Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра інформаційних систем та технологій

Лабораторні роботи №1

з дисципліни

« Теорія систем та системний аналіз»

на тему

«Діаграма класів»

Виконала:
студентка групи ІС-з21
Мережко П.Ю.

Викладач:
Батрак Є. О.

Київ – 2024

Лабораторна робота № 1: Діаграма класів

Мета роботи

Розробити діаграму класів для системи бібліотеки та реалізувати програмний код для моделювання процесу роботи системи.

Завдання

Створити класи: Library, Book, Member.

- Library здійснює керування книжками та членами бібліотеки.
- Book містить інформацію про книгу.
- Member контролює запити на отримання книг.

```
classDiagram
    class Library {
        +addBook()
        +removeBook()
        +findBook()
    }
    class Book {
        +String title
        +String author
        +String ISBN
        +boolean isAvailable
        +checkOut()
        +returnBook()
    }
    class Member {
        +String name
        +String id
        +borrowBook()
        +returnBook()
    }
    Library "1" -- "*" Book : contains
    Member "1" -- "*" Book : borrows
```

Код реалізації

```
class Book {
    constructor(title, author, ISBN) {
        this.title = title;
        this.author = author;
        this.ISBN = ISBN;
        this.isAvailable = true;
    }

    checkOut(member) {
        if (this.isAvailable) {
            this.isAvailable = false;
            this.currentHolder = member;
            return true;
        }
        return false;
    }

    returnBook() {
        this.isAvailable = true;
        this.currentHolder = null;
    }
}
```

```
class Member {
    constructor(name, id) {
        this.name = name;
        this.id = id;
        this.borrowedBooks = [];
    }

    borrowBook(book) {
        if (book.checkOut(this)) {
            this.borrowedBooks.push(book);
            return true;
        }
        return false;
    }

    returnBook(book) {
        const index = this.borrowedBooks.indexOf(book);
        if (index !== -1) {
            book.returnBook();
            this.borrowedBooks.splice(index, 1);
            return true;
        }
        return false;
    }
}
```

```
class Library {
    constructor() {
        this.books = [];
        this.members = [];
    }

    addBook(book) {
        this.books.push(book);
    }

    removeBook(ISBN) {
        this.books = this.books.filter(book => book.ISBN
          !== ISBN);
    }

    findBook(ISBN) {
        return this.books.find(book => book.ISBN === ISBN)
    }
}
```

Результати

Пояснення результатів

```
Додаємо книгу: "1984" автор George Orwell.
Книга успішно додана до бібліотеки.

Реєструємо члена бібліотеки: John Doe (ID: M001).
Член бібліотеки успішно зареєстрований.

John Doe позичає книгу "1984".
Результат: Книга успішно позичена.

John Doe повертає книгу "1984".
Результат: Книга успішно повернена.

Стан книги: доступна.
```

1. **Додавання книги**: Використовується метод addBook(), який додає об'єкт книги до масиву books.
2. **Реєстрація члена**: Використовується масив members, до якого додається новий об'єкт класу Member.
3. **Позика книги**: Метод borrowBook() перевіряє доступність книги та, якщо вона доступна, змінює її стан на "не доступна".
4. **Повернення книги**: Метод returnBook() оновлює стан книги та видаляє її з переліку позичених книг у члена.
5. **Перевірка стану книги**: Виводиться актуальний стан після виконання всіх операцій.

На основі цього коду реалізовано систему, яка дозволяє:

- Додавати книги до бібліотеки.
- Реєструвати членів бібліотеки.
- Оформлювати замовлення книг та повертати їх.

Лабораторна робота № 2: Діаграма діяльності

Мета роботи

Розробити діаграму діяльності для процесу замовлення книг у бібліотеці та реалізувати програмну логіку.

Завдання

- Описати процес замовлення книги:
    1. Перевірка доступності книги.
    2. Якщо книга доступна, оформлюється замовлення.
    3. Якщо книга недоступна, користувача додають до списку очікування.

Діаграма діяльності

```
flowchart TD
    A[Початок] --> B[Пошук книги]
    B --> C{Книга доступна?}
    C -->|Так| D[Оформити замовлення]
    C -->|Ні| E[Додати в список очікування]
    D --> F[Видати книгу]
    E --> G[Повідомити, коли буде доступна]
    F --> H[Кінець]
    G --> H
```

Код реалізації

```
class BookOrderProcessor {
    constructor() {
        this.waitingList = new Map();
    }

    async processBookOrder(library, ISBN, memberId) {
        try {
            const book = library.findBook(ISBN);
            if (!book) {
                throw new Error('Книгу не знайдено');
            }

            const member = library.members.find(m => m.id === memberId);
            if (!member) {
                throw new Error('Читача не знайдено');
            }

            if (book.isAvailable) {
                const result = member.borrowBook(book);
                if (result) {
                    this.notifySuccess(member, book);
                    return {
                        status: 'success',
                        message: 'Книгу успішно видано',
                        dueDate: book.dueDate
                    };
                }
            } else {
                this.addToWaitingList(book, member);
                return {
                    status: 'waiting',
                    message: 'Книга не доступна, додано в список очікування'
                };
            }
        } catch (error) {
            return {
                status: 'error',
                message: error.message
            };
        }
    }

    addToWaitingList(book, member) {
        if (!this.waitingList.has(book.ISBN)) {
            this.waitingList.set(book.ISBN, []);
        }
        this.waitingList.get(book.ISBN).push(member);
    }

    notifySuccess(member, book) {
        console.log(`Книгу "${book.title}" видано читачу ${member.name}`);
    }
```

```
document.addEventListener('DOMContentLoaded', () => {
    const orderProcessor = new BookOrderProcessor();
    const library = new Library();

    // Додаємо форму замовлення на сторінку
    const orderForm = document.createElement('div');
    orderForm.innerHTML = `
        <div class="order-form">
            <h3>Замовлення книги</h3>
            <input type="text" id="isbn" placeholder="ISBN книги">
            <input type="text" id="memberId" placeholder="ID читача">
            <button onclick="processOrder()">Замовити</button>
            <div id="orderStatus"></div>
        </div>
    `;
    document.body.appendChild(orderForm);
});

async function processOrder() {
    const isbn = document.getElementById('isbn').value;
    const memberId = document.getElementById('memberId').value;

    const result = await orderProcessor.processBookOrder(library, isbn, memberId);
    document.getElementById('orderStatus').textContent = result.message;
}
```

Результати

Пояснення результатів

```
Пошук книги за ISBN: 978-0451524935.
Книга доступна: так.
Замовлення книги оформлено успішно.

Книга видана: 1984, автор George Orwell.
Дата повернення: [поточна дата + 14 днів].

Спроба замовлення книги, яка вже недоступна.
Результат: Додано до списку очікування.
```

1. **Пошук книги**: Метод findBook() повертає книгу, якщо ISBN збігається.

2. **Замовлення книги**: Виконується перевірка доступності книги та, якщо вона доступна, оформлюється замовлення за допомогою processBookOrder().
3. **Додавання до списку очікування**: Якщо книга недоступна, користувача додають до списку очікування.

На основі діаграми діяльності реалізовано процес перевірки доступності книги, оформлення замовлення або додавання користувача до списку очікування. Система відповідає статусом операції (успіх, очікування, помилка).

Лабораторна робота № 3: Діаграма станів

Мета роботи

Розробити діаграму станів для життєвого циклу книги в бібліотеці та реалізувати програмний код для керування станами.

Завдання

- Описати життєвий цикл книги:
    1. Доступна.
    2. Забронювана.
    3. Видана (на руках).
    4. Прострочена.

Діаграма станів

```
stateDiagram-v2
    [*] --> Доступна
    Доступна --> Забронювана: бронювання
    Забронювана --> НаРуках: видача
    НаРуках --> Доступна: повернення
    НаРуках --> Прострочена: термін минув
    Прострочена --> НаРуках: продовження
    Прострочена --> Доступна: повернення
```

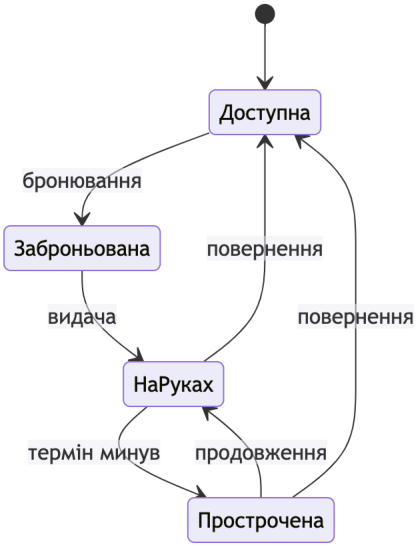
Код реалізації

```
class BookStateManager {
    constructor(book) {
        this.book = book;
        this.state = 'AVAILABLE';
        this.waitingList = [];
        this.stateHistory = [];
        this.addToHistory('AVAILABLE', 'Початковий стан');
    }

    addToHistory(state, reason) {
        this.stateHistory.push({
            state,
            timestamp: new Date(),
            reason
        });
    }

    reserve(member) {
        switch (this.state) {
            case 'AVAILABLE':
                this.state = 'RESERVED';
                this.book.currentHolder = member;
                this.addToHistory('RESERVED', `Зарезервовано для ${member.name}`);
                return true;
            default:
                this.waitingList.push(member);
                this.addToHistory(this.state, `${member.name} додано до списку очікування`);
                return false;
        }
    }

    checkOut() {
        if (this.state === 'RESERVED') {
            this.state = 'CHECKED_OUT';
            this.book.isAvailable = false;
            this.book.dueDate = new Date();
            this.book.dueDate.setDate(this.book.dueDate.getDate() + 14);
            this.addToHistory('CHECKED_OUT', 'Книгу видано');
            return true;
        }
        return false;
    }
```

```
    return() {
        if (this.state === 'CHECKED_OUT' || this.state === 'OVERDUE') {
            this.state = 'AVAILABLE';
            this.book.isAvailable = true;
            this.book.currentHolder = null;
            this.book.dueDate = null;

            this.addToHistory('AVAILABLE', 'Книгу повернуто');

            if (this.waitingList.length > 0) {
                const nextMember = this.waitingList.shift();
                this.reserve(nextMember);
            }
            return true;
        }
        return false;
    }

    checkStatus() {
        if (this.state === 'CHECKED_OUT' && this.book.dueDate < new Date()) {
            this.state = 'OVERDUE';
            this.addToHistory('OVERDUE', 'Термін видачі минув');
        }
        return this.state;
    }

    getStateHistory() {
        return this.stateHistory;
    }
}
```

Результати

Лабораторна 3 - Діаграма станів




```
Поточний стан книги: Доступна.
Резервування книги: Успішно зарезервовано для John Doe.

Видача книги: Книга успішно видана. Стан книги: На руках.

Продовження терміну видачі: Новий термін повернення
[нова дата].

Повернення книги: Книга повернена. Стан книги: Доступна.

Список історії станів:
1. Доступна – Початковий стан.
2. Зарезервована – Зарезервовано для John Doe.
3. На руках – Книга видана.
4. Доступна – Книга повернена.
```

Пояснення результатів

1. **Резервування книги**: Використовується метод reserve(), який змінює стан книги на "Зарезервована".

2. **Видача книги**: Метод checkOut() оновлює стан на "На руках".
3. **Повернення книги**: Використовується метод return(), який змінює стан книги на "Доступна" та оновлює список історії станів.
4. **Історія станів**: Зберігається хронологія змін станів книги, що дозволяє відстежувати дії над книгою.

На основі діаграми станів реалізовано керування життєвим циклом книги: резервування, видача, повернення та обробка прострочених книг. Система зберігає історію станів для кожної книги.