Kourosh Maissamyazad

ID: 2440557

Course Management System

Course: 420-SF2-RE DATA STRUCTURES AND OBJECT-ORIENTED PROGRAMMING

sect. 00001 – (Tue 13:00, Thu 14:00) Teacher: Yi Wang

Link to the Git repository:
https://github.com/KouroshM2006/FinalProjectKouroshMaissamyazad.git

Table of Contents

---

1. Project Description

The **Course Management System (CMS)** is a Java application designed for educational institutions to manage courses, students, teachers, and assignments. It fulfills the following requirements:

- **Class hierarchies** (2+ layers)
- **User-defined interface** (Creatable)
- **Runtime polymorphism** (e.g., getRole(), getType())
- **Text I/O** (importing users from CSV files and writing schedules)
- **Comparable/Comparator** (AcademicItem.compareTo, CourseComparator)
- **Junit testing**

---

2. Program Features & Implementation

Class Hierarchies and Methods

1. User Hierarchy (User **Abstract Class)**

Methods:

- getRole() (Abstract): Returns role-specific strings ("Administrator", "Teacher", "Student").

Subclasses:

- Administrator (implements Creatable interface):
    - createTeacher()**/**createStudent(): adds users to the system.
    - importUsers(): Reads users from CSV (Text I/O requirement).
    - createCourse(): creates courses and checks for duplicate course titles before creation
- Teacher:
    - createAssignment(): Creates assignments for their courses.
    - gradeAssignment(): Adds feedback to submissions
    - modifyAssignmentDueDate(): changes the due date of an assignment to one chosen
- Student:
    - enrollCourse(): enrolls student into a course, checks capacity before enrollment
    - downloadSchedule(): Writes course schedule to CSV (Text I/O).
    - submitAssignment(): submits an assignment, validates due date before submission
    - viewFeedback(): Checks is submission is graded before displaying feedback

2. AcademicItem Hierarchy (AcademicItem **Abstract Class)**

Methods:

- getType() (Abstract): Returns "Course" or "Assignment".
- compareTo(): Implements Comparable<AcademicItem> for sorting by createdDate.

Subclasses:

- Course:
    - addStudent(): Manages enrollment (checks capacity).
    - CourseComparator: Static nested class to sort courses by given parameter (Comparator requirement).
- Assignment:

- addSubmission(): adds a new submission to the assignment with the student as its key.
- addFeedback(): adds Feedback and a given score to a submission.

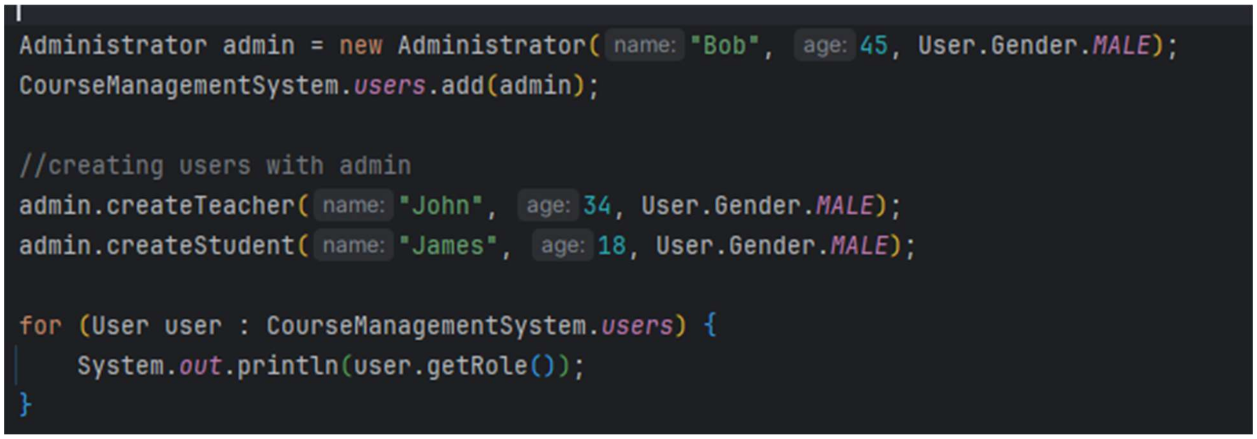
3. Supporting Classes

- Submission: Stores file, timestamp, and feedback.
- Gender: Enum for user gender.

Key Functionalities with Code Examples

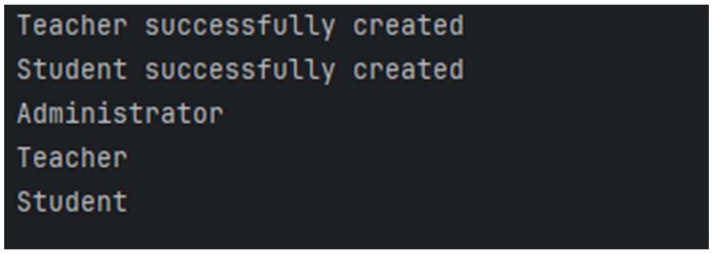
1. **Runtime Polymorphism with getRole() or getType() methods:** administrators can create user and add them to the system and each user has a getRole() method that shows their role

Example:



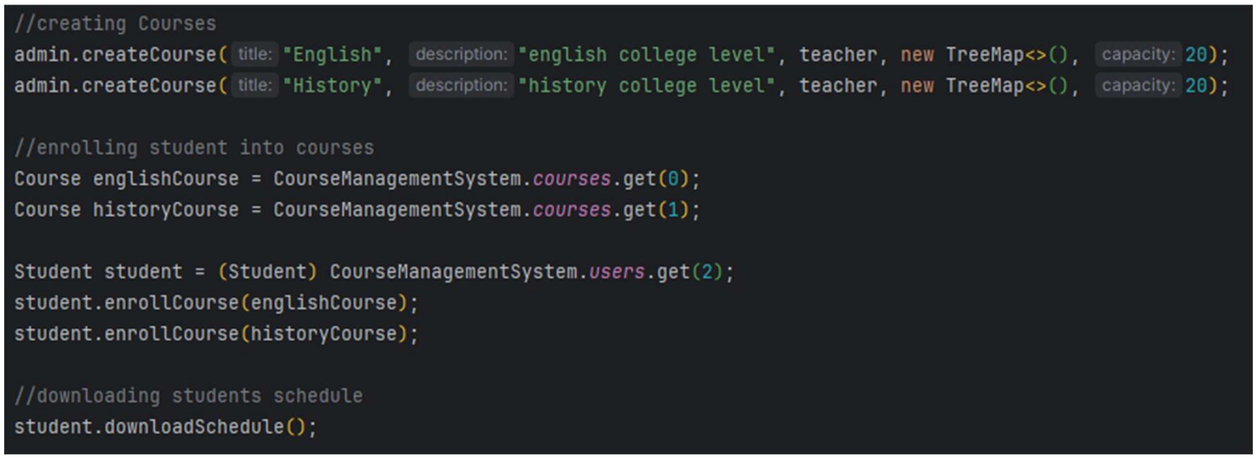
```
Administrator admin = new Administrator( name: "Bob",  age: 45, User.Gender.MALE);
CourseManagementSystem.users.add(admin);

//creating users with admin
admin.createTeacher( name: "John",  age: 34, User.Gender.MALE);
admin.createStudent( name: "James",  age: 18, User.Gender.MALE);

for (User user : CourseManagementSystem.users) {
    System.out.println(user.getRole());
}
```

Output:



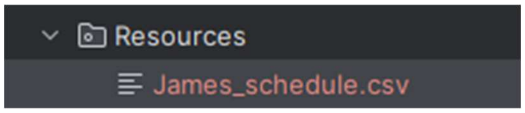
```
Teacher successfully created
Student successfully created
Administrator
Teacher
Student
```

2. **Text I/O writing with downloadSchedule method:** students can enroll into courses and later download their course schedule into a csv file under their name:
   Example:

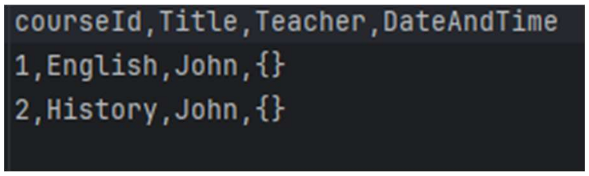


```
//creating Courses
admin.createCourse( title: "English",  description: "english college level", teacher, new TreeMap<>(),  capacity: 20);
admin.createCourse( title: "History",  description: "history college level", teacher, new TreeMap<>(),  capacity: 20);

//enrolling student into courses
Course englishCourse = CourseManagementSystem.courses.get(0);
Course historyCourse = CourseManagementSystem.courses.get(1);

Student student = (Student) CourseManagementSystem.users.get(2);
student.enrollCourse(englishCourse);
student.enrollCourse(historyCourse);

//downloading students schedule
student.downloadSchedule();
```

   Output:



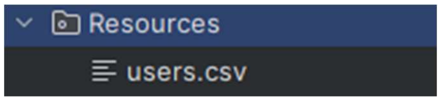
```
Resources
    James_schedule.csv
```



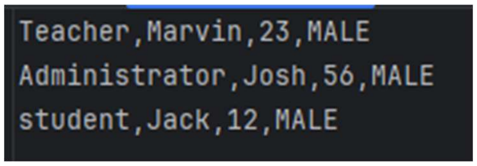
```
courseId,Title,Teacher,DateAndTime
1,English,John,{}
2,History,John,{}
```

3. **Text I/O Reading with importUser method:** administrators can call the importUser method to add users to the system from the user.csv file in the Resources folder

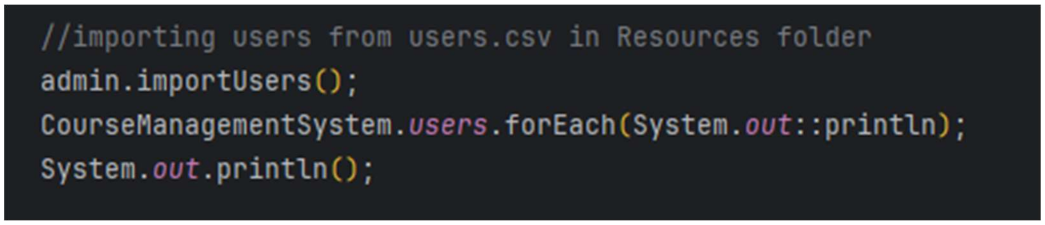
   Example:



```
Resources
    users.csv
```

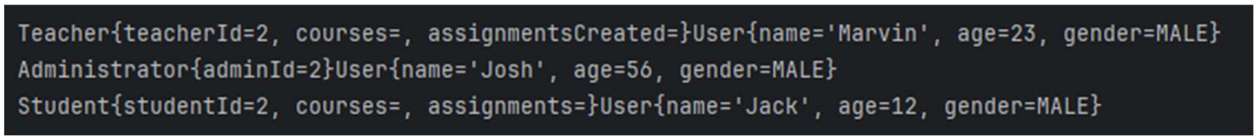


```
Teacher,Marvin,23,MALE
Administrator,Josh,56,MALE
student,Jack,12,MALE
```



```
//importing users from users.csv in Resources folder
admin.importUsers();
CourseManagementSystem.users.forEach(System.out::println);
System.out.println();
```

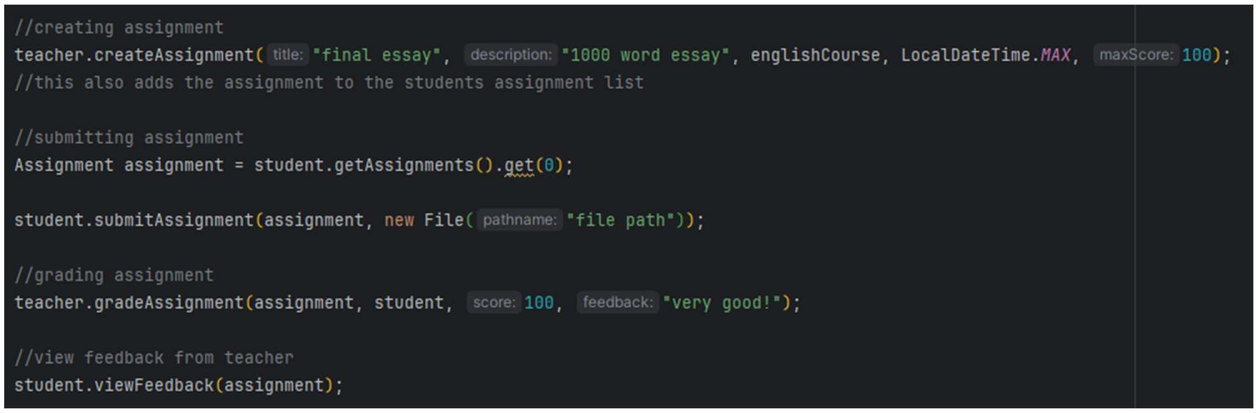
Output:



```
Teacher{teacherId=2, courses=, assignmentsCreated=}User{name='Marvin', age=23, gender=MALE}
Administrator{adminId=2}User{name='Josh', age=56, gender=MALE}
Student{studentId=2, courses=, assignments=}User{name='Jack', age=12, gender=MALE}
```

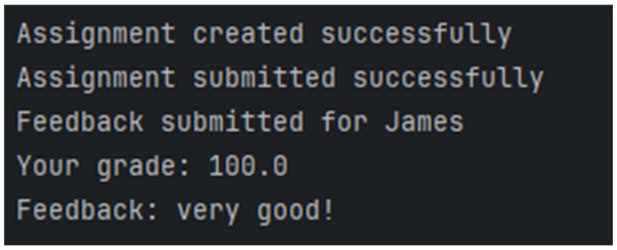
4. **Interaction between user types**: Teachers can create assignments and grade assignments while students can submit assignments and view feedback from their teachers

Example:



```
//creating assignment
teacher.createAssignment( title: "final essay", description: "1000 word essay", englishCourse, LocalDateTime.MAX, maxScore: 100);
//this also adds the assignment to the students assignment list

//submitting assignment
Assignment assignment = student.getAssignments().get(0);

student.submitAssignment(assignment, new File( pathname: "file path"));

//grading assignment
teacher.gradeAssignment(assignment, student, score: 100, feedback: "very good!");

//view feedback from teacher
student.viewFeedback(assignment);
```

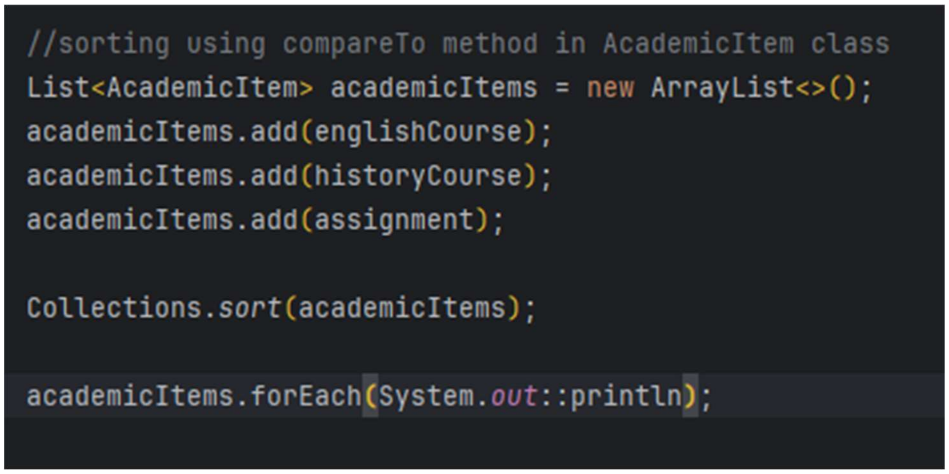
Output:



```
Assignment created successfully
Assignment submitted successfully
Feedback submitted for James
Your grade: 100.0
Feedback: very good!
```

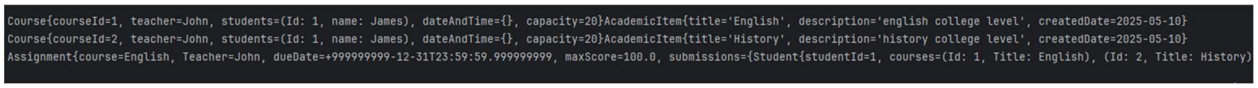
5. **Sorting with Comparable/Comparator:** AcademicItems can be sorted by createdDate through the compareTo method in the AcademicItem class and Courses can also be sorted by a given parameter using the courseComparator subclass in the Course class

Example 1:



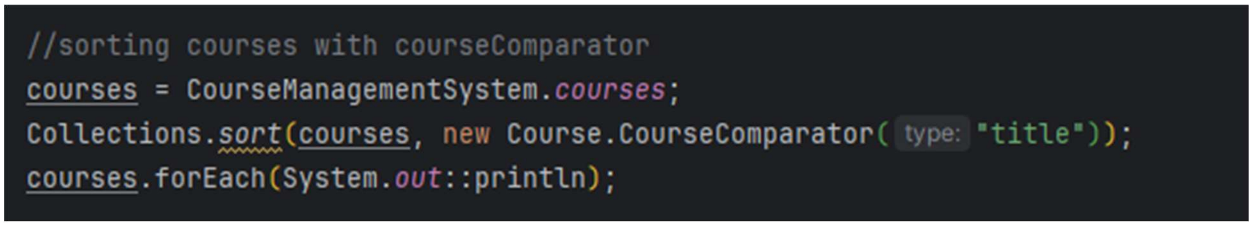
```
//sorting using compareTo method in AcademicItem class
List<AcademicItem> academicItems = new ArrayList<>();
academicItems.add(englishCourse);
academicItems.add(historyCourse);
academicItems.add(assignment);

Collections.sort(academicItems);

academicItems.forEach(System.out::println);
```

Output:



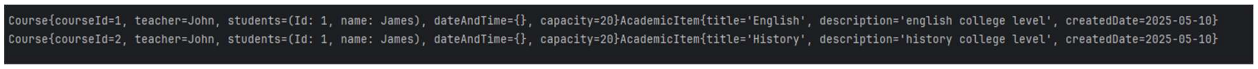
```
Course{courseId=1, teacher=John, students=(Id: 1, name: James), dateAndTime={}, capacity=20}AcademicItem{title='English', description='english college level', createdDate=2025-05-10}
Course{courseId=2, teacher=John, students=(Id: 1, name: James), dateAndTime={}, capacity=20}AcademicItem{title='History', description='history college level', createdDate=2025-05-10}
Assignment{course=English, Teacher=John, dueDate=+999999999-12-31T23:59:59.999999999, maxScore=100.0, submissions={Student{studentId=1, courses=(Id: 1, Title: English), (Id: 2, Title: History)
```

Example 2:



```
//sorting courses with courseComparator
courses = CourseManagementSystem.courses;
Collections.sort(courses, new Course.CourseComparator( type: "title"));
courses.forEach(System.out::println);
```

Output:



```
Course{courseId=1, teacher=John, students=(Id: 1, name: James), dateAndTime={}, capacity=20}AcademicItem{title='English', description='english college level', createdDate=2025-05-10}
Course{courseId=2, teacher=John, students=(Id: 1, name: James), dateAndTime={}, capacity=20}AcademicItem{title='History', description='history college level', createdDate=2025-05-10}
```

Requirements Fulfillment :

Requirement	**Implementation Example**
Class hierarchies	User → Teacher/Student; AcademicItem → Course/Assignment
Interface	Creatable with createTeacher()/createStudent(), importUser(), createCourse();
Runtime polymorphism	getRole(), getType(), compareTo()
Text I/O	Student.downloadSchedule() writes CSV Administrator.importUser() reads CSV
Comparable/Comparator	AcademicItem.compareTo(), CourseComparator

Challenges

Due to bidirectional relationships between classes (e.g., Course ↔ Student) I had to customize the equals and hashCode methods as well as the toString methods for all subclasses because they would throw stack overflow errors, however the rest of the project was pretty straight forward.

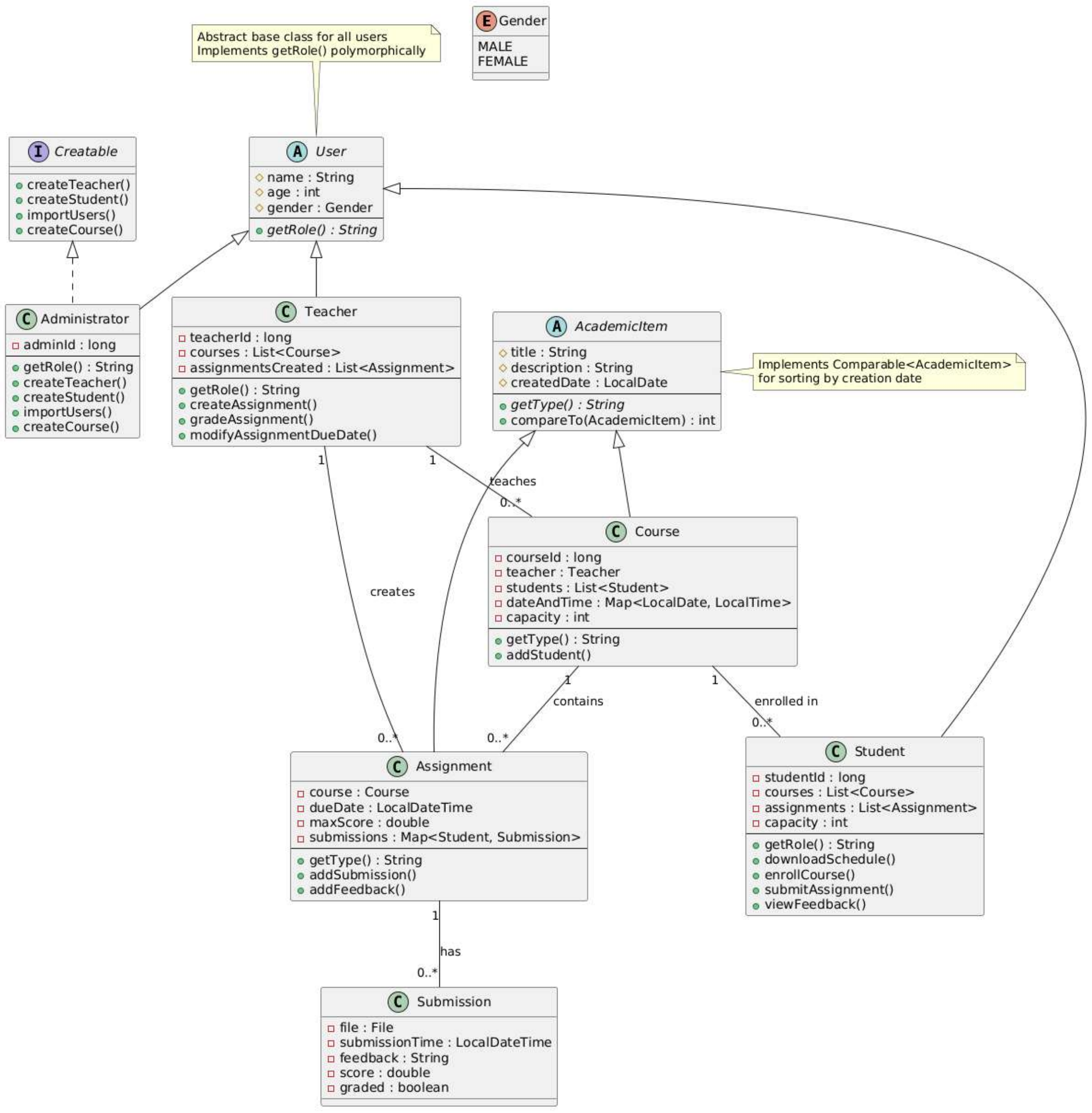
I also decided to not implement one of the original methods called uploadSchedule in the Student class because I realized that students can only be given schedules by administrators and not by themselves and so I decided to replace that method with the importUser method currently implemented in the Administrator class.

Learning outcomes

I learned a few tricks during the making of the project such as using List.of() when making new Arraylists. I learned how to use ByteArrayOutputStream in Junit testing to check the output of System.out.println.

The project Also helped me developpe more my skills in inheritance, interfaces and polymorphism. I learned more about planning and organizing projects and Debugging errors in my code

New class diagram for the project:



Abstract base class for all users
Implements getRole() polymorphically
E Gender
MALE
FEMALE
I Creatable
createTeacher()
createStudent()
importUsers()
createCourse()
A User
name : String
age : int
gender : Gender
getRole() : String
C Administrator
adminId : long
getRole() : String
createTeacher()
createStudent()
importUsers()
createCourse()
C Teacher
teacherId : long
courses : List<Course>
assignmentsCreated : List<Assignment>
getRole() : String
createAssignment()
gradeAssignment()
modifyAssignmentDueDate()
A AcademicItem
title : String
description : String
createdDate : LocalDate
getType() : String
compareTo(AcademicItem) : int
Implements Comparable<AcademicItem>
for sorting by creation date
1
1
teaches
0..*
creates
C Course
courseId : long
teacher : Teacher
students : List<Student>
dateAndTime : Map<LocalDate, LocalTime>
capacity : int
getType() : String
addStudent()
1
contains
1
enrolled in
0..*
0..*
0..*
C Assignment
course : Course
dueDate : LocalDateTime
maxScore : double
submissions : Map<Student, Submission>
getType() : String
addSubmission()
addFeedback()
C Student
studentId : long
courses : List<Course>
assignments : List<Assignment>
capacity : int
getRole() : String
downloadSchedule()
enrollCourse()
submitAssignment()
viewFeedback()
1
has
0..*
C Submission
file : File
submissionTime : LocalDateTime
feedback : String
score : double
graded : boolean