Курсов проект по Обектно Ориентирано Програмиране на Java

Въведение

Целта на този проект е да учениците да практикуват основните принципи на обектно ориентираното програмиране (ООП) на Java, като създадат система за управление на библиотека. Проектът включва създаване на класове, методи и интерфейси, които ще ви помогнат да разберете как се структурират и взаимодействат обектите в една програма.

Задание

СЕДМИЦА_01 ::

Част 1: Клас `Book`

Създайте клас `Book`, който представлява книга. Класът трябва да съдържа следните атрибути:

- `title`: заглавие на книгата
- `author`: автор на книгата
- `isbn`: уникален идентификационен номер на книгата
- `yearPublished`: година на публикуване

Инициализирайте атрибутите чрез конструктор и създайте методи `getters` и `setters` за всеки от тях.

Част 2: Интерфейс `BookRepository`

Създайте интерфейс `BookRepository`, който дефинира следните методи:

- `void addBook(Book book)`
- `boolean removeBook(String isbn)`
- `List<Book> findBookByTitle(String title)`
- `List<Book> findBookByAuthor(String author)`

СЕДМИЦА_02 ::

Част 3: Клас `InMemoryBookRepository`

Създайте клас `InMemoryBookRepository`, който реализира интерфейса `BookRepository` и съхранява книгите в **паметта**. Създайте подходяща структура от данни за съхранение в паметта на компютъра.

СЕДМИЦА_03 ::

Част 4: Клас `DatabaseBookRepository`

Създайте клас `DatabaseBookRepository`, който реализира интерфейса `BookRepository` и съхранява книгите в **база данни (SQLite)**.

СЕДМИЦА_04 ::

Част 5: Главен клас (`Main`)

Създайте клас `Main`, който тества функционалността на вашата библиотека. Текстово мвню за добавяне, редактиране и премахване на елементи. Добавете няколко книги, потърсете ги по заглавие и автор, и премахнете някоя от тях. Позволете на потребителя да избира между `InMemoryBookRepository` и `DatabaseBookRepository`.

Част 6: Обработка на грешки

Добавете обработка на грешки за всяка операция. Хвърлете изключения при опит за добавяне на книга с вече съществуващ ISBN или при премахване на книга, която не съществува.

Част 7: Многопоточност - НЕ !

Имплементирайте многопоточност, за да позволите на потребителите да търсят книги едновременно без блокиране на основната нишка.

СЕДМИЦА_05 ::

Част 8: GUI интерфейс

Създайте графичен потребителски интерфейс (GUI) за взаимодействие с библиотеката като използвате SWING - вградена функционалност на IntelliJ IDEA за създаване на графичен интерфейс.

Част 9: Тестове - НЕ !

Напишете тестове за всеки метод с помощта на JUnit или друг тестов фреймуърк. Тестовете трябва да покриват нормални и гранични случаи.

СЕДМИЦА_06 ::

Част 10: Документация

Създайде подробна документация на проекта, включително UML диаграми, описание на класовете и методите, както и ръководство за потребителя.

Примерен код

```
// Book class
public class Book {
    private String title;
    private String author;
    private String isbn;
    private int yearPublished;

    public Book(String title, String author, String isbn, int
yearPublished) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
        this.yearPublished = yearPublished;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public String getISBN() {
        return isbn;
    }

    public int getYearPublished() {
        return yearPublished;
    }
}

// Interface BookRepository
public interface BookRepository {
    void addBook(Book book);
    boolean removeBook(String isbn);
```

```
    List<Book> findBookByTitle(String title);
    List<Book> findBookByAuthor(String author);
}

// InMemoryBookRepository class
public class InMemoryBookRepository implements BookRepository {
    private List<Book> books;

    public InMemoryBookRepository() {
        books = new ArrayList<>();
    }

    @Override
    public void addBook(Book book) {
        books.add(book);
    }

    @Override
    public boolean removeBook(String isbn) {
        for (int i = 0; i < books.size(); i++) {
            if (books.get(i).getISBN().equals(isbn)) {
                books.remove(i);
                return true;
            }
        }
        return false;
    }

    @Override
    public List<Book> findBookByTitle(String title) {
        List<Book> result = new ArrayList<>();
        for (Book book : books) {
            if (book.getTitle().equalsIgnoreCase(title)) {
                result.add(book);
            }
        }
        return result;
    }
```

```
    @Override
    public List<Book> findBookByAuthor(String author) {
        List<Book> result = new ArrayList<>();
        for (Book book : books) {
            if (book.getAuthor().equalsIgnoreCase(author)) {
                result.add(book);
            }
        }
        return result;
    }
}

// DatabaseBookRepository class
public class DatabaseBookRepository implements BookRepository {
    // Implementation details omitted for brevity
}

// Main class for testing
public class Main {
    public static void main(String[] args) {
        BookRepository repository = chooseRepositoryType(); //
Choose between InMemory and Database repositories

        repository.addBook(new Book("The Great Gatsby", "F.
Scott Fitzgerald", "978-0743273565", 1925));
        repository.addBook(new Book("To Kill a Mockingbird",
"Harper Lee", "978-0061120084", 1960));

        System.out.println("Books by F. Scott Fitzgerald:");
        for (Book book : repository.findBookByAuthor("F. Scott
Fitzgerald")) {
            System.out.println(book.getTitle());
        }

        System.out.println("\nRemoved book with ISBN 978-
0743273565: " + repository.removeBook("978-0743273565"));
    }
```

```
private static BookRepository chooseRepositoryType() {
  ( продължете сами...)
```