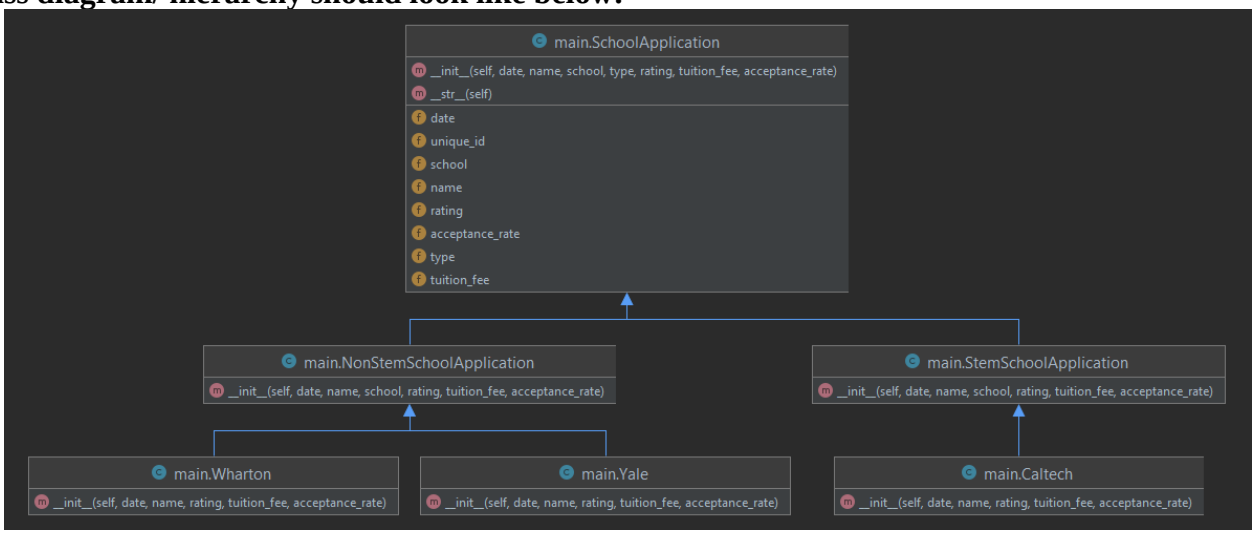
Appendix:
Follow the instructions carefully to make sure the dataset can be loaded properly:

Scenario #4: Data about SchoolApplications of candidates from an aggregator website

1. Create a class called SchoolApplication with the following attributes in its constructor (__init__ method) **in this order**:
    a. date: Application date.
    b. name: Major name.
    c. school: School's name.
    d. type: School's type.
    e. rating: Candidates's rating for the school.
    f. tuition_fee: Tuition fee (in USD).
    g. acceptance_rate: School's acceptance rate for the candidate.
2. In the constructor, add another attribute **unique_id** (Unique Identifier) and set the attribute's value to **id(self)** – the memory id of the instance of the class.
3. Implement the __str__ method within the SchoolApplication class to return a formatted string containing the **values** of all the attributes in this format: "unique_id,date,name,school,type,rating,tuition_fee,acceptance_rate" (no spaces)
4. Create two subclasses: StemSchoolApplication and NonStemSchoolApplication, both inheriting from the SchoolApplication class
5. The constructors in the StemSchoolApplication and NonStemSchoolApplication classes should:
    a. Take input **in this order**: date,name,school,rating,tuition_fee,acceptance_rate
    b. Use super() to call the constructor of the parent SchoolApplication class, passing the type attribute as "Stem" for StemSchoolApplication and "non-Stem" for NonStemSchoolApplication with rest of the input.
6. Create three additional subclasses: Caltech, Wharton, Yale, inheriting from StemSchoolApplication and NonStemSchoolApplication, NonStemSchoolApplication respectively.
7. The constructors in the Caltech, Wharton, and Yale classes should:
    a. Take input **in this order**: date,name,rating,tuition_fee,acceptance_rate
    b. Set the school as class name (i.e. school = "Wharton" if class is Wharton).
    c. Use super() to call the constructor of the parent class with the rest of the input.
8. Once the classes are ready, test your code with the examples shown below:
    a. WhartonSchoolApplication = Wharton("2022-11-15","nameA", 2, 641, 86.06)
    b. print(str(WhartonSchoolApplication))
    c. should return "**[unique_id],2022-11-15, nameA,Wharton,non-Stem,2,641,86.06**"
9. Once that is verified, retrieve your objects by loading the pickle file provided in this link.
10. Create a csv file with the steps below:
    a. Write a first line that denotes the column headers as "uniqueId,date,name,school,type,rating,tuition_fee,acceptance_rate" (no spaces)
    b. Loop through all the objects retrieved from the above pickle file and use str method to print formatted string as mentioned in step 3. Refer to snippet of code provided below.
11. Use the above generated csv file to create visualizations in Python.

Class diagram/ hierarchy should look like below:



Snippet for step 10:

```
with open('data.csv', 'w') as f:
    f.write("uniqueId,date,name,school,type,rating,tuition_fee,acceptance_rate\n")
    for obj in objects:
        f.write(str(obj)+'\n')
```