# Ethan Booker

CS 746 Assignment 2

# Notes

My class diagrams have like a “minimize” button on the top left corner of each box, so please ignore those as that came with the software tool I used.

- Tool: draw.io

# Problem 1, Strategy Pattern Problem

UWL library hire: librarian, front desk helper, stacker positions

- 1 form
- basic info: age, eligibility to work, etc
- librarian: bachelors in library science, 1 yr at another library
- front desk helper: high school completion, attending / graduated with degree
- stacker: high-school completion, pass physical exam set by UWL

# Problem 1, No Design Pattern

- A possible solution would be to have a form class that requires the basic requirements such as age and eligibility to be filled out with valid values.

- Then there could be a function that states what position someone would like then also function that accepts a list of additional qualifications.

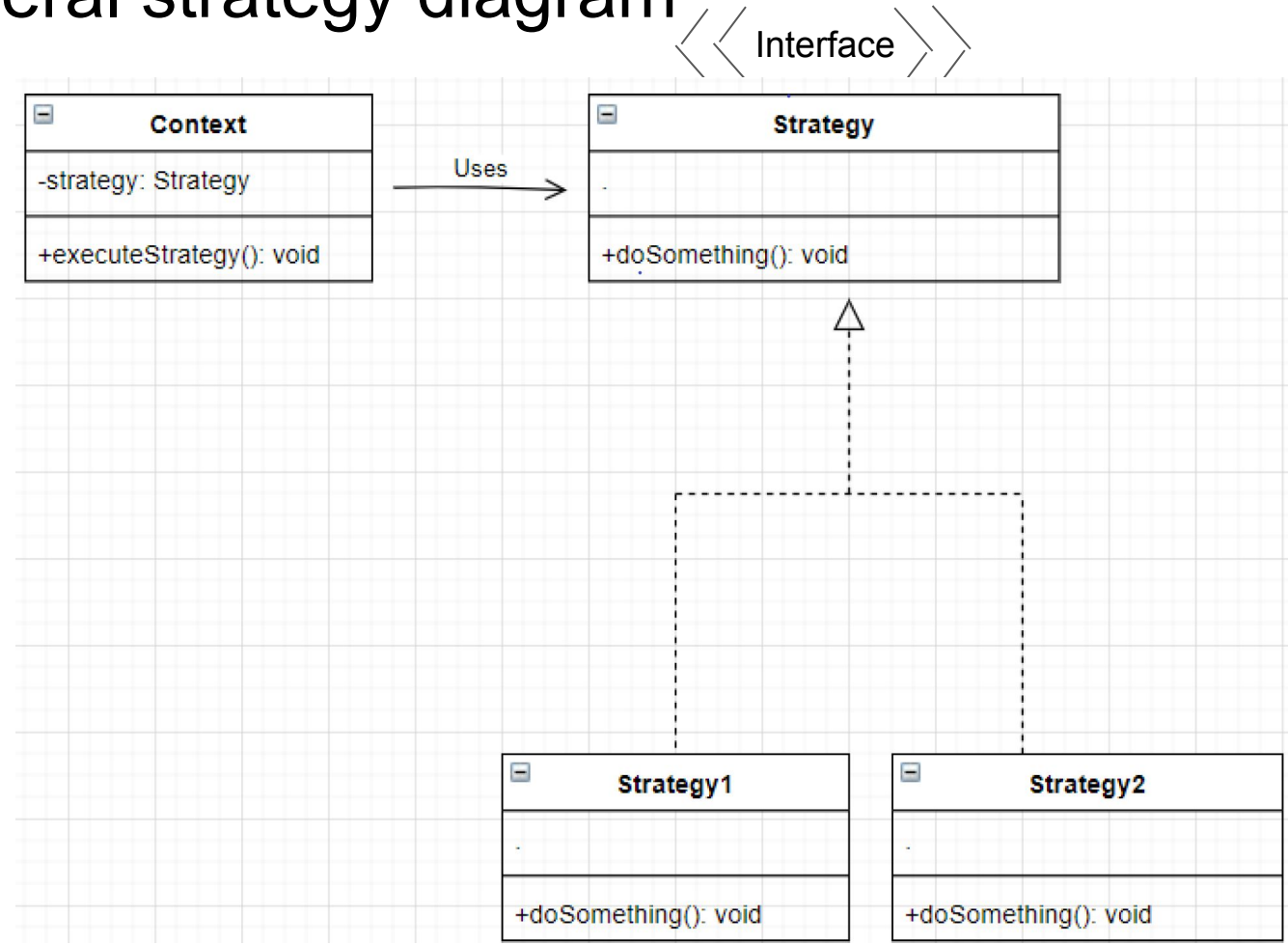
- Then lastly, there would be a submit function that returns a boolean value depending on if the information provided is valid and is their qualifications meet.

# Problem 1: Strategy Pattern

- For this problem, there are additional checks on qualifications that must be made depending on the desired role.
- So a way to look at this is to have a list of qualifications such that depending on your desired role you can use a strategy to evaluate the list of qualifications.
- This pattern is useful for changing run-time behavior of a certain property, so the form values will be evaluated differently depending on the desired role.

# Problem 1, general strategy diagram

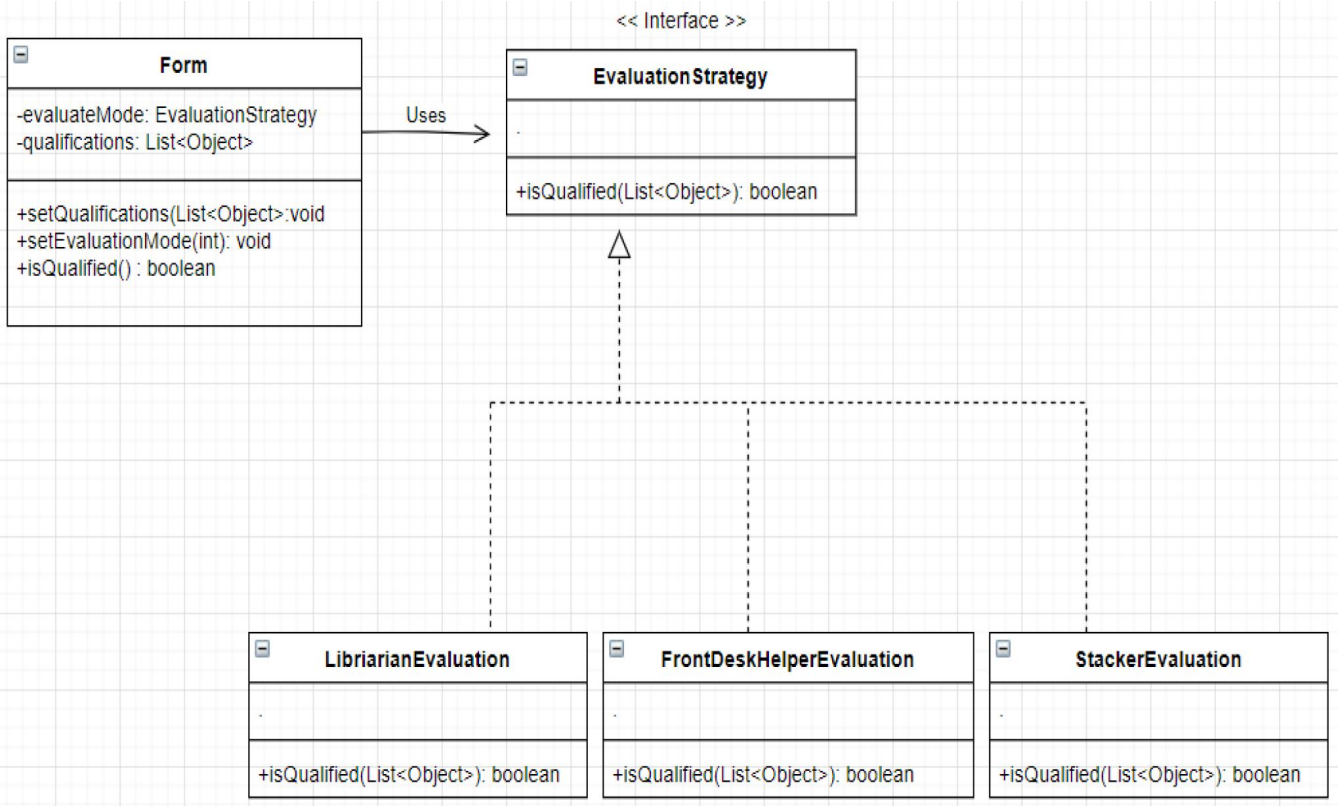
- Strategy is an <<interface>> but I accidentally cropped it out of the diagram.



# Problem 1, class diagram

- One note, if you are questioning why I'm using a **List<Object>**, it is because I'm lazy to write out all the qualifications, however if you think about it, a form is just a massive list of qualification items that can be read.

- **Each isQualified methods will do the additional checks needed.**



# Problem 2, Mediator Pattern Problem

UWL graduate applications

- Admissions Office first checks basic information
- Appropriate department evaluates
  - If approve, return to Admissions Office → International Education Office
- Graduate Assistantship Request field
  - Additional information must be provided
  - After approval by department → Financial Aid Office

# Problem 2, No Design Pattern

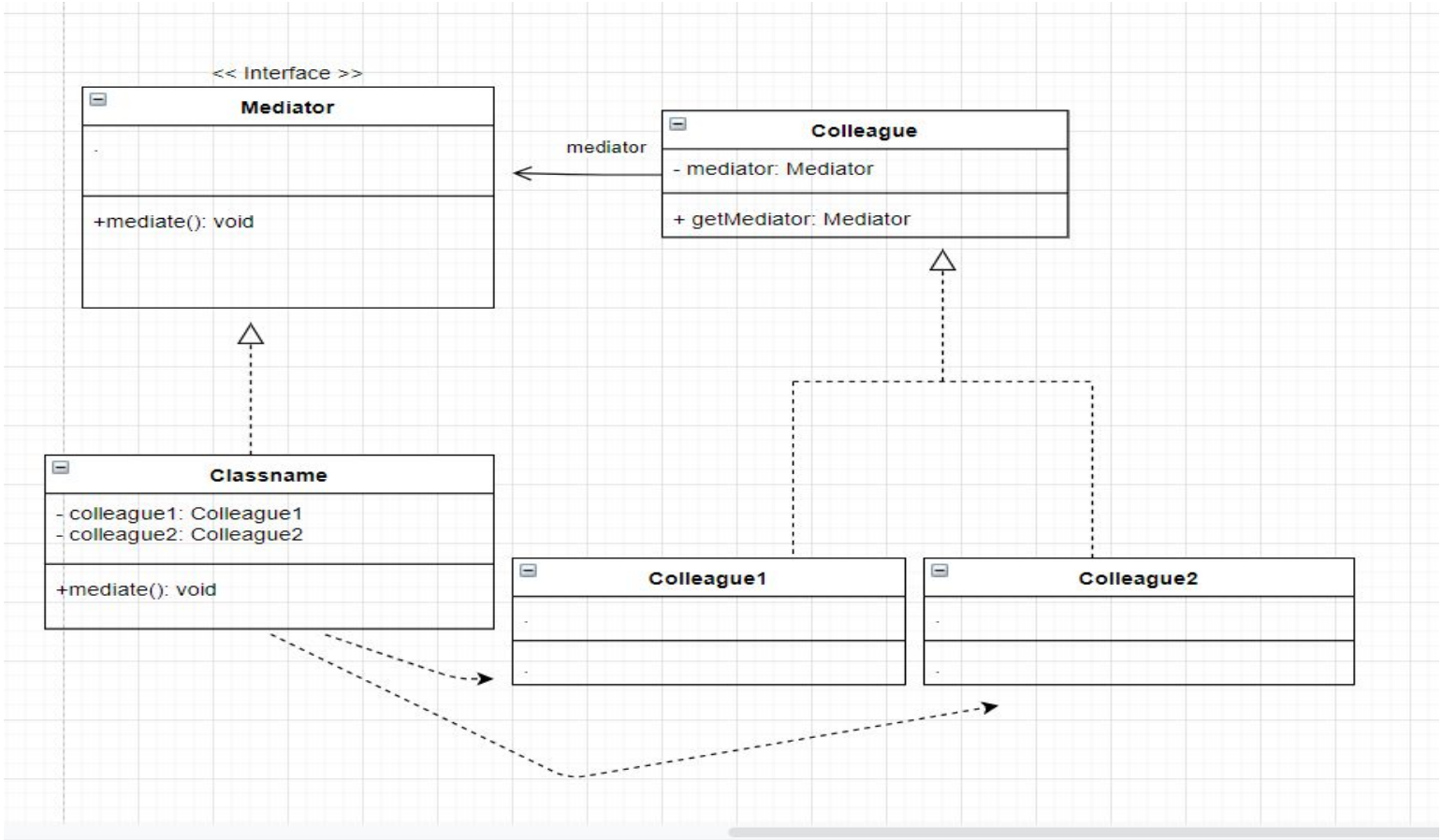
- A possible solution would be to have an application class that contains all the application info and a boolean value for “Graduate Assistantship Request”.

- The Admissions Office class validates the application object.
  - If fails, reject the application
  - else (passes)
    - Send to appropriate department class object for evaluation. Wait for approval response.
      - If approved
        - If Graduate Assistantship Request → send application to Financial Aids Office
        - Else → route application to International Education Office
      - Else reject application

# Problem 2: Mediator Pattern

- For this problem, the admissions office class is tightly coupled with the department class, financial aid, and international office. So, it’s best that the admissions office because a mediator to communicate the results.

# Problem 2, general mediator diagram



# Problem 2, class diagram

I used boolean values as I was unsure what the return value should be for responses.