

registration Management System

Name	ID	Role
كري عماد الدين محمد يوسف	2401249596	1 , 2 , 3 , 4
عبد الرحمن محمد عبد المنعم	2401244002	5 ,6 ,7 ,8
عبد الرحمن عمر زكى	2401246024	14 ,15 ,16
عبد الرحمن محمد ابراهيم	2401241678	9 ,10
عبد الرحمن اسامة الدسوقي	2401242849	11 , 12 , 13

Printmenu function

```
public static void printMenu() {  
    System.out.println("=====");  
    System.out.println("[ 1. Add Student ]");  
    System.out.println("[ 2. Add Course ]");  
    System.out.println("[ 3. Enroll Student in Course ]");  
    System.out.println("[ 4. Remove Student from Course ]");  
    System.out.println("[ 5. enroll course to student ]");  
    System.out.println("[ 6. remove course from student ]");  
    System.out.println("[ 7. List Students by Course ]");  
    System.out.println("[ 8. List Courses by Student ]");  
    System.out.println("[ 9. Undo Last Action ]");  
    System.out.println("[ 10. Redo Last Action ]");  
    System.out.println("[ 11. Sort Students ]");  
    System.out.println("[ 12. Sort Courses ]");  
    System.out.println("[ 13. Check if Course is Full ]");  
    System.out.println("[ 14. check if student is normal student ]");  
    System.out.println("[ 15. Exit ]");  
    System.out.println("=====");  
}
```

Output :

```
Welcome to the registration Management System!
```

```
=====
```

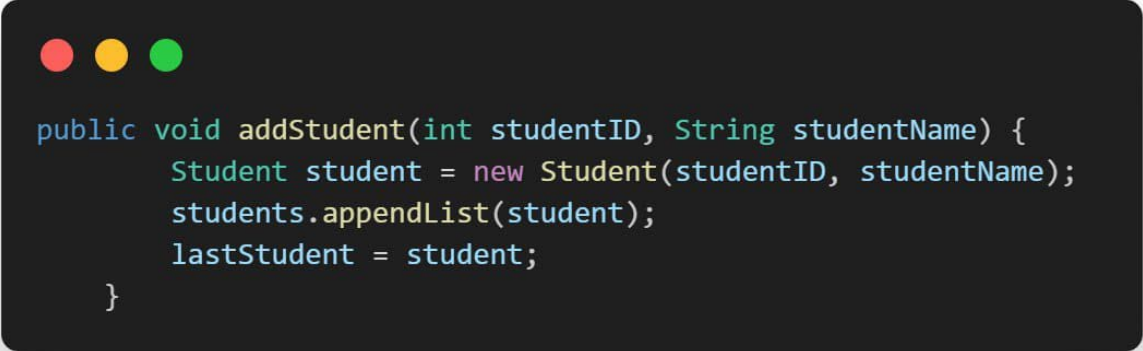
- [1. Add Student]
- [2. Add Course]
- [3. Enroll Student in Course]
- [4. Remove Student from Course]
- [5. enroll course to student]
- [6. remove course from student]
- [7. List Students by Course]
- [8. List Courses by Student]
- [9. Undo Last Action]
- [10. Redo Last Action]
- [11. Sort Students]
- [12. Sort Courses]
- [13. Check if Course is Full]
- [14. check if student is normal student]
- [15. Exit]

```
=====
```

```
Enter your choice: 
```

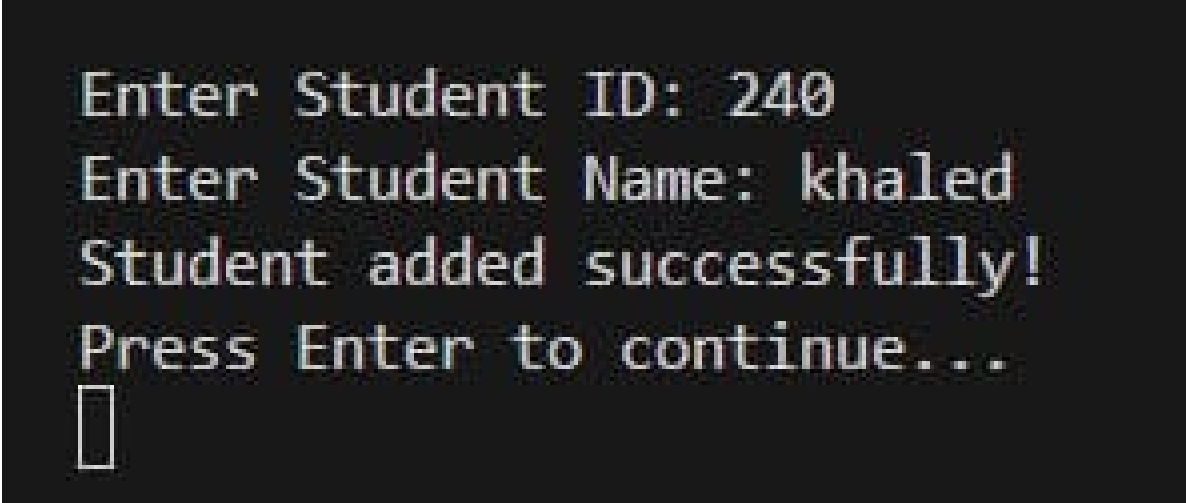
“Actions”

1- addStudent() :



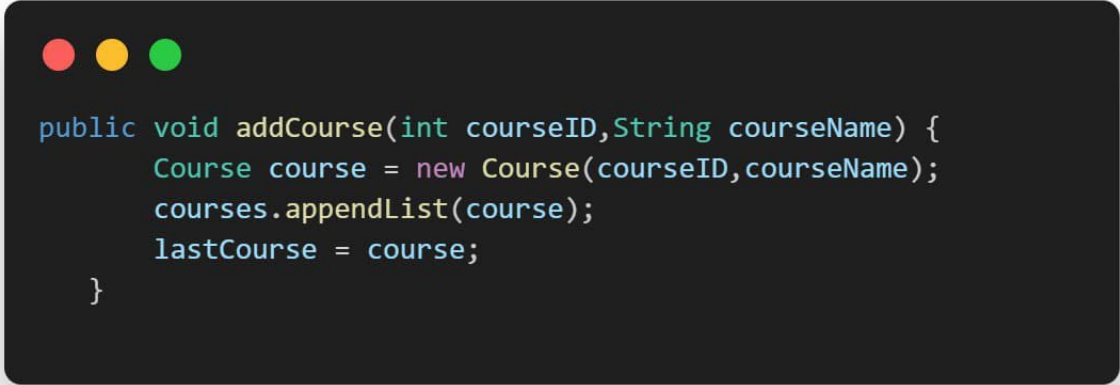
```
public void addStudent(int studentID, String studentName) {  
    Student student = new Student(studentID, studentName);  
    students.appendList(student);  
    lastStudent = student;  
}
```

output :



```
Enter Student ID: 240  
Enter Student Name: khaled  
Student added successfully!  
Press Enter to continue...  
█
```

2- addCourse() :



```
public void addCourse(int courseID,String courseName) {  
    Course course = new Course(courseID,courseName);  
    courses.appendList(course);  
    lastCourse = course;  
}
```

Output :

```
Enter Course ID: 240  
Enter Course Name: cs101  
Course added successfully!  
Press Enter to continue...  
□
```

3- enrollStudent() :

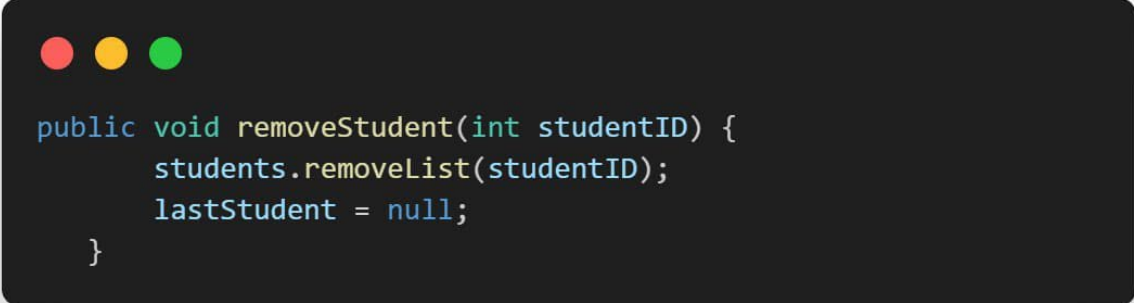
```
public void enrollStudent(int studentID, int courseID, String studentName, String courseName) {  
    Student student = findNode(students, studentID, Student.class);  
    Course course = findNode(courses, courseID, Course.class);  
    if (student != null && course != null) {  
        student.enrolledCourse.appendList(new Node(courseID, courseName));  
        course.enrolledStudents.appendList(new Node(studentID, studentName));  
        undoStack.push(new Action(student.id, student.Name, course.id, course.Name, "enroll"));  
        redoStack.clear();  
    }  
}
```

Output :

```
Enter Course ID: 140  
Enter Course Name: cs  
Student enrolled in course successfully!  
Press Enter to continue...
```

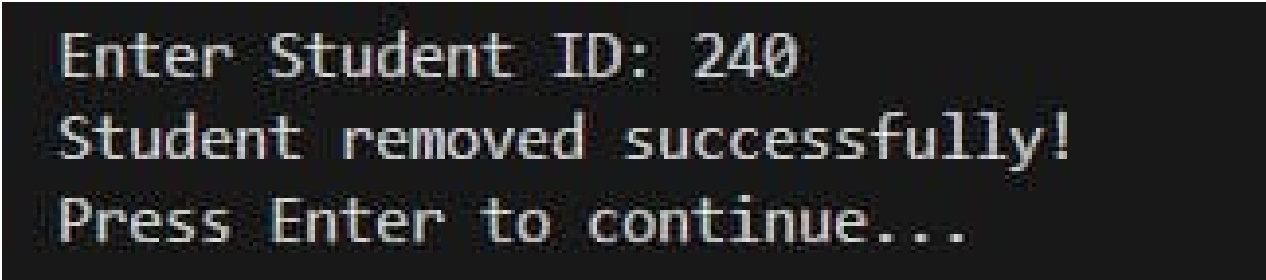


4- removeStudent():



```
public void removeStudent(int studentID) {  
    students.removeList(studentID);  
    lastStudent = null;  
}
```

Output :



```
Enter Student ID: 240  
Student removed successfully!  
Press Enter to continue...
```

5- Enroll course to student (enrollStudent()):

```
public void enrollStudent(int studentID, int courseID, String studentName, String courseName) {  
    Student student = findNode(students, studentID, Student.class);  
    Course course = findNode(courses, courseID, Course.class);  
    if (student != null && course != null) {  
        student.enrolledCourse.appendList(new Node(courseID, courseName));  
        course.enrolledStudents.appendList(new Node(studentID, studentName));  
        undoStack.push(new Action(student.id, student.Name, course.id, course.Name, "enroll"));  
        redoStack.clear();  
    }  
}
```

Output :

```
Enter Course ID: 140  
Enter Course Name: cs  
Course enrolled to student successfully!  
Press Enter to continue...
```



6- removeCourse():

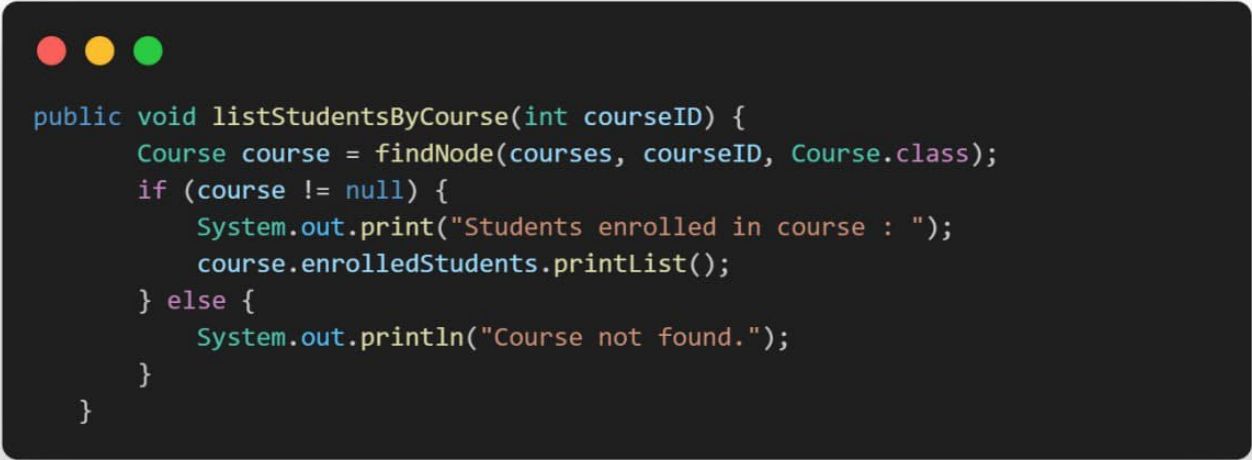


```
public void removeCourse(int courseID) {  
    courses.removeList(courseID);  
    lastCourse = null;  
}
```

Output :

```
Enter Course ID: 240  
Course removed successfully!  
Press Enter to continue...
```

7- listStudentByCourse():



```
public void listStudentsByCourse(int courseID) {  
    Course course = findNode(courses, courseID, Course.class);  
    if (course != null) {  
        System.out.print("Students enrolled in course : ");  
        course.enrolledStudents.printList();  
    } else {  
        System.out.println("Course not found.");  
    }  
}
```

Output :

```
Enter Course ID: 140  
Students enrolled in course : [ khaled , ahmed , abdelrahman]  
Press Enter to continue...
```

8- listCourseByStudent():

```
public void listCoursesByStudent(int studentID) {  
    Student student = findNode(students, studentID, Student.class);  
    if (student != null) {  
        System.out.print("Courses enrolled by student : ");  
        student.enrolledCourse.printList();  
    } else {  
        System.out.println("Student not found.");  
    }  
}
```

Output :

```
Enter Student ID: 240  
Courses enrolled by student : [ cs , cs , ai , calc]  
Press Enter to continue...
```

```
=====
```

9- Undo():

```
public void undo() {
    if (!undoStack.isEmpty()) {
        Action action = undoStack.pop();
        switch (action.ActionType) {
            case "enroll":
                removeEnrollment(action.StudentID, action.CourseID);
                redoStack.push(new Action(action.StudentID, action.StudentName, action.CourseID, action.CourseName
, "remove"));
                break;
            case "remove":
                enrollStudent(action.StudentID, action.CourseID, action.StudentName, action.CourseName);
                redoStack.push(new Action(action.StudentID, action.StudentName, action.CourseID, action.CourseName
, "enroll"));
                break;
            default:
                System.out.println("Unknown action type for undo.");
        }
    } else {
        System.out.println("No actions to undo.");
    }
}
```

Output :

```
No actions to undo.
Last action undone successfully!
Press Enter to continue...
```

```
=====
```

10- Redo():


```
public void redo() {
    if (!redoStack.isEmpty()) {
        Action action = redoStack.pop();
        switch (action.ActionType) {
            case "enroll":
                enrollStudent(action.StudentID, action.CourseID, action.StudentName, action.CourseName);
                undoStack.push(new Action(action.StudentID, action.StudentName, action.CourseID, action.CourseName
, "enroll"));
                break;
            case "remove":
                removeEnrollment(action.StudentID, action.CourseID);
                undoStack.push(new Action(action.StudentID, action.StudentName, action.CourseID, action.CourseName
, "remove"));
                break;
            default:
                System.out.println("Unknown action type for redo.");
        }
    } else {
        System.out.println("No actions to redo.");
    }
}
```

Output :

```
No actions to redo.
Last action redone successfully!
Press Enter to continue...
```

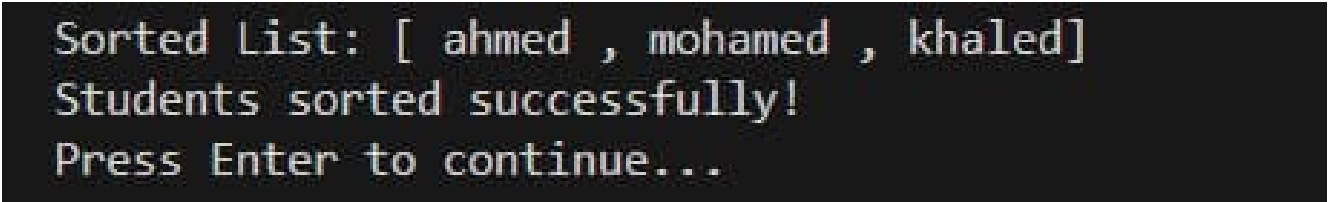
```
=====
Enroll Student 1 in Course 1
```

11- sortStudents() :




```
public void sortStudents() {  
    students.sortList();  
    students.printList();  
}
```

Output :



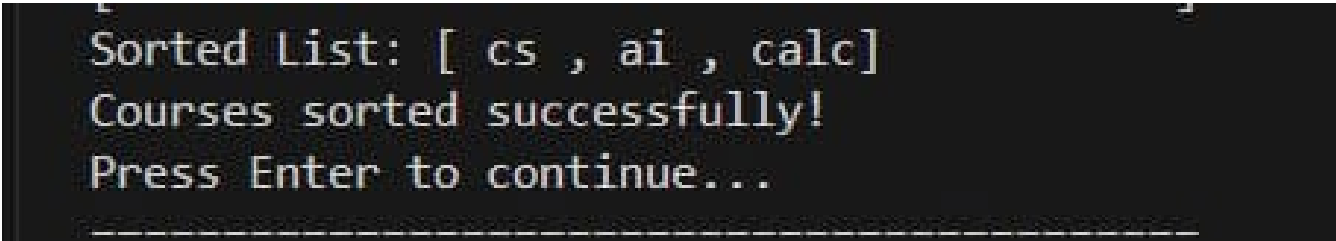
```
Sorted List: [ ahmed , mohamed , khaled]  
Students sorted successfully!  
Press Enter to continue...
```

12- sortCourses() :



```
public void sortCourses() {  
    courses.sortList();  
    courses.printList();  
}
```

Output :



```
Sorted List: [ cs , ai , calc]  
Courses sorted successfully!  
Press Enter to continue...  
_____
```

13- isFullCourse() :

```
public boolean isFullCourse(int courseID) {  
    Course course = findNode(courses, courseID, Course.class);  
    if (course != null) {  
        return course.enrolledStudents.list_length >= 30;  
    }  
    return false;  
}
```

Output :

```
Enter Course ID: 100  
The course is not full.  
Press Enter to continue...
```


14- isNormalStudent() :

```
public boolean isNormalStudent(int studentID) {  
    Student student = findNode(students, studentID, Student.class);  
    if (student != null) {  
        int n = student.enrolledCourse.list_length;  
        return n >= 2 && n <= 7;  
    }  
    return false;  
}
```

Output :

```
Enter Student ID: 10  
The student is a normal student.  
Press Enter to continue...  
=====
```