_title()}, Available: YES")

else:

print(f"Title: {book.get\_title()}, Available: NO")

def get\_book\_by\_title(self, title):#get method for borrwing or returning books

for book in self.\_\_books:

if book.get\_title().lower() == title.lower() and book.is\_available():

return book

return None

library = Library()#creating library object

fiction\_book = FictionBook("Harry Potter", "J.K. Rowling", 1997, "Fantasy")

fiction\_book1 = FictionBook("The Chronicles of Narnia", "C.S. Lewis,", 1950, "Fantasy")

fiction\_book2 = FictionBook("Percy Jackson & the Olympians", "Rick Riordan", 2005, "Fantasy")

nonfiction\_book = NonFictionBook("Sapiens", "Yuval Noah Harari", 2011, "History")

nonfiction\_book1 = NonFictionBook("Harry Potter and History", " Nancy R. Reagin", 2011, "History")

nonfiction\_book2 = NonFictionBook("J.K. Rowling: A Biography", "Sean Smith", 2002, "Biography")

library.add\_book(fiction\_book)#adding the fiction book for library

library.add\_book(fiction\_book1)

library.add\_book(fiction\_book2)

library.add\_book(nonfiction\_book)#addinig the non-fiction book for library

library.add\_book(nonfiction\_book1)

library.add\_book(nonfiction\_book2)

user\_name = input("Enter your Name: ")#getting user's name

ID\_no = "A" + str(random.randint(100, 999))#creating a user's ID

new\_user = LibraryMember(user\_name, ID\_no)# Creating a LibraryMember object

print(f"Hello {user\_name}! \nYour member ID is {ID\_no}")# Printing user's name and member ID

print("List of available Books")#printing list of all books in the library

library.display\_books()

while True:#loop for user iteration

action = input("Enter '1' to borrow a book, '2' to return a book, or '3' to exit: ")#prompt for user action

if action == "1":#if user wants to borrow book

select\_book\_title = input("Enter the title of the book you want to borrow: ")

selected\_book = library.get\_book\_by\_title(select\_book\_title)

if selected\_book:

new\_user.borrow(selected\_book)#borrowing the book

else:

print("The book you selected is not available or does not exist.")

elif action == "2":

if new\_user.\_LibraryMember\_\_borrowed\_books: # Accessing private attribute

return\_books\_title = input("Enter the title of the book you want to return: ")

new\_user.return\_book(return\_books\_title) # Modified to pass book title

else:

print("You have not borrowed any books.")

elif action == "3":# if the user wants to exit

print("Exiting...")

break

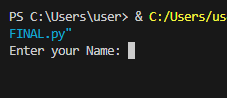
else:

print("Invalid input! Please try again.")

**User Guide:**

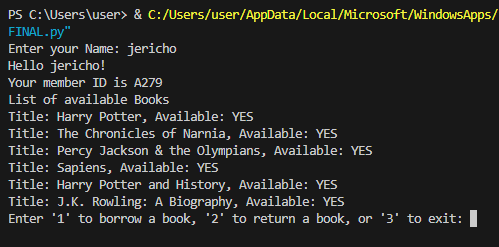
**Step 1:**

* Start the program by running the Python script or executable file (Vscode).



**Step 2:**

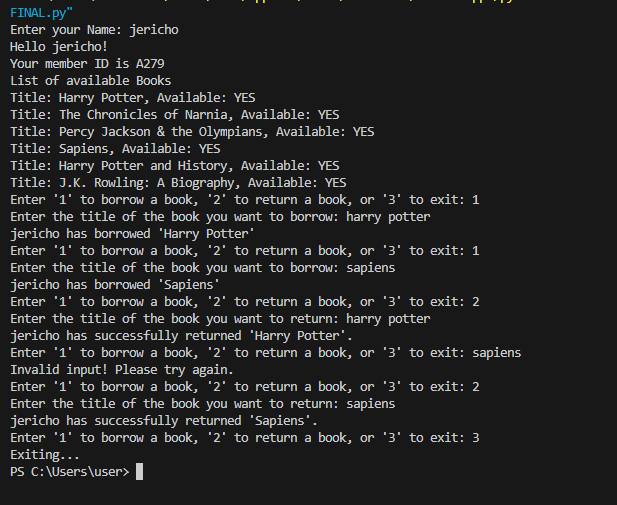
* Input your desired name.
* After choice “1” if you want to borrow books, “2” if you want to return a book, and “3” if you want to exit the program.



**Step 3:**

* I enter 1 to borrow the books Harry Potter and Sapiens.
* After I borrowed the I entered 2 to return the books.
* After I am satisfied with the interaction I entered 3 to exit the program.

**Output:**



**Description:**

This program analyzes student preferences in mobile devices, presenting the results through a user-friendly GUI and generating insightful reports for business and also for education. Program concepts such as object-oriented design, graphical interface development, data analysis, and error handling is a valuable tool for understanding and interpreting survey data related to mobile technology preferences among students. The necessary libraries such as tkinter for GUI development, pandas for data manipulation, matplotlib for data visualization, filedialog for file handling, and PIL for image processing.

**Reference:**

Code assistant [ChatGPT](https://chatgpt.com/?oai-dm=1)

Grammar corrector [Paraphrasing Tool - QuillBot AI](https://quillbot.com/)

GUI guide <https://www.pythontutorial.net/tkinter/>

Abstract Sources <https://www.geeksforgeeks.org/abstract-classes-in-python/>