**Кросплатформенне програмування**

## **Лабораторна робота №1**

«Розробка міні-проекту »

*Виконав Заяць Ілля, група ІПЗ-41*

**Мета:**

Розробити міні-проект із використанням ООП та основних методів із бібліотек Котліну.

Хід роботи

1.)- Створити проект, що імітує просту систему керування бібліотекою, де користувачі (читачі) можуть шукати, позичати та повертати книги. Система має керувати інвентаризацією книг у бібліотеці, відстежувати, які книги взято, і забезпечувати виконання дат повернення. Створити звіт.

Посилання на гіт-хаб репозиторій:  
<https://github.com/db-user/kpLR1>

## *Завдання*

## 1.)- Пояснення структури і роботи, опис домену.

***!!!Оскільки об’єм роботи дуже великий, то в текст були додані заголовки, за допомогою яких можна переміщуватись по документу!!!***

Проект «Бібліотека» імітує систему керування бібліотекою, яка дозволяє бібліотекарям і читачам виконувати основні операції, пов’язані з книгою, читачами і позичанням книг. У цій системі є кілька ключових учасників та сутностей, кожен з яких виконує певні функції в межах бібліотеки.

1. Ключові учасники:

*Читачі (Member)* — це люди, які можуть брати книги на час, поки вони не будуть повернуті. Читачі мають свої унікальні ідентифікатори, ім'я та ліміт на кількість книг, які вони можуть позичити одночасно. Кожен читач може позичити обмежену кількість книг, після чого йому буде заборонено брати більше до повернення хоча б однієї.

*Бібліотекар (Librarian)* — це адміністративна роль, відповідальна за керування бібліотечним інвентарем і користувачами. Бібліотекар може додавати нові книги в бібліотеку, видаляти старі або застарілі книги, а також реєструвати нових читачів. Це основна роль для управління бібліотечною системою на вищому рівні.

2. Ключові сутності та їх функції:

*Книга (Book)* — це основний елемент бібліотечної системи. Кожна книга має:

*Назву (title) –* назва книги.

*Автора (author) –* автор книги.

*ISBN (isbn) –* унікальний ідентифікатор для кожної книги.

*Доступність (available) –* вказує, чи доступна книга для позичання.

*Позичальник (borrower) –* користувач бібліотеки, який позичив цю книгу (якщо вона позичена).

Книга може бути позичена тільки тоді, коли вона доступна, і коли позичальник має змогу позичити ще книги згідно зі своїм лімітом. Коли книга повертається, її статус змінюється на "доступна".

*Читач бібліотеки (Member)* — це особа, яка має доступ до бібліотечної системи для позичання книг. Читачі мають:

*Ім'я (name) –* ім'я читача.

*ID (memberId) –* унікальний ідентифікатор читача.

*Список позичених книг (borrowedBooks) –*  список книг, які читач позичив.

*Максимальна кількість книг (maxBooks) –* ліміт на кількість книг, які може позичити кожен читач одночасно.

Читачі можуть позичати книгу, якщо вона доступна і якщо вони ще не перевищили свій ліміт на кількість книг.

*Бібліотека (Library)* — є центральною сутністю, яка організовує всі книги та читачів. Вона зберігає список книг та читачів, а також надає функції для пошуку книг, додавання або видалення книг, реєстрації або скасування реєстрації читачів, а також для позичання та повернення книг.

3. Основні операції:

*Додавання книг* — бібліотекар додає нові книги до інвентарю бібліотеки. Це важливе завдання для підтримки актуальності бібліотечного фонду.

*Видалення книг* — коли книга стає застарілою, пошкодженою або більше не потрібною, бібліотекар може видаляти її з інвентарю бібліотеки.

*Реєстрація та скасування реєстрації читачів* — бібліотекар може реєструвати нових користувачів (читачів), а також видаляти їх з реєстру, якщо вони більше не є читачами.

*Пошук книг* — читачі та бібліотекар можуть здійснювати пошук книг за назвою, автором або ISBN. Це допомагає швидко знаходити необхідні ресурси в бібліотеці.

*Позичання та повернення книг* — читачі можуть позичати книги, якщо вони доступні та у них є дозвіл на позичання. Книга позичається на певний термін, і після повернення вона знову стає доступною для інших читачів.

4. Підрахунок та статистика:

Бібліотека може надавати статистичні дані, такі як загальна кількість книг у бібліотеці, кількість доступних книг, кількість книг за автором або за назвою. Це дозволяє бібліотекарям та читачам мати загальне уявлення про стан бібліотечного фонду.

5. Інтерфейси взаємодії:

Читачі (Members) взаємодіють з системою через можливість позичати та повертати книги.

Бібліотекарі (Librarians) мають більший доступ до функцій бібліотеки, таких як додавання або видалення книг, реєстрація нових членів і ведення статистики.

Система (Library) виступає як центральний координатор, що забезпечує всю логіку операцій, перевірки та стеження за станом бібліотечного фонду.

## 2.)- Блок-схема ієрархії класів

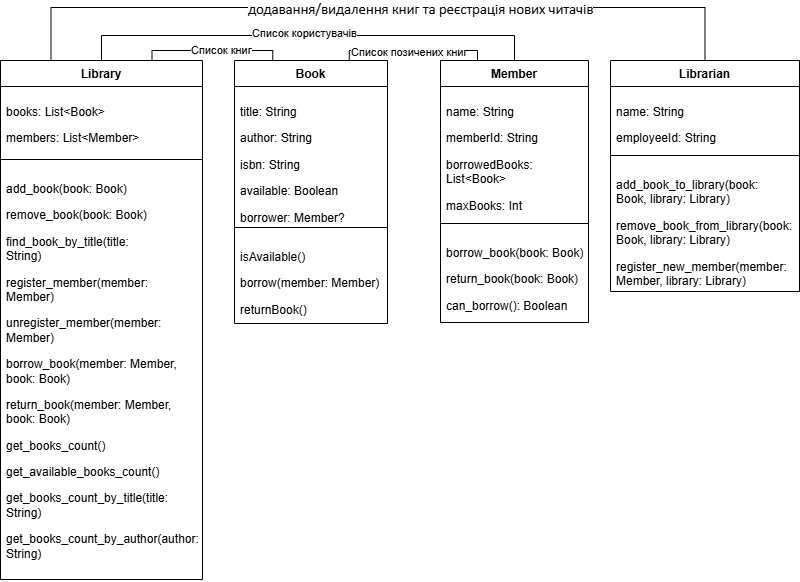


Рисунок 1 - UML діаграма класів

***Опис UML діаграми класів:***

**Library (Бібліотека)**

Атрибути:

books: List<Book>

members: List<Member>

Методи:

add\_book(book: Book)

remove\_book(book: Book)

find\_book\_by\_title(title: String)

register\_member(member: Member)

unregister\_member(member: Member)

borrow\_book(member: Member, book: Book)

return\_book(member: Member, book: Book)

get\_books\_count()

get\_available\_books\_count()

get\_books\_count\_by\_title(title: String)

get\_books\_count\_by\_author(author: String)

**Book (Книга)**

Атрибути:

title: String

author: String

isbn: String

available: Boolean

borrower: Member?

Методи:

isAvailable()

borrow(member: Member)

returnBook()

**Member (Член бібліотеки)**

Атрибути:

name: String

memberId: String

borrowedBooks: List<Book>

maxBooks: Int

Методи:

borrow\_book(book: Book)

return\_book(book: Book)

can\_borrow(): Boolean

**Librarian (Бібліотекар)**

Атрибути:

name: String

employeeId: String

Методи:

add\_book\_to\_library(book: Book, library: Library)

remove\_book\_from\_library(book: Book, library: Library)

register\_new\_member(member: Member, library: Library)

**Взаємозв'язки між класами:**

Library має асоціації з Book та Member (вона містить списки книг і членів бібліотеки).

Member має асоціацію з Book через список позичених книг. Кожен член може позичити кілька книг, але має ліміт на їх кількість.

Librarian має асоціацію з Library, адже бібліотекар взаємодіє з бібліотекою для додавання/видалення книг та реєстрації нових членів.

**Ієрархія:**

Library містить багато Book і багато Member.

Member може позичати багато Book.

Librarian управляє Library.

## 3.)- Код:

data class Book(

val title: String,

val author: String,

val isbn: String,

var available: Boolean = true,

var borrower: Member? = null

) {

fun isAvailable(): Boolean = available

fun borrow(member: Member) {

if (available) {

available = false

borrower = member

member.borrow\_book(this)

} else {

println("Book is not available.")

}

}

fun returnBook() {

available = true

borrower?.return\_book(this)

borrower = null

}

}

class Member(

val name: String,

val memberId: String,

val maxBooks: Int = 3

) {

val borrowedBooks: MutableList<Book> = mutableListOf()

fun borrow\_book(book: Book) {

if (can\_borrow()) {

borrowedBooks.add(book)

} else {

println("$name can't borrow more books.")

}

}

fun return\_book(book: Book) {

borrowedBooks.remove(book)

}

fun can\_borrow(): Boolean = borrowedBooks.size < maxBooks

}

class Library {

val books: MutableList<Book> = mutableListOf()

val members: MutableList<Member> = mutableListOf()

fun add\_book(book: Book) {

books.add(book)

}

fun remove\_book(book: Book) {

books.remove(book)

}

fun find\_book\_by\_title(title: String): Book? {

return books.find { it.title == title }

}

fun register\_member(member: Member) {

members.add(member)

}

fun unregister\_member(member: Member) {

members.remove(member)

}

fun borrow\_book(member: Member, book: Book) {

if (book.isAvailable() && member.can\_borrow()) {

book.borrow(member)

} else {

println("Cannot borrow book.")

}

}

fun return\_book(member: Member, book: Book) {

book.returnBook()

}

fun get\_books\_count(): Int = books.size

fun get\_available\_books\_count(): Int = books.count { it.isAvailable() }

fun get\_books\_count\_by\_title(title: String): Int {

return books.count { it.title == title }

}

fun get\_books\_count\_by\_author(author: String): Int {

return books.count { it.author == author }

}

}

class Librarian(

val name: String,

val employeeId: String

) {

fun add\_book\_to\_library(book: Book, library: Library) {

library.add\_book(book)

}

fun remove\_book\_from\_library(book: Book, library: Library) {

library.remove\_book(book)

}

fun register\_new\_member(member: Member, library: Library) {

library.register\_member(member)

}

}

fun main() {

val library = Library()

val librarian = Librarian("John Doe", "L123")

val member = Member("Alice", "M001")

librarian.register\_new\_member(member, library)

val book1 = Book("Kotlin Programming", "JetBrains", "123-456-789")

val book2 = Book("Advanced Kotlin", "JetBrains", "987-654-321")

librarian.add\_book\_to\_library(book1, library)

librarian.add\_book\_to\_library(book2, library)

println("Total books in library: ${library.get\_books\_count()}")

println("Available books: ${library.get\_available\_books\_count()}")

librarian.add\_book\_to\_library(book1, library)

println("Total books by JetBrains: ${library.get\_books\_count\_by\_author("JetBrains")}")

// Alice borrows a book

library.borrow\_book(member, book1)

println("Books borrowed by Alice: ${member.borrowedBooks.size}")

println("Is 'Kotlin Programming' available? ${book1.isAvailable()}")

// Alice returns the book

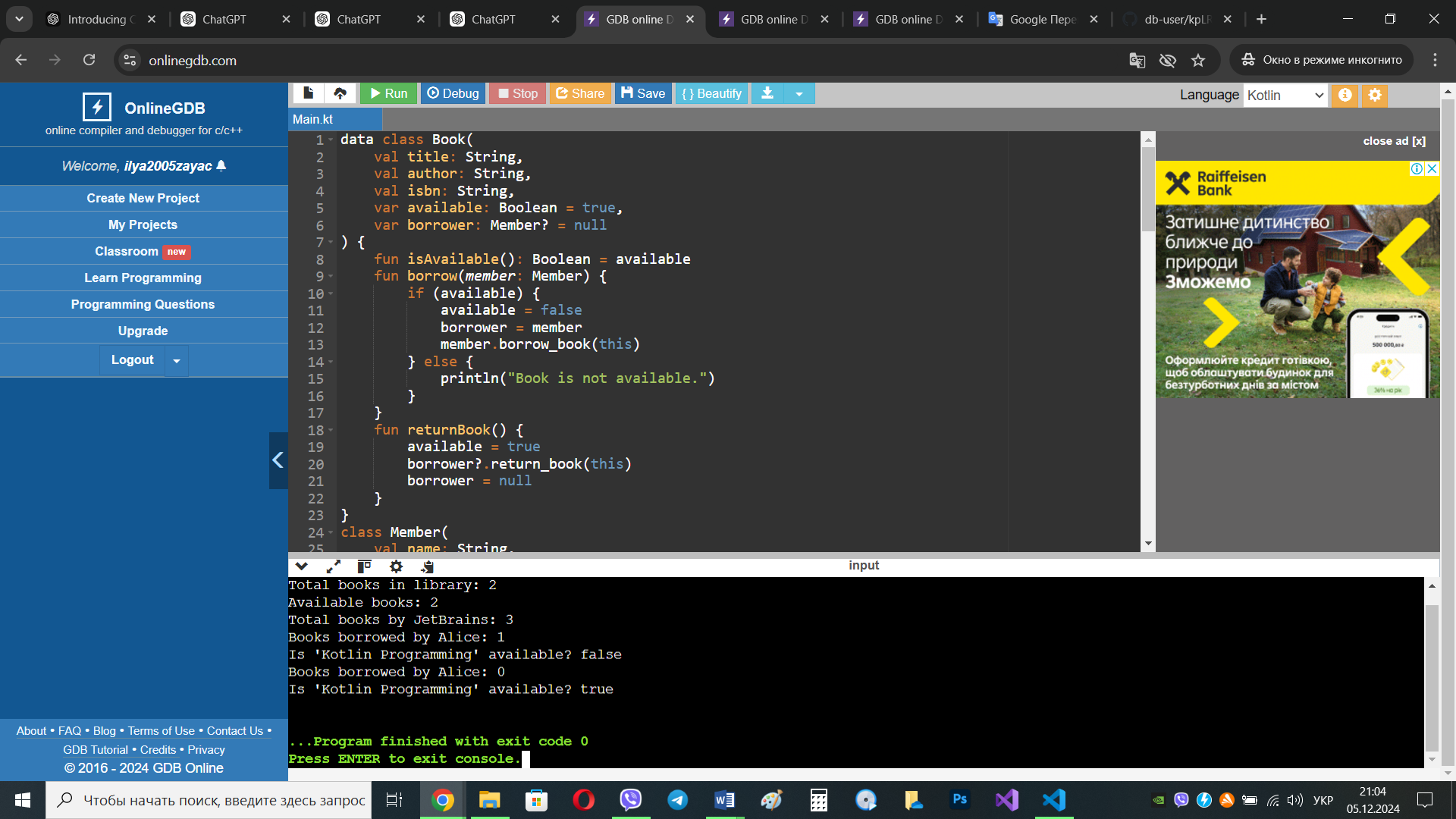
library.return\_book(member, book1)

println("Books borrowed by Alice: ${member.borrowedBooks.size}")

println("Is 'Kotlin Programming' available? ${book1.isAvailable()}")

}

## 3.)- Приклад виконання (з онлайн-компілятора onlinegdb):



## **Висновок:**

Проект "Бібліотека" — це проста система для управління книгами та членами бібліотеки, яка реалізує основні операції, пов’язані з позичанням, поверненням і статистикою книг. Вона включає кілька типів учасників: читачів бібліотеки, бібліотекарів і саму бібліотеку як центральну точку для взаємодії. Система допомагає організувати та автоматизувати ці процеси для покращення користування бібліотечними ресурсами.