

Access Modifiers in this diagram may be wrong

You choose the right Access Modifiers

+ = public
= protected
- = default
~ = private

These public methods are the interface and must these methods so we return a object of the internal system like: StudentNode, Cell or History
internal system must be inaccessible

RegistrationSystem
<ul style="list-style-type: none">- studentsList: StudentsList- coursesList: CoursesList- final maxStudentsForCourse: int- final maxCoursesForStudent: int- final minCourseForStudent: int <p>+ RegistrationSystem(maxStudentsForCourse: int, minCourseForStudent: int, maxCoursesForStudent: int): void // Constructor</p> <p>+ addStudent(studentId: int): void</p> <p>+ addCourse(courseId: int): void</p> <p>+ removeStudent(studentId: int): void</p> <p>+ removeCourse(courseId: int): void</p> <p>+ getLatestStudentAddedId(): int // returns the id</p> <p>+ getLatestCourseAddedId(): int // returns the id</p> <p>+ getAllStudentsList(): int[] // Sorted by Id</p> <p>+ getAllCoursesList(): int[] // Sorted by Id</p> <p>+ displayAllStudentsList(): void // Sorted by Id</p> <p>+ displayAllCoursesList(): void // Sorted by Id</p> <p>+ addStudentToCourse(int StudentId, int courseId): void</p> <p>+ removeStudentFromCourse(int StudentId, int courseId): void</p> <p>+ getCoursesOfStudent(studentId: int): int[]</p> <p>+ getStudentsOfCourse(courseId: int): int[]</p> <p>+ displayCoursesOfStudent(studentId: int): void</p> <p>+ displayStudentsOfCourse(courseId: int): void</p> <p>+ isFullCourse(courseId: int): boolean</p> <p>+ isNormalStudent(courseId: int): boolean</p>

Node
<p># next: Node // pointer to the next node</p> <p># id: int (final variable, prevent wrong edit)</p> <p>- firstCell: Cell</p> <p>No thing</p>

StudentNode extends Node
<p>- coursesCount: int // with Getter</p> <p>- getCourses(): int[] // returns a list of IDs</p> <p>- removeAllCoursesOfMe(): void // need it when remove a student</p>

CourseNode extends Node
<p>- studentsCount: int // with Getter</p> <p>- getStudents(): int[] // returns a list of IDs</p> <p>- removeAllStudentsOfMe(): void // need it when remove a course</p>

In addStudent() and addCourse()
you as a programmer must check that the given id has not used before, so you will loop over all list items and this is O(n)
why not make the list is sorted and when add you make loop over the items to the current id and if after it this makes the list is sorted always but still O(n)

LinkedList int
<p>// This is special Singly linked list with some features i need</p> <p># head: Node</p> <p>- lastAdded: int // the id of the last added node // with Getter // default value is 0 because there is non-positive id number</p> <p>+ add(): void</p> <p>+ remove(): void</p> <p>+ getAllTheList(): int[] // returns array of IDs</p>

StudentList extends LinkedList int
<p># head: StudentNode</p> <p>...</p>

CoursesList extends LinkedList int
<p># head: CourseNode</p> <p>...</p>

History Management part

History
<p>// for enrollment operations only</p> <p># undoStack: Stack</p> <p># redoStack: Stack</p> <p>+ Undo(): void</p> <p>+ Redo(): void</p> <p>// These methods takes the appropriate operations according to the event happened</p> <p>+ studentAddedToCourse(studentId): void // or courseAddedToStudent</p> <p>+ studentRemovedFromCourse(studentId): void // courseRemovedFromStudent</p> <p>+ studentWasRemovedFromCourse(studentId: int, courseId: int): void</p> <p>+ courseWasRemovedFromCourse(studentId: int, studentId: int): void</p>

when you remove a student or course may some actions stored in the history can't be back
Need a solution or system design for this part

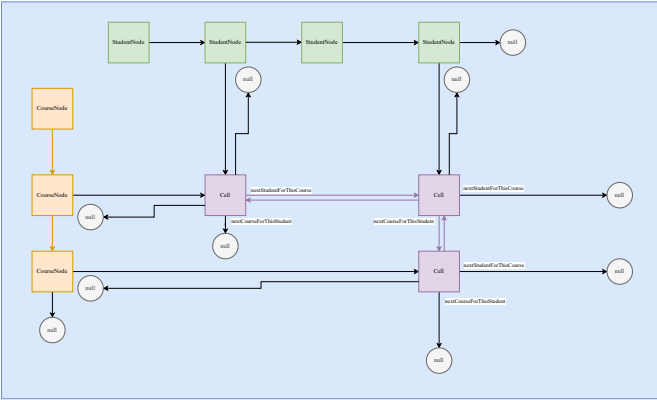
Action
<p>+ final actionType: int OR byte // stores the type of action "add" or "remove"</p> <p>- final studentId: int</p> <p>- final courseId: int</p> <p>+ Action(actionType: int, studentId: int, courseId: int) // Constructor</p> <p>+ doTheAction(): void</p> <p>+ doTheReverseAction(): void</p>

Stack
<p>// This is Stack using Singly linked list</p> <p>+ push(actions: Action[]): void</p> <p>+ pop(): Action</p> <p>+ pop(): Action[]</p> <p>+ clear() // Empty all the stack</p>

the idea of Action class you can replace it with your idea that give the correct final output

When remove a student, you will remove all the course in the stack
This is the reason why array of Actions are in the Stack

Sparse Table Example



Cell is also a Node as you know in data structures but i renamed it to cell to be easier name only but Cell is a Node

Cell // You may say EnrollmentNode
<p>// There no info part, only need know to connect this student with this course</p> <p># studentId: StudentNode (final variable)</p> <p># courseId: CourseNode (final variable)</p> <p>- nextCourseForThisStudent: Cell // if equals null, there is no next cell or enrollment</p> <p>- nextStudentForThisCourse: Cell // if equals null, there is no next cell or enrollment</p> <p>- previousCourseForThisStudent: Cell // if equals null, the previous is the StudentNode it self</p> <p>- previousStudentForThisCourse: Cell // if equals null, the previous is the CourseNode it self</p> <p>Nothing</p>