Final Project Object Oriented Programming Guideline

Lecture: **Ir. Galih Wasis Wicaksono, S.Kom., M.Cs.**
Class: 2A, 2B

1. Project Description

Students are tasked with designing and implementing a campus library information system using Java. The system will include collection management (books), borrowing and returning processes, and a graphical user interface (JavaFX) for user interactions. This project integrates all Java Foundations materials and assesses students' software development competencies.

1.1 Group Work & Submission Guidelines on GitHub

- **Team Formation:** Each team should consist of 2-3 students, max 3 students.
- **Roles & Responsibilities:**
    - Lead Developer, Documentation & Testing: Coordinate module integration, write technical docs, and unit tests
    - Backend Specialist: develop data models & CRUD logic.
    - Frontend Specialist: design and implement JavaFX UI.
- **GitHub Repository**:
    - Create a public repo named "OOPLibrarySystem_TeamX".
    - Folder structure:
        - **/src** for source code.
        - **/data** for sample CSV files.
        - **/docs** for documentation.
        - **/presentation** for slides.
    - Initial commit: **git init** and add **README.md** with project overview.
    - Use **main** for releases and **dev** for ongoing development.

1.2. Submission

- Push all changes to **main** before the deadline.
- Tag the final release as **v1.0** and submit the GitHub link via the LMS.
- The due date for the final project and submission to the LMS is **June 19th, 2025**.
- Offline/online presentation will be held on **June 20-26, 2025**.

2. Learning Outcomes by Course Section

Students are expected to demonstrate the following competencies aligned with each curriculum section:

Section	Learning Outcome
1	Explain Java's history, configure JDK/JRE, and set up an IDE for development.
2	Implement a development model (spiral/iterative) and apply basic debugging techniques in Java code.
3	Use primitive and object data types (boolean, int, double, String) and perform data conversions.
4	Write custom methods, invoke library methods (String, Math, Random), and manage imports/packages.
5	Apply control structures (if/else, switch) for business logic such as validation and fine calculation.
6	Use loop constructs (for, while, do-while) with break & continue to manage iterative flows.
7	Design Java classes following OOP principles: encapsulation, inheritance, polymorphism, and static members.
8	Manage collections (ArrayList), handle file I/O, and process exceptions (FileNotFoundException, IOException).
9	Build interactive interfaces with JavaFX: layout, TableView, event handling, and notifications.
10	Combine all concepts into the final project, complete with documentation and presentation.

3. Case Study: Campus Library Service

Students will build a library information system with the following details:

3.1 Background

UMM Library wants to enhance accessibility and efficiency for students and staff. Current manual processes cause delays and data errors. Your system should replace manual workflows with a user-friendly digital interface.

3.2 Key Features

1. **User Authentication**
    - Log in for **Members** (students/staff) and **Admins** (librarians).
2. **Book Catalog Management**
    - CRUD (Create, Read, Update, Delete) operations for books.

    - Search by ISBN, title, author, or category.
3. **Member Registration**
    - Registration form for new members (ID, full name, major, email).
    - Validate duplicate ID or email.
4. **Borrowing Process**
    - Members select books from the catalog and borrow them.
    - System records borrow date, return date (default 7 days), and status.
5. **Return Process**
    - Admin processes returns and calculates/displays fines for late returns.
    - Update the book status back to “available.”
6. **Reports & Statistics**
    - Report of currently borrowed books.
    - Monthly statistics: total borrows, returns, and fines collected.
7. **Notifications & Dialogs**
    - Confirmation dialogs for critical actions (delete data, return book).
    - Error/success alerts using JavaFX Alert.

3.3 Use Cases & Interaction Flow

Use Case	Actor	Brief Description
Login	Member	Enter credentials to access member features.
Member Registration	User	Fill out the registration form to become a library member.
Manage Book Catalog	Admin	Add, modify, delete, and search books.
Borrow Book	Member	Select a book, confirm borrowing, data is recorded.
Return Book	Admin	Process return, calculate fine, update status.
View Reports & Statistics	Admin	Display usage reports and charts.

3.4 Entities and Sample Data

- **Book**:
    - ISBN: `978-602-73156-1-0`, Title: "Pemrograman Java Dasar", Author: "Agus Salim", Quantity: `5`
- **Member**:
    - ID: `M001`, Name: "Budi Santoso", Major: "Informatics", Email: "budi@umm.ac.id"
- **Transaction**:
    - ID: `T1001`, MemberID: `M001`, ISBN: `978-602-73156-1-0`, BorrowDate: `2025-01-15`, ReturnDate: `2025-01-22`, Status: `Borrowed`

4. Technical Specifications

4.1 CRUD Operations

The system must implement the following basic data operations:

- **Create**: Add new records (books, members, transactions).
- **Read**: Retrieve and display records (book lists, borrow history).
- **Update**: Modify existing records (edit book/member details, transaction status).
- **Delete**: Remove records (obsolete books, former members).

4.2 MVC Architecture

The codebase should follow the Model-View-Controller pattern:

- **Model**:
    - Represents data and business logic (classes Book, Member, Transaction, LibraryManager).
    - Handles file I/O for reading/writing CSV data.
- **View**:
    - User interface built with JavaFX (FXML and GUI controllers).
    - UI components: forms, TableView, dialogs, charts.
- **Controller**:
    - Mediates between Model and View.
    - Processes user events (button clicks, form inputs), invokes Model methods, and updates the View.

4.3 Detailed Technical Requirements

1. **Setup & Project**: Java 11+, IDE (IntelliJ/Eclipse).
2. **Data Storage**: using CSV files with CRUD via File I/O or Database System.
3. **OOP**: Minimum four (4) classes with clear relationships.
4. **Business Logic**: Availability checks, fine calculation, book search (title/ISBN).
5. **Exception Handling**: Handle FileNotFoundException, IOException, and input format errors.
6. **JavaFX GUI**: Main menu, input forms, TableView, confirmation dialogs.
7. **Testing**: Unit tests for CRUD methods and fine calculation logic.

5. Scope & Limitations

- **Minimum**: Core CRUD, borrowing/returning features, basic GUI.
- **Optional**: Login/authentication, CSV/PDF export, JavaFX Charts statistics.

6. Deliverables

1. **Complete Source Code (.java, .fxml)**
   Well-structured Java code with FXML files for the GUI.
2. **Sample Data Files (books.csv, members.csv)**
   CSV files containing initial book and member data for testing CRUD functionality.
3. **Technical Documentation (5–7 pages)**
   Explains system architecture, class diagrams, flowcharts, and user instructions.
4. **Demo Video (≤ 5 minutes)**
   Short recording showcasing installation, key features (CRUD, borrowing/returning), and the GUI.
5. **Presentation Slides (10–12 slides)**
   Summarize background, architecture, feature demo, testing results, and future enhancements.

7. Timeline & Milestones

This final project will be completed in **four weeks (starting now until June 19th, 2025)** as follows:

Phase	Week	Output
Design & Setup	1	IDE & JDK configuration; UI mockups; MVC class diagrams
Core & CRUD Implementation	2	Model classes (Book, Member, Transaction, LibraryManager); CRUD features for books & members
GUI Integration & Business Logic	3	JavaFX UI (forms, TableView, dialogs); borrowing, returning, and fine calculation logic
Testing, Documentation & Presentation	4	Unit testing & debugging; final technical documentation; preparation of slides & demo video

8. Evaluation Criteria & Weighting

Assessment is designed to evaluate each Learning Outcome from Sections 1–10.

Aspect & Section	Weight (%)	Description
Section 1: Setup & Configuration	5	Correct IDE, JDK, and GitHub setup
Section 2: Model & Debugging	10	Effective use of spiral/iterative model and debugging techniques.

Section 3: Data Types & Conversion	5	Appropriate use of primitive and object data types.
Section 4: Methods & Libraries	10	Custom methods and library usage (String, Math, Random).
Section 5: Control Structures	10	Business logic via if/else and switch statements.
Section 6: Loop Constructs	5	Correct use of loops (for, while, do-while) with break & continue.
Section 7: OOP Principles	15	Class design, encapsulation, inheritance, and polymorphism.
Section 8: Collections & Exception Handling	10	Use of ArrayList, file I/O, and exception handling.
Section 9: JavaFX GUI	15	UI design, TableView implementation, event handling.
Section 10: Integration & Documentation	15	Integration of all concepts, clear documentation, and final presentation.
Total	100	

9. Final Presentation

- **Duration**: 10-minute presentation + 5-minute Q&A.
- **Schedule**: June 20-26, 2025
- **Content of Slides:**
    1. **Background & Objectives (Sections 1–2)**
       Describe the manual workflow challenges and project goals.
    2. **System Architecture & Class Diagram (Sections 7–8)**
       Display the MVC structure and relationships of core classes.
    3. **Feature Demo: CRUD & Borrowing (Sections 3–6)**
       Showcase book addition, member registration, borrowing, and returning.
    4. **UI & User Flow (Section 9)**
       Explain navigation, form layouts, and event responses.
    5. **Testing Results & Challenges (Sections 2 & 8)**
       Present unit test outcomes, bugs encountered, and fixes.
    6. **Future Enhancements (Optional)**
       Propose additional features like authentication, report exports, or charts.