مهندسی نرم‌افزار چیست؟

نرم‌افزار محصولی پیچیده‌ای است. بنابراین برای ساخت یک نرم‌افزار باید اصول مهندسی در آن رعایت شود. هدف در مهندسی نرم‌افزار ساخت یک نرم‌افزار با کیفیت است.

برای پیاده‌سازی یک سیستم نرم‌افزاری باید یک‌سری فعالیت‌ها انجام شود این فعالیت‌ها شامل:

۱. مهندسی نیازها: شروع توسعه نرم‌افزار با مهندسی نیازها است. در مهندسی نیازها امکاناتی که نرم‌افزار باید فراهم‌کنند مشخص می‌شود.

۲. طراحی: در طراحی نرم‌افزار باید ٤ بخش طراحی شود که شامل:

۱. طراحی معماری نرم‌افزار: معماری نرم‌افزار یک دید سطح بالا است که مشخص می‌کند نرم‌افزار دارای چه اجزایی است و این اجزا چگونه با یکدیگر ارتباط دارند. معماری که استفاده می‌کنیم معماری لایه‌ای است.

۲. طراحی رابط کاربر: هر سیستم نرم‌افزاری باید دارای رابط کاربر باشد که بوسیله‌ی آن کاربر بتواند با سیستم کار کند.

۳. طراحی مدل داده: مشخص می‌کند چه داده‌هایی باید ذخیره شوند و ساختار ذخیره‌سازی انها چگونه باشد.

٤. طراحی اجزاء نرم‌افزار: شامل کلاس‌ها و ارتباط بین آنها می‌باشد.

۳. پیاده‌سازی: طرح‌های نرم‌افزار به کدنویس‌ها داده شده تا آن را پیاده‌سازی کنند.

٤. تست و ارزیابی: بعد از پیاده‌سازی باید نرم‌افزار نصب و ارزیابی شود.

به این فعالیت‌ها و نحوه‌ی انجام آنها مهندسی نرم‌افزار گویند. مهندس نرم‌افزار، کسی است که نیازهای نرم‌افزار را تحلیل و سپس نرم‌افزار را طراحی می‌کند سپس آن طرح‌ها (نقشه‌ها) را به کدنویس‌ها میدهد تا پیاده‌سازی کنند.

در ادامه پروژه‌ی اول کدنویسی شما شرح داده خواهد شد و بخش‌های آن توضیح داده شده و طراحی می‌شوند. شما باید با توجه به طرح‌هایی که در این سند آمده است پروژه را پیاده‌سازی (کدنویسی) کنید.

پروژه برنامه نویسی پیشرفته

نام پروژه: سیستم نرم‌افزاری مدیریت کتابخانه‌ی دانشگاه

هدف پروژه: پیاده‌سازی یک سیستم مدیریت کتابخانه برای دانشگاه است. این سیستم امکانات لازم برای مدیریت کتاب، مدیریت اعضاء و مدیریت امانت را باید داشته باشد. در ادامه مهندسی نیازها و طراحی آن خواهد آمد.

مهندسی نیازها

سیستم مدیریت کتابخانه باید دارای امکانات زیر باشد.

۱.مدیریت کتاب:

- امکان تعریف کتاب‌های جدید شامل عنوان، نویسنده، شابک (ISBN)، سال انتشار و وضعیت امانت.
- امکان جستجوی کتاب‌ها بر اساس عنوان و شابک.
- امکان حذف کتاب‌ها از سیستم.
- نمایش لیست کتاب‌های موجود.

۲.مدیریت دسته‌بندی:

- امکان تعریف دسته‌بندی‌های جدید برای کتاب‌ها (مانند علوم کامپیوتر، ادبیات، تاریخ و غیره).
- امکان جستجو و نمایش دسته‌بندی‌ها.

۳.مدیریت اعضا (دانشجویان):

- امکان تعریف دانشجویان در دو مقطع کارشناسی و تحصیلات تکمیلی.
- برای هر دانشجو اطلاعات شماره دانشجویی، نام، نام خانوادگی، رشته تحصیلی و سال ورود ثبت می‌شود.
- برای دانشجویان تحصیلات تکمیلی علاوه بر آن نام استاد راهنما و عنوان پایان‌نامه ثبت می‌شود.
- امکان جستجوی دانشجویان بر اساس شماره دانشجویی.
- امکان حذف دانشجویان از سیستم.

٤.امانت و بازگشت کتاب:

- امکان امانت دادن کتاب به دانشجویان با ثبت تاریخ امانت و تاریخ سررسید.
- امکان ثبت بازگشت کتاب از دانشجو.

۵.گزارش‌گیری:

- لیست کتاب‌های موجود در هر دسته.
- لیست کتاب‌های که یک دانشجو به امانت گرفته.
- لیست دانشجویان یک رشته‌ی خاص.

طراحی نرم‌افزار

نرم‌افزار به صورت کامل در این سند طراحی شده است و شما کافی است فقط پیاده‌سازی کنید. برای این منظور بخش‌های معماری، رابط کاربر، مدل داده و مولفه در ادامه توضیح داده شده است.

طراحی معماری سیستم

معماری نرم‌افزار یک دید سطح بالا از سیستم است که مشخص می‌کند، سیستم دارای چه اجزایی است و آنها چگونه با هم ارتباط دارند. در این پروژه از معماری لایه‌ای MVC استفاده می‌شود.

معماری (MVC) Model-View-Controller یک الگوی طراحی نرم‌افزار است که برای جداسازی منطق برنامه به سه بخش متصل به هم استفاده می‌شود. این جداسازی، توسعه، نگهداری و تست برنامه را آسان‌تر می‌کند. هنگامی که صحبت از لایه میشود یعنی به صورت منطقی کلاس‌ها رو تو لایه‌های مختلف قرار میدیم. هر لایه به صورت یک پوشه (Folder) است که کلاس‌های مربوط به آن لایه در آن پوشه قرار می‌گیرند. طراحی پروژه سیستم مدیریت کتابخانه مبتنی بر معماری لایه‌ای انجام می‌شود و شامل لایه‌های زیر است.

۱.**لایه View (نمایش):**

- **مسئولیت:** نمایش اطلاعات به کاربر.
- **شامل:** کلاس `UserInterface` با متدهایی برای نمایش منوهای مختلف (اصلی، مدیریت اعضا، مدیریت کتاب، امانت، گزارش‌گیری) و دریافت انتخاب‌ها است.

۲.**لایه Controller (کنترل‌گر):**

- **مسئولیت:** دریافت ورودی از لایه View، تعامل با لایه Model برای انجام عملیات درخواستی، و ارسال نتایج به لایه View برای نمایش.
- **شامل:** کلاس `Library` (به عنوان کنترل‌گر اصلی) و احتمالاً کنترل‌گرهای فرعی برای مدیریت بخش‌های خاص (مانند `MemberController`, `BookController` و غیره). کلاس `Library` متدهایی خواهد داشت که از لایه View فراخوانی می‌شوند و پس از پردازش درخواست با استفاده از لایه Model، نتایج را برای نمایش به لایه View ارسال می‌کند.

۳.**لایه Model (مدل):**

- **مسئولیت:** منطق برنامه در این لایه تعریف میشود.
- **شامل:** کلاس‌های `Book`, `UndergraduateStudent`, `GraduateStudent`, `Category`, `Loan` که ساختار داده‌های سیستم را تعریف می‌کنند.
  متدهایی مانند افزودن کتاب، جستجوی دانشجو، امانت دادن کتاب و غیره در کلاسهای این لایه تعریف میشود.

٤.**لایه Data Access (دسترسی به داده):**

- **مسئولیت:** مدیریت نحوه ذخیره‌سازی و بازیابی داده‌ها ازفایلها
- **شامل:** کلاس `FileDataModel` با متدهایی برای خواندن و نوشتن اطلاعات در فایل‌ها. لایه Model برای دسترسی به داده‌ها از طریق این لایه با منابع داده تعامل می‌کند.

برای درک بهتر جریان کاری زیر اتفاق می‌افتد.

۱.زمانیکه برنامه اجرا میشود با استفاده از کلاس‌های لایه **View** منوی اصلی نشان داده میشود.

۲.لایه **View** ورودی کاربر را به لایه **Controller** ارسال می‌کند.
۳.لایه **Controller** ورودی را دریافت کرده و تصمیم می‌گیرد که کدام عملیات در لایه **Model** باید انجام شود.
٤.لایه **Controller** متدهای مربوطه را در لایه **Model** فراخوانی می‌کند.
٥.لایه **Model** (در صورت نیاز) از لایه **Data Access** برای خواندن یا نوشتن داده‌ها استفاده می‌کند.
٦.لایه **Model** کار مربوطه را اجرا کرده و نتایج را به لایه **Controller** برمی‌گرداند.
۷.لایه **Controller** نتایج را دریافت کرده و آن‌ها را برای نمایش به لایه **View** ارسال می‌کند.
۸.لایه **View** نتایج را به کاربر نمایش می‌دهد.

این معماری لایه‌ای مبتنی بر MVC باعث می‌شود که تغییرات در یک لایه (مثلاً نحوه نمایش در View) تأثیر کمی بر سایر لایه‌ها داشته باشد، زیرا مسئولیت‌ها به خوبی از هم جدا شده‌اند. این امر نگهداری، توسعه و تست سیستم را تسهیل می‌کند.

طراحی رابط کاربر

رابط کاربر سیستم به صورت متنی و به صورت زیر است.

ابتدا که نرم‌افزار اجراء می‌شود باید منوی زیر نشان داده شود.

***** سیستم مدیریت کتابخانه دانشگاه *****

لطفاً یکی از بخش‌های زیر را انتخاب کنید:

۱- مدیریت اعضا
۲- مدیریت کتاب
۳- بخش امانت
۴- گزارش‌گیری
۰- خروج از برنامه

با زدن هر کدام از شماره زیرمنوی مربوطه نشان داده شود.

زیرمنوی مدیریت اعضا (در صورت انتخاب ۱ از منوی اصلی):

***** مدیریت اعضا *****

لطفاً یکی از عملیات زیر را انتخاب کنید:

۱- تعریف دانشجوی جدید (کارشناسی)
۲- تعریف دانشجوی جدید (تحصیلات تکمیلی)
۳- جستجوی دانشجو بر اساس شماره دانشجویی
۴- حذف دانشجو بر اساس شماره دانشجویی
۵- مشاهده لیست تمامی دانشجویان
۶- بازگشت به منوی اصلی

زیرمنوی مدیریت کتاب (در صورت انتخاب ۲ از منوی اصلی):

***** مدیریت کتاب *****

لطفاً یکی از عملیات زیر را انتخاب کنید:

۱- تعریف کتاب جدید
۲- حذف کتاب بر اساس شابک
۳- جستجوی کتاب
۴- مشاهده لیست تمامی کتاب‌های موجود
۵- بازگشت به منوی اصلی

زیرمنوی جستجوی کتاب (در صورت انتخاب ۳ از زیرمنوی مدیریت کتاب):

***** جستجوی کتاب *****

لطفاً نوع جستجو را انتخاب کنید:

۱- جستجو بر اساس عنوان
۲- جستجو بر اساس شابک
۳- بازگشت به منوی مدیریت کتاب

زیرمنوی بخش امانت (در صورت انتخاب ۳ از منوی اصلی):

***** بخش امانت *****

لطفاً یکی از عملیات زیر را انتخاب کنید:

۱- امانت دادن کتاب به دانشجو
۲- بازگرداندن کتاب توسط دانشجو
۳- بازگشت به منوی اصلی

زیرمنوی گزارش‌گیری (در صورت انتخاب ٤ از منوی اصلی):

***** گزارش‌گیری *****

لطفاً نوع گزارش مورد نظر را انتخاب کنید:

۱- لیست کتاب‌های موجود در یک دسته‌بندی خاص
۲- لیست کتاب‌های امانت گرفته شده توسط یک دانشجو
۳- لیست دانشجویان یک رشته‌ی خاص
۴- بازگشت به منوی اصلی

طراحی کلاس‌های سیستم مدیریت کتابخانه دانشگاه

. در زیر، کلاس‌های اصلی هر لایه و متدهای کلیدی آن‌ها شرح داده شده‌اند:

۱. لایه View (نمایش): کلاس `UserInterface`

مسئولیت: نمایش اطلاعات به کاربر و دریافت ورودی از کاربر.

- **متدها:**
- `displayMainMenu()`: نمایش منوی اصلی برنامه و دریافت انتخاب کاربر.
- `displayMemberMenu()`: نمایش زیرمنوی مدیریت اعضا و دریافت انتخاب کاربر.
- `displayBookMenu()`: نمایش زیرمنوی مدیریت کتاب و دریافت انتخاب کاربر.
- `displayLoanMenu()`: نمایش زیرمنوی بخش امانت و دریافت انتخاب کاربر.
- `displayReportMenu()`: نمایش زیرمنوی گزارش‌گیری و دریافت انتخاب کاربر.
- `getUndergraduateStudentDetails()`: دریافت اطلاعات لازم برای ایجاد دانشجوی کارشناسی از کاربر.
- `getGraduateStudentDetails()`: دریافت اطلاعات لازم برای ایجاد دانشجوی تحصیلات تکمیلی از کاربر.
- `getStudentIdForSearch()`: دریافت شناسه دانشجو برای جستجو.
- `getBookDetailsForCreation()`: دریافت اطلاعات لازم برای ایجاد کتاب جدید از کاربر.
- `getBookISBNForOperation()`: دریافت شابک کتاب برای حذف یا جستجو.
- `getBookTitleForSearch()`: دریافت عنوان کتاب برای جستجو.
- `getStudentIdForLoan()`: دریافت شناسه دانشجو برای عملیات امانت.
- `getBookISBNForLoan()`: دریافت شابک کتاب برای عملیات امانت.
- `displayStudentDetails(Student student)`: نمایش جزئیات یک دانشجو.
- `displayBookDetails(Book book)`: نمایش جزئیات یک کتاب.
- `displayAllStudents(List<Student> students)`: نمایش لیست تمامی دانشجویان.
- `displayAllBooks(List<Book> books)`: نمایش لیست تمامی کتاب‌ها.
- `displayLoanDetails(Loan loan)`: نمایش جزئیات یک رکورد امانت.
- `displayReport(List<?> results)`: نمایش نتایج گزارش‌ها.
- // ... سایر متدهای نمایش و دریافت ورودی

۲. لایه Controller (کنترل‌گر): کلاس `LibraryController`

مسئولیت: دریافت ورودی از View، تعامل با Model برای انجام عملیات، و ارسال نتایج به View.

- **متدها:**
- `addUndergraduateStudent(String studentId, String firstName, String lastName, String major, int enrollmentYear)`: ایجاد و افزودن دانشجوی کارشناسی.
- `addGraduateStudent(String studentId, String firstName, String lastName, String major, String supervisor, String thesisTitle)`: ایجاد و افزودن دانشجوی تحصیلات تکمیلی.
- `searchStudent(String studentId)`: جستجوی دانشجو بر اساس شناسه.
- `removeStudent(String studentId)`: حذف دانشجو بر اساس شناسه.
- `displayAllStudents()`: درخواست نمایش لیست تمامی دانشجویان.
- `addBook(String title, String author, String isbn, int publicationYear)`: ایجاد و افزودن کتاب جدید.
- `removeBook(String isbn)`: حذف کتاب بر اساس شابک.
- `searchBookByTitle(String title)`: جستجوی کتاب بر اساس عنوان.
- `searchBookByISBN(String isbn)`: جستجوی کتاب بر اساس شابک.
- `displayAvailableBooks()`: درخواست نمایش لیست کتاب‌های موجود.
- `borrowBook(String bookISBN, String studentId)`: امانت دادن کتاب به دانشجو.
- `returnBook(String bookISBN)`: ثبت بازگشت کتاب.
- `displayBorrowedBooksByStudent(String studentId)`: درخواست نمایش کتاب‌های امانت گرفته شده توسط یک دانشجو.
- `displayBooksByCategory(String categoryName)`: درخواست نمایش کتاب‌های موجود در یک دسته‌بندی خاص.
- `displayStudentsByMajor(String major)`: درخواست نمایش دانشجویان یک رشته‌ی خاص.
- // ... سایر متدهای کنترل عملیات مربوط به کتاب، امانت و گزارش‌گیری
- `saveData()`: درخواست ذخیره‌سازی تمامی داده‌ها در فایل.
- `loadData()`: درخواست بارگیری تمامی داده‌ها از فایل.

۳. لایه Model (مدل): کلاس‌های موجودیت و منطق تجاری

مسئولیت: نگهداری داده‌های برنامه و پیاده‌سازی منطق تجاری.

- **کلاس Book**:

- **ویژگی‌ها**: `title`, `author`, `isbn`, `publicationYear`, `isBorrowed`.
- **متدها**: `borrowBook()`, `returnBook()`, `getDetails()`.

- **کلاس انتزاعی `Student`**:
  - **ویژگی‌ها**: `studentId`, `firstName`, `lastName`, `major`.
  - **متدهای انتزاعی**: `getStudentDetails()`.

- **کلاس `UndergraduateStudent` (ارث‌بر از `Student`)**:
  - **ویژگی‌ها**: `enrollmentYear`.
  - **متدها**: `getStudentDetails()` (پیاده‌سازی).

- **کلاس `GraduateStudent` (ارث‌بر از `Student`)**:
  - **ویژگی‌ها**: `supervisor`, `thesisTitle`.
  - **متدها**: `getStudentDetails()` (پیاده‌سازی).

- **کلاس `Category`**:
  - **ویژگی‌ها**: `name`, `description`.
  - **متدها**: `getName()`, `getDescription()`.

- **کلاس `Loan`**:
  - **ویژگی‌ها**: `book`, `student`, `loanDate`, `dueDate`.
  - **متدها**: `getBook()`, `getStudent()`, `getLoanDate()`, `getDueDate()`.

- **کلاس `Library` (مدیریت مجموعه‌های داده و منطق اصلی)**:
  - **ویژگی‌ها**: `books` (لیست کتاب‌ها), `students` (لیست دانشجویان), `categories` (لیست دسته‌بندی‌ها), `loans` (لیست سوابق امانت).
  - **متدها**: `addBook(Book book)`, `removeBook(String isbn)`, `searchBookByTitle(String title)`, `searchBookByISBN(String isbn)`, `getAvailableBooks()`, `addStudent(Student student)`, `removeStudent(String studentId)`, `searchStudentById(String studentId)`, `getAllStudents()`, `borrowBook(Book book, Student student)`, `returnBook(Book book)`, `getBorrowedBooksByStudent(String studentId)`, `getBooksByCategory(String categoryName)`, `getStudentsByMajor(String major)`.

٤. لایه Data Access (دسترسی به داده): کلاس `FileDataModel`

مسئولیت: مدیریت نحوه ذخیره‌سازی و بازیابی داده‌ها از فایل.

- **متدها:**
  - `loadBooks(String filePath)`: بارگیری اطلاعات کتاب‌ها از فایل.
  - `saveBooks(List<Book> books, String filePath)`: ذخیره اطلاعات کتاب‌ها در فایل.
  - `loadStudents(String filePath)`: بارگیری اطلاعات دانشجویان از فایل.
  - `saveStudents(List<Student> students, String filePath)`: ذخیره اطلاعات دانشجویان در فایل.
  - `loadCategories(String filePath)`: بارگیری اطلاعات دسته‌بندی‌ها از فایل.
  - `saveCategories(List<Category> categories, String filePath)`: ذخیره اطلاعات دسته‌بندی‌ها در فایل.
  - `loadLoans(String filePath)`: بارگیری اطلاعات سوابق امانت از فایل.
  - `saveLoans(List<Loan> loans, String filePath)`: ذخیره اطلاعات سوابق امانت در فایل.
  - // ... متدهای بارگیری و ذخیره برای سایر موجودیت‌ها

نحوه تعامل لایه‌ها:

- لایه `UserInterface` با دریافت ورودی از کاربر، متدهای مربوطه را در لایه `LibraryController` فراخوانی می‌کند.
- لایه `LibraryController` با استفاده از کلاس‌های موجود در لایه Model (`Book`, `Student`, `Library` و غیره) منطق تجاری را پیاده‌سازی می‌کند و داده‌ها را مدیریت می‌کند. برای ذخیره‌سازی و بازیابی داده‌ها، `LibraryController` از لایه `FileDataModel` استفاده می‌کند.
- لایه Model شامل ساختار داده‌ها (کلاس‌های موجودیت) و منطق کسب و کار است.
- لایه `FileDataModel` جزئیات مربوط به نحوه تعامل با فایل‌ها را کپسوله می‌کند.

توضیحات تکمیلی:

هنگامی که برنامه جاوا اجرا می‌شود و متد `main` آن فراخوانی می‌گردد، اولین قدم ایجاد یک شیء از کلاس `UserInterface` (لایه View) است. سپس، از طریق این شیء، متدی که مسئول نمایش منوی اصلی به کاربر است، صدا زده می‌شود.

```
public class Main {
    public static void main(String[] args) {

        UserInterface ui = new UserInterface();

        ui.displayMainMenu();
    }
}
```

متد `displayMainMenu` در کلاس `UserInterface` مسئول نمایش منوی اصلی و همچنین مدیریت تعامل اولیه کاربر برای انتخاب یک بخش از سیستم است. این متد باید منتظر ورودی کاربر بماند و سپس بر اساس آن، جریان برنامه را هدایت کند.

```
public class UserInterface {
      private Scanner scanner;
      private LibraryController libraryController; // شیء از کنترل‌گر اصلی
      public UserInterface() {
            this.scanner = new Scanner(System.in);
            this.libraryController = new LibraryController();
      }
      public void displayMainMenu() {
            int choice;
            do {
                  System.out.println("\n*** سیستم مدیریت کتابخانه دانشگاه ***");
                  System.out.println("لطفاً یکی از بخش‌های زیر را انتخاب کنید:");
                  System.out.println("1- مدیریت اعضا");
                  System.out.println("2- مدیریت کتاب");
                  System.out.println("3- بخش امانت");
                  System.out.println("4- گزارش‌گیری");
                  System.out.println("0- خروج از برنامه");
                  System.out.print("انتخاب شما: ");
                  choice = scanner.nextInt();

                  switch (choice) {
                        case 1:
                              displayMemberMenu();
                        break;
                        case 2:
                              displayBookMenu();
                        break;
                        case 3:
                              displayLoanMenu();
                        break;
                        case 4:
                              displayReportMenu();
                        break;
                        case 0:
                              System.out.println("خروج از برنامه...");
                        break;
                        default:
                              System.out.println("انتخاب نامعتبر. لطفاً دوباره تلاش کنید.");
                  }
      }
```

توضیحات:

۱. **ایجاد `Scanner`**: برای دریافت ورودی از کاربر.

۲. **ایجاد `LibraryController`**: یک شیء از کنترل‌گر اصلی ایجاد می‌شود تا `UserInterface` بتواند برای انجام عملیات با لایه Model تعامل کند. (ممکن است در پروژه‌های بزرگ‌تر، کنترل‌گرهای فرعی نیز در اینجا ایجاد شوند یا از طریق `LibraryController` به آن‌ها دسترسی پیدا شود).

۳. **حلقه `do-while`**: برای نمایش منوی اصلی به صورت مداوم تا زمانی که کاربر گزینه خروج (۰) را انتخاب کند.

٤. **نمایش منو**: چاپ گزینه‌های مختلف به کاربر.

۵. **دریافت ورودی**: خواندن عدد وارد شده توسط کاربر با استفاده از `scanner.nextInt()`.

٦. **مصرف `nextLine()`**: بعد از `nextInt()`, یک `scanner.nextLine()` فراخوانی می‌شود تا کاراکتر newline که بعد از وارد کردن عدد در بافر ورودی باقی می‌ماند، مصرف شود. این کار از بروز مشکلات در خواندن رشته‌های بعدی جلوگیری می‌کند.

۷. **ساختار `switch-case`**: بر اساس مقدار `choice` (انتخاب کاربر)، متد مربوط به نمایش زیرمنوی آن بخش فراخوانی می‌شود.

۸. **گزینه خروج**: اگر کاربر ۰ را وارد کند، حلقه `do-while` خاتمه یافته و برنامه به پایان می‌رسد.

۹. **پیام خطا**: در صورت وارد کردن عدد نامعتبر، یک پیام خطا به کاربر نمایش داده می‌شود.

۱۰. **متدهای زیرمنو**: متدهای `displayMemberMenu()`, `displayBookMenu()`, `displayLoanMenu()`, و `displayReportMenu()` به طور مشابه عمل می‌کنند: نمایش گزینه‌های زیرمنو، دریافت انتخاب کاربر، و سپس فراخوانی متدهای مربوطه در لایه Controller برای انجام عملیات مورد نظر.

برای فهم بیشتر در ادامه خلاصه متد `displayMemberMenu` آمده است.

```
private void displayMemberMenu() {
     int choice;

     do {
         System.out.println("\n*** مدیریت اعضا ***");
         System.out.println("لطفاً یکی از عملیات زیر را انتخاب کنید:");
         System.out.println("1- تعریف دانشجوی جدید (کارشناسی)");
         System.out.println("2- تعریف دانشجوی جدید (تحصیلات تکمیلی)");
         System.out.println("3- جستجوی دانشجو بر اساس شماره دانشجویی");
         System.out.println("4- حذف دانشجو بر اساس شماره دانشجویی");
         System.out.println("5- مشاهده لیست تمامی دانشجویان");
         System.out.println("6- بازگشت به منوی اصلی");
         System.out.print("انتخاب شما: ");

         choice = scanner.nextInt();
         scanner.nextLine(); // مصرف کاراکتر newline

         switch (choice) {
             case 1:
                 // دریافت اطلاعات دانشجوی کارشناسی از کاربر
                 System.out.print("شماره دانشجویی: ");
                 String undergraduateId = scanner.nextLine();
                 System.out.print("نام: ");
                 String undergraduateFirstName = scanner.nextLine();
                 System.out.print("نام خانوادگی: ");
                 String undergraduateLastName = scanner.nextLine();
                 System.out.print("رشته تحصیلی: ");
                 String undergraduateMajor = scanner.nextLine();
                 System.out.print("سال ورود: ");
                 int undergraduateEnrollmentYear = scanner.nextInt();
                 scanner.nextLine();

                 // فراخوانی متد کنترل‌گر برای ایجاد دانشجوی کارشناسی
                 libraryController.addUndergraduateStudent
                     (undergraduateId, undergraduateFirstName,
                     undergraduateLastName, undergraduateMajor,
                     undergraduateEnrollmentYear);
                 break;
             case 2:
                 // دریافت اطلاعات دانشجوی تحصیلات تکمیلی از کاربر
                 System.out.print("شماره دانشجویی: ");
                 String graduateId = scanner.nextLine();
                 System.out.print("نام: ");
                 String graduateFirstName = scanner.nextLine();
                 System.out.print("نام خانوادگی: ");
                 String graduateLastName = scanner.nextLine();
                 System.out.print("رشته تحصیلی: ");
                 String graduateMajor = scanner.nextLine();
                 System.out.print("استاد راهنما: ");
                 String graduateSupervisor = scanner.nextLine();
                 System.out.print("عنوان پایان‌نامه: ");
                 String graduateThesisTitle = scanner.nextLine();

                 // فراخوانی متد کنترل‌گر برای ایجاد دانشجوی تحصیلات تکمیلی
                 libraryController.addGraduateStudent
                     (graduateId, graduateFirstName, graduateLastName,
                     graduateMajor, graduateSupervisor, graduateThesisTitle);
                 break;
             case 3:
                 // دریافت شماره دانشجویی برای جستجو
                 System.out.print("شماره دانشجویی مورد نظر برای جستجو: ");
                 String searchId = scanner.nextLine();
                 // فراخوانی متد کنترل‌گر برای جستجوی دانشجو
                 libraryController.searchStudent(searchId);
```

```
                    break;
                case 4:
                    // دریافت شماره دانشجویی برای حذف
                    System.out.print("شماره دانشجویی مورد نظر برای حذف: ");
                    String deleteId = scanner.nextLine();
                    // فراخوانی متد کنترل‌گر برای حذف دانشجو
                    libraryController.removeStudent(deleteId);
                    break;
                case 5:
                    // فراخوانی متد کنترل‌گر برای نمایش لیست همه دانشجویان
                    libraryController.displayAllStudents();
                    break;
                case 6:
                    System.out.println("بازگشت به منوی اصلی...");
                    return; // خروج از متد و بازگشت به منوی اصلی
                default:
                    System.out.println("انتخاب نامعتبر. لطفاً دوباره تلاش کنید.");
            }
        } while (true); // تا زمانی که کاربر گزینه بازگشت را انتخاب نکند
    }
}
```

متد `addGraduateStudent` در کلاس `LibraryController` وظیفه دریافت اطلاعات مربوط به یک دانشجوی تحصیلات تکمیلی از لایه View، ایجاد یک شیء `GraduateStudent` (که بخشی از لایه Model است) و سپس درخواست ذخیره‌سازی این شیء را از طریق لایه Model (یا یک سرویس در لایه Model که با Data Access تعامل دارد) دارد.

نمای شماتیک از کد متد `addGraduateStudent` در `LibraryController`:

```
public class LibraryController {
    private List<Student> students;
    private FileDataModel fileDataModel;

    public LibraryController() {
        this.students = new ArrayList<>();
        this.fileDataModel = new FileDataModel("students.txt");
        loadStudentsFromFile();
    }

    public void addGraduateStudent(String studentId, String firstName, String
            lastName, String major, String supervisor, String thesisTitle) {
        // 1. ایجاد یک شیء GraduateStudent (مدل داده)
        GraduateStudent newGraduateStudent = new GraduateStudent(studentId,
                firstName, lastName, major, supervisor, thesisTitle);

        // 2. افزودن شیء دانشجو به مجموعه (مدل)
        students.add(newGraduateStudent);

        // 3. درخواست از لایه Model (یا سرویس) برای ذخیره‌سازی اطلاعات دانشجو
        saveStudentsToFile();

        System.out.println("دانشجوی تحصیلات تکمیلی با شماره دانشجویی " + studentId + " با
موفقیت ثبت شد.");
    }

    // سایر متدهای مربوط به مدیریت دانشجو (جستجو، حذف، نمایش) و کتاب و امانت و غیره

    private void saveStudentsToFile() {
        // این متد باید منطق تبدیل لیست دانشجویان به قالبی قابل ذخیره در فایل
```

```
        // را برای نوشتن در فایل پیاده‌سازی کند fileDataModel.writeData() و سپس استفاده از
        // مثال:
        // List<String> studentData = convertStudentsToStrings(students);
        // fileDataModel.writeData(studentData, "students.txt");
        System.out.println("اطلاعات دانشجویان در فایل ذخیره شد."); // پیام موقت
    }

    private void loadStudentsFromFile() {
        // این متد باید منطق خواندن اطلاعات دانشجویان از فایل با استفاده از
        // fileDataModel.readData() و تبدیل آن‌ها به اشیاء GraduateStudent و
UndergraduateStudent
        // را پیاده‌سازی کند students و افزودن آن‌ها به لیست.
        System.out.println("اطلاعات دانشجویان از فایل بارگیری شد."); // پیام موقت
    }
}
```

این فایل را مطالعه بفرمایید

سرکلاس صحبت خواهیم کرد

موفق باشید

سید مهرداد اسلامی