Library Management System

Masterclass in Java

Paata Gogishvili

Doctor of Informatics, Associate Professor at Ilia State University

paatagog@gmail.com, paata.gogishvili@iliauni.edu.ge

Description

Create Library Management System (LMS) in Java. LMS is widely used software. It can be any complexity. Our example is basic one, which have the following features:

1. storage for the books
2. ability to add the book in the library
3. ability remove the book from the library
4. ability to print the library book information on the console

LMS structure

We will need the following classes for the software:

1. Book – the book itself.
2. LMS – library management system.
3. LibraryTester – the tester class. This class will be used to test our management system.

Class Book	
String	title
String	Author

Class LMS	
List <Book>	storage
void addBook(Book)	
boolean removeBook(Book)	
Void printStorage()	

Class Book

The class `Book` should have several fields, including `title` and `author`. This class can be implemented in the following way:

```
package library;

public class Book {
    private String title, author;


    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
```

```
    }

    public void setAuthor(String author) {
        this.author = author;
    }

}
```

Pay attention to the setters and getters of the fields. In general, all the fields are private (unless the special requirements are stated) and the access functions are implemented such as setters and getters.

Read about `toString()` function and implement it for `Book` class.

Class LMS

The library management system should have an inner structure for storing books. The management system should have methods for adding the new books and removing the old ones. It should have the ability to print the entire library content when needed. The class can be implemented in the following way:

```
package library;

import java.util.ArrayList;
import java.util.List;

public class LMS {

    // Mapping with Book and the number of this book in the library
    private List<Book> storage = new ArrayList<Book>();

    // adds the book to the library
    public void addBook(Book book) {
        storage.add(book);
    }

    // removes the book from the library
    public boolean removeBook(Book book) {

        boolean removed = false;

        for (int i = 0; i < storage.size(); i++) {
            Book b = storage.get(i);
            if (b.getTitle().equals(book.getTitle()) && b.getAuthor().equals(book.getAuthor())) {
                storage.remove(i);
                removed = true;
                break;
            }
        }

        return removed;
    }

    public void printStorage() {

        if (storage.isEmpty()) {
            System.out.println("The storage is empty");
        } else {
            for (Book b: storage) {
                System.out.println(b.getAuthor() + ", " + b.getTitle());
                System.out.println();
            }
        }
    }

}
```

Pay attention to the usage of the `ArrayList` class, `for` loops for the lists, `break` clause and the `String` object comparison. It is a good point to understand how `Interface` works. Usage of the `boolean` variables can also be observed in this example.

LMS Tester class

Now let's test our management system. First, create some books. Then create LLM and add those books to the library using the LLM. Then try to remove some of the books.

```
package library;

public class LibraryTester {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setName("Maka");
        s1.setSurname("Kapanadze");
        s1.setPn("12345678912");

        Student s2 = new Student();
        s2.setName("Giorgi");
        s2.setSurname("Giorgadze");
        s2.setPn("111111111111");

        Book b1 = new Book();
        b1.setTitle("Lord of the Rings");
        b1.setAuthor("Tolkien");

        Book b2 = new Book();
        b2.setTitle("Introduction to OOP");
        b2.setAuthor("Paata Gogishvili");

        LMS lms = new LMS();

        lms.addBook(b1);
        lms.addBook(b1);
        lms.addBook(b2);

        lms.removeBook(b1);

        lms.printStorage();

    }

}
```

We print the state of the library to check if all the methods are working properly.

Future Improvements

This example above is the very basic for the LMS. It can be improved by adding convenient features. Let's define some of them. Consider adding some packages if convenient.

1. Add the ability to say how many of each book is available in the library.

2. Introduce the class for student and add the ability to LLM to make accounting of the process of borrowing and returning the books. Implement the following methods for LLM:

a) `boolean borrowBook(Book book, Student student)`

b) `boolean returnBook(Book book, Student student)`

c) `void printStudentBooks(Student student)`

d) `void printBookInformation(Book book)`