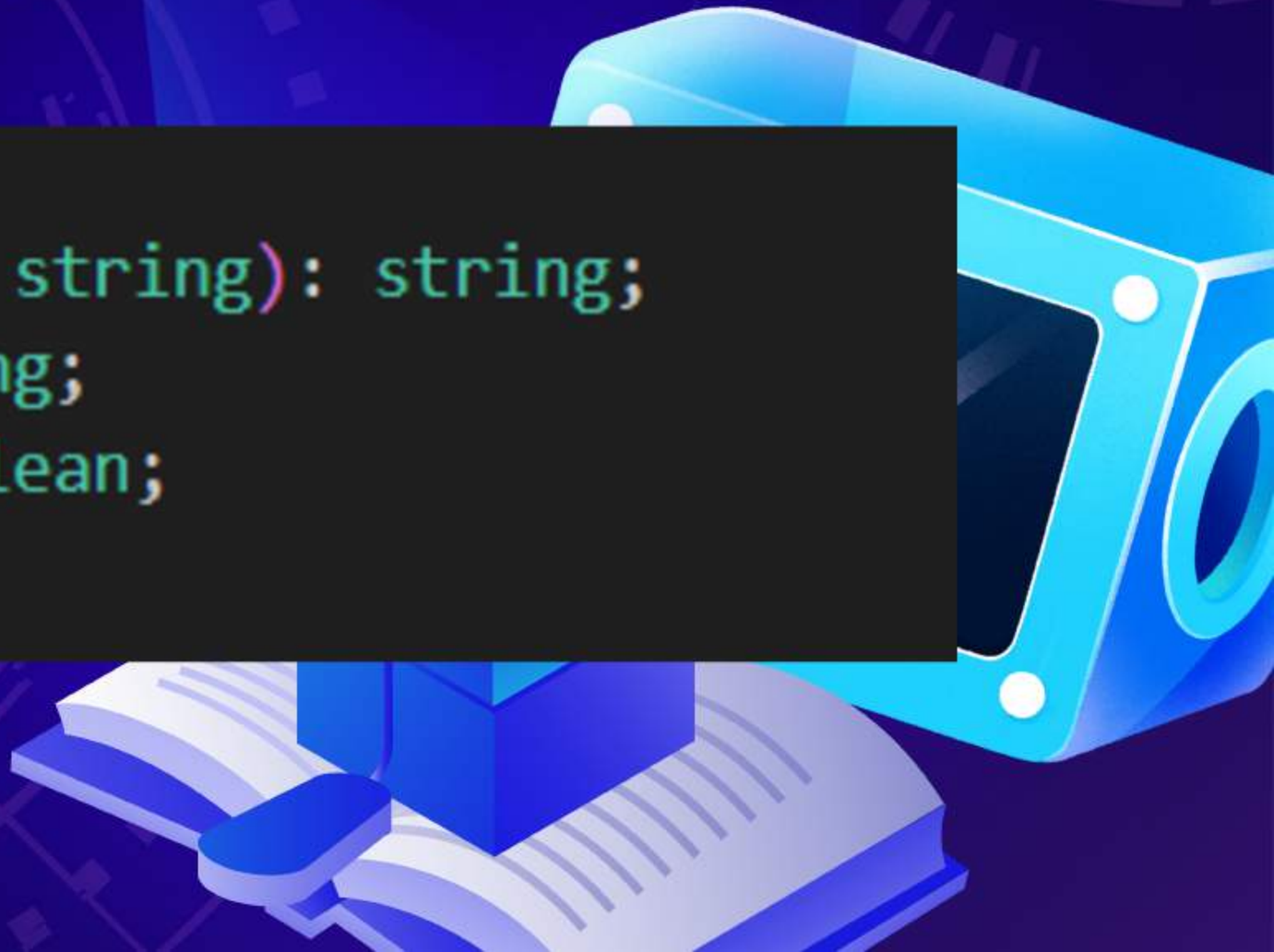
OOP_TEST

กลุ่ม ตะวันฉาย
67162110414-4 นายธนธรณ์ เม่ากลาง ไกค์ *18*
67162110530-9 พีรพัฒน์ มันกระโทก เชน *24*

1.Interface: Borrowable

Interface คือสัญญา *(Contract)* ว่าหากมีคลาสใดๆก็ตามที่ *implement* ไปต้องมีฟังก์ชันของ *Interface* ไปด้วย เช่น *borrow()* คือฟังก์ชันยืมรายการ ส่วน*returnItem()* ก็คือฟังก์ชันคืนรายการ และ*isAvailable()* คือการตรวจสอบว่ารายการพร้อมให้ยืมหรือไม่ ประโยชน์คือการทำให้แน่ใจว่าทุก *LibraryItem* มีฟังก์ชันยืมและฟังก์ชันคืน

```
interface Borrowable {
    borrow(memberName: string): string;
    returnItem(): string;
    isAvailable(): boolean;
}
```



2.Abstract Class: LibraryItem

Abstract class คือคลาสที่ไม่สามารถสร้าง *object* ตรงๆได้จำเป็นต้องมี *subclass* ที่*implement* และเก็บคุณสมบัติพื้นฐานอย่างเช่น *title, itemId, available* และก็มีฟังก์ชันทั่วไปคือ *borrow()* คือการยืมถ้าขึ้น *available = false*แล้วคืนข้อความว่าไม่สามารถยืมได้ ถ้า*returnItem()* แล้วคืนรายการและเปลี่ยนสถานะ *available isAvailable()* แล้วตรวจสอบว่าสถานะ *(getDetails())* ถ้าเป็น *abstract function* ต้องมีการ*implement* ไปใน*subclass* ประโยชน์คือการลดการซ้ำของโค้ดสำหรับหนังสือทุกประเภทในรายการของห้องสมุด

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
get title(): string {
    return this._title;
}

Windsurf: Refactor | Explain | Generate JSDoc | ×
get itemId(): string {
    return this._itemId;
}

Windsurf: Refactor | Explain | Generate JSDoc | ×
set available(status: boolean) {
    this._available = status;
}
```

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
isAvailable(): boolean {
    return this._available;
}

Windsurf: Refactor | Explain | Generate JSDoc | ×
borrow(memberName: string): string {
    if (!this._available) {
        return `Item not available`;
    }
    this._available = false;
    return `${this._title} borrowed by ${memberName}`;
}

returnItem(): string {
    this._available = true;
    return `${this._title} returned`;
}

abstract getDetails(): string;
```

3.Subclasses: Book, Magazine, DVD, Newspaper, Thesis

Subclass จะเพิ่มคุณสมบัติเฉพาะของแต่ละประเภท เช่น *Book* คือ *author*, *Magazine* คือ *issueDate*, *DVD* คือ *duration*, *Newspaper* คือ *date*, *Thesis* คือ *researcher* แล้วก็*Implement getDetails()* มาใช้สำหรับการแสดงข้อมูลรายการ ประโยชน์คือสามารถสร้างได้หลายประเภทรายการที่แตกต่างกันแต่ก็ยังใช้โค้ดร่วมกันได้

ตัวอย่าง

```
// ---------- Subclasses ----------
Windsurf: Refactor | Explain
class Book extends LibraryItem {
    private _author: string;

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    constructor(title: string, itemId: string, author: string) {
        super(title, itemId);
        this._author = author;
    }

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    getDetails(): string {
        return `Book: ${this.title} by ${this._author} (ID: ${this.itemId})`;
    }
}
```

4.LibraryMember

เป็นที่เก็บข้อมูลของสมาชิกของห้องสมุด นอกจากนี้ยังทำหน้าที่เก็บรายการของการยืมและการคืนของหนังสือของสามาชิก *borrowedItems* กับ *returnedItems* โดยจะมีฟังก์ชันสำคัญคือ *borrowItem()* คือยืม *item* และเพิ่มในรายการ *borrowedItems* ส่วนการ *returnItem()* คือการคืน *item* และก็ลบออกจากรายการ *borrowedItems* และไปเพิ่มใน *returnedItems*แทน ใน *listBorrowedItems()* และ *listReturnedItems()* มีประโยชน์ในการติดตามการยืมและคืนหนังสือของสมาชิกแต่ละคน

```
class LibraryMember {
    private _memberName: string;
    private _memberId: string;
    private _borrowedItems: LibraryItem[];

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    constructor(memberName: string, memberId: string) {
        this._memberName = memberName;
        this._memberId = memberId;
        this._borrowedItems = [];
    }

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    get memberName(): string {
        return this._memberName;
    }

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    get memberId(): string {
        return this._memberId;
    }
```

```
borrowItem(item: LibraryItem): string {
    const msg = item.borrow(this._memberName);
    if (msg !== "Item not available") {
        this._borrowedItems.push(item);
    }
    return msg;
}

Windsurf: Refactor | Explain | Generate JSDoc | ×
returnItem(itemId: string): string {
    const index = this._borrowedItems.findIndex(i => i.itemId === itemId);
    if (index !== -1) {
        const item = this._borrowedItems[index];
        this._borrowedItems.splice(index, 1);
        return item.returnItem();
    }
    return `Item not found in borrowed list`;
}
```

5.Library

เราใช้ *Library* เป็นคลาสหลักในการใช้เพิ่มรายการและใช้เพิ่มสมาชิก *(addItem, addMember)* และใช้ในการยืมและคืนหนังสือ *(borrowItem, returnItem)* ฟังก์ชันช่วยสรุปใช้ในการสรุปว่ามีใครที่ยืมหนังสือและคืนหนังสือบ้าง*(getLibrarySummary)*โดยจะแสดงผล รายการทั้งหมด สมาชิก รายการที่สมาชิกยืมไป รายการที่สมาชิกคืนไป ประโยชน์ของ *Library* คือการเป็นตัวกลางที่คอยควบคุมระบบทั้งหมด ทำให้โค้ดง่ายต่อการขยายเพิ่ม

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
addItem(item: LibraryItem): void {
    this.items.push(item);
}

Windsurf: Refactor | Explain | Generate JSDoc | ×
addMember(member: LibraryMember): void {
    this.members.push(member);
}
```

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
borrowItem(memberId: string, itemId: string): string {
    const member = this.members.find(m => m.memberId === memberId);
    const item = this.items.find(i => i.itemId === itemId);
    if (member && item) {
        return member.borrowItem(item);
    }
    return "Member or Item not found";
}

Windsurf: Refactor | Explain | Generate JSDoc | ×
returnItem(memberId: string, itemId: string): string {
    const member = this.members.find(m => m.memberId === memberId);
    if (member) {
        return member.returnItem(itemId);
    }
    return "Member not found";
}
```

6.ตัวอย่างการใช้งาน

สร้าง *new Library* โดยการสร้างสมาชิกและรายการใหม่เพิ่มลงใน *Library* สมาชิกยืมรายการสถานะจะเปลี่ยนเป็นไม่ *available* และถ้าสมาชิกคืนรายการสถานะจะเปลี่ยนเป็น *available*

```
const lib = new Library();
const member = new LibraryMember("Alice", "M1001");
const book = new Book("The Hobbit", "B001", "Tolkien");

lib.addItem(book);
lib.addMember(member);
console.log(lib.borrowItem("M1001", "B001")); // ยืม
console.log(lib.returnItem("M1001", "B001")); // คืน
```

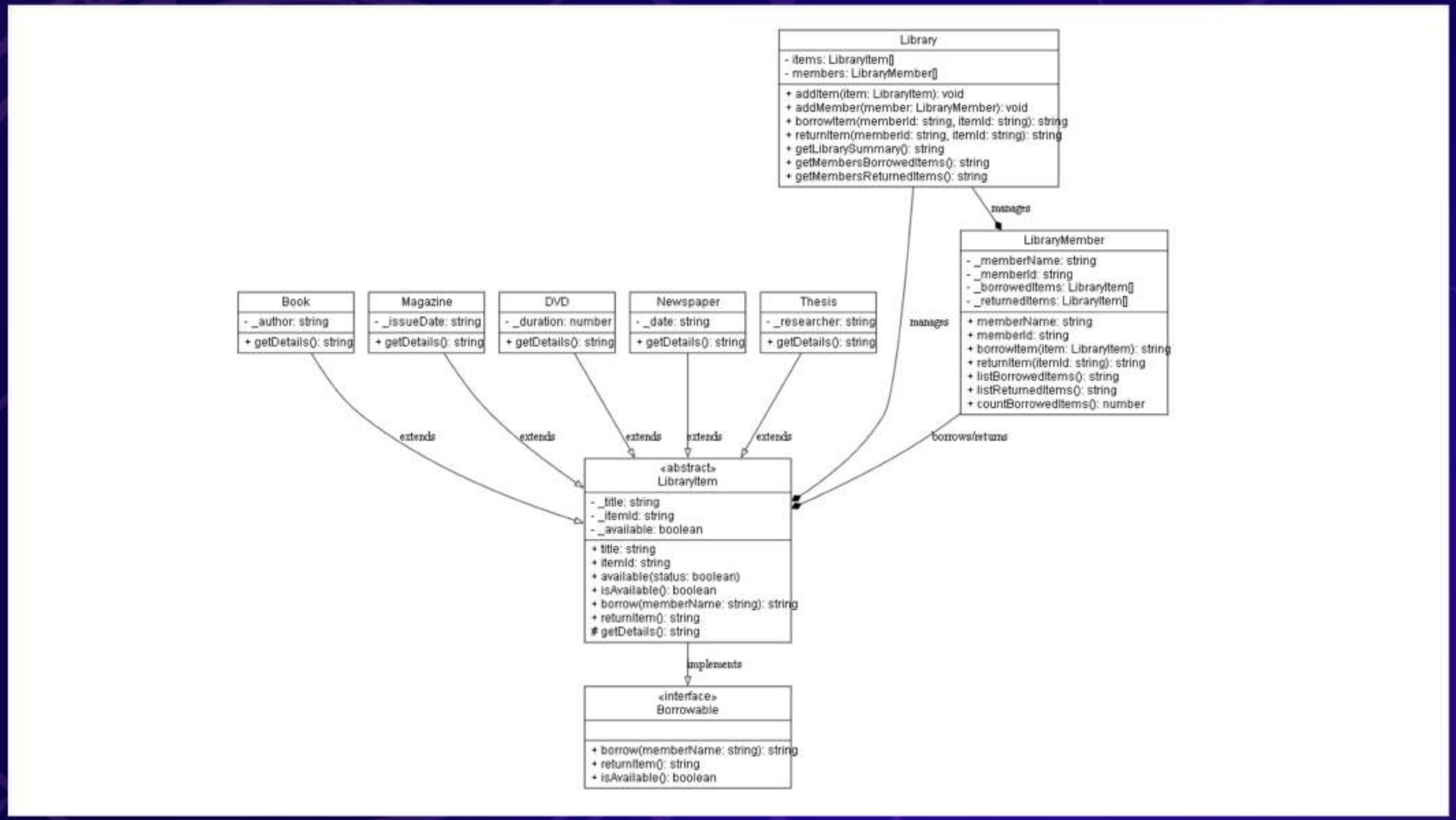
7.สรุป

เราใช้ *OOP(Interface, Abstract Class, Inheritance, Encapsulation,Aggregation,Composition)* ทำให้รองรับได้หลายประเภทรายการ เช่น *Book, Magazine, DVD, Newspaper, Thesis* เป็นต้น ส่วนใน *LibraryMember* คือการติดตามรายการที่ยืมและคืน และ*Library* เป็นตัวกลางในการควบคุมและสรุปข้อมูลทั้งหมดของโค้ด ซึ่งทำให้ในอนาคตสามารถขยายระบบง่าย

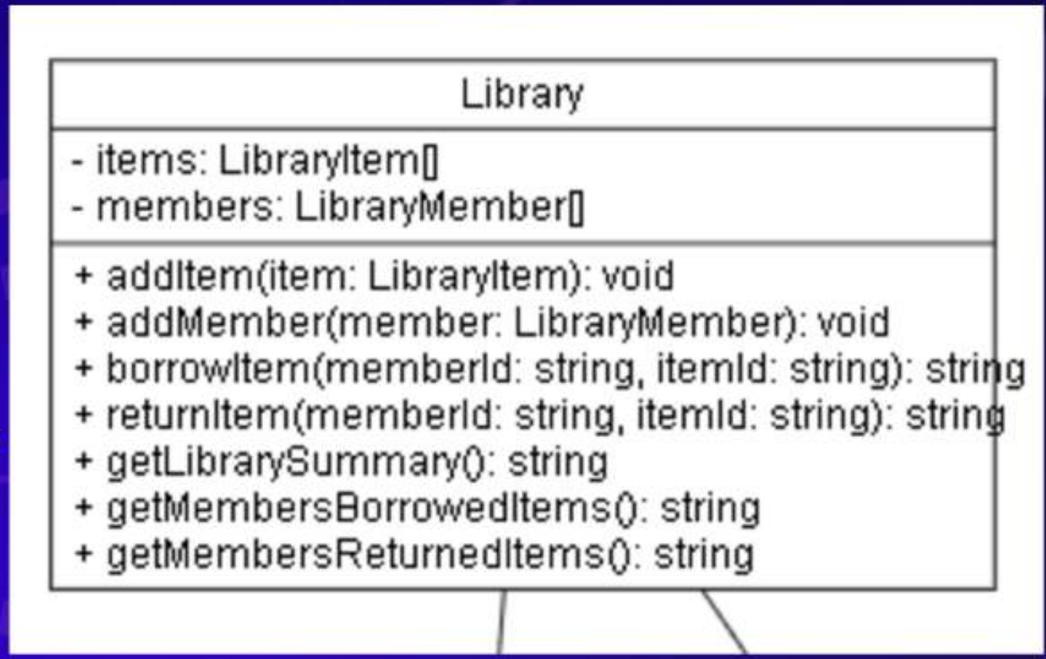
```
const lib = new Library();
const member = new LibraryMember("Alice", "M1001");
const book = new Book("The Hobbit", "B001", "Tolkien");

lib.addItem(book);
lib.addMember(member);
console.log(lib.borrowItem("M1001", "B001")); // ยืม
console.log(lib.returnItem("M1001", "B001")); // คืน
```

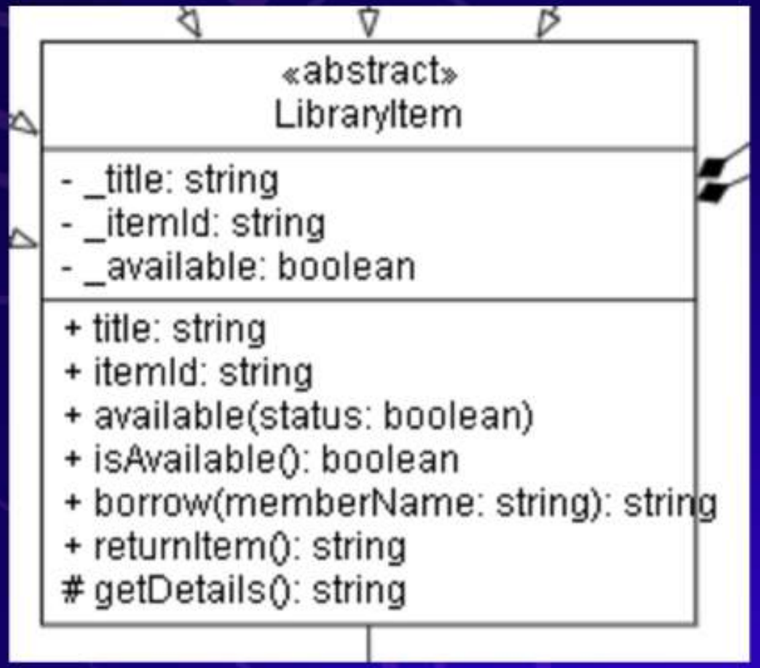
Class Diagram



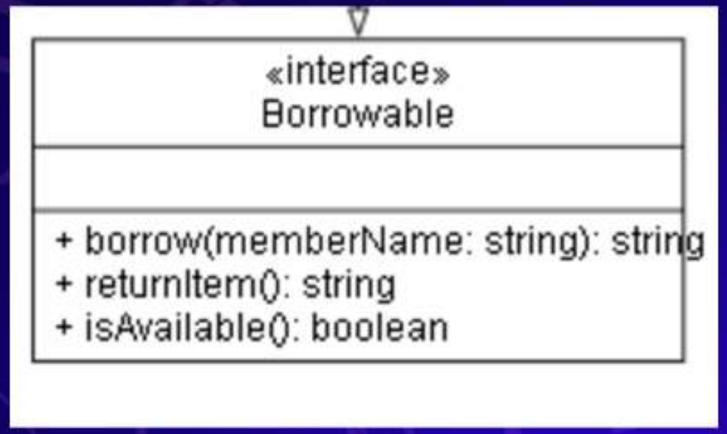
1. Library (ห้องสมุด)
คลาสหลักที่จัดการระบบห้องสมุด
คุณสมบัติ:
items: LibraryItem[] – รายการสื่อทั้งหมด
members: LibraryMember[] – รายการสมาชิก
เมธอด:
addItem(item: LibraryItem): เพิ่มสื่อ
addMember(member: LibraryMember): เพิ่มสมาชิก
borrowItem(memberId: string, itemId: string): ให้สมาชิกยืมสื่อ
returnItem(memberId: string, itemId: string): ให้สมาชิกคืนสื่อ
getLibrarySummary(): สรุปข้อมูลห้องสมุด
getMembersBorrowedItems(): ดูสื่อที่สมาชิกยืม
getMembersReturnedItems(): ดูสื่อที่สมาชิกคืน



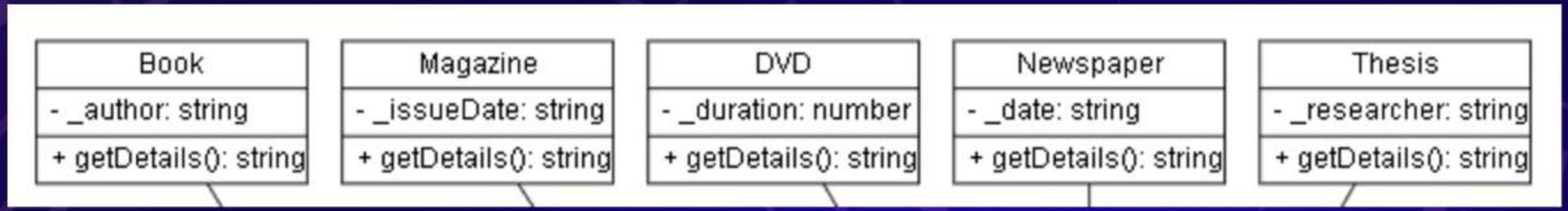
Library manages LibraryMember → ห้องสมุดจัดการสมาชิก
Library manages LibraryItem → ห้องสมุดจัดการสื่อ
LibraryMember borrows/returns LibraryItem → สมาชิกยืมและคืนสื่อ



2. LibraryItem (คลาสแบบนามธรรม - Abstract Class)
เป็น คลาสแม่ (Superclass) ที่ถูกสืบทอดโดยคลาสต่าง ๆ ของสื่อในห้องสมุด
มีคุณสมบัติ (Attributes):
_title: string – ชื่อของสื่อ
_itemId: string – รหัสประจำสื่อ
_available: boolean – สถานะว่าสามารถยืมได้หรือไม่
มีเมธอด (Methods):
title, itemId: getter
available(status: boolean): เปลี่ยนสถานะการใช้งาน
isAvailable(): ตรวจสอบว่ามีอยู่ไหม
borrow(memberName: string): ยืมสื่อ
returnItem(): คืนสื่อ
#getDetails(): เมธอดปิดกั้น (protected) สำหรับแสดงรายละเอียดเฉพาะ



3. Borrowable (Interface)
เป็น อินเทอร์เฟซ ที่กำหนดพฤติกรรมที่ "สามารถยืมได้"
ประกอบด้วยเมธอด:
borrow(memberName: string): string
returnItem(): string
isAvailable(): boolean



4. คลาสประเภทสื่อ (Concrete Classes ที่สืบทอดจาก LibraryItem)
แต่ละคลาสนี้เป็น สื่อประเภทต่าง ๆ ในห้องสมุด และสืบทอดจาก LibraryItem พร้อมเพิ่มข้อมูลเฉพาะตัว

5. LibraryMember (สมาชิกห้องสมุด)
คลาสที่เก็บข้อมูลสมาชิกห้องสมุด
คุณสมบัติ:
_memberName: string
_memberId: string
_borrowedItems: LibraryItem[] – สื่อที่ยืมอยู่
_returnedItems: LibraryItem[] – สื่อที่คืนแล้ว
เมธอด:
borrowItem(item: LibraryItem): ยืมสื่อ
returnItem(itemId: string): คืนสื่อ
listBorrowedItems(): แสดงรายการสื่อที่ยืม
listReturnedItems(): แสดงรายการสื่อที่คืน
countBorrowedItems(): นับจำนวนสื่อที่ยืม


Thank You
Embracing the Future Together