Project Part 4: Progress Report 2

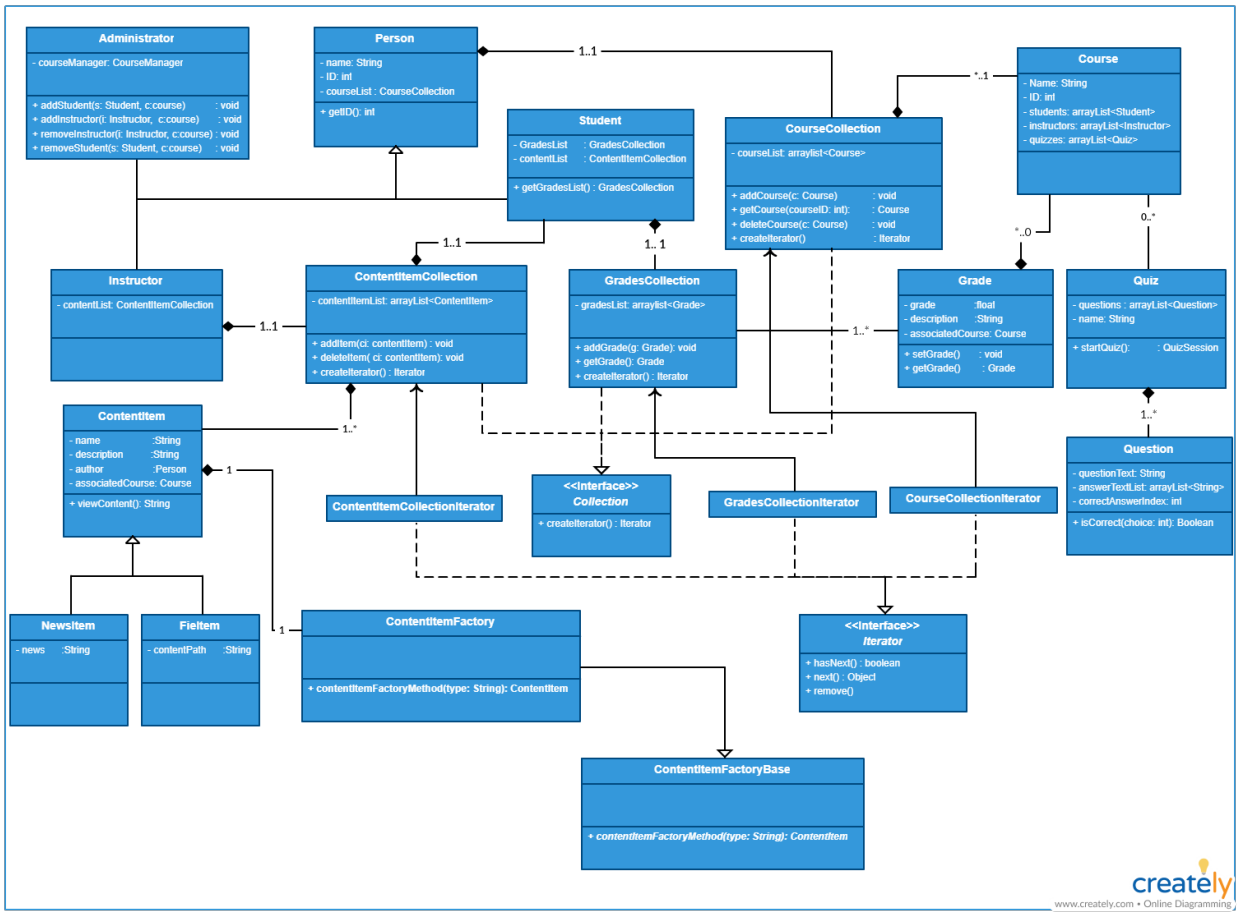
1. **Project**:

Team 42: Erik Holbrook (github:zephyr1999) and Ahmed Al Hasani (github:AhmedAlHasani)

Schoodle is a lightweight, web-based college course management application. It allows students to view grades and course content, and complete assignments. Instructors can upload content and assign grades. Administrators manage enrollment of students, creation of courses, and assigning instructors to courses.

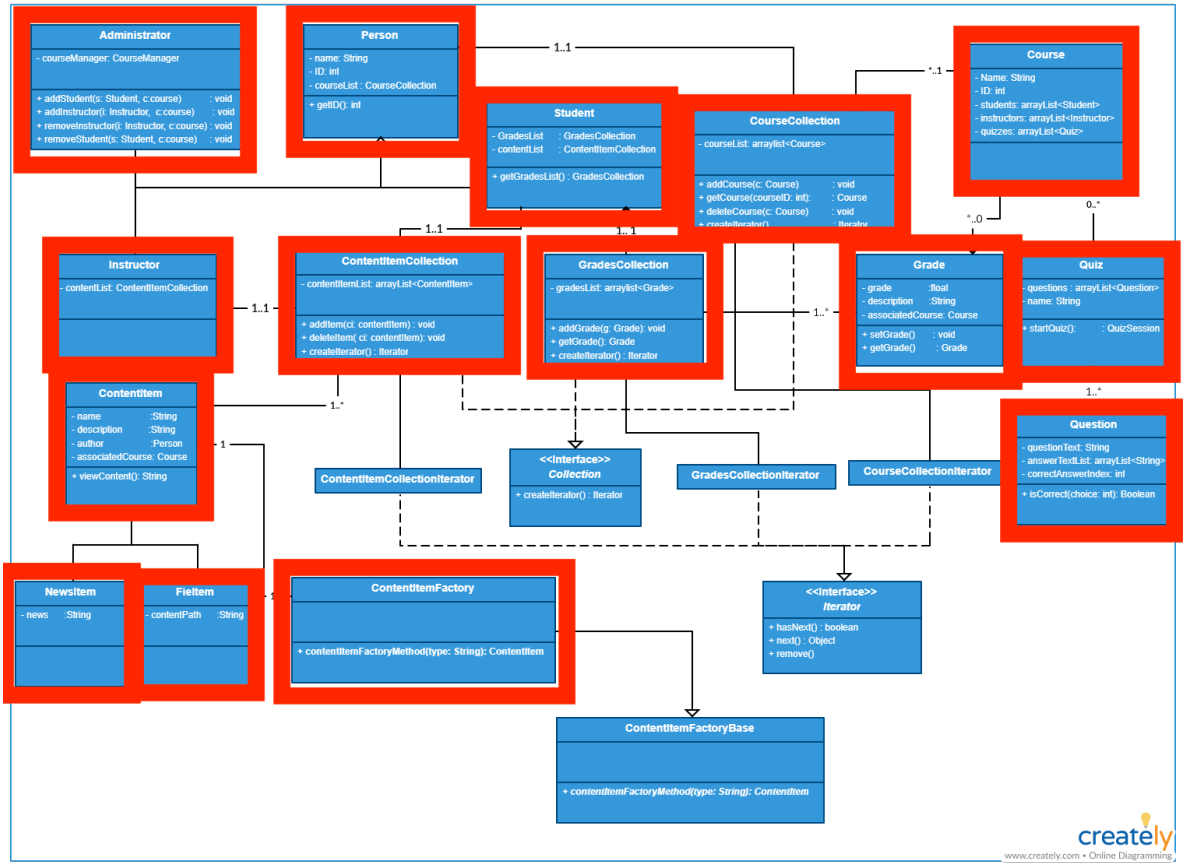
2. **Previous Class Diagram**

This diagram has been updated from progress report 1 with classes following the design patterns in GoF (iterator, factory method).



3. **Completed class diagram**

Classes outlined in red have been implemented.



4. **Summary:**

The past two weeks have been spent solely in the implementation of classes in the “models” folder of the github repo. At the time of the previous progress report, only 2 classes had been (partially) implemented: Course and Person. The intervening weeks have seen the creation of the remaining 12 classes, and partial implementation of the utilized design patterns.

5. **Breakdown:**

Erik worked on implementing and refactoring the following classes:

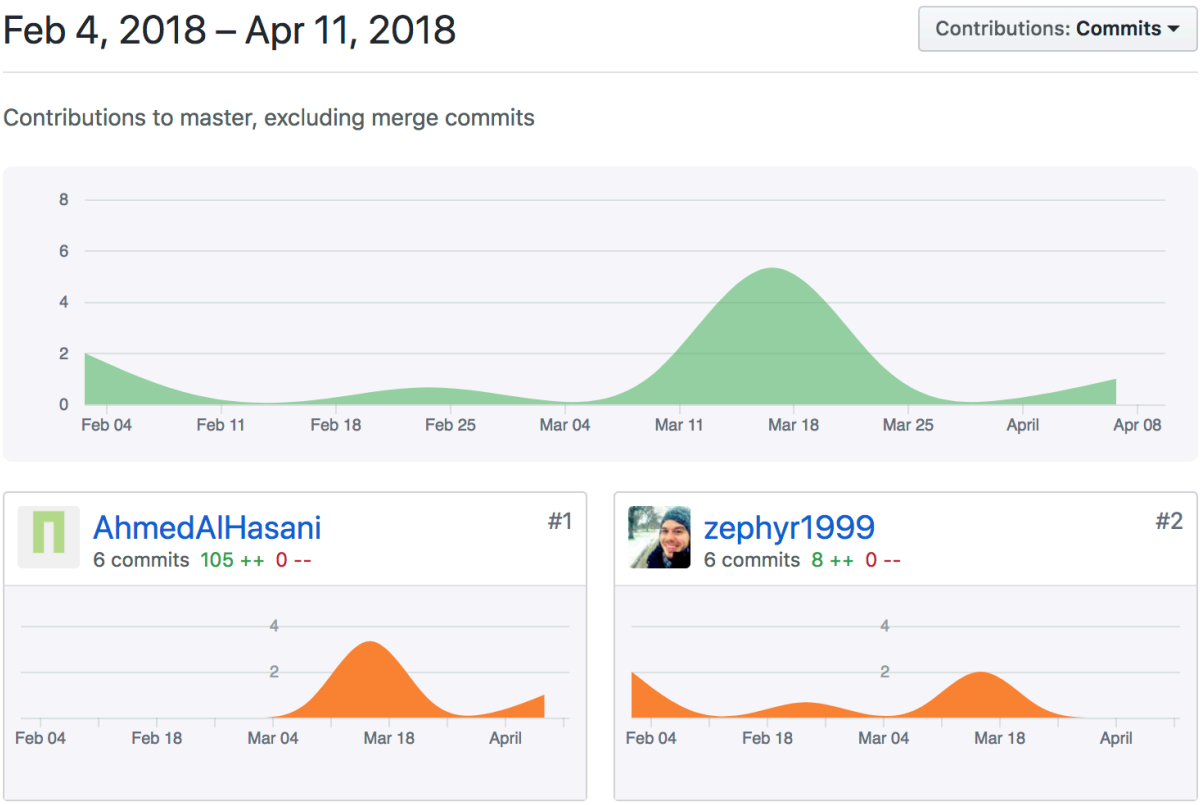
- Person
- Student
- Instructor

- Administrator
- CourseManager
- Course
- CourseCollection

Ahmed worked on implementing and refactoring the following classes:

- ContentItem
- FileItem
- NewsItem
- ContentItemFactory
- Grade
- GradesCollection
- Quiz
- Question

6. **Github Contribution Graph**



7. **Estimate Remaining Effort:**

As reported in the previous progress report, our goal was to finish the class implementation for this progress report. What remains is the interface implementation, which amounts to little

more than creating and connecting Django View objects to the correct Models, and configuring the Django URL Routing. We estimate this encompasses about **1 week** of work to implement entirely, though further refinement of the UI could take considerably more time (more on this below).

Additionally, we have a significant amount of refactoring and bug squashing. Development with proper OOD principles inside of the context of Django is somewhat convoluted; Django requires specific object code beyond the normal Python syntax, like the @-syntax annotations for Java's Hibernate. Correct delegation and composition of objects can be tedious; we have largely ignored in-depth testing and bug squashing to this point for the sake of completing classes. We estimate this refactoring will also take **1 week** of effort.

From the above estimates, we have approximately **a total of 2 weeks** of coding-focused effort to complete our project, which will allow us several additional days to prepare the presentation. **Any additional time** will be used to refine the appearance of the user-interface, though since this is an ungraded portion of the project, it is not a high priority.

8. **Design Patterns:**

Each of the following patterns correlates by number to the diagrams in **part 9.**

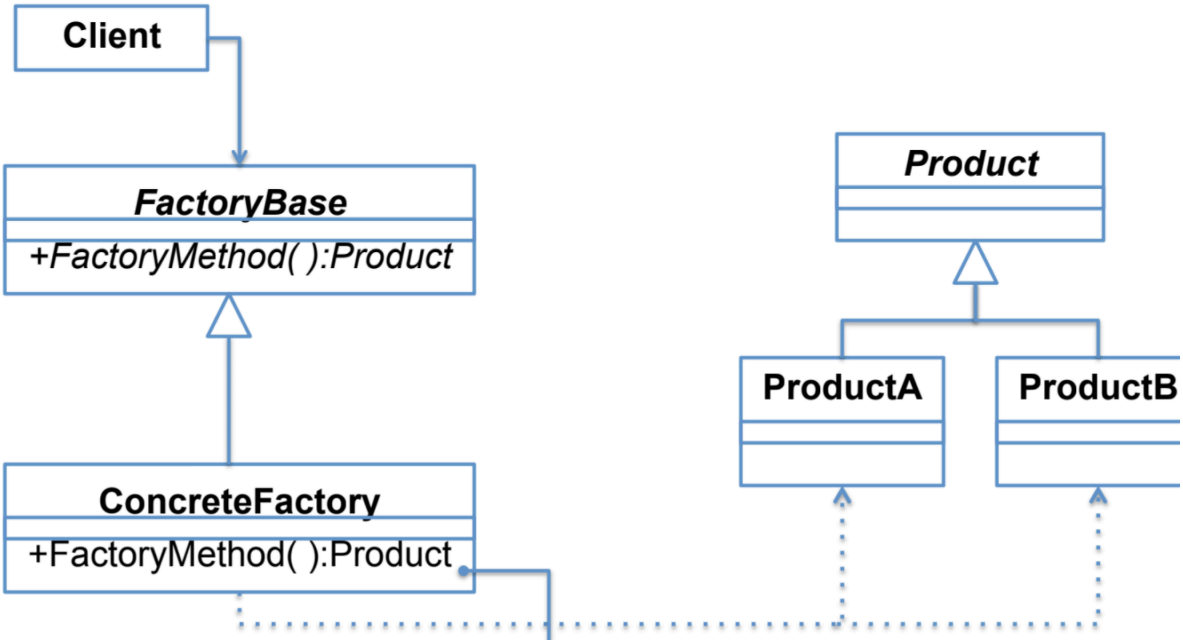
1. **Factory Method pattern**: This pattern has been implemented with the ContentItem Superclass and the ContentItemFactory to allow the system to dynamically create the correct child class.
2. **Iterator pattern**: We make heavy use of this pattern with our Collection objects: CourseCollection, ContentItemCollection, GradesCollection. Each of these classes encapsulates a list of items and contains methods which iterate over all of the items.

We only implement 2 distinct patterns, but the iterator appears in several different places.

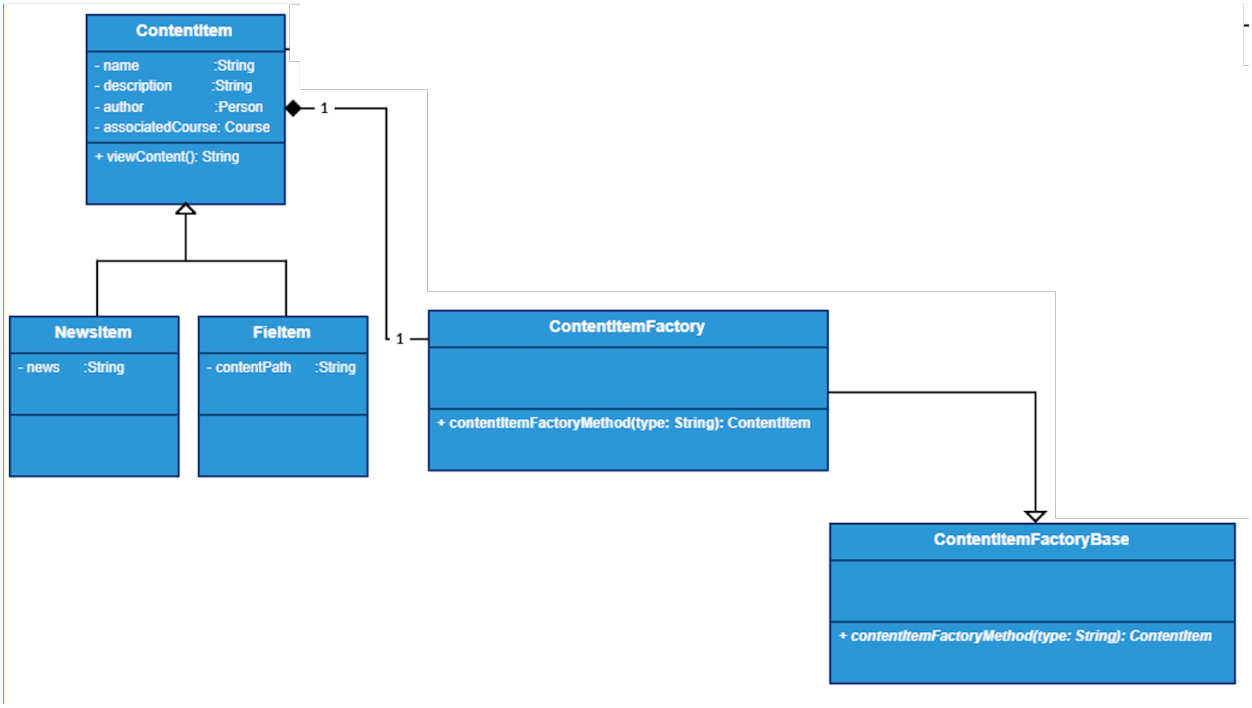
9. **Design Pattern Diagrams:**

1. **Factory pattern:**

a. Actual pattern from GOF:

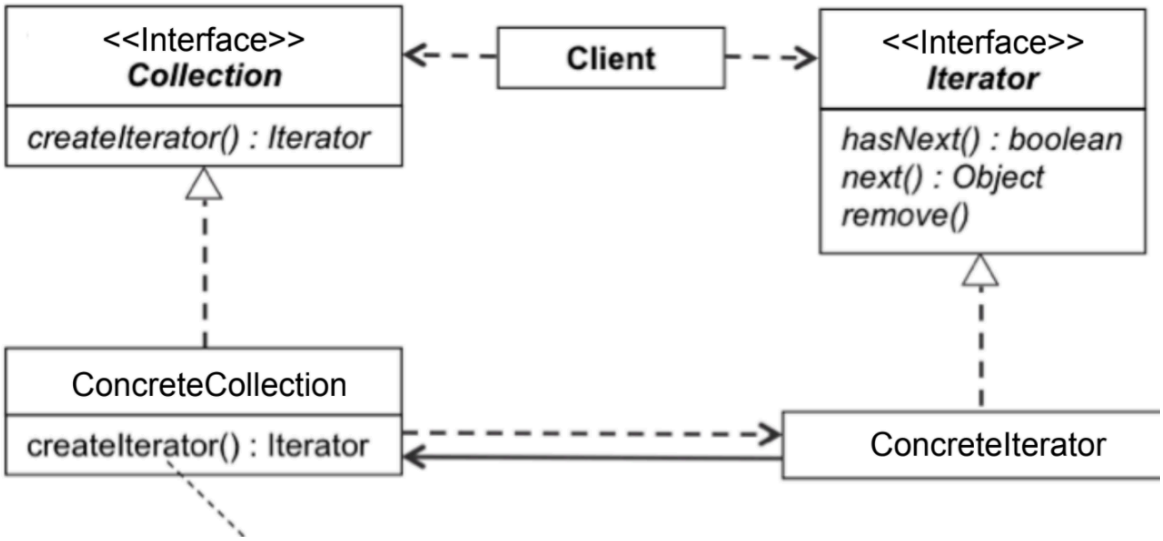


b. Portion of our diagram:

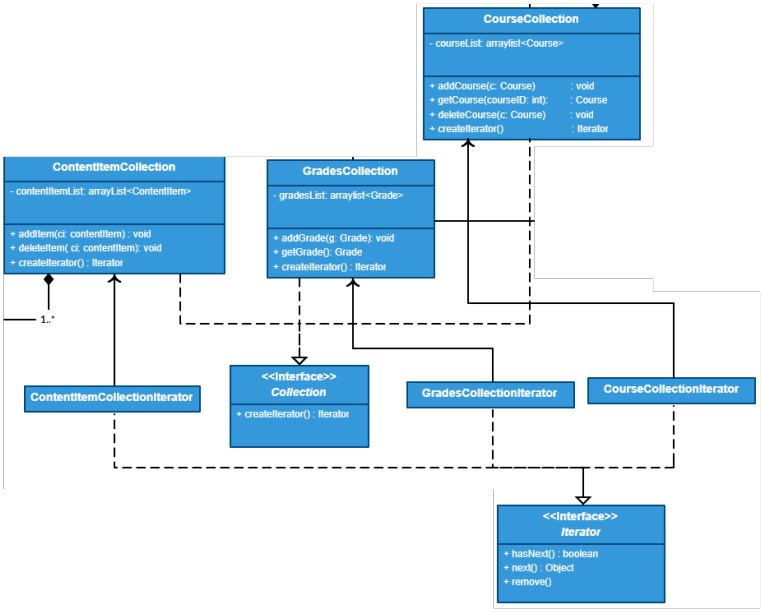


c. Participants: ContentItemFactoryBase, ContentItemFactory, ContentItem, FileItem,NewsItem

2. **Iterator pattern:**
    a. Actual pattern from GOF:



    b. Portion of our diagram:



c. Participants: Collection,Iterator, CourseCollection, ContentItemCollection, GradesCollection, CourseIterator, ContentItemIterator, GradesIterator