**King Fahd University of Petroleum & Minerals**
**College of Computing and Mathematics**
**Information and Computer Science Department**
**ICS 202 – Data Structures**
**First Semester 2024-2025 (241)**

# Lab Project

# Multi-Index Data Structure: Student Records Management System

You can work in teams of 2 or individually.
Deadline is Saturday 14th/December/2024, and the demo day is Sunday 15th/December/2024 and will be arranged with your lab instructor.

## Understanding Database Indexing

In database management systems, indexing is a fundamental technique used to optimize data retrieval operations.

Consider searching for a student by their last name in a dataset of 10,000 records:

- Without an index: Must scan all 10,000 records (Linear search, O(n))
- With an index: Can use efficient data structure(Typically O(log n))

## Multiple Indices in Practice

Different queries require different access patterns. For example:

- Academic advisors might search by student level
- Registration office might search by student ID
- Faculty might search by last name

Instead of maintaining separate sorted lists (which wastes memory and risks inconsistency), a multi-index structure maintains references to a single data copy.

## Project Requirements

Develop a Java-based data structure implementation that efficiently manages student records using multiple indices or data structure based on all the attributes of the student class, similar to database management systems. The structure should maintain data consistency while providing quick access through different sorting criteria. The attached csv file contains a randomly generated list of students.

Your data structure should provide the following operation:

- Efficient insertion maintaining all indices
- Deletion with proper reference cleanup
- Updates with automatic index maintenance
- Retrieval through any index, those retrievals will provide the capabilities to search By
    - By exact student ID
    - By last name
    - By First name
- Retrieve a list for an academic level

Develop a command-based interactive interface for all operations, ensure data validation, and handle errors.

The interface should have a main menu with these options

1. Search Student
2. Add New Student
3. Show students in an academic level
4. Exit

Under "Search Student", display all possible ways to retrieve a student. Once retrieved, prompt whether the user wants to:

- Edit the student
- Delete the student
- Return to main menu.

## Question and Answers

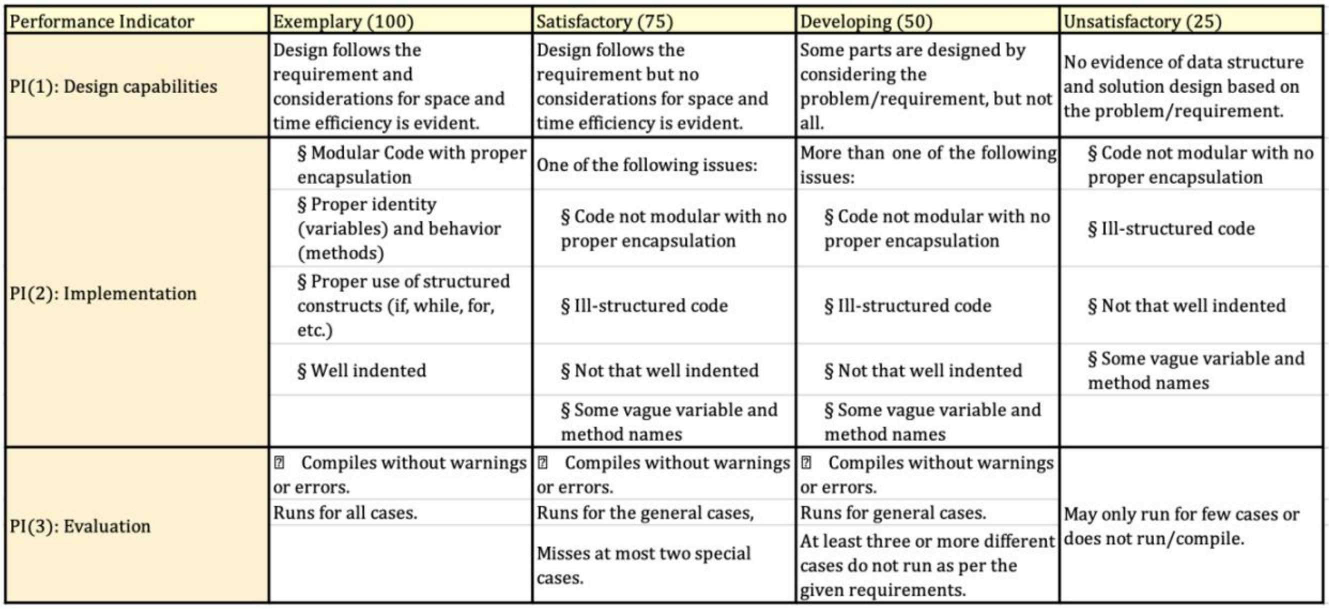
Use the following link to submit your questions, all questions will be shared and respond to publicly to keep it fair for everyone.

Questions Form: https://forms.office.com/r/tW0gW79NHS

Answers Page(will be updated periodically): Google Doc Link

## Submission Details

- Submit all Java files.
- Use comments in your code
- Cheating and copying are strictly prohibited and evidence of it will result in a zero grade. Make sure you understand the submitted code completely.
- A report containing the following:
- Zip all the items in a folder and named as YourkfupmID_Sec#
- Consider the points mentioned in the rubrics below:



| Performance Indicator | Exemplary (100) | Satisfactory (75) | Developing (50) | Unsatisfactory (25) |
|---|---|---|---|---|
| PI(1): Design capabilities | Design follows the requirement and considerations for space and time efficiency is evident. | Design follows the requirement but no considerations for space and time efficiency is evident. | Some parts are designed by considering the problem/requirement, but not all. | No evidence of data structure and solution design based on the problem/requirement. |
| PI(2): Implementation | § Modular Code with proper encapsulation | One of the following issues: | More than one of the following issues: | § Code not modular with no proper encapsulation |
| | § Proper identity (variables) and behavior (methods) | § Code not modular with no proper encapsulation | § Code not modular with no proper encapsulation | § Ill-structured code |
| | § Proper use of structured constructs (if, while, for, etc.) | § Ill-structured code | § Ill-structured code | § Not that well indented |
| | § Well indented | § Not that well indented | § Not that well indented | § Some vague variable and method names |
| | | § Some vague variable and method names | § Some vague variable and method names | |
| PI(3): Evaluation | □ Compiles without warnings or errors. | □ Compiles without warnings or errors. | □ Compiles without warnings or errors. | May only run for few cases or does not run/compile. |
| | Runs for all cases. | Runs for the general cases, | Runs for general cases. | |
| | | Misses at most two special cases. | At least three or more different cases do not run as per the given requirements. | |