

OOP TEST

SIMPLE LIBRARY SYSTEM



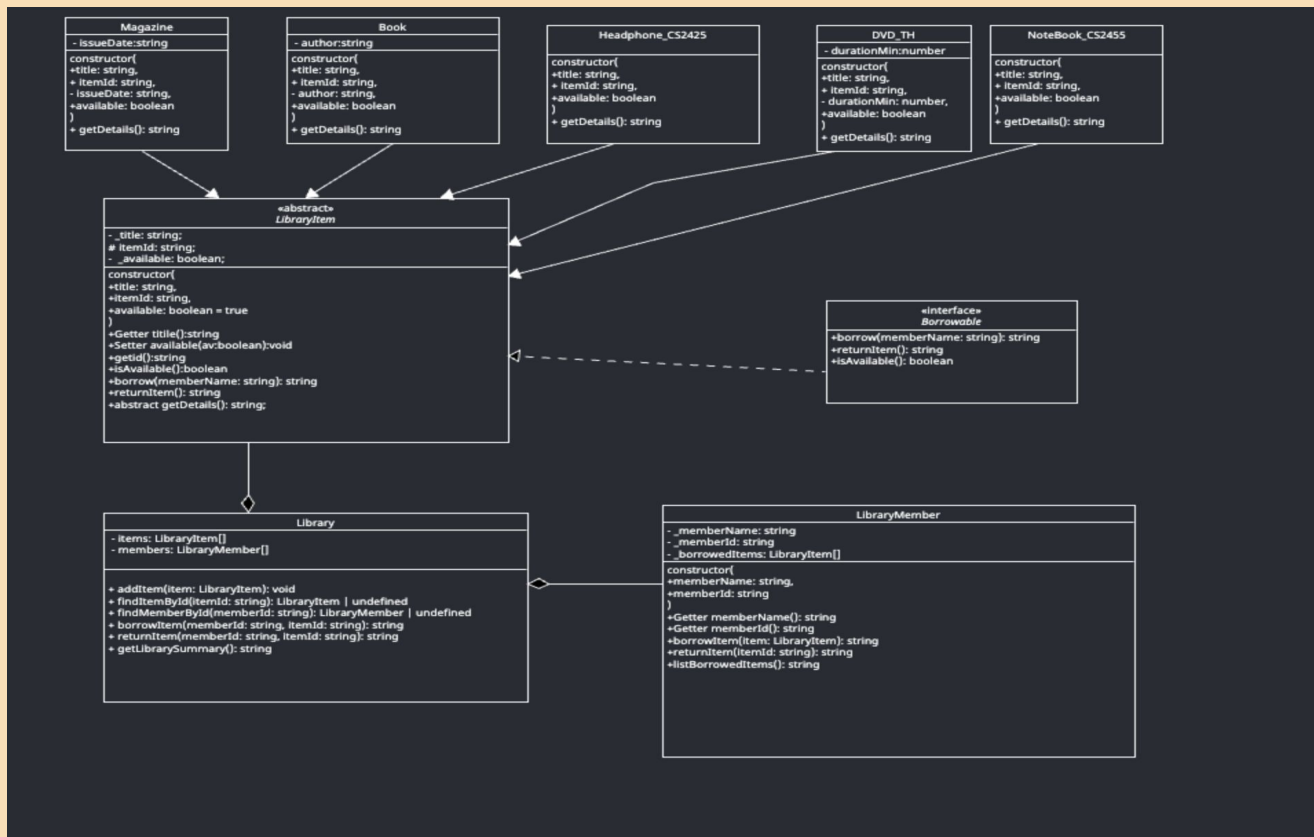
FEATURE



- ยืม-คืนหนังสือ, แมกกาซีน, DVD, Notebook, หูฟัง ฯลฯ
- มีสมาชิก (Library Member) ที่สามารถยืม/คืนได้
- Library เก็บข้อมูลรายการ (Library Item) และสมาชิก
- มีการตรวจสอบว่าของยังสามารถยืมได้อยู่หรือไม่

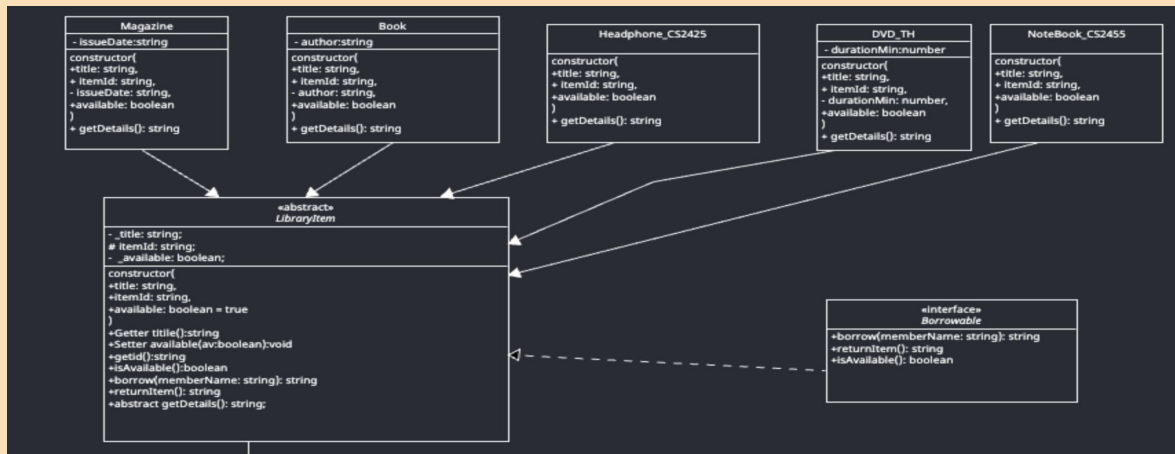


CLASS DIAGRAM





CLASS DIAGRAM



Inheritance (สืบทอด):

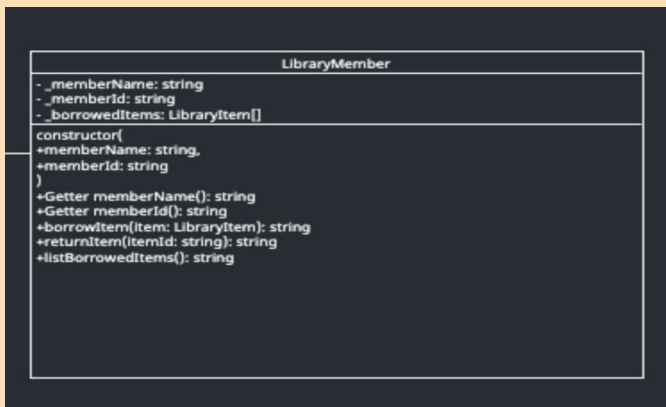
- Book, Magazine, DVD_TH, NoteBook_CS2455, Headphone_CS2425 → สืบทอดจาก LibraryItem

Interface:

- Borrowable (มี method: borrow(), returnItem(), isAvailable())



CLASS DIAGRAM

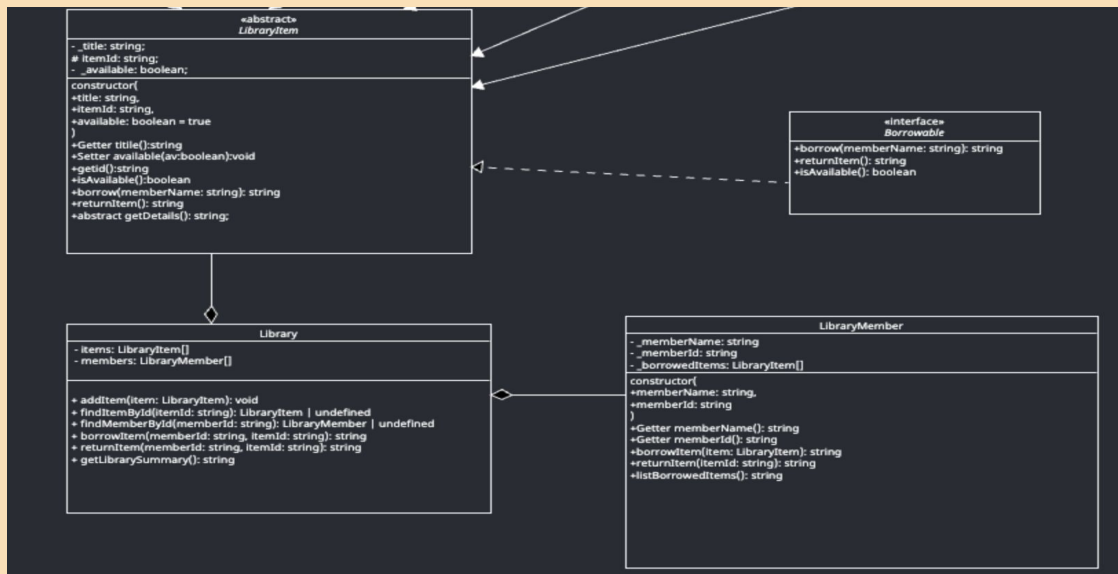


Encapsulation :

- ใช้ private/protected field เช่น `_title`, `_available`, `_memberId`
- มี getter/setter (`getTitle()`, `setAvailable()`)



CLASS DIAGRAM



Library มี `items: LibraryItem[]` และ `members: LibraryMember[]`



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



กำหนดพฤติกรรมการยืมสิ่งของ



```
export abstract class LibraryItem implements Borrowable {  
    // encapsulated attributes  
    private _title: string;  
    protected itemId: string;  
    private _available: boolean;  
  
    constructor(title: string, itemId: string, available: boolean = true) {  
        this._title = title;  
        this.itemId = itemId;  
        this._available = available;  
    }  
  
    // getter for title  
    public get title(): string {  
        return this._title;  
    }  
  
    // getter for id  
    public getId(): string {  
        return this.itemId;  
    }  
  
    // setter for available (could include extra validation)  
    public set available(av: boolean) {  
        this._available = av;  
    }  
}
```

```
// implement borrow(memberName)  
public borrow(memberName: string): string {  
    if (!this._available) {  
        return `Item ${this.getId()} ("${this._title}") is not available`;  
    }  
    this._available = false;  
    return `${this.getDetails()} borrowed by ${memberName}`;  
}  
  
// implement returnItem()  
public returnItem(): string {  
    if (this._available) {  
        return `${this.getDetails()} was not borrowed`;  
    }  
    this._available = true;  
    return `${this.getDetails()} returned`;  
}  
  
// subclass must override to give specific details  
public abstract getDetails(): string;
```




Sub Class



```
export class Book extends LibraryItem {
  private author: string;

  constructor(title: string, itemId: string, author: string, available: boolean = true) {
    super(title, itemId, available);
    this.author = author;
  }

  public getDetails(): string {
    return `Book: ${this.title} by ${this.author} (ID: ${this.getId()})`;
  }
}

export class Magazine extends LibraryItem {
  private issueDate: string;

  constructor(title: string, itemId: string, issueDate: string, available: boolean = true) {
    super(title, itemId, available);
    this.issueDate = issueDate;
  }

  public getDetails(): string {
    return `Magazine: ${this.title} (Issue: ${this.issueDate}) (ID: ${this.getId()})`;
  }
}
```



Sub Class



```
export class DVD_TH extends LibraryItem {
  private durationMin: number;

  constructor(title: string, itemId: string, durationMin: number, available: boolean = true) {
    super(title, itemId, available);
    this.durationMin = durationMin;
  }

  public getDetails(): string {
    return `DVD: ${this.title} (${this.durationMin} min) (ID: ${this.getId()})`;
  }
}

export class NoteBook_CS2455 extends LibraryItem {


  constructor(title: string, itemId: string, available: boolean = true) {
    super(title, itemId, available);
  }

  public getDetails(): string {
    return `NoteBook: ${this.title} (ID: ${this.getId()})`;
  }
}

export class Headphone_CS2425 extends LibraryItem {

  constructor(title: string, itemId: string, available: boolean = true) {
    super(title, itemId, available);
  }

  public getDetails(): string {
    return `HeadPhone: ${this.title}(ID: ${this.getId()})`;
  }
}
```





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

  constructor(memberName: string, memberId: string) {
    this._memberName = memberName;
    this._memberId = memberId;
  }

  public get memberName(): string {
    return this._memberName;
  }

  public get memberId(): string {
    return this._memberId;
  }

  // borrowItem: attempt to borrow a LibraryItem
  // return message string (per spec)
  public borrowItem(item: LibraryItem): string {
    // validate availability
    if (!item.isAvailable()) {
      return 'Item not available';
    }

    // Ask item to borrow (item will update its state)
    const msg = item.borrow(this._memberName);

    // If borrow succeeded (item becomes unavailable), add to member's borrowed list
    if (!item.isAvailable()) {
      this._borrowedItems.push(item);
    }

    return msg;
  }
}
```

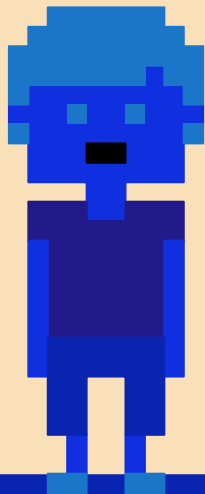
```
// returnItem by itemId
public returnItem(itemId: string): string {
  const idx = this._borrowedItems.findIndex(it => it.getId() === itemId);
  if (idx === -1) {
    return `Member ${this._memberName} does not have item ${itemId}`;
  }
  const item = this._borrowedItems[idx];
  const msg = item.returnItem();

  // remove from borrowed list if returned successfully (i.e., item.isAvailable() === true)
  if (item.isAvailable()) {
    this._borrowedItems.splice(idx, 1);
  }
  return msg;
}

// listBorrowedItems returns a string (can be used in UI or slide)
public listBorrowedItems(): string {
  if (this._borrowedItems.length === 0) return `${this._memberName} has no borrowed items`;
  const lines = this._borrowedItems.map(it => it.getDetails());
  return `${this._memberName} borrowed:\n- ${lines.join("\n- ")}`;
}
```



TEST!



```
// test library and member
const lib = new Library();
lib.addItem(new Book("TypeScript Guide", "B001", "John Doe"));
lib.addItem(new Magazine("Tech Monthly", "M001", "2023-09"));
lib.addItem(new DVD_TH("Interstellar", "D001", 169));
lib.addItem(new Headphone_CS2425("Interstellar", "H001"));
lib.addItem(new Headphone_CS2425("HeadPhone.com/bommerrangthailand", "H002"));

const userName = prompt("Enter your name:");
let member = lib.findMemberById(userName!);
if (!member && userName) {
    alert(`Welcome new member, ${userName}!`);
    member = new LibraryMember(userName, userName);
    lib.addMember(member);
}
```

```
let exit = false;
while (!exit) {
    const choice = prompt(
        "==== Library Menu =====\n" +
        "1. ตรวจสอบของที่มีทั้งหมด\n" +
        "2. ยืมหนังสือ\n" +
        "3. คืนหนังสือ\n" +
        "4. ออกจากระบบ\n" +
        "เลือกเมนู (1-4):"
    );

    switch (choice) {
        case "1":
            alert(`📖 รายการหนังสือทั้งหมด: \n` + lib.getLibrarySummary());
            break;

        case "2":
            const borrowId = prompt("ใส่รหัสหนังสือที่ต้องการยืม:");
            if (borrowId) {
                alert(lib.borrowItem(member!.memberId, borrowId));
            }
            break;

        case "3":
            const returnId = prompt("ใส่รหัสหนังสือที่ต้องการคืน:");
            if (returnId) {
                alert(lib.returnItem(member!.memberId, returnId));
            }
            break;

        case "4":
            alert("ออกจากระบบแล้ว 🏠");
            exit = true;
            break;

        default:
            alert("❌ เลือกเมนูไม่ถูกต้อง");
    }
}
```