Iteration One: 8 October – 20 October

On-going iteration log

Functions: Add\remove teacher to class, add\remove student to class.

Goals: Analysis and design of functions, application tier implementation and test definition.

Analysis

Use cases for object creation and deletion :

Create/Delete Class

The user indicates that they wish to create a class. The data management system creates a new row for the class. The system then takes the users specified information about the new class and adds it to the data management system. The new class is now able to be displayed in the browser.

The user indicates they wish to delete a class. After the user specifies the class they wish to be removed the system will disallow access to this class for a set period of time in case the users wish to restore this class. Once the time period is up and the user has not restored the class the data management system deletes the class permanently.

Add/Delete Student to Class

The user indicates that they wish to add a student to a class. The system takes two parameters the student id and class id. It then goes to the data management system, in the table/file where the class's enrolment information is stored. The system checks if these two parameters already exist together. If it does exists an appropriate error message is displayed, if the entry does not exist it is then added to the data management system.

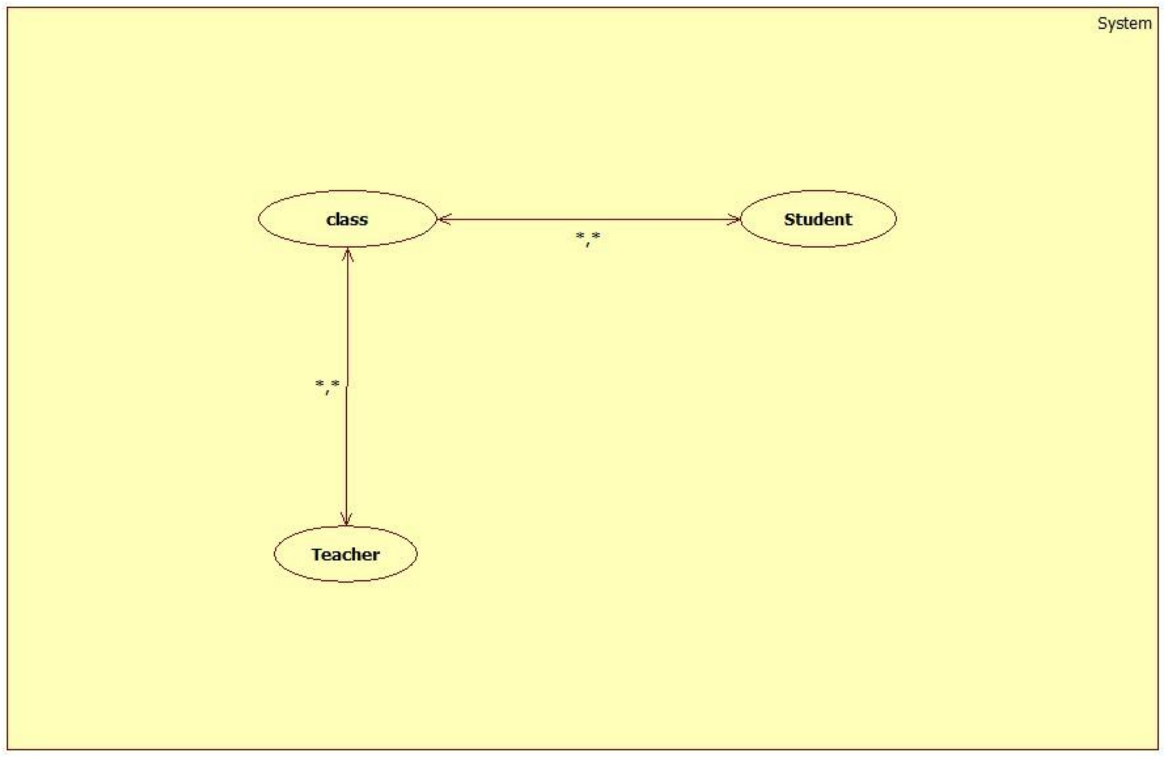
The user indicates that they wish to delete a student from a class. The system looks for the student id and class id in the class enrolment table/file. If it finds an entry it is removed to a temporary table in case the user made a mistake. After a set period of time the student is then permanently removed from the data management system.

Add/Delete Teacher to Class

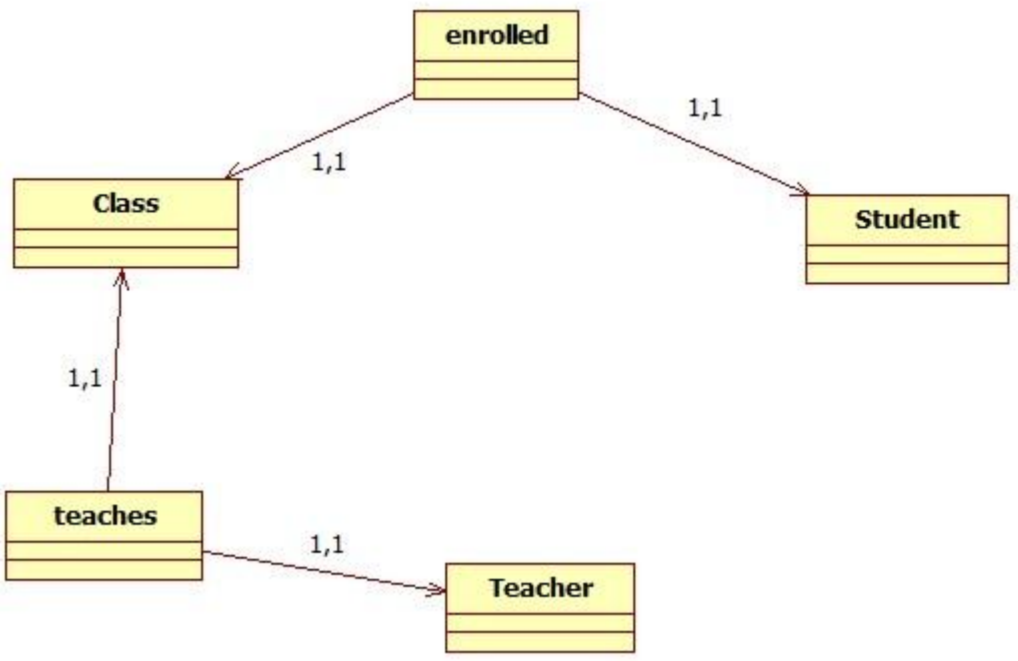
The user indicates that they wish to add a teacher to a class. The system takes two parameters the teacher id and class id. It then goes to the data management system, in the table/file where the class's enrolment information is stored. The system checks if these two parameters already exist together. If it does exists an appropriate error message is displayed, if the entry does not exist it is then added to the data management system.

The user indicates that they wish to delete a teacher from a class. The system looks for the teacher id and class id in the class enrolment table/file. If it finds an entry it is removed to a temporary table in case the user made a mistake. After a set period of time the teacher is then permanently removed from the data management system.

Underlying the use case at the most abstract level is the following object model – three major elements (Class, Student, Teacher) and two many-to-many links.

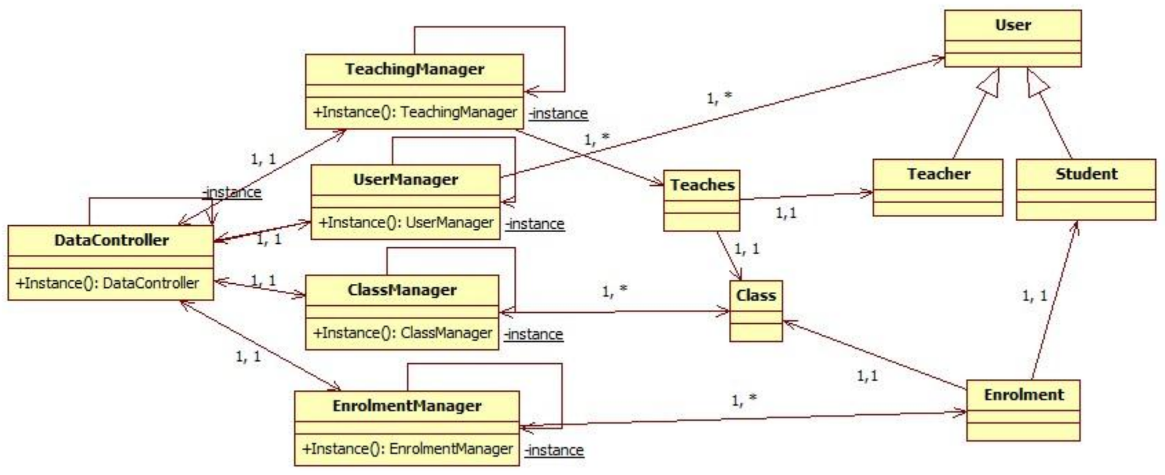


Design



In the first iteration of design we use association classes “Enrolled” and “Teaches” to link Students to Classes and Teachers to Classes.

The need is clear for some structure to store and manage collections of the objects in the system, hence the addition of several manager classes and another “DataController” class. All five of these use the Singleton design pattern as there is never any need to have more than one of these objects. Manager classes are responsible for storing all instances of their component objects, controlling external access to these objects, adding and removing objects (ensuring duplicates are not allowed to be added). Additionally, commonality from the Student and Teacher classes has been abstracted into a User base class, which also conveniently allows further extension in later iterations of development.



Implementation and Testing

With requirements and a preliminary model in place tests can be defined and implementation can be begun in parallel.

One minor change arose as an immediate result implementation: to alleviate on-going confusion the "Class" class was renamed to "ClassGroup" – the changed terminology is used from here on in. Furthermore, a TimeOutList class has been created to manage lists of deleted data to be stored for a time before final deletion. This functionality is not completed, as will rely of the database management system which is not a part of the application layer we focused on in this iteration.

Test conditions for the objects created in this iteration:

- Enrolment
  - On object creation
    - Student Reference must be non-null
    - Class reference must be non null
    - Check that valid inputs result in an object
    - Check any combination of invalid inputs results in an exception
  - Object use
    - Check two objects referring to the same Student/Class objects are identified as equal

- Teaches
  - On object creation
    - Teacher Reference must be non-null
    - Class reference must be non null
    - Check that valid inputs result in an object
    - Check any combination of invalid inputs results in an exception
  - Object use
    - Check two objects referring to the same Teacher/Class objects are identified as equal

- User (Teacher & Student)
  - On object creation
    - name property must be a non null and non empty String
    - Check that valid inputs result in an object
    - Check any combination of invalid inputs results in an exception
  - Object use
    - Check two users the same sub class and identity values are identified as equal

- Test plan ClassGroup

- On object creation
  - name and description properties must be a non-null and non-empty String
  - Check that valid inputs result in an object
  - Check any combination of invalid inputs results in an exception
- Object use
  - Check two ClassGroup objects with the same identity(name) values are identified as equal

- EnrolmentManager
  - Ensure only one instance exists (Singleton)
  - Ensure an object can be added to the list correctly if a duplicate does not exist
    - Check the list of enrolments contains the added object after adding an object
    - Check the list of deleted enrolments does not contain the added object after adding an object
    - Check an object is not added to the list if a duplicate is already in the list
  - Ensure an object can be deleted correctly
    - Check the list of enrolments does not contain the deleted object after deleting an object
    - Check the list of deleted enrolments does contain the deleted object after deleting an object
  - Ensure an object can be restores correctly
    - Check the list of enrolments does contain the deleted object after restoring an object
    - Check the list of deleted enrolments does not contain the deleted object after restoring an object

- TeachingManager
  - Ensure only one instance exists (Singleton)
  - Ensure an object can be added to the list correctly if a duplicate does not exist
    - Check the list of Teaches objects contains the added object after adding an object
    - Check the list of deleted Teaches objects does not contain the added object after adding an object
    - Check an object is not added to the list if a duplicate is already in the list
  - Ensure an object can be deleted correctly
    - Check the list of Teaches objects does not contain the deleted object after deleting an object
    - Check the list of deleted Teaches objects does contain the deleted object after deleting an object
  - Ensure an object can be restores correctly
    - Check the list of Teaches objects does contain the deleted object after restoring an object

        - Check the list of deleted Teaches objects does not contain the deleted object after restoring an object

- UserManager
    - Ensure only one instance exists (Singleton)
    - Ensure an object can be added to the list correctly if a duplicate does not exist
        - Check the list of Users contains the added object after adding an object
        - Check the list of deleted Users does not contain the added object after adding an object
        - Check an object is not added to the list if a duplicate is already in the list

- ClassManager
    - Ensure only one instance exists (Singleton)
    - Ensure an object can be added to the list correctly if a duplicate does not exist
        - Check the list of Classes contains the added object after adding an object
        - Check the list of deleted Classes does not contain the added object after adding an object
        - Check an object is not added to the list if a duplicate is already in the list

Testing has identified several errors made in code mostly missing functions, especially class specific overrides of the generic Object.equals() method.

As of the end of the iteration, all current code passes tests and can be released in its current form.