รายงาน

Library by OOP

เสนอ

อาจารย์ศักดิ์ระพี ไพศาลนันทน์

จัดทำโดย

นาย ธีรเดช สุขเกษม 630910864

นาย สุทธิพงษ์ เลิศปัญญาวุฒิกุล 630910880

นางสาว อณัศยา พวงทรัพย์ 630910881

รายงานนี้เป็นส่วนหนึ่งของวิชา OBJECT-ORIENTED SYSTEM DESIGN FOR ENGINEERS รหัสวิชา 618445 มหาวิทยาลัยศิลปากร ภาคเรียนที่2 ปีการศึกษา2566

คำนำ

รายงานเล่มนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของวิชา OBJECT-ORIENTED SYSTEM DESIGN FOR ENGINEERS โดยมีเนื้อหาเกี่ยวกับการสร้างโปรแกรมเดสก์ทอป Library จาก OOP เพื่อให้ผู้ที่ต้องการศึกษาหาความรู้ในเรื่องการเขียนโค้ด ได้ศึกษาอย่างเข้าใจเพื่อเป็นประโยชน์กับการเรียน

ผู้จัดทำหวังว่ารายงานนี้จะมีข้อมูลที่เป็นประโยชน์กับผู้อ่านหรือนักเรียน นักศึกษาที่กำลังหาข้อมูลเรื่องนี้อยู่หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้ด้วย

คณะผู้จัดทำ
นาย ธีรเดช สุขเกษม 630910864
นาย สุทธิพงษ์ เลิศปัญญาวุฒิกุล 630910880
นางสาว อณัศยา พวงทรัพย์ 630910881

สารบัญ

App.py

เป็นโค้ด Python ที่ใช้สร้างหน้าต่างแอปพลิเคชัน PyQt6 โดยใช้คลาส MainWindow ที่สืบทอดมาจาก QWidget ซึ่งมีการกำหนดคุณสมบัติต่าง ๆ ของหน้าต่างและปุ่มต่าง ๆ ที่ใช้ในการจัดการกับข้อมูลในหน้าต่างนี้

Import Libraries

```
from PyQt6.QtCore import QSize, Qt, QRectF
from PyQt6.QtGui import QImage, QPalette, QBrush , QPainter , QColor ,QFont
from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QHBoxLayout, QVBoxLayout, QLabel, QInputDialog , QLineEdit , QDialog ,QMessageBox ,QComboBox
from library_system import *
from datetime import datetime, timedelta
```

- from PyQt6.QtCore import QSize, Qt, QRectF: นำเข้า classes และ constants ที่เกี่ยวข้องกับการทำงานของ PyQt6 ในส่วนของการจัดการขนาด (QSize), ค่าคงที่ที่เกี่ยวข้องกับการจัดตำแหน่งขององค์ประกอบ (Qt), และการจัดการตำแหน่งของสี่เหลี่ยมผืนผ้า (QRectF) เพื่อใช้ในการเขียนโปรแกรมต่อไป
- from PyQt6.QtGui import QImage, QPalette, QBrush, QPainter, QColor, QFont: นำเข้า classes ที่เกี่ยวข้องกับการสร้างและจัดการกับองค์ประกอบกราฟิกต่างๆ เช่น ภาพ (QImage), พาเล็ต (QPalette), แปรง (QBrush), ตัวเขียน (QPainter), สี (QColor), และแบบอักษร (QFont) เพื่อใช้ในการสร้างและกำหนดลักษณะต่างๆ ขององค์ประกอบใน GUI
- from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QHBoxLayout, QVBoxLayout, QLabel, QInputDialog, QLineEdit, QDialog, QMessageBox, QComboBox: นำเข้า classes และ widgets ที่ใช้ในการสร้าง GUI โดยสำคัญ เช่น แอปพลิเคชัน (QApplication), หน้าต่าง (QWidget), ปุ่ม (QPushButton), การจัดวางแนวนอน (QHBoxLayout), การจัดวางแนวตั้ง (QVBoxLayout), ป้ายกำกับ (QLabel), การนำเข้าข้อมูล (QInputDialog), ช่องใส่ข้อความ (QLineEdit), หน้าต่างโต้ตอบ (QDialog), กล่องข้อความแจ้งเตือน (QMessageBox), และกล่องควบคุมเลือก (QComboBox) เพื่อใช้ในการสร้างและจัดการกับส่วนต่างๆ ของ GUI
- from library_system import *: นำเข้าโมดูลหรือไลบรารีชื่อ "library_system" ซึ่งอาจจะมี classes หรือฟังก์ชันที่เกี่ยวข้องกับระบบห้องสมุด (library system) เพื่อใช้ในการดำเนินการต่างๆ ในระบบห้องสมุด เช่น การจัดการข้อมูลหนังสือ การยืม-คืนหนังสือ ฯลฯ
- from datetime import datetime, timedelta: นำเข้า classes และ functions ที่เกี่ยวข้องกับการจัดการเวลา (datetime) และการคำนวณเวลา (timedelta) เพื่อใช้ในการดำเนินการต่างๆ ที่เกี่ยวข้องกับเวลา เช่น การเก็บข้อมูลเวลาที่ทำการยืมหนังสือ การคำนวณกำหนดส่งคืนหนังสือ เป็นต้น

สร้างคลาส MainWindow

```
class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Library")
        self.setFixedSize(QSize(1920, 1080))
        # สร้าง class lib
        self.library = Library()


        # กำหนดพื้นหลังเป็นรูปภาพ
        palette = self.palette()
        image = QImage("classic.png")
        palette.setBrush(QPalette.ColorRole.Window, QBrush(image))
        self.setAutoFillBackground(True)
        self.setPalette(palette)

        # สร้าง QVBoxLayout และ QHBoxLayout สำหรับปุ่มและ Label
        vbox = QVBoxLayout(self)
        vbox.setAlignment(Qt.AlignmentFlag.AlignTop)

        label_layout = QHBoxLayout()
        label_layout.setAlignment(Qt.AlignmentFlag.AlignCenter)
        label_layout.setContentsMargins(0, 20, 0, 20)  # กำหนดระยะห่างด้านบนและด้านล่าง

        button_layout = QHBoxLayout()
        button_layout.setAlignment(Qt.AlignmentFlag.AlignCenter)
        button_layout.setSpacing(20)

        vbox.addLayout(label_layout)
        vbox.addLayout(button_layout)
```

```
# เพิ่ม QLabel ด้านบน
label = QLabel("Library")
label.setAlignment(Qt.AlignmentFlag.AlignCenter)
label.setStyleSheet("font-size: 24px; font-weight: bold; color: white; background-color: rgba(0, 0, 0, 0.5);")
label_layout.addWidget(label)

# สร้างและแสดงปุ่ม
self.display_button("Add new member", button_layout)
self.display_button("Search member", button_layout)
self.display_button("Add new book", button_layout)
self.display_button("Search book", button_layout)
self.display_button("Borrow a book", button_layout)
self.display_button("Return the book", button_layout)
self.display_button("Check The book is overdue", button_layout)
```

- กำหนดหัวเรื่องของหน้าต่างด้วย self.setWindowTitle("Library")
- กำหนดขนาดคงที่ของหน้าต่างด้วย self.setFixedSize(QSize(1920, 1080))
- กำหนดพื้นหลังของหน้าต่างด้วยรูปภาพ ("classic.png") โดยใช้ QImage, QPalette, และ QBrush
- สร้างเลย์เอาต์แนวดิ่งและแนวนอน (QVBoxLayout, QHBoxLayout) เพื่อจัดวางปุ่มและป้ายกำกับในหน้าต่าง

- ใช้ฟังก์ชัน display_button เพื่อสร้างปุ่มต่าง ๆ และเชื่อมต่อกับฟังก์ชันต่าง ๆ เมื่อปุ่มถูกคลิก เช่น เพิ่มสมาชิกใหม่, ค้นหาสมาชิก, เพิ่มหนังสือใหม่, ค้นหาหนังสือ, ยืมหนังสือ, คืนหนังสือ ซึ่งจะช่วยให้ผู้ใช้สามารถทำงานกับระบบห้องสมุดได้อย่างสะดวกและรวดเร็ว

```
def paintEvent(self,event):
    painter = QPainter(self)
    painter.setRenderHint(QPainter.RenderHint.Antialiasing)

    # กำหนดสีและความโปร่งแสงของสี่เหลี่ยม
    painter.setPen(Qt.GlobalColor.black)
        # กำหนดสีของพื้นหลัง
    painter.setBrush(Qt.GlobalColor.white)  # เลือกสีเขียวเป็นตัวอย่าง

    # วาดสี่เหลี่ยมที่ 1
    rect1 = QRectF(60, 180, 400, 600)
    painter.drawRect(rect1)
    # ปรับพิกัดของสี่เหลี่ยมเล็กลงเพื่อทำให้ข้อความอยู่ในตำแหน่งที่ต้องการ
    margin = 10
    text_rect1 = QRectF(rect1.x() + margin, rect1.y() + margin, rect1.width() - 2 * margin, rect1.height() - 2 * margin)
    # เขียนข้อความ
    painter.setFont(QFont("Arial", 12))
    painter.drawText(text_rect1, Qt.AlignmentFlag.AlignHCenter, "MEMBER")
    # display member
    self.display_detail_member(painter , rect1)

    rect2 = QRectF(55*9, 180, 550, 600)
    painter.drawRect(rect2)
    margin = 10
    text_rect2 = QRectF(rect2.x() + margin, rect2.y() + margin, rect2.width() - 2 * margin, rect2.height() - 2 * margin)
    # เขียนข้อความ
    painter.setFont(QFont("Arial", 12))
    painter.drawText(text_rect2, Qt.AlignmentFlag.AlignHCenter, "BOOK")
    # display book ใน lip
    self.display_detail_book(painter , rect2)
```

การกำหนด Painter และ RenderHint

- painter = QPainter(self): สร้างอ็อบเจกต์ Painter โดยใช้ QWidget เป็นพื้นที่ที่จะวาด
- painter.setRenderHint(QPainter.RenderHint.Antialiasing): กำหนด RenderHint เพื่อให้การวาดมีความคมชัด และไม่มีเส้นต่อ
- การกำหนดสีและวาดสี่เหลี่ยม
- กำหนดสีของเส้นขอบสี่เหลี่ยม (pen) เป็นสีดำ (Qt.GlobalColor.black)
- กำหนดสีของพื้นหลังสี่เหลี่ยม (brush) เป็นสีขาว (Qt.GlobalColor.white)
- วาดสี่เหลี่ยมโดยใช้ painter.drawRect(rect) และกำหนดพิกัดและขนาดของสี่เหลี่ยมแต่ละอัน (rect1, rect2, rect3, rect4, rect5)

```
rect2 = QRectF(55*9, 180, 550, 600)
painter.drawRect(rect2)
margin = 10
text_rect2 = QRectF(rect2.x() + margin, rect2.y() + margin, rect2.width() - 2 * margin, rect2.height() - 2 * margin)
# เขียนข้อความ
painter.setFont(QFont("Arial", 12))
painter.drawText(text_rect2, Qt.AlignmentFlag.AlignHCenter, "BOOK")
# display book ใน lip
self.display_detail_book(painter , rect2)

rect3 = QRectF(60*18, 180, 400, 600)
painter.drawRect(rect3)
margin = 10
text_rect3 = QRectF(rect3.x() + margin, rect3.y() + margin, rect3.width() - 2 * margin, rect3.height() - 2 * margin)
# เขียนข้อความ
painter.setFont(QFont("Arial", 12))
painter.drawText(text_rect3, Qt.AlignmentFlag.AlignHCenter, "BORROWING")
self.display_detail_borrow(painter , rect3)

rect4 = QRectF(60*25, 180, 400, 600)
painter.drawRect(rect4)
margin = 10
text_rect4 = QRectF(rect4.x() + margin, rect4.y() + margin, rect4.width() - 2 * margin, rect4.height() - 2 * margin)
# เขียนข้อความ
painter.setFont(QFont("Arial", 12))
painter.drawText(text_rect4, Qt.AlignmentFlag.AlignHCenter, "OVERDUE")
self.display_detail_over_due(painter,rect4)
```

```
rect5 = QRectF(60, 200 *4, 1500, 150)
painter.drawRect(rect5)
margin = 10
text_rect5 = QRectF(rect5.x() + margin, rect5.y() + margin, rect5.width() - 2 * margin, rect5.height() - 2 * margin)
# เขียนข้อความ
painter.setFont(QFont("Arial", 12))
painter.drawText(text_rect5, Qt.AlignmentFlag.AlignHCenter, "Overdue")
self.display_detail_popular(painter ,rect5)
```

การเขียนข้อความบนสี่เหลี่ยม

- ใช้ painter.drawText เพื่อเขียนข้อความบนสี่เหลี่ยมที่วาดขึ้นมา โดยกำหนดว่าจะแสดงข้อความอยู่ที่ไหนของสี่เหลี่ยม และใช้ฟอนต์และขนาดตัวอักษรที่ต้องการ
- ตัวอย่างการเขียนข้อความได้แก่ "MEMBER", "BOOK", "BORROWING", "OVERDUE", "Overdue" ซึ่งแสดงข้อมูลต่าง ๆ ที่เกี่ยวข้องกับสมาชิก, หนังสือ, การยืม, และหนังสือที่ค้างคืน

การแสดงข้อมูล

- ใช้ฟังก์ชัน display_detail_member, display_detail_book, display_detail_borrow, display_detail_over_due, และ display_detail_popular เพื่อแสดงข้อมูลของสมาชิก, หนังสือ, การยืม, ค้างคืน, และหนังสือยอดนิยมภายในสี่เหลี่ยมที่วาดไว้

```python
# แสดงข้อมูล member ภายใน lib
def display_detail_member(self , painter , rect):
    mem_list = self.library.get_member()
    for index, mem in enumerate(mem_list):
       margin = 10
       text_rect = QRectF(rect.x() + margin , rect.y() + (margin*5 * (index+1)), rect.width() - 2 * margin, rect.height() - 2 * margin)
       painter.setFont(QFont("Arial", 12))
       # ตัวอย่างการแสดงข้อมูลสมาชิก
       painter.drawText(text_rect, Qt.AlignmentFlag.AlignLeft, f"{index+1}   Name: {mem.get_name()}, Contact: {mem.get_contact()} , ID : {mem.get_id()}")
# แสดงข้อมูล book ภายใน lib
def display_detail_book(self , painter , rect):
   pub_list = self.library.get_publication()
   for index, pub in enumerate(pub_list):
     margin = 10
     text_rect = QRectF(rect.x() + margin , rect.y() + (margin*5 * (index+1)), rect.width() - 2 * margin, rect.height() - 2 * margin)
     painter.setFont(QFont("Arial", 12))
     # ตัวอย่างการแสดงข้อมูลสมาชิก
     painter.drawText(text_rect, Qt.AlignmentFlag.AlignLeft, f"{index+1}  Title: {pub.get_title()}, Author: {pub.get_author()} , Genre : {pub.get_genre()} , Ye
# แสดงข้อมูล borrow ภายใน lib
def display_detail_borrow(self , painter , rect):
   loan_list = self.library.get_loan()
   for index, loan in enumerate(loan_list):
     margin = 10
     text_rect = QRectF(rect.x() + margin , rect.y() + (margin*5 * (index+1)), rect.width() - 2 * margin, rect.height() - 2 * margin)
     painter.setFont(QFont("Arial", 12))
     # ตัวอย่างการแสดงข้อมูลสมาชิก
     painter.drawText(text_rect, Qt.AlignmentFlag.AlignLeft, f"{index+1}  Name: {loan.get_member().get_name()}, Title: {loan.get_publication().get_title()} , I
```

display_detail_member(painter, rect)

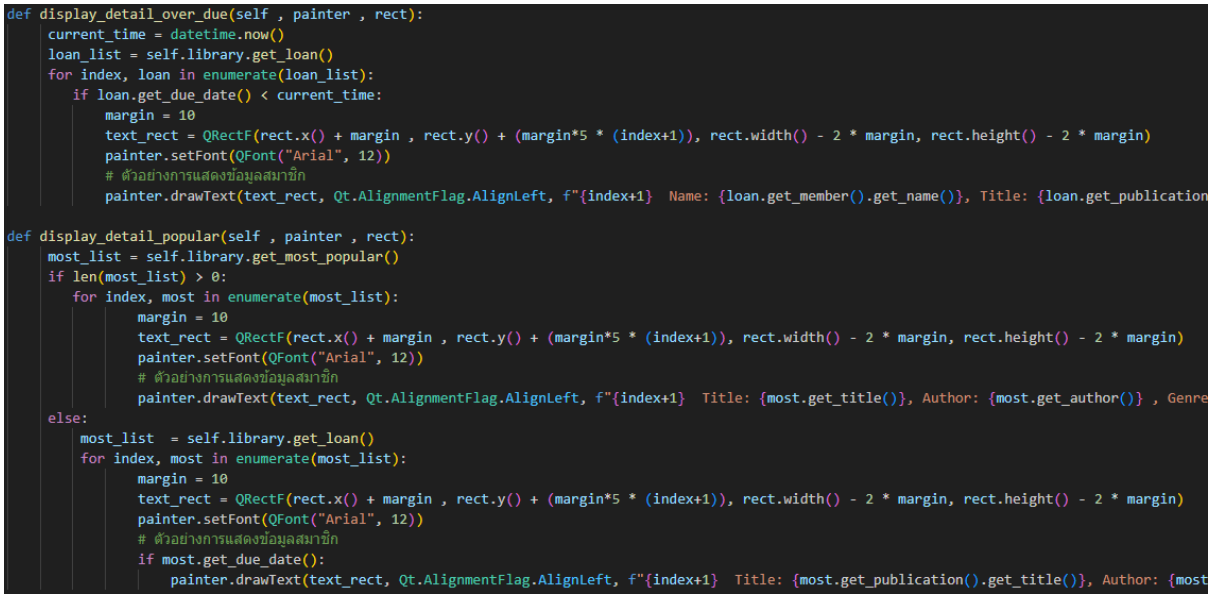
- ฟังก์ชันนี้ใช้สำหรับแสดงข้อมูลของสมาชิกภายในห้องสมุด (Library) โดยรับพารามิเตอร์เป็น painter และ rect ซึ่งเป็น QPainter และ QRectF ตามลำดับ
- ฟังก์ชันนี้จะดึงข้อมูลสมาชิกจากห้องสมุดด้วย self.library.get_member() และแสดงข้อมูลของแต่ละสมาชิกด้วยการวนลูปผ่าน mem_list
- สำหรับแต่ละสมาชิก mem ใน mem_list จะสร้าง text_rect สำหรับการแสดงข้อความ และใช้ painter.drawText() เพื่อแสดงข้อมูลสมาชิกนั้นๆ

display_detail_book(painter, rect)

- ฟังก์ชันนี้ใช้สำหรับแสดงข้อมูลหนังสือภายในห้องสมุด (Library) โดยรับพารามิเตอร์เป็น painter และ rect เช่นเดียวกับฟังก์ชันก่อนหน้า
- จะดึงข้อมูลหนังสือจากห้องสมุดด้วย self.library.get_publication() และวนลูปผ่าน pub_list เพื่อแสดงข้อมูลแต่ละเรื่อง

display_detail_borrow(painter, rect)

- ฟังก์ชันนี้ใช้สำหรับแสดงข้อมูลการยืมหนังสือภายในห้องสมุด (Library) โดยรับพารามิเตอร์เป็น painter และ rect เช่นเดียวกับฟังก์ชันก่อนหน้า
- จะดึงข้อมูลการยืมหนังสือจากห้องสมุดด้วย self.library.get_loan() และวนลูปผ่าน loan_list เพื่อแสดงข้อมูลแต่ละรายการการยืม



```
def display_detail_over_due(self , painter , rect):
    current_time = datetime.now()
    loan_list = self.library.get_loan()
    for index, loan in enumerate(loan_list):
        if loan.get_due_date() < current_time:
            margin = 10
            text_rect = QRectF(rect.x() + margin , rect.y() + (margin*5 * (index+1)), rect.width() - 2 * margin, rect.height() - 2 * margin)
            painter.setFont(QFont("Arial", 12))
            # ตัวอย่างการแสดงข้อมูลสมาชิก
            painter.drawText(text_rect, Qt.AlignmentFlag.AlignLeft, f"{index+1}  Name: {loan.get_member().get_name()}, Title: {loan.get_publication

def display_detail_popular(self , painter , rect):
    most_list = self.library.get_most_popular()
    if len(most_list) > 0:
        for index, most in enumerate(most_list):
                margin = 10
                text_rect = QRectF(rect.x() + margin , rect.y() + (margin*5 * (index+1)), rect.width() - 2 * margin, rect.height() - 2 * margin)
                painter.setFont(QFont("Arial", 12))
                # ตัวอย่างการแสดงข้อมูลสมาชิก
                painter.drawText(text_rect, Qt.AlignmentFlag.AlignLeft, f"{index+1}  Title: {most.get_title()}, Author: {most.get_author()} , Genre
    else:
        most_list  = self.library.get_loan()
        for index, most in enumerate(most_list):
                margin = 10
                text_rect = QRectF(rect.x() + margin , rect.y() + (margin*5 * (index+1)), rect.width() - 2 * margin, rect.height() - 2 * margin)
                painter.setFont(QFont("Arial", 12))
                # ตัวอย่างการแสดงข้อมูลสมาชิก
                if most.get_due_date():
                    painter.drawText(text_rect, Qt.AlignmentFlag.AlignLeft, f"{index+1}  Title: {most.get_publication().get_title()}, Author: {most
```

display_detail_over_due(painter, rect)

- ฟังก์ชันนี้ใช้สำหรับแสดงข้อมูลการยืมหนังสือที่ครบกำหนดคืนแล้วภายในห้องสมุด (Library) โดยรับพารามิเตอร์เป็น painter และ rect เช่นเดียวกับฟังก์ชันก่อนหน้า
- ฟังก์ชันนี้จะเช็คว่าวันครบกำหนดคืนของการยืมหนังสือมีค่าน้อยกว่าเวลาปัจจุบันหรือไม่ และแสดงข้อมูลนี้ออกทาง UI

display_detail_popular(painter, rect)

- ฟังก์ชันนี้ใช้สำหรับแสดงข้อมูลหนังสือยอดนิยมในห้องสมุด (Library) โดยรับพารามิเตอร์เป็น painter และ rect เช่นเดียวกับฟังก์ชันก่อนหน้า
- จะดึงข้อมูลหนังสือยอดนิยมจากห้องสมุดด้วย self.library.get_most_popular() และแสดงข้อมูลนั้นออกทาง UI

```python
# หน้าต่าง ของ add member
def box_add_new_member(self):
    dialog = QDialog(self)
    dialog.setWindowTitle("Add New Member")

    layout = QVBoxLayout()

    name_label = QLabel("Name:")
    layout.addWidget(name_label)
    name_input = QLineEdit()
    layout.addWidget(name_input)


    id_label = QLabel("Id:")
    layout.addWidget(id_label)
    id_input = QLineEdit()
    layout.addWidget(id_input)



    contact_label = QLabel("Contact:")
    layout.addWidget(contact_label)
    contact_input = QLineEdit()
    layout.addWidget(contact_input)



    btn_ok = QPushButton("OK")
    btn_ok.clicked.connect(lambda: self.on_ok_mem(dialog, name_input.text(), contact_input.text() , id_input.text()))
    layout.addWidget(btn_ok)

    dialog.setLayout(layout)
    dialog.exec()
```

box_add_new_member() ฟังก์ชันนี้ใช้สำหรับเปิดหน้าต่างเพื่อเพิ่มสมาชิกใหม่ลงในระบบ โดยจะสร้างหน้าต่างใหม่ที่มีช่องให้กรอกข้อมูล และปุ่ม "OK" เพื่อยืนยันการเพิ่มข้อมูล

การทำงานของฟังก์ชั่นอื่นๆ

```python
# รันแอปพลิเคชัน
app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

สร้างแอปพลิเคชัน QApplication สร้างหน้าต่างหลักของแอปพลิเคชันด้วยคลาส MainWindow ที่สร้างขึ้น เริ่มการทำงานของแอปพลิเคชันและรอจนกว่าผู้ใช้จะปิดโปรแกรม

Library_system.py

โค้ด Python นี้มีโครงสร้างพื้นฐานเกี่ยวกับระบบห้องสมุด (Library System) ที่มีคลาสหลักคือ Publication, Member, Book, Loan, และ Library อธิบายการทำงานของโค้ดนี้ตามลำดับขั้นตอนนี้

Import Libraries

```
from abc import ABC, abstractmethod
from datetime import datetime, timedelta
```

นำเข้าคลาส ABC และmethod abstractmethod จากโมดูล abc เพื่อใช้ในการสร้างคลาสแบบ Abstract และmethod Abstract นำเข้าคลาส datetime และ timedelta จากโมดูล datetime เพื่อใช้ในการจัดการเกี่ยวกับเวลาและวันที่ในโปรแกรม

สร้างคลาส Publication

```python
# สร้างคลาส Publication เป็น abstract class
class Publication(ABC):
    #  กำหนด property ต่างๆ
    def __init__(self , title , author, year):
        self.__title = title
        self.__author = author
        self.__year = year

    def get_title(self):
        return self.__title
    def get_author(self):
        return self.__author

    def set_title(self,title):
        self.__title = title

    def set_author(self,author):
        self.__author = author

    def set_year(self,year):
        self.__year = year

    def get_year(self):
        return self.__year

    # abstractmethod ต่างๆเพื่อให้คลาสลูกมี method นี้
    @abstractmethod
    def display_detail(self):
       raise NotImplementedError("Subclasses must implement display_detail method.")
```

```python
@abstractmethod
def search_item(self,lip):
   raise NotImplementedError("Subclasses must implement search_item method.")

@abstractmethod
def update_item(self,lip):
   raise NotImplementedError("Subclasses must implement update_item method.")
```

เริ่มต้นการสร้างคลาส Publication และระบุให้มันเป็นคลาสแบบ Abstract ด้วยการสืบทอดจาก ABC ซึ่งมีmethod Abstract อย่างน้อยหนึ่งmethodในคลาส สร้างmethod __init__ เพื่อกำหนดค่าเริ่มต้นสำหรับอ็อบเจกต์ของคลาส Publication โดยรับพารามิเตอร์ title, author, และ year สร้างmethodเพื่อให้สามารถเข้าถึงข้อมูล title และ author ของอ็อบเจกต์ได้ กำหนดค่า title และ author ของอ็อบเจกต์ ระบุmethodที่เป็น Abstract ในคลาส Publication โดยในที่นี้มีmethod display_detail, add_item, search_item, และ update_item เป็น Abstract ซึ่งต้องถูกโอเวอร์ไรด์ในคลาสที่สืบทอด

สร้างคลาส Library

```
class Library:
    def __init__(self):
        self.__member = []
        self.__publication = []
        self.__loan = []
    # เพิ่ม member
    def add_member(self,mem):
        for mem_lib in self.__member:
            if mem_lib.get_id() == mem.get_id():
                return False
        self.__member.append(mem)
        return True

    def get_member(self):
        return self.__member

    def get_publication(self):
        return self.__publication

    def add_publication(self,pub):
        for pub_lip in self.__publication:
            if pub_lip.get_isbn() == pub.get_isbn():
                return False
        self.__publication.append(pub)
        return True


    def get_publication(self):
        return self.__publication

    def remove_publication(self,pub):
        self.__publication.remove(pub)
```

เริ่มต้นสร้างคลาส Library ซึ่งเป็นส่วนหนึ่งของระบบห้องสมุด โดยมีเมทอดและคุณสมบัติต่างๆ ที่ใช้ในการจัดการข้อมูลสมาชิก (member), หนังสือ (publication), และการยืม-คืนหนังสือ (loan) เมทอดสร้างอ็อบเจกต์ของคลาส Library โดยกำหนดให้มีรายการสมาชิก (member), หนังสือ (publication), และการยืม-คืนหนังสือ (loan) ว่างเปล่าเมื่อสร้างอ็อบเจกต์ใหม่ของคลาสนี้ เพิ่มสมาชิกเข้าในรายการสมาชิก โดยตรวจสอบว่าสมาชิกที่ต้องการเพิ่มมี ID ที่ไม่ซ้ำกับสมาชิกที่มีอยู่แล้วหรือไม่ เพิ่มหนังสือเข้าในรายการหนังสือ โดยตรวจสอบว่าหนังสือที่ต้องการเพิ่มมี ISBN ที่ไม่ซ้ำกับหนังสือที่มีอยู่แล้วหรือไม่ ลบหนังสือออกจากรายการหนังสือ

```
def remove_publication(self,pub):
    self.__publication.remove(pub)

def search_member(self , id):
    mem_list = self.__member
    for mem in mem_list:
        if mem.get_id() == id:
            return mem
    return False

def search_book(self , title , author , genre):
    pub_list = self.__publication
    for pub in pub_list:
        if pub.get_title() == title and pub.get_author() == author and pub.get_genre() == genre:
            return pub
    return False

def search_book_isbn(self , isbn):
    print(isbn)
    pub_list = self.__publication
    for pub in pub_list:
       print(pub.get_isbn())
       if pub.get_isbn() == isbn:
            return pub
    return False

def add_loan(self , loan):
    self.__loan.append(loan)

def get_loan(self):
    return self.__loan
```

ลบหนังสือออกจากรายการหนังสือ ค้นหาสมาชิกโดยใช้ ID ในรายการสมาชิก ค้นหาหนังสือโดยใช้ชื่อ, ผู้เขียน, และประเภทหนังสือในรายการหนังสือ ค้นหาหนังสือโดยใช้ ISBN ในรายการหนังสือ เพิ่มการยืม-คืนหนังสือเข้าในรายการยืม-คืนหนังสือ คืนค่ารายการยืม-คืนหนังสือ

```
def search_loan(self , mem , pub):
    loan_list =  self.__loan
    for loan in loan_list:
        if loan.get_member() == mem and loan.get_publication().get_isbn() == pub:
            return loan
    return False

def remove_loan(self ,loan):
    self.__loan.remove(loan)

def get_most_popular(self):
    pub_list = self.__publication
    if len(pub_list) > 0:
        max_rating = max(pub_list, key=lambda pub: pub.get_rating()).get_rating()
        pub_most = [pub for pub in pub_list if pub.get_rating() == max_rating]
        return pub_most
    return []
```

ค้นหาการยืม-คืนหนังสือโดยใช้สมาชิกและหนังสือที่เกี่ยวข้องในรายการยืม-คืนหนังสือ ลบการยืม-คืนหนังสือออกจากรายการยืม-คืนหนังสือ นค่าหนังสือที่มีคะแนนยอดนิยมสูงสุดในรายการหนังสือ

สร้างคลาส Member

```
class Member(Publication):
    # กำหนด property ต่างๆ
    def __init__(self,name,id,contact):
        self.__name = name
        self.__id = id
        self.__contact = contact


    def search_item(self , lip):
        mem_list = lip.get_member()
        if self in mem_list:
                return self.display_detail()
        print('ไม่พบข้อมูล')

    def display_detail(self):
        print(f'name : {self.__name}\n id : {self.__id}\n contact : {self.__contact}')
        message = f'name : {self.__name}\n id : {self.__id}\n contact : {self.__contact}'
        return message

    def get_name(self):
        return self.__name

    def get_contact(self):
        return self.__contact

    def update_item(self,name,contact):
        if name and contact:
            self.__name = name
            self.__contact = contact
        else :
            print('update ไม่สำเร็จ')
```

เริ่มต้นสร้างคลาส Member ที่สืบทอดมาจาก Publication ซึ่งเป็นสมาชิกในห้องสมุด สร้างmethod __init__ ของคลาส Member เพื่อกำหนดค่าเริ่มต้นสำหรับอ็อบเจกต์ของคลาส Member โดยรับพารามิเตอร์ เช่น ชื่อ, รหัส, และข้อมูลติดต่อ สร้างmethodสำหรับเพิ่มสมาชิกเข้าสู่ระบบห้องสมุด ค้นหาสมาชิกในระบบห้องสมุด แสดงรายละเอียดของสมาชิก และอัปเดตข้อมูลของสมาชิก

สร้างคลาส Book

```
class Book(Publication):

    def __init__(self, title =None, author=None, year=None , isbn=None , genre=None):
        super().__init__(title, author, year)
        self.__isbn = isbn
        self.__genre = genre
        self.__rating = 0

    def search_item(self,lip):
        pub_list = lip.get_publication()
        if self in pub_list:
            return self.display_detail()
        else:
            print('ไม่พบหนังสือที่ท่านต้องการ')

    def display_detail(self):
        message = f'title : {super().get_author()}\nauthor : {super().get_title()}\ngenre : {self.__genre}'
        return message

    def set_genre(self,genre):
        self.__genre = genre

    def update_item(self,title,author,genre):
        super().set_title(title)
        super().set_author(author)
        self.set_genre(genre)

    def set_book(self,book):
        self = book

    def get_genre(self):
        return self.__genre
```

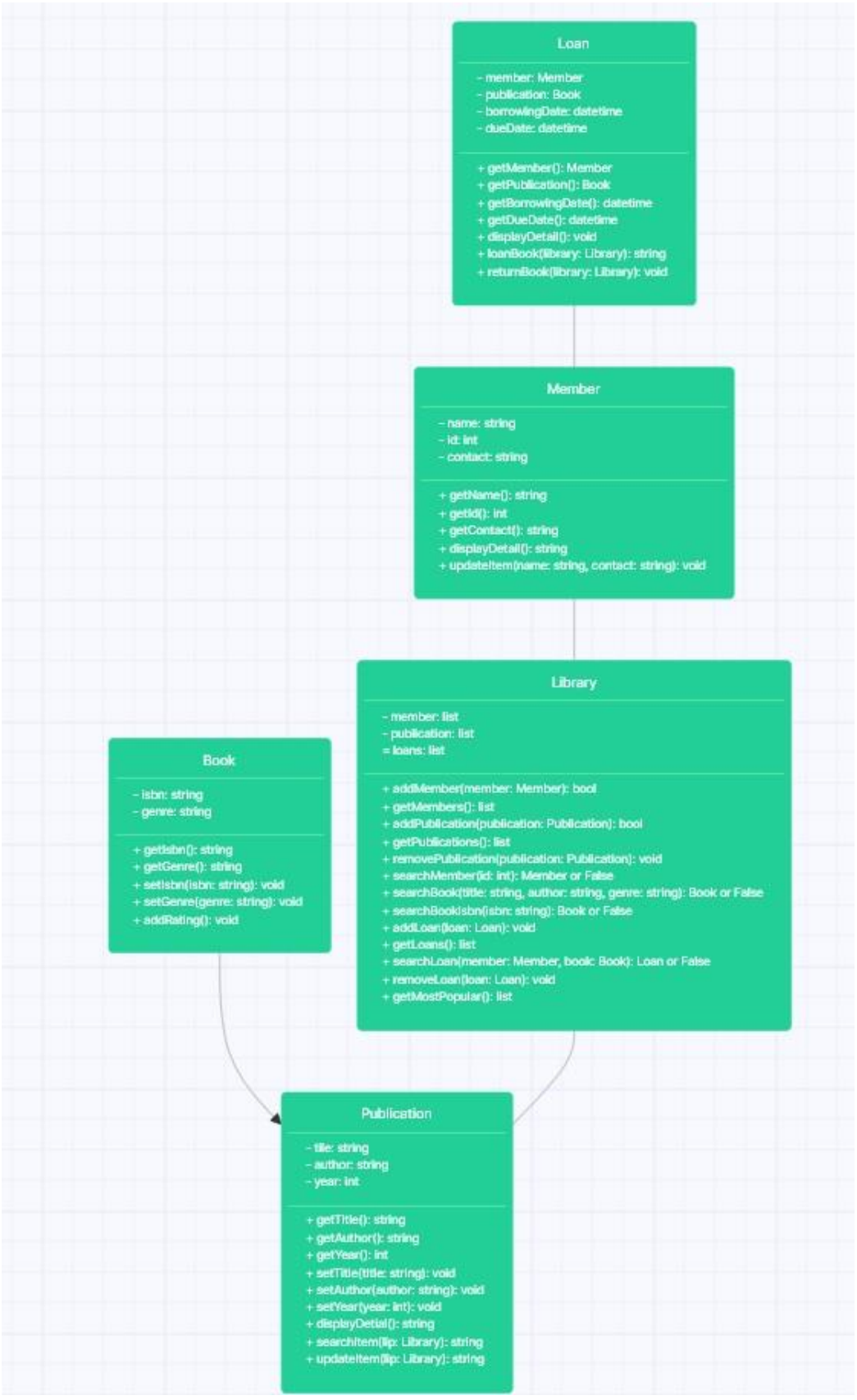
เริ่มต้นสร้างคลาส Book ที่สืบทอดมาจาก Publication ซึ่งเป็นหนังสือในห้องสมุด สร้างmethod __init__ ของคลาส Book เพื่อกำหนดค่าเริ่มต้นสำหรับอ็อบเจกต์ของคลาส Book โดยรับพารามิเตอร์ต่าง ๆ เช่น ชื่อหนังสือ, ผู้เขียน, ปีที่พิมพ์, ISBN, และประเภทของหนังสือ สร้างmethodสำหรับเพิ่มหนังสือเข้าสู่ระบบห้องสมุด ค้นหาหนังสือในระบบห้องสมุด แสดงรายละเอียดของหนังสือ และอัปเดตข้อมูลของหนังสือ

สร้างคลาส Loan

```
class Loan:
    def __init__(self , member , publication , borrowing_date , due_to):
        self.__member = member
        self.__publication = publication
        self.__borrowing_date = borrowing_date
        self.__due_to = due_to

    def display_detail(self):
        print(f'member : {self.__member.display_detail()}\n -- -- -- \npublication : {self.__publicatio

    # method ยืมหนังสือ ถ้าหนังสือมีใน lib จะสามารถยืมได้
    def loan_book(self,lip):
        pub_list = lip.get_publication()
        if self.__publication in pub_list:
            lip.remove_publication(self.__publication)
            lip.add_loan(self)
            self.__publication.add_rating()
            message ='Loan successfully'
            return message
        else:
            message ='Failed to loan'

    # คืนหนังสือกลับ lib
    def return_booK(self,lip):
        lip.add_publication(self.__publication)
        lip.remove_loan(self)
        print('คืนสำเร็จ')

    def get_member(self):
        return self.__member

    def get_publication(self):
        return self.__publication
```

เริ่มต้นสร้างคลาส Loan ซึ่งเป็นการยืมหนังสือในระบบห้องสมุด method __init__ ของคลาส Loan เพื่อกำหนดค่าเริ่มต้นสำหรับอ็อบเจกต์ของคลาส Loan โดยรับพารามิเตอร์เช่น สมาชิก, หนังสือ, วันที่ยืม, และกำหนดการคืน สร้างmethodสำหรับแสดงรายละเอียดของการยืมหนังสือ ทำการยืมหนังสือ และทำการคืนหนังสือ

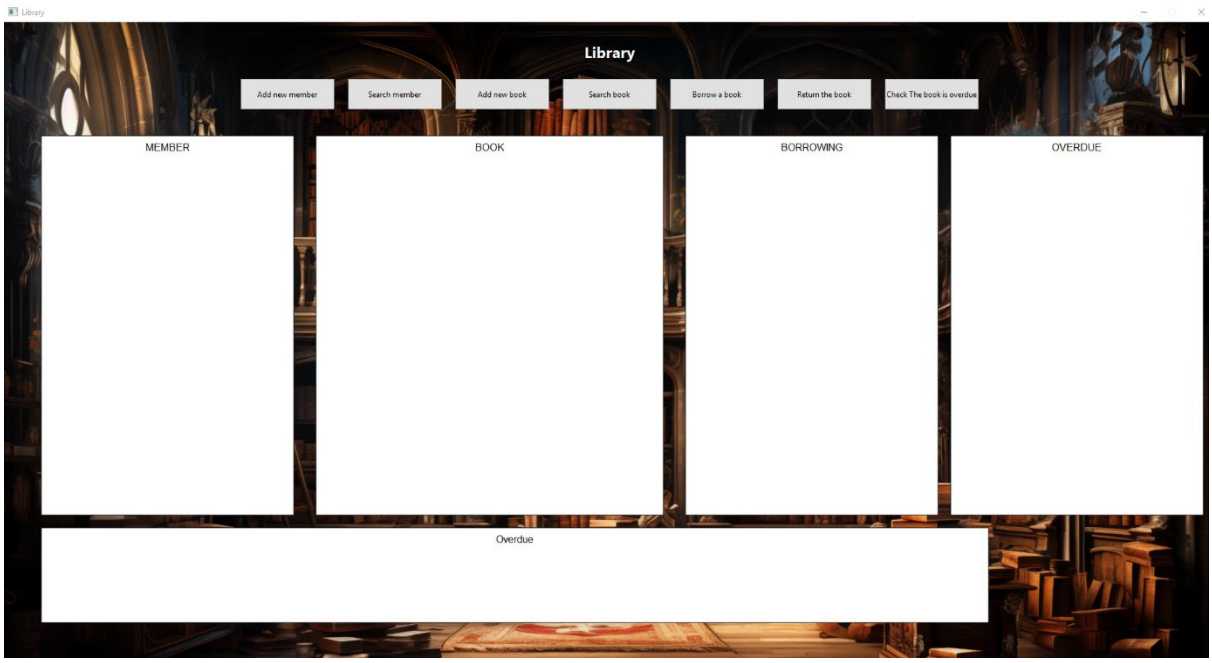
UML class diagram



- Publication เป็น Abstract Class: คลาส Publication เป็นคลาสแบบ abstract ซึ่งหมายถึงว่ามีเมทอด abstract ที่ต้องถูกสืบทอดและโอเวอร์ไรด์ในคลาสลูก
- Member เป็น Library User: Member เป็นคลาสที่แสดงถึงผู้ใช้งานของห้องสมุด (Library) โดยมีความสัมพันธ์แบบ "has-a" กับ Library
- Book เป็น Publication: Book เป็นคลาสที่สืบทอดจาก Publication และมีความสัมพันธ์แบบ "is-a" กับ Publication
- Loan เชื่อมต่อกับ Member และ Publication: Loan เป็นคลาสที่เชื่อมโยงระหว่าง Member (ผู้ยืม) และ Publication (หนังสือที่ถูกยืม) โดย Loan มีความสัมพันธ์ "has-a" กับ Member และ Publication
- Library เก็บ Member, Publication, และ Loan: Library เป็นคลาสที่มีความสัมพันธ์ "has-a" กับ Member (ผู้ใช้งาน), Publication (หนังสือ), และ Loan (การยืม) โดย Library เป็นตัวกลางที่ใช้ในการจัดการข้อมูลและการทำงานในระบบห้องสมุด

ดังนั้น ความสัมพันธ์ในโค้ดมีลักษณะเชิงวัตถุและความสัมพันธ์ที่ชัดเจนตามหลักการของ Object-Oriented Programming (OOP) โดยแต่ละคลาสมีบทบาทและความสัมพันธ์ที่ต่างกันในระบบที่สร้างขึ้น

ผลการรัน

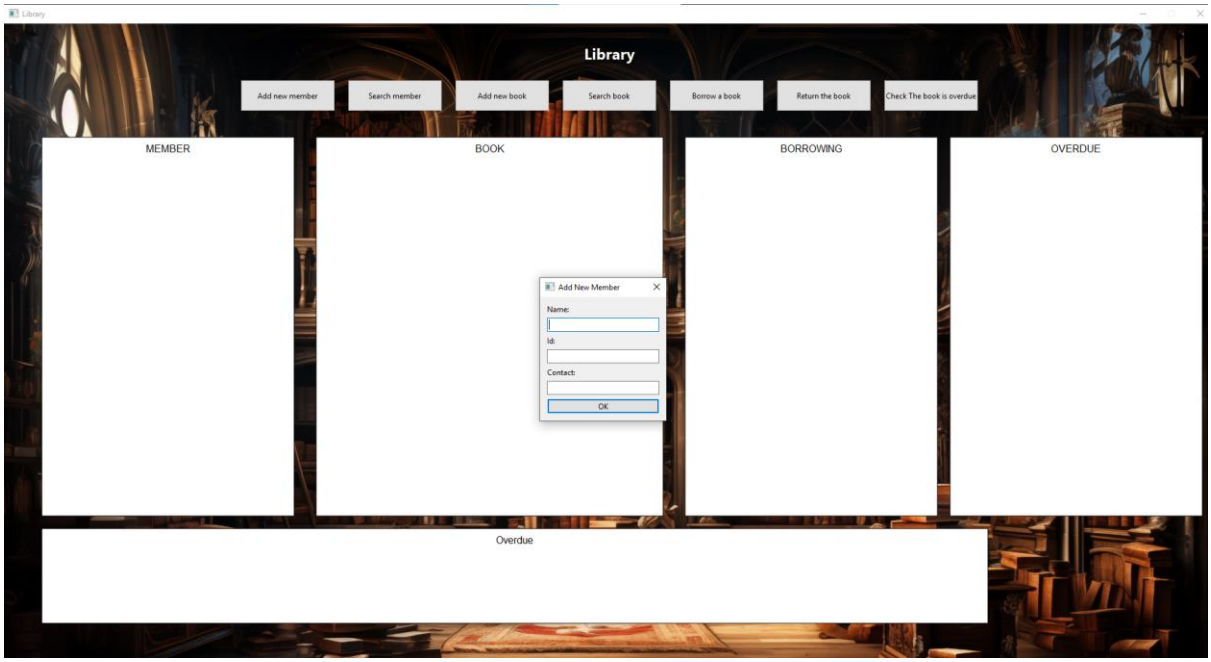


หน้า interface ประกอบด้วย

-ปุ่ม (Add new member, Search member, Add a Book, Return the book, check the book is overdue)

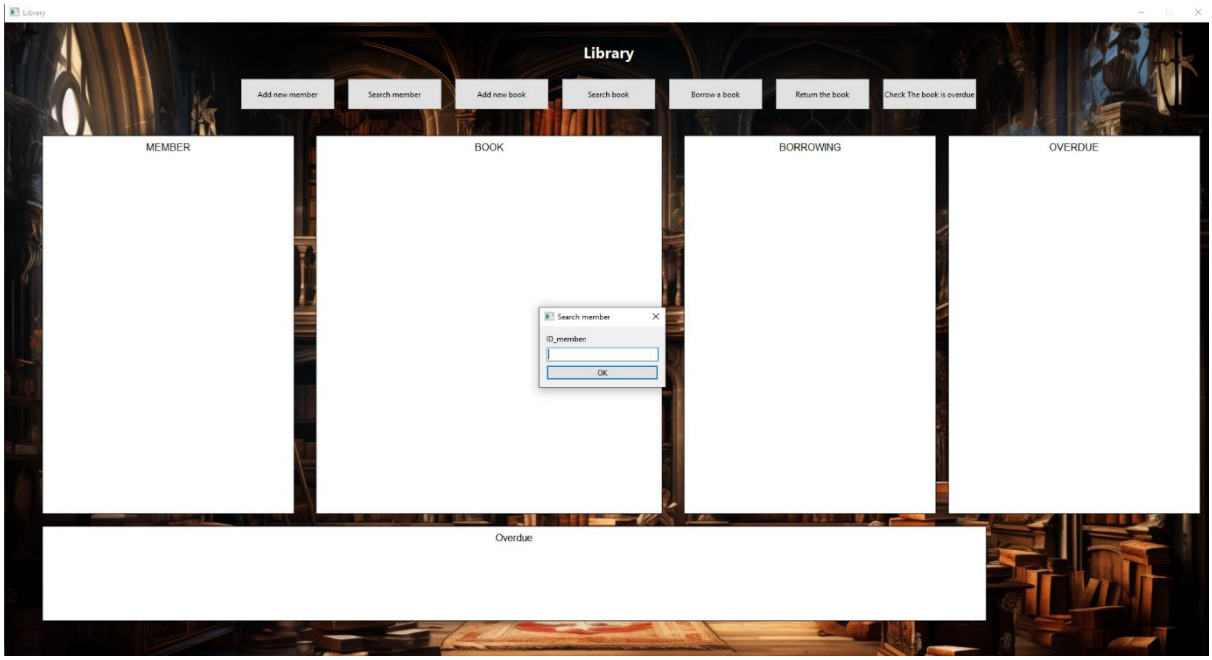
-หน้าต่างแสดงข้อมูลรายละเอียด (MEMBER, BOOK, BORROWING, OVERDUE)

ผลการรัน



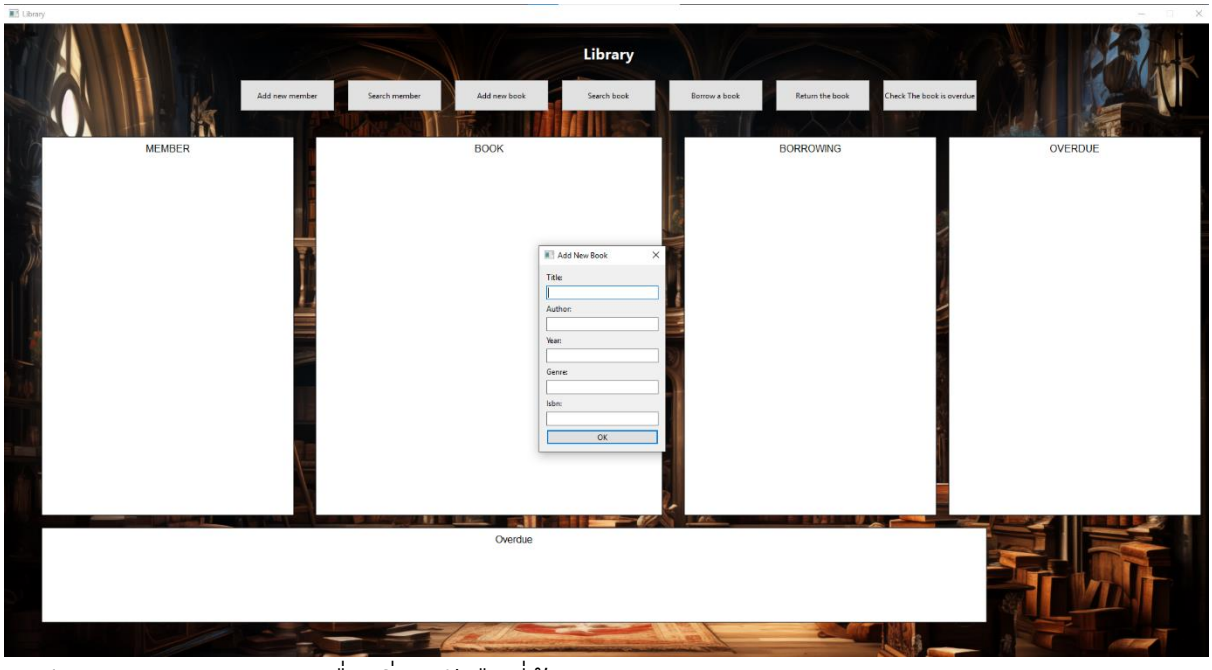
กดปุ่ม Add new member

- ใส่ (ชื่อ, id, contact) กด ok
- เสร็จสิ้นการเพิ่มสมาชิก



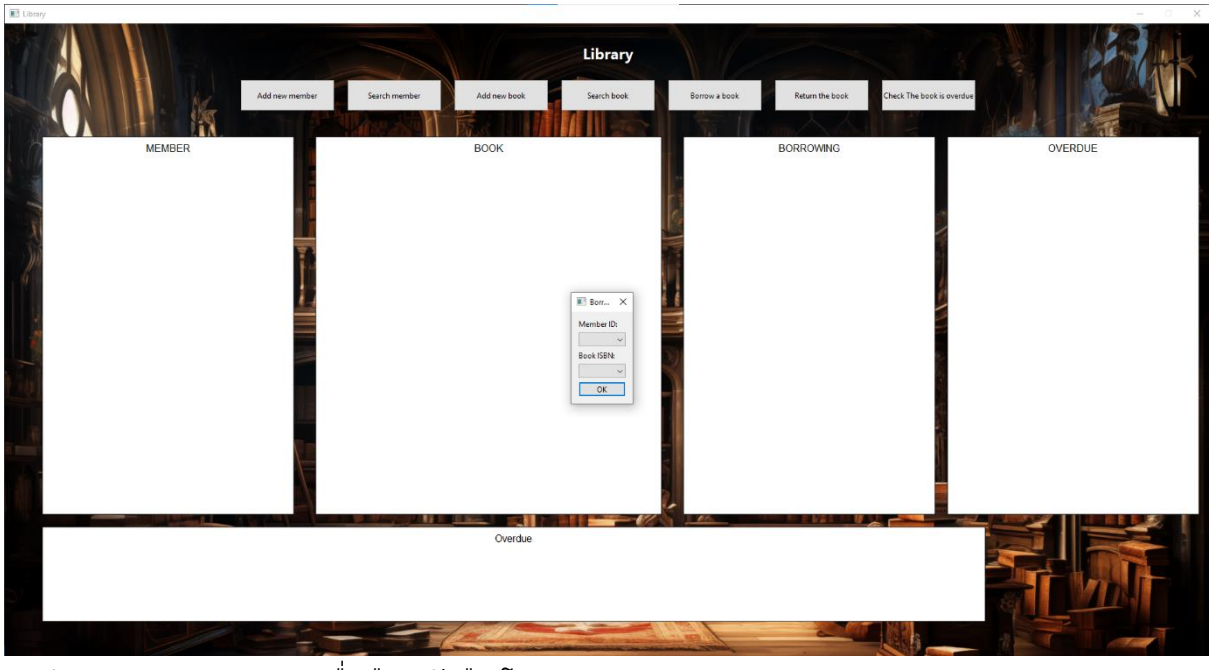
กดปุ่ม Search member

-ใส่เลข ID ของ member ในหน้าต่าง หน้าต่างจะแสดงข้อมูล member ของ ID นั้น



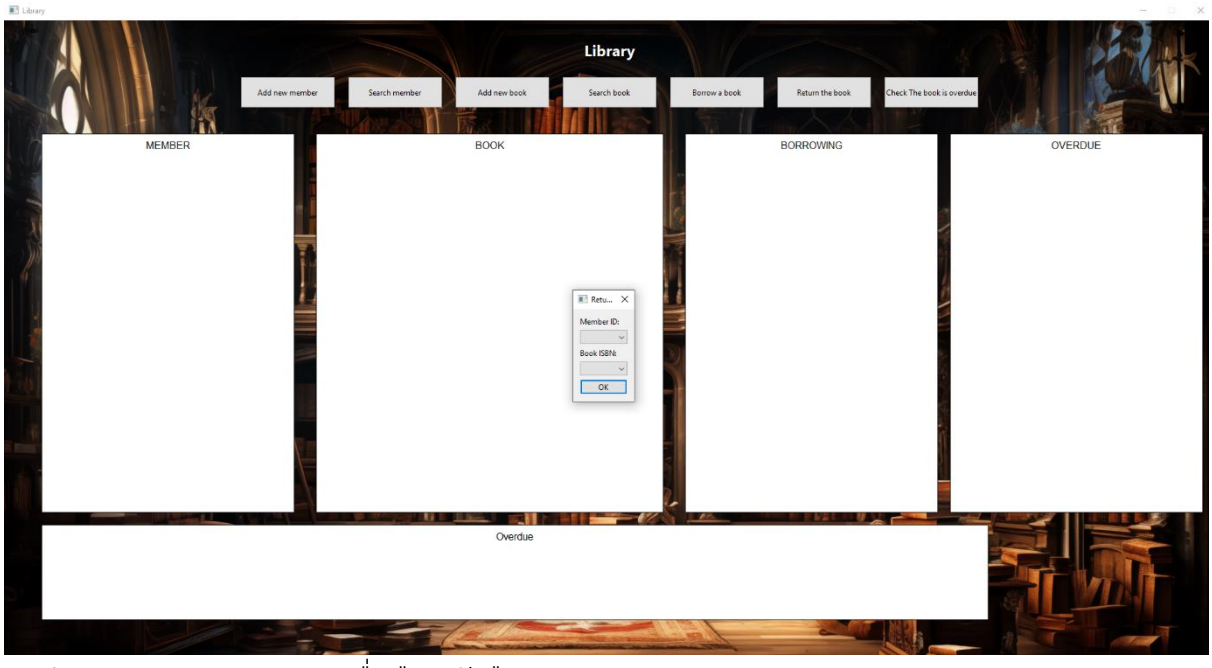
กดปุ่ม Add new book เพื่อเพิ่มหนังสือที่ต้องการ

-ใส่ข้อมูล(ชื่อหนังสือ, ชื่อผู้แต่ง, ปีที่พิมพ์, ประเภท, รหัสQRcode) กด ok



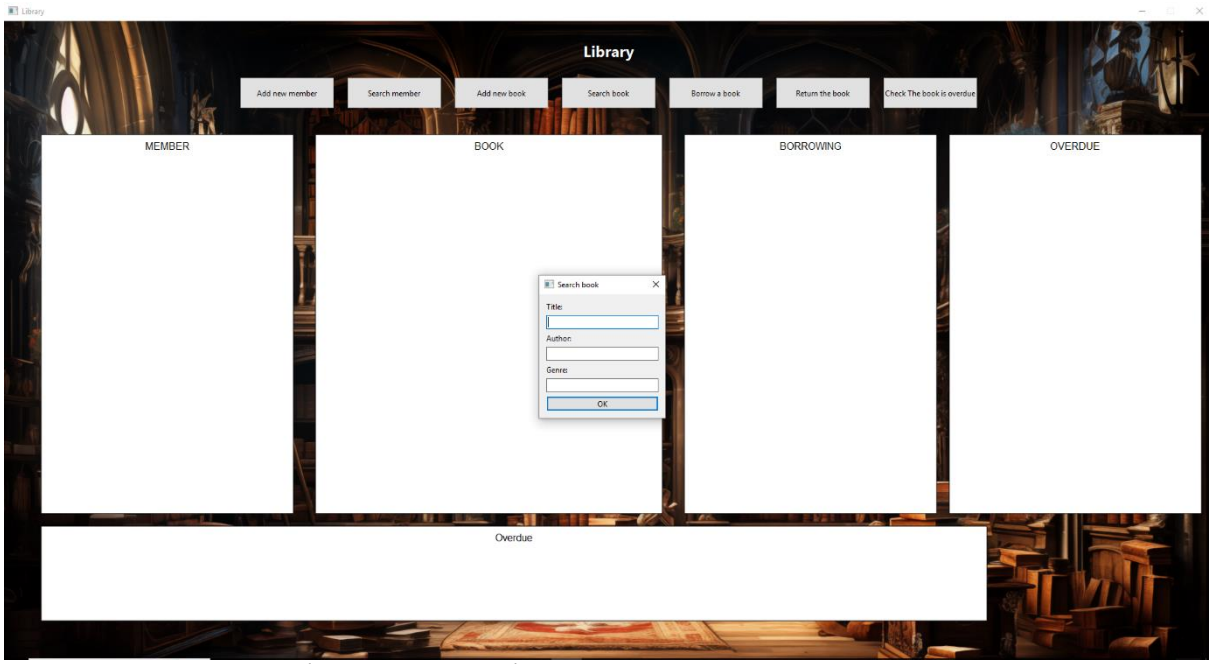
กดปุ่ม Borrow a book เพื่อยืมหนังสือ โดย

-เลือก ID member ที่ต้องการจะยืมเลือกหนังสือที่ต้องการจะยืม กด ok



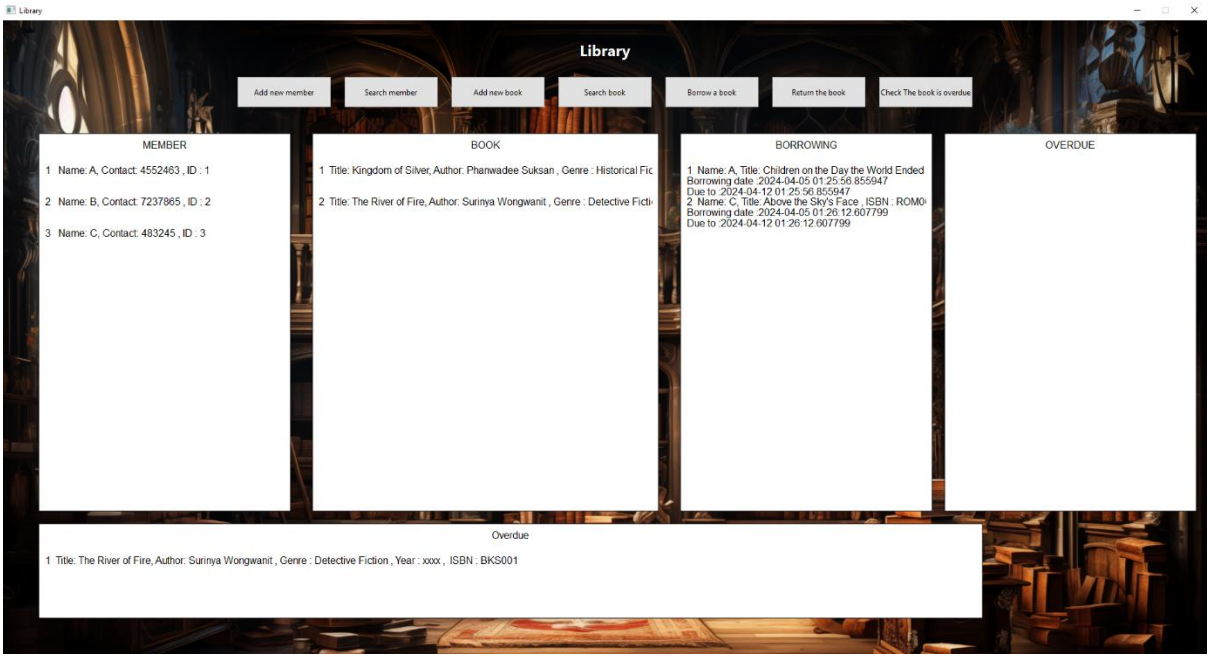
กดปุ่ม Return the book เพื่อคืนหนังสือ

-เลือก Member ID ที่ต้องการจะคืนหนังสือเลือกหนังสือที่ต้องการจะคืน กด ok



กดปุ่ม Search book เพื่อค้นหาหนังสือที่ต้องการ

-ใส่ข้อมูลหนังสือ (ชื่อหนังสือ, ผู้แต่ง, ประเภทหนังสือ) กด ok



หน้า interface แสดงตัวอย่างการใช้งานโดย

-เพิ่มสมาชิก

-เพิ่มหนังสือ

-ระบุหนังสือที่ member มีการยืม

-ระบุหนังสือที่เกินเวลาในการคืน