物件導向程式設計—期末報告
電影訂票系統

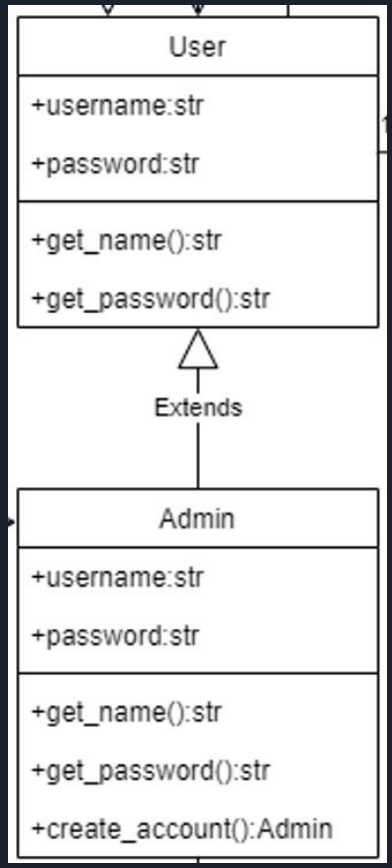
應數四甲 10911107 高博彥

系統介紹—電影訂票系統

- 支援用戶和管理員操作
- 用戶：註冊、登入/出、訂票、查看訂票紀錄
- 管理員：登入/出、新增電影、修改電影、刪除電影、查看電影列表
- 使用CSV模組
- 系統中管理員預設帳號：admin 預設密碼：admin_password
- 登入時為了隱私，輸入密碼不會顯示在螢幕上。

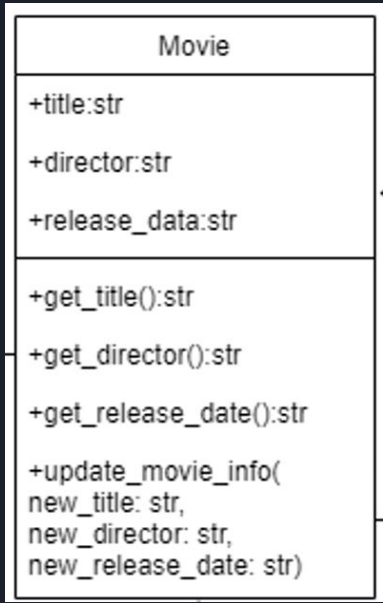
Class介紹—User

```python
# 定義 User 類別
class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password

    def get_name(self):
        return self.username

    def get_password(self):
        return self.password

# 定義 Admin 類別，繼承自 User
class Admin(User):
    def __init__(self, username, password):
        super().__init__(username, password)

    @staticmethod
    def create_account():
        new_admin = Admin("admin", "admin_password")
        return new_admin
```



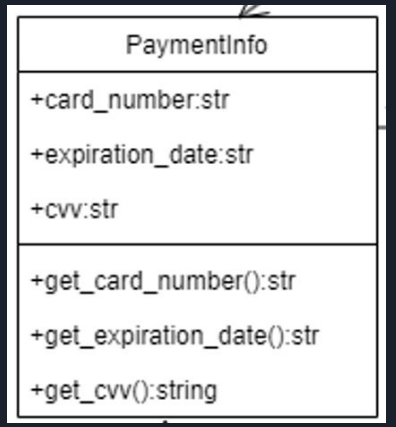
Class介紹—Movie

```python
# 定義 Movie 類別
class Movie:
    def __init__(self, title, director, release_date):
        self.title = title
        self.director = director
        self.release_date = release_date

    def get_title(self):
        return self.title

    def get_director(self):
        return self.director

    def get_release_date(self):
        return self.release_date

    def update_movie_info(self, new_title, new_director, new_release_date):
        self.title = new_title
        self.director = new_director
        self.release_date = new_release_date
```



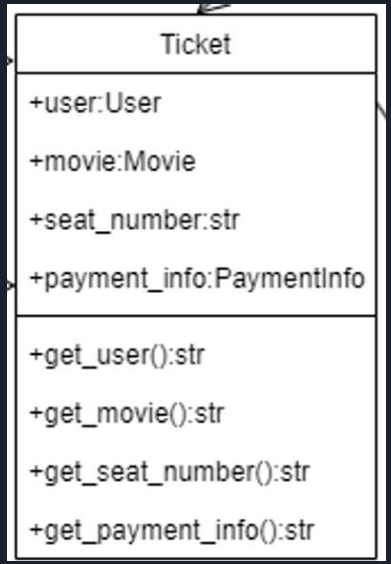
Class介紹—PaymentInfo

```python
# 定義 PaymentInfo 類別
class PaymentInfo:
    def __init__(self, card_number, expiration_date, cvv):
        self.card_number = card_number
        self.expiration_date = expiration_date
        self.cvv = cvv

    def get_card_number(self):
        return self.card_number

    def get_expiration_date(self):
        return self.expiration_date

    def get_cvv(self):
        return self.cvv
```



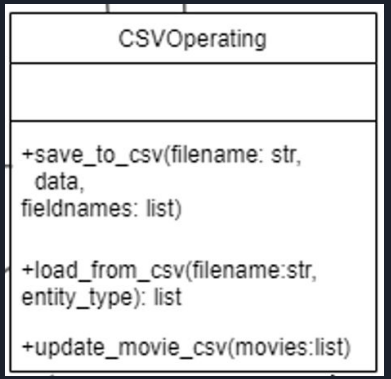
Class介紹—Ticket

```python
# 定義 Ticket 類別
class Ticket:
    def __init__(self, user, movie, seat_number, payment_info):
        self.user = user
        self.movie = movie
        self.seat_number = seat_number
        self.payment_info = payment_info

    def get_user(self):
        return self.user

    def get_movie(self):
        return self.movie

    def get_seat_number(self):
        return self.seat_number

    def get_payment_info(self):
        return self.payment_info
```



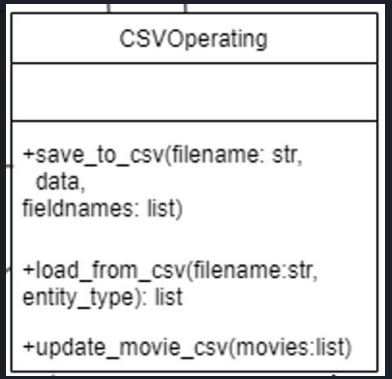
Class介紹—CSVOperating

```python
# 定義 CSVOperating 類別
class CSVOperating:
    @staticmethod
    def save_to_csv(filename, data, fieldnames):
        with open(filename, mode='a', newline='', encoding='utf-8') as file:
            writer = csv.DictWriter(file, fieldnames=fieldnames)
            if file.tell() == 0:
                writer.writeheader()

            if isinstance(data, (User, Admin, Movie, PaymentInfo)):
                writer.writerow(data.__dict__)
            elif isinstance(data, Ticket):
                writer.writerow({
                    'user': data.get_user().__dict__,
                    'movie': data.get_movie().__dict__,
                    'seat_number': data.get_seat_number(),
                    'payment_info': data.get_payment_info().__dict__
                })
            elif isinstance(data, dict):
                writer.writerow(data)
            else:
                print(f"不支援的類型: {type(data)}")
```



Class介紹—CSVOperating

```python
@staticmethod
def load_from_csv(filename, entity_type):
    data = []
    try:
        with open(filename, mode='r', encoding='utf-8') as file:
            reader = csv.DictReader(file)
            for row in reader:
                if entity_type == User and 'username' in row and 'password' in row:
                    data.append(User(row['username'], row['password']))

                elif entity_type == Admin and 'username' in row and 'password' in row:
                    data.append(Admin(row['username'], row['password']))

                elif entity_type == Movie and all(key in row for key in ['title', 'director', 'release_date']):
                    data.append(Movie(row['title'], row['director'], row['release_date']))

                elif entity_type == PaymentInfo and all(key in row for key in ['card_number', 'expiration_date', 'cvv']):
                    data.append(PaymentInfo(row['card_number'], row['expiration_date'], row['cvv']))

                elif entity_type == Ticket and all(key in row for key in ['user', 'movie', 'seat_number', 'payment_info']):
                    user_data, movie_data, payment_info_data = map(eval, [row['user'], row['movie'], row['payment_info']])
                    user_obj = User(user_data['username'], user_data['password'])
                    movie_obj = Movie(movie_data['title'], movie_data['director'], movie_data['release_date'])
                    payment_info_obj = PaymentInfo(payment_info_data['card_number'],
                    payment_info_data['expiration_date'], payment_info_data['cvv'])
                    ticket_obj = Ticket(user_obj, movie_obj, int(row['seat_number']), payment_info_obj)
                    data.append(ticket_obj)

    except FileNotFoundError:
        print(f"檔案 {filename} 不存在。")
    except Exception as e:
        print(f"讀取檔案時發生錯誤：{e}")
    return data
```

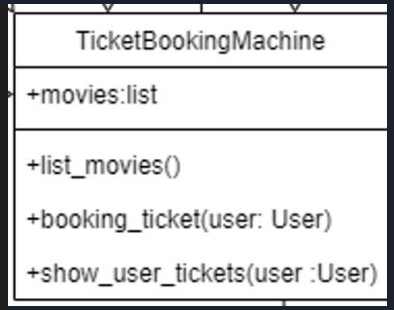


Class介紹—CSVOperating

```python
@staticmethod
def update_movie_csv(movies):
    with open("movies.csv", mode='w', newline='', encoding='utf-8') as file:
        writer = csv.DictWriter(file, fieldnames=["title", "director", "release_date"])
        writer.writeheader()
        for movie in movies:
            writer.writerow(movie.__dict__)
```

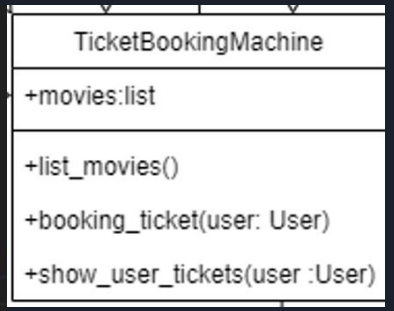
Class介紹—TicketBookingMachine

```python
# 定義 TicketBookingMachine 類別
class TicketBookingMachine:
    movies = CSVOperating.load_from_csv("movies.csv", Movie)

    @staticmethod
    def list_movies():
        print("目前上映的電影：")
        for i, movie in enumerate(TicketBookingMachine.movies, start=1):
            print(f"{i}. {movie.get_title()}\t[導演:{movie.get_director()}，上映日期:{movie.get_release_date()}]")

    @staticmethod
    def book_ticket(user):
        try:
            TicketBookingMachine.list_movies()
            selected_movie_index = int(input("請選擇欲訂票的電影編號：")) - 1
            selected_movie = TicketBookingMachine.movies[selected_movie_index]
            seat_number = int(input("請選擇座位號碼(1~200,1列20個座位)："))

            card_number = str(input("請輸入信用卡卡號(16碼)："))
            expiration_date = str(input("請輸入信用卡到期日(MM/YY)："))
            cvv = str(input("請輸入ccv(3碼)："))

            payment_info = PaymentInfo(card_number, expiration_date, cvv)
            ticket = Ticket(user, selected_movie, seat_number, payment_info)

            CSVOperating.save_to_csv("tickets.csv", ticket, fieldnames=["user", "movie", "seat_number", "payment_info"])
            print("付款資訊儲存成功")

        except (ValueError, IndexError):
            print("輸入無效，請重新選擇。")
```



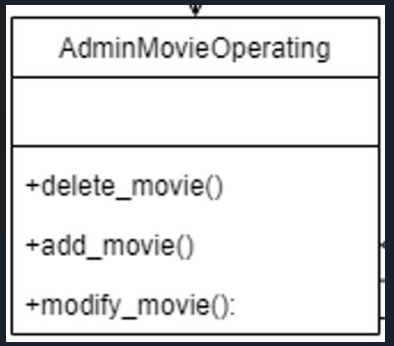
Class介紹—TicketBookingMachine

```python
@staticmethod
def show_user_tickets(user):
    tickets = CSVOperating.load_from_csv("tickets.csv", Ticket)
    user_tickets = [ticket for ticket in tickets if ticket.get_user().get_name() == user.get_name()]

    if user_tickets:
        print(f"{user.get_name()} 的訂票資訊：")
        for i, ticket in enumerate(user_tickets, start=1):
            print(f"{i}. {ticket.get_movie().get_title()} ({ticket.get_movie().get_release_date()})
                  座位號: {ticket.get_seat_number()}")
    else:
        print(f"{user.get_name()} 目前沒有訂票紀錄。")
```



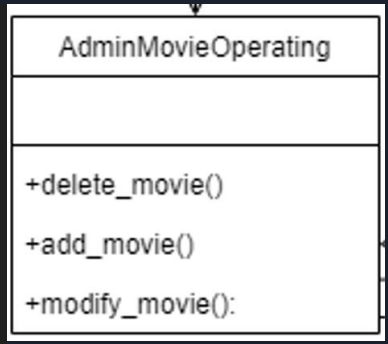
Class介紹—AdminMovieOperating

```python
# 定義 AdminMovieOperating 類別
class AdminMovieOperating:
    @staticmethod
    def add_movie():
        new_movie = Movie(input("請輸入電影名稱："), input("請輸入導演："), input("請輸入上映日期(YYYY-MM-DD)："))
        TicketBookingMachine.movies.append(new_movie)
        CSVOperating.update_movie_csv(TicketBookingMachine.movies)
        print(f"電影:{new_movie.get_title()}新增成功")

    @staticmethod
    def delete_movie():
        TicketBookingMachine.list_movies()
        try:
            movie_index = int(input("請選擇欲刪除的電影編號：")) - 1
            deleted_movie = TicketBookingMachine.movies.pop(movie_index)

            # 更新 movies.csv 檔案
            CSVOperating.update_movie_csv(TicketBookingMachine.movies)
            print(f"電影 {deleted_movie.get_title()} 已刪除。")

        except (ValueError, IndexError):
            print("輸入無效，請重新選擇。")
```



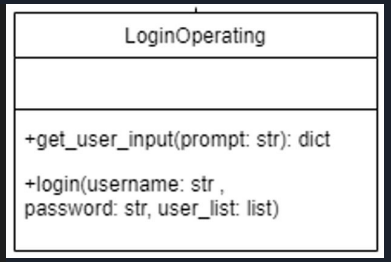
Class介紹—AdminMovieOperating

```python
@staticmethod
def modify_movie():
    TicketBookingMachine.list_movies()
    try:
        movie_index = int(input("請選擇欲修改的電影編號：")) - 1
        selected_movie = TicketBookingMachine.movies[movie_index]

        # 輸入新的電影資訊
        print("若無需修改，直接換行即可")
        new_title = input(f"請輸入新的電影名稱（原名：{selected_movie.get_title()}）：")
        new_director = input(f"請輸入新的導演（原導演：{selected_movie.get_director()}）：")
        new_release_date = input(f"請輸入新的上映日期（原日期：{selected_movie.get_release_date()}）：")

        # 更新電影資訊
        new_title = new_title if new_title else selected_movie.get_title()
        new_director = new_director if new_director else selected_movie.get_director()
        new_release_date = new_release_date if new_release_date else selected_movie.get_release_date()

        selected_movie.update_movie_info(new_title, new_director, new_release_date)

        # 更新 movies.csv 檔案
        CSVOperating.update_movie_csv(TicketBookingMachine.movies)
        print(f"電影 {selected_movie.get_title()} 已修改。")

    except (ValueError, IndexError):
        print("輸入無效，請重新選擇。")
```



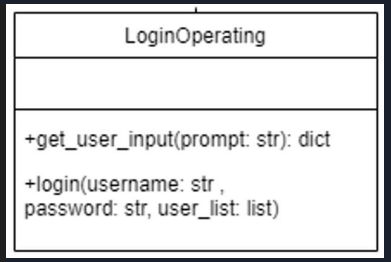
Class介紹—LoginOperating

```python
# 定義 LoginOperating 類別
class LoginOperating:
    #檢查輸入是否為str
    @staticmethod
    def get_user_input(prompt, input_type=str):
        while True:
            try:
                user_input = input(prompt)
                return input_type(user_input)
            except ValueError:
                print("輸入無效，請重新輸入。")

    @staticmethod
    def login(username, password, user_list):
        for user_dict in user_list:
            if user_dict.get_name() == username and user_dict.get_password() == password:
                return user_dict
        return None
```



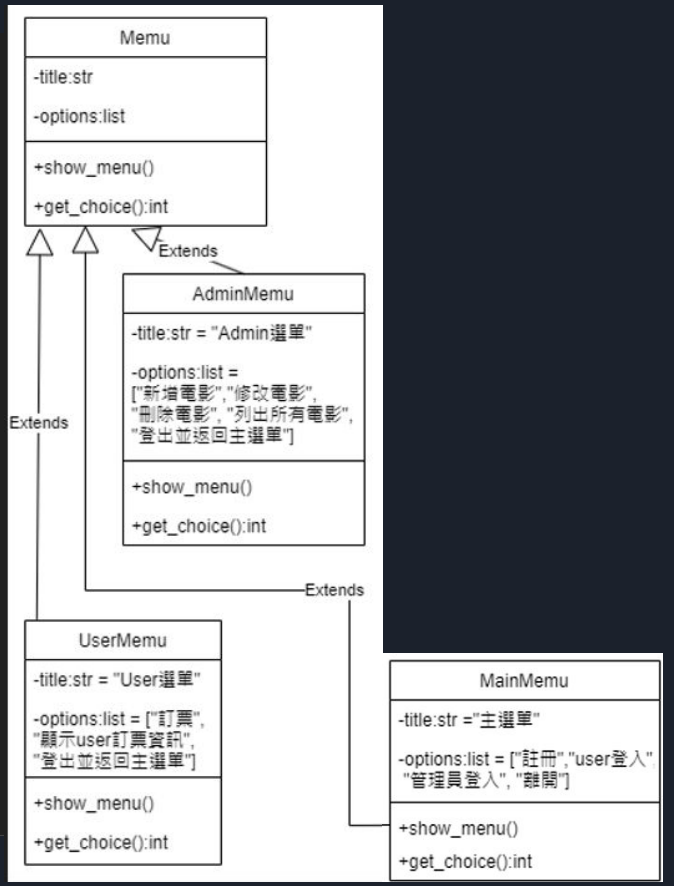
Class介紹—LoginOperating

```python
# 定義 LoginOperating 類別
class LoginOperating:
    #檢查輸入是否為str
    @staticmethod
    def get_user_input(prompt, input_type=str):
        while True:
            try:
                user_input = input(prompt)
                return input_type(user_input)
            except ValueError:
                print("輸入無效，請重新輸入。")

    @staticmethod
    def login(username, password, user_list):
        for user_dict in user_list:
            if user_dict.get_name() == username and user_dict.get_password() == password:
                return user_dict
        return None
```



Class介紹—Menu

```python
class Menu:
    def __init__(self, title, options):
        self.__title = title
        self.__options = options

    def show_menu(self):
        print(f"\n{self.__title}:")
        for i, option in enumerate(self.__options, start=1):
            print(f"{i}. {option}")

    def get_choice(self):
        while True:
            try:
                choice = int(input("請輸入操作編號："))
                if 1 <= choice <= len(self.__options):
                    return choice
                else:
                    print("請輸入有效的選項編號。")
            except ValueError:
                print("請輸入有效的數字。")

class MainMenu(Menu):
    def __init__(self):
        super().__init__("主選單", ["註冊", "user登入", "管理員登入", "離開"])

class UserMenu(Menu):
    def __init__(self):
        super().__init__("User選單", ["訂票", "顯示user訂票資訊", "登出並返回主選單"])

class AdminMenu(Menu):
    def __init__(self):
        super().__init__("Admin選單", ["新增電影", "修改電影", "刪除電影", "列出所有電影", "登出並返回主選單"])
```



Classe間的關係

- ‘’is a’’(Implementation)的關係在本系統中較多是在操作時才會呈現

- 舉例來說:當使用者註冊時，輸入完帳號密碼，會在程式中建立一個new_user的物件，此時new_user ‘’is a’’ User

Classe間的關係

- ‘’part of’’ 關係 :
  - User與TicketBookingMachine
  - Movie與TicketBookingMachine
  - User與Ticket
  - Movie與Ticket
  - PaymentInfo與Ticket

- 皆為Aggregation而無composition, 因為他們之間的生命周期皆互相獨立。

Class Diagram