**Курс:** Шаблоны проектирования приложений

**Тема:** Модуль 11 Диаграммы вариантов использования. Диаграмма классов

*Баллы:*

**Цель работы:**

Разработка системы управления библиотекой с учетом принципов объектно-ориентированного программирования и проектирования, с использованием диаграммы классов и последующей реализации на языке C#.

**Описание задания:**

Необходимо спроектировать и реализовать систему управления библиотекой, которая включает следующие основные функции:

1. Управление пользователями библиотеки (читатели, библиотекари).
2. Управление книгами (добавление, редактирование, удаление, поиск книг).
3. Управление операциями выдачи и возврата книг.
4. Отчеты по использованию библиотеки (популярность книг, активность читателей и т.д.).

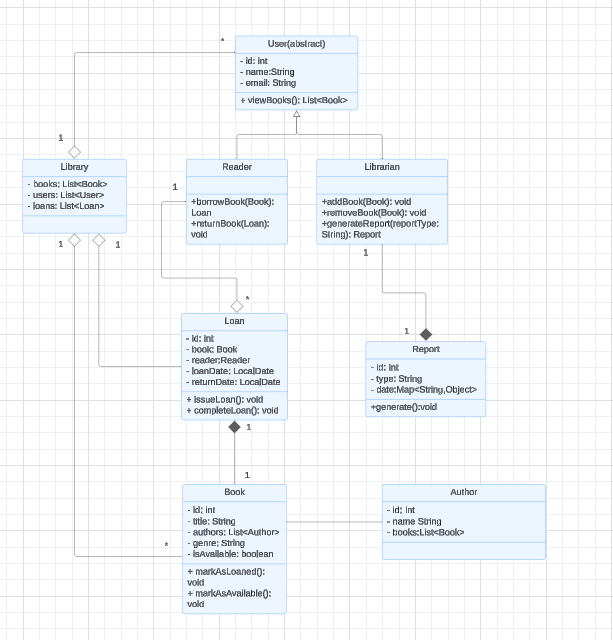
**Этапы выполнения работы:**

**1. Проектирование диаграммы классов:**

Создайте диаграмму классов, отражающую основные сущности и их взаимосвязи. Основные сущности:

* **Book** (Книга)
* **User** (Пользователь)
  + **Reader** (Читатель)
  + **Librarian** (Библиотекарь)
* **Loan** (Выдача книги)
* **Library** (Библиотека)
* **Author** (Автор)
* **Report** (Отчеты)

Каждый класс должен содержать необходимые свойства и методы для выполнения функций управления библиотекой.

**2. Реализация классов на языке C#:**

import java.util.List;

public class Book {

private int id;

private String title;

private List<Author> authors;

private String genre;

private boolean isAvailable;

public Book(int id, String title, List<Author> authors, String genre) {

this.id = id;

this.title = title;

this.authors = authors;

this.genre = genre;

this.isAvailable = true;

}

public void markAsLoaned() {

this.isAvailable = false;

}

public void markAsAvailable() {

this.isAvailable = true;

}

public boolean isAvailable() {

return isAvailable;

}

**}**

import java.util.List;

public class Author {

private int id;

private String name;

private List<Book> books;

public Author(int id, String name, List<Book> books) {

this.id = id;

this.name = name;

this.books = books;

}

}

public abstract class User {

protected int id;

protected String name;

protected String email;

public User(int id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

public abstract void viewBooks();

}

class Reader extends User {

public Reader(int id, String name, String email) {

super(id, name, email);

}

public Loan borrowBook(Book book) {

if (!book.isAvailable()) {

throw new IllegalStateException("Book is not available.");

}

book.markAsLoaned();

return new Loan(this, book);

}

public void returnBook(Loan loan) {

loan.completeLoan();

loan.getBook().markAsAvailable();

}

}

class Librarian extends User {

public Librarian(int id, String name, String email) {

super(id, name, email);

}

public void addBook(Book book) {

// Add book to library

}

public void removeBook(Book book) {

// Remove book from library

}

public Report generateReport(String reportType) {

return new Report(reportType);

}

}

import java.time.LocalDate;

public class Loan {

private static int counter = 1;

private int id;

private Book book;

private Reader reader;

private LocalDate loanDate;

private LocalDate returnDate;

public Loan(Reader reader, Book book) {

this.id = counter++;

this.reader = reader;

this.book = book;

this.loanDate = LocalDate.now();

}

public void completeLoan() {

this.returnDate = LocalDate.now();

}

public Book getBook() {

return book;

}

}

import java.util.Map;

public class Report {

private static int counter = 1;

private int id;

private String type;

private Map<String, Object> data;

public Report(String type) {

this.id = counter++;

this.type = type;

}

public void generate() {

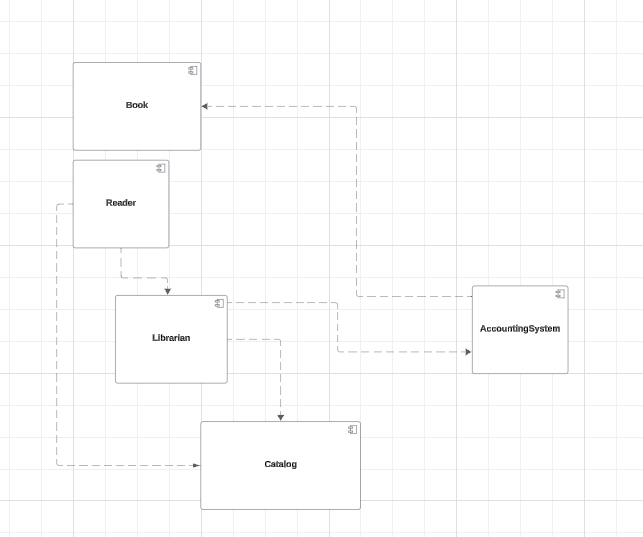
// Generate report based on type

}

}

**Задание:**  
Создайте простое консольное приложение на языке C#, которое моделирует систему управления библиотекой. Приложение должно включать компоненты, представляющие основные сущности системы: Книга, Читатель, Библиотекарь, Каталог, Учетная система.

1. **Описание компонентов:**
   * **Книга** - компонент, представляющий сущность книги в библиотеке. Включает такие свойства, как Название, Автор, Жанр, ISBN.
   * **Читатель** - компонент, представляющий читателя библиотеки. Включает свойства Имя, Фамилия, Номер билета.
   * **Библиотекарь** - компонент, отвечающий за выдачу и возврат книг. Обеспечивает операции по работе с книгами и учетной системой.
   * **Каталог** - компонент, представляющий каталог библиотеки. Включает методы поиска и фильтрации книг по жанру, автору, названию.
   * **Учетная система** - компонент, который ведет учет всех выданных и возвращенных книг, фиксирует информацию о том, кто и когда взял книгу.
2. **Создайте диаграмму компонентов**



1. **Реализуйте основные классы**

public class Book {

private String title;

private String author;

private String genre;

private String isbn;

private boolean isAvailable;

public Book(String title, String author, String genre, String isbn) {

this.title = title;

this.author = author;

this.genre = genre;

this.isbn = isbn;

this.isAvailable = true;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public String getGenre() {

return genre;

}

public String getIsbn() {

return isbn;

}

public boolean isAvailable() {

return isAvailable;

}

public void setAvailable(boolean available) {

isAvailable = available;

}

@Override

public String toString() {

return "Book{" +

"title='" + title + '\'' +

", author='" + author + '\'' +

", genre='" + genre + '\'' +

", isbn='" + isbn + '\'' +

", isAvailable=" + isAvailable +

'}';

}

}

public class Reader {

private String firstName;

private String lastName;

private int ticketNumber;

public Reader(String firstName, String lastName, int ticketNumber) {

this.firstName = firstName;

this.lastName = lastName;

this.ticketNumber = ticketNumber;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public int getTicketNumber() {

return ticketNumber;

}

@Override

public String toString() {

return "Reader{" +

"firstName='" + firstName + '\'' +

", lastName='" + lastName + '\'' +

", ticketNumber=" + ticketNumber +

'}';

}

}

import java.util.Date;

public class Librarian {

private AccountingSystem accountingSystem;

public Librarian(AccountingSystem accountingSystem) {

this.accountingSystem = accountingSystem;

}

public void issueBook(Book book, Reader reader) {

if (book.isAvailable()) {

book.setAvailable(false);

accountingSystem.recordLoan(book, reader, new Date());

System.out.println("Book issued: " + book.getTitle());

} else {

System.out.println("Book is not available.");

}

}

public void returnBook(Book book, Reader reader) {

book.setAvailable(true);

accountingSystem.recordReturn(book, reader, new Date());

System.out.println("Book returned: " + book.getTitle());

}

}

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

public class Catalog {

private List<Book> books = new ArrayList<>();

public void addBook(Book book) {

books.add(book);

}

public List<Book> searchByTitle(String title) {

return books.stream()

.filter(book -> book.getTitle().equalsIgnoreCase(title))

.collect(Collectors.toList());

}

public List<Book> searchByAuthor(String author) {

return books.stream()

.filter(book -> book.getAuthor().equalsIgnoreCase(author))

.collect(Collectors.toList());

}

public List<Book> searchByGenre(String genre) {

return books.stream()

.filter(book -> book.getGenre().equalsIgnoreCase(genre))

.collect(Collectors.toList());

}

}

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

public class AccountingSystem {

private Map<Book, String> loans = new HashMap<>();

public void recordLoan(Book book, Reader reader, Date date) {

loans.put(book, "Issued to " + reader.getFirstName() + " on " + date.toString());

}

public void recordReturn(Book book, Reader reader, Date date) {

loans.remove(book);

System.out.println("Book returned by " + reader.getFirstName() + " on " + date.toString());

}

public void printLoans() {

loans.forEach((book, info) -> System.out.println(book.getTitle() + ": " + info));

}

}

public class LibraryApplication {

public static void main(String[] args) {

Catalog catalog = new Catalog();

AccountingSystem accountingSystem = new AccountingSystem();

Librarian librarian = new Librarian(accountingSystem);

Book book1 = new Book("1984", "George Orwell", "Dystopian", "123456789");

Book book2 = new Book("Brave New World", "Aldous Huxley", "Dystopian", "987654321");

catalog.addBook(book1);

catalog.addBook(book2);

Reader reader = new Reader("John", "Doe", 101);

System.out.println("Search results for '1984':");

catalog.searchByTitle("1984").forEach(System.out::println);

librarian.issueBook(book1, reader);

librarian.issueBook(book1, reader);

librarian.returnBook(book1, reader);

accountingSystem.printLoans();

}

}