Prerequisite Checker - 100 course points

Overview

This assignment focuses on directed graphs and adjacency list storage. It requires substantial time and practice to complete effectively. Students will map out course prerequisites as a Directed Acyclic Graph (DAG) to understand the relationships between courses and their prerequisites.

Prerequisites are represented by directed edges from advanced courses to their prerequisites. For example, CS112 has an edge to CS111, signifying that CS111 is a prerequisite for CS112. Courses must have all their prerequisites satisfied before they can be taken.

Implementation Overview

The assignment provides Java classes for various tasks, including adjacency list construction, valid prerequisite checks, and identifying eligible courses. All implementation should adhere to specific structural and coding guidelines.

Tasks

- AdjList: Constructs and outputs an adjacency list from course prerequisites.
- ValidPrereq: Verifies if adding a prerequisite makes the course structure invalid.
- Eligible: Determines the courses a student is eligible to take based on completed prerequisites.

Guidelines

- DO NOT use static variables in your code.
- Ensure all Java classes read and write from specified input and output files.
- Adhere strictly to the provided file structure and naming conventions.
- Use provided libraries like StdIn and StdOut for file operations.
- Avoid using System.exit() in your code.

Using StdIn and StdOut

StdIn and StdOut are used to handle file operations. Set input files using StdIn.setFile(fileName) and output files using StdOut.setFile(fileName). Methods like StdIn.readInt() and StdOut.print() facilitate efficient data handling.

Additional Notes

The project allows flexibility in designing additional classes under the src/prereqchecker folder. However, ensure all classes follow proper naming conventions and include package statements.