	School of Engineering & Technology	
	Department: SOET	**Session: 2025-26**
	Programme: B.Tech. , BCA, BSc.	**Semester: 3rd**
	Course Code: ENCS201, ENCA 203, ENBC205	**Number of students:**
	Course Name: Java Programming	**Faculty: Vishwanil S.**

Assignment Number: 04

Instructions:

- Assignment needs to be submitted by given date.
- Assignment must be submitted on (https://lms.krmangalam.edu.in/).
- Use of ChatGPT and similar tools is strictly prohibited.
- Assignment must be written on dedicated assignment copy for Java Programming subject.
- All assignments must be prepared as per the format shared in classes
- Assignment needs to be submitted by each individual.
- Total marks: 5 marks.
- This assignment contributes to total 10% of internal evaluation.
- Submission Requirements: Submit individually via **GitHub** and provide the link of submission by **25th October,2025.**
- Assignments will be evaluated on the basis of the following metrics.
  - Originality
  - Correctness
  - Completeness

Sr.no	Assignment Details	COs
1.	**Project Title: City Library Digital Management System** **Problem Statement:** The City Central Library has decided to digitize its operations to make book management, membership services, and data tracking more efficient. They need a Java-based application to handle book records, member details, and transactions using **File Handling** and the **Java Collections Framework**. The system must be capable of: • Adding new books and members. • Issuing and returning books. • Storing and retrieving data from files for persistence. • Using collections to efficiently search, sort, and manage records.	CO1

Project Objectives:

- Apply **File Handling** concepts to store and retrieve library data from text and binary files.
- Implement **Java Collections Framework** (List, Set, Map, Queue) to manage records dynamically.
- Use **Comparable** and **Comparator** for sorting books and members by different criteria.
- Practice **Buffered I/O**, **Character Streams**, and **Byte Streams** for efficient file operations.
- Apply **Generics** in collections for type-safe data handling.

Learning Outcomes (COs):

- **CO4.1**: Use Java's File Handling API for persistent storage of real-world data.
- **CO4.2**: Implement the Java Collections Framework for dynamic data management.
- **CO4.3**: Demonstrate sorting and searching using Comparable/Comparator.
- **CO4.4**: Integrate I/O operations with collection-based data handling for real-world scenarios.

Project Instructions:

1. Class Design

Book Class

- Attributes:
  - bookId (Integer, unique ID)
  - title (String)
  - author (String)
  - category (String)
  - isIssued (Boolean)
- Methods:
  - displayBookDetails() – Shows book details.
  - markAsIssued() – Updates isIssued to true.
  - markAsReturned() – Updates isIssued to false.

Member Class

- Attributes:
  - memberId (Integer, unique ID)
  - name (String)
  - email (String)
  - issuedBooks (List of book IDs)
- Methods:
  - displayMemberDetails()
  - addIssuedBook(int bookId)
  - returnIssuedBook(int bookId)

LibraryManager Class

- Attributes:
  - Map<Integer, Book> books
  - Map<Integer, Member> members
- Methods:
  - addBook() – Adds a new book to the collection and saves it to file.
  - addMember() – Adds a new member and saves to file.
  - issueBook() – Marks a book as issued and updates both book and member records in file.
  - returnBook() – Marks a book as returned.
  - searchBooks() – Search by title, author, or category using **Collections**.
  - sortBooks() – Sort by title/author using **Comparable/Comparator**.
  - loadFromFile() – Loads all data at startup.
  - saveToFile() – Saves all data before exit.

2. File Handling Requirements

- Use FileReader/FileWriter for text-based data storage.
- Use BufferedReader/BufferedWriter for faster reading/writing.
- Use FileInputStream/FileOutputStream for binary storage of serialized objects (optional advanced).
- Ensure files are created if not present.
- Store books in books.txt and members in members.txt.

3. Collection Framework Requirements

- **List**: Store issued book IDs for each member.
- **Set**: Maintain a set of unique categories.
- **Map**: Store books and members using IDs as keys.
- **Queue (Optional Advanced)**: Maintain a waiting list for popular books.
- **Comparable**: Sort books by title.
- **Comparator**: Sort books by author or category.

4. Implementation Steps

1. Design Book and Member classes with required fields and methods.
2. Implement LibraryManager with collections to store data in memory.
3. Load existing records from files when the application starts.
4. Implement menu options for all operations (add/search/sort/issue/return).
5. Save updated data to files after each operation.

5. Sample Interaction

Welcome to City Library Digital Management System

1. Add Book
2. Add Member
3. Issue Book
4. Return Book
5. Search Books
6. Sort Books
7. Exit

Enter your choice: 1

Enter Book Title: Java Programming Mastery Enter Author:
John Smith

Enter Category: Programming

Book added successfully with ID: 101

Evaluation Highlight Rubrics (5 points)

Criterion	Points	Description
File Handling Implementation	1	Proper use of file streams, buffered I/O, character & byte streams.
Use of Collections Framework	1	Efficient use of List, Set, Map, Queue with correct operations.
Sorting & Searching	1	Correct use of Comparable & Comparator for sorting, searching functionality implemented.
Input Validation & Exception Handling	0.5	Validates data (unique IDs, email format, book availability).
Code Structure & Modularity	0.5	Classes/methods well-structured and reusable.
Code Documentation & Readability	0.5	Meaningful comments, naming conventions followed.
Completeness & Real-life Relevance	0.5	Fully functional, realistic simulation of library management.