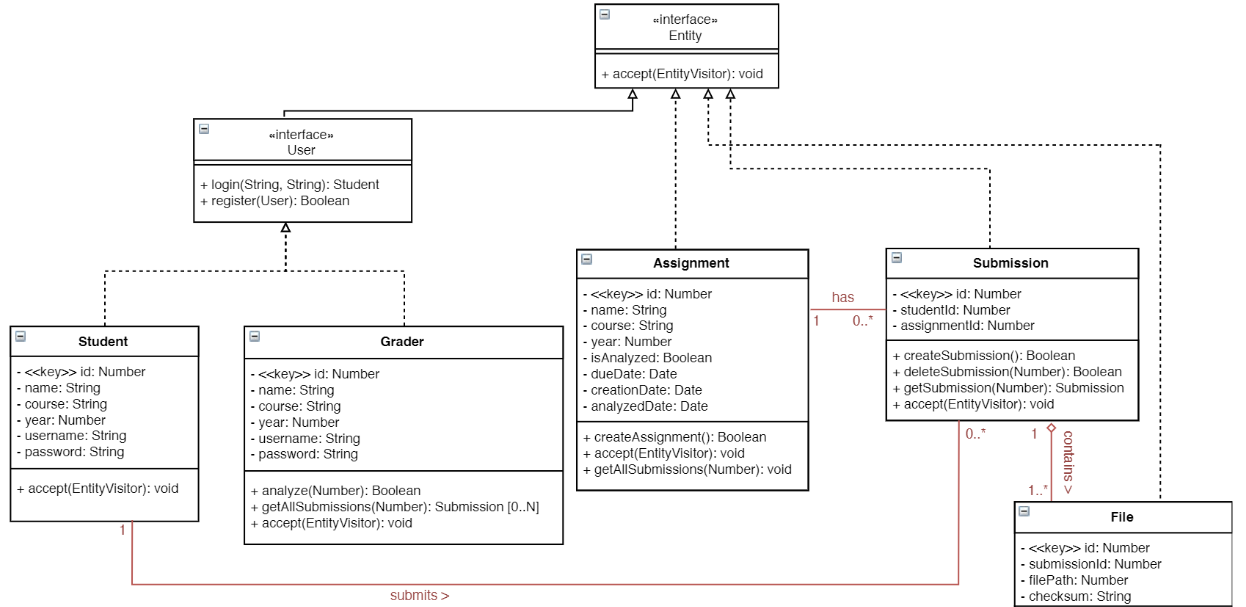
# UML README

This file explains the two of the class diagrams

1. User File Class Diagram



* There are 4 conceptual **Entities** as shown above
  + User
  + Assignment
  + Submission
  + File
* Each **Assignment** can have 0 or 1 **Submission**.
* Each **Submission** has-a **File**.
* Each **Student** can have 0 or 1 **Submission**.
* User: User Entity has 2 subclasses:
* Student
  + Can Login and Register in the system
* Grader
  + Can Login and Register in the system
  + Can ‘**analyze**’ a given assignment by its id
  + Can ‘**getAllSubmissions**’ for a given assignment by its id

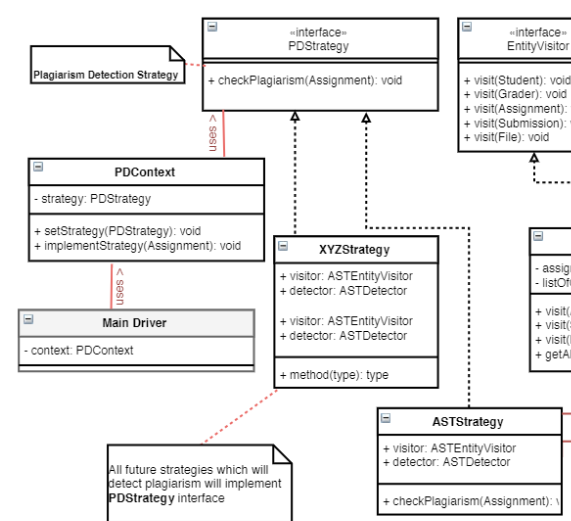
Note: **Grader**’s **analyze** method will trigger **Main Driver** Class explained in point 2 below

1. Core Application + AST Class Diagram

Note: As the figure is too big is to fit in here, it is divided into multiple images.

Please lookup full image [here](https://github.ccs.neu.edu/cs5500/team-27/blob/master/phaseB/PhaseB_UML.pdf) on pg3.

Fig 2.1

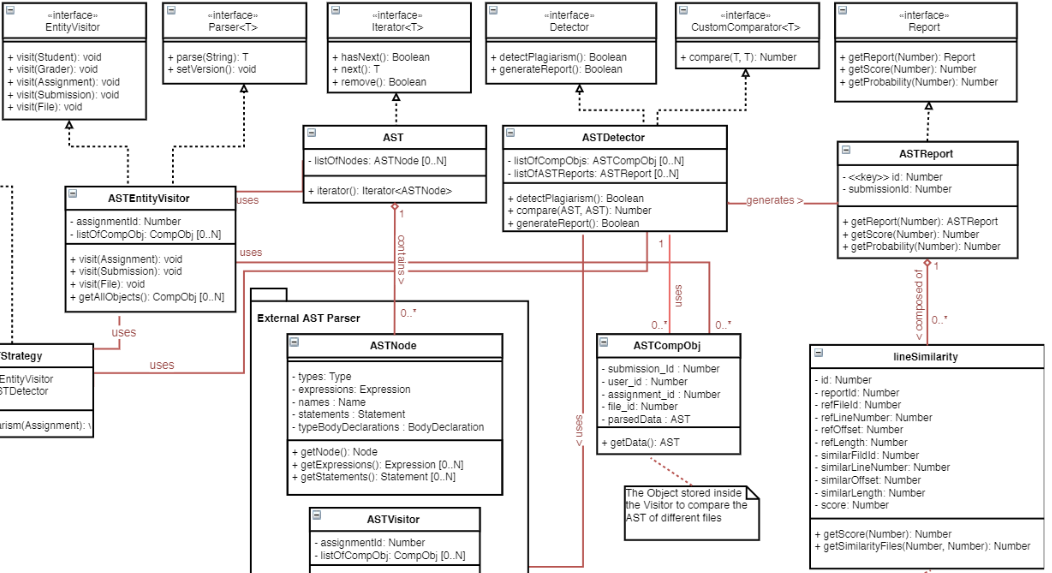


* Grader’s Analyze method will **Main Driver**
* There is an interface called **PDStrategy (Plagiarism Detection Strategy)**
* All subclasses implementing this interface, for example **ASTStrategy**, will actually contain implementation logic for that particular strategy.
* There is also **PDContext** class which uses **PDStrategy.** It has two method **setStrategy** and **implementStrategy**
* The **Main Driver** uses this **PDContext** class in turn to drive through the plagiarism detection.

These all classes implement **Strategy Design Pattern.**

* In future, if there is a new strategy to incorporate to detect plagiarism then all needs to be done create a new **<XYZ>Strategy** class which implements **PDStrategy** interface.

Fig 2.2



* **ASTEntityVisitor** visits assignment and gets all submission and files for that assignment
* We then parse the files and create **ASTCompObj** which is stored in the visitor
* **ASTEntityVisitor** then passes this list to **ASTDetector.**
* **ASTDetector** takes the list and compares files of each submission with others and outputs a score
* Once that comparison is done, store that as **lineSimilarity** to database.
* Generate **ASTReport** object for each file using **lineSimilarity**