Object-Oriented Programming

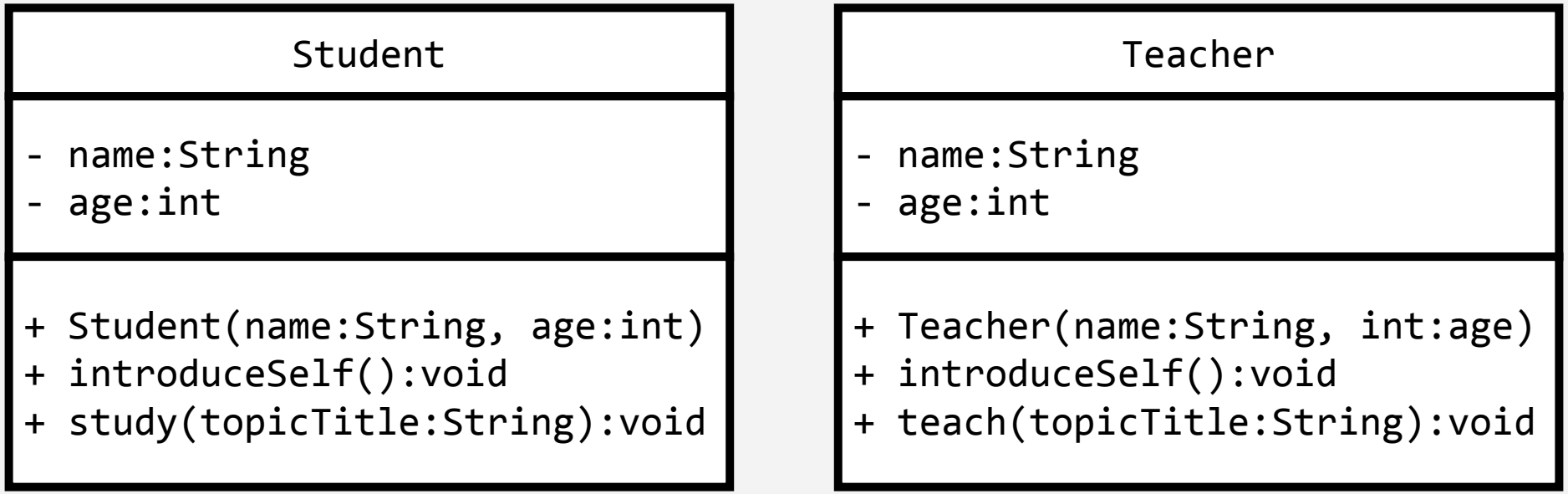
Cont. of Inheritance

Overriding

Practice Exercise 8

- Class Diagram of a Teacher + Student + University Scenario
  - How many classes did you create?
  - What is your University + Teacher relationship?
  - What is your University + Student relationship?
  - What is your Teacher + Student relationship?

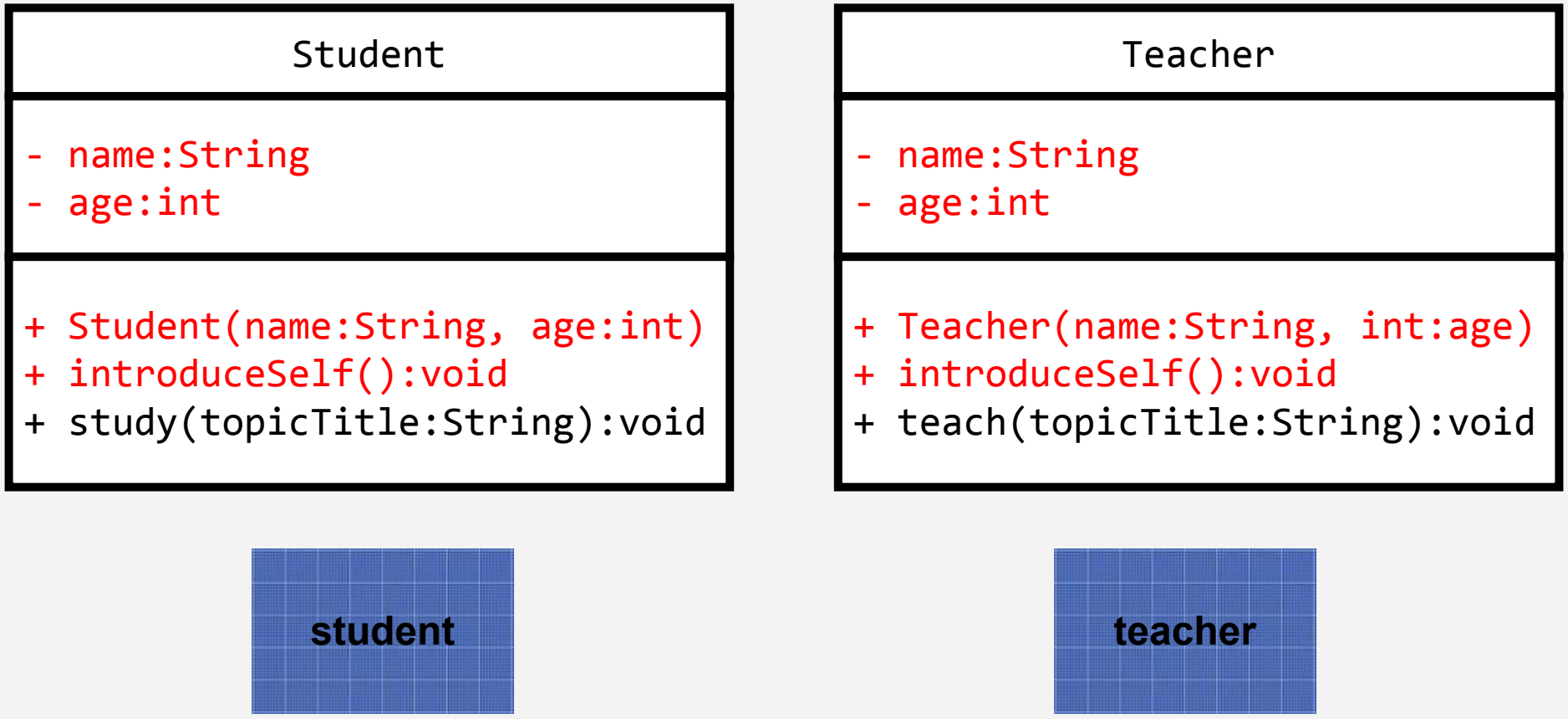
Practice Exercise 8



student

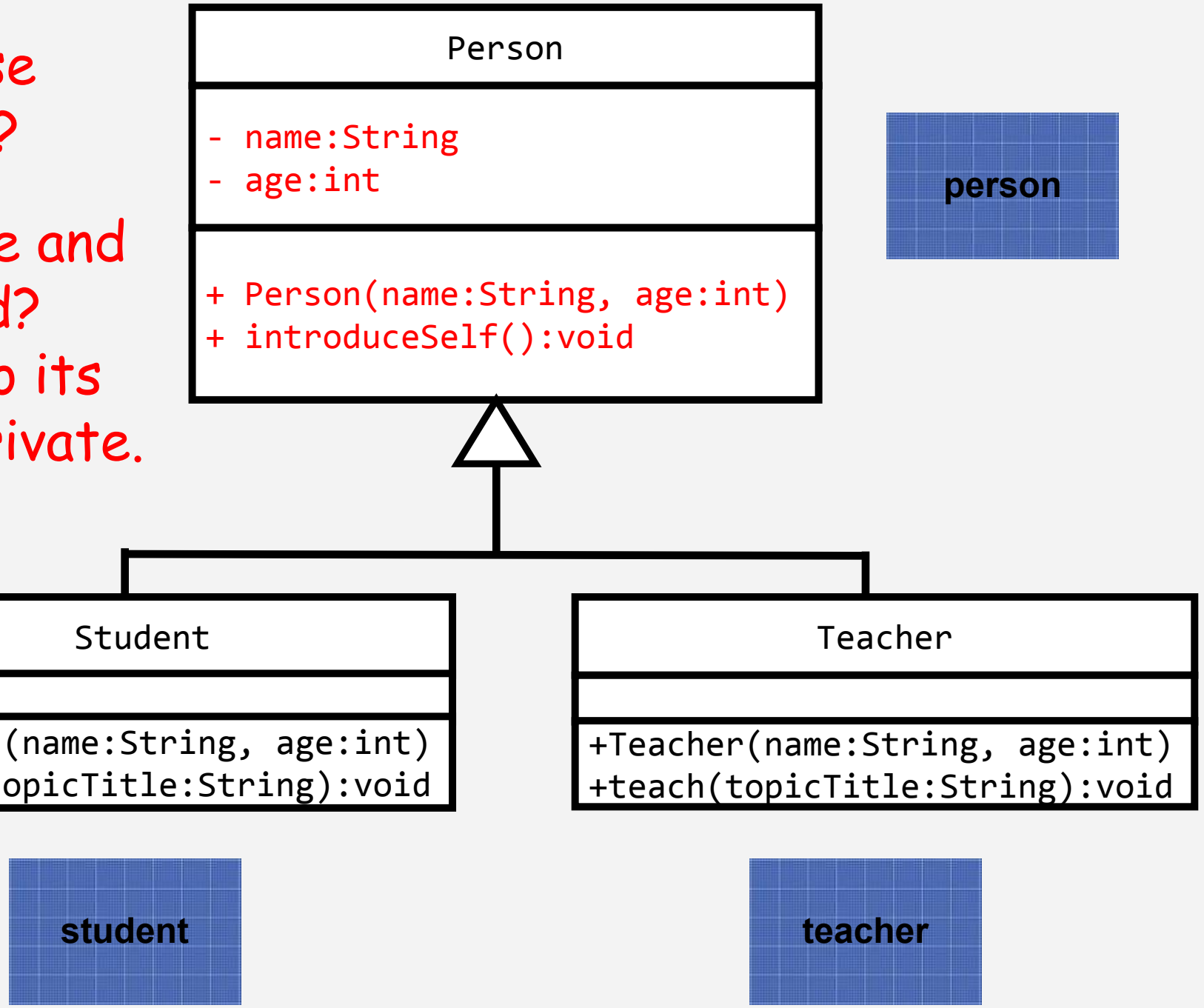
teacher

Practice Exercise 8

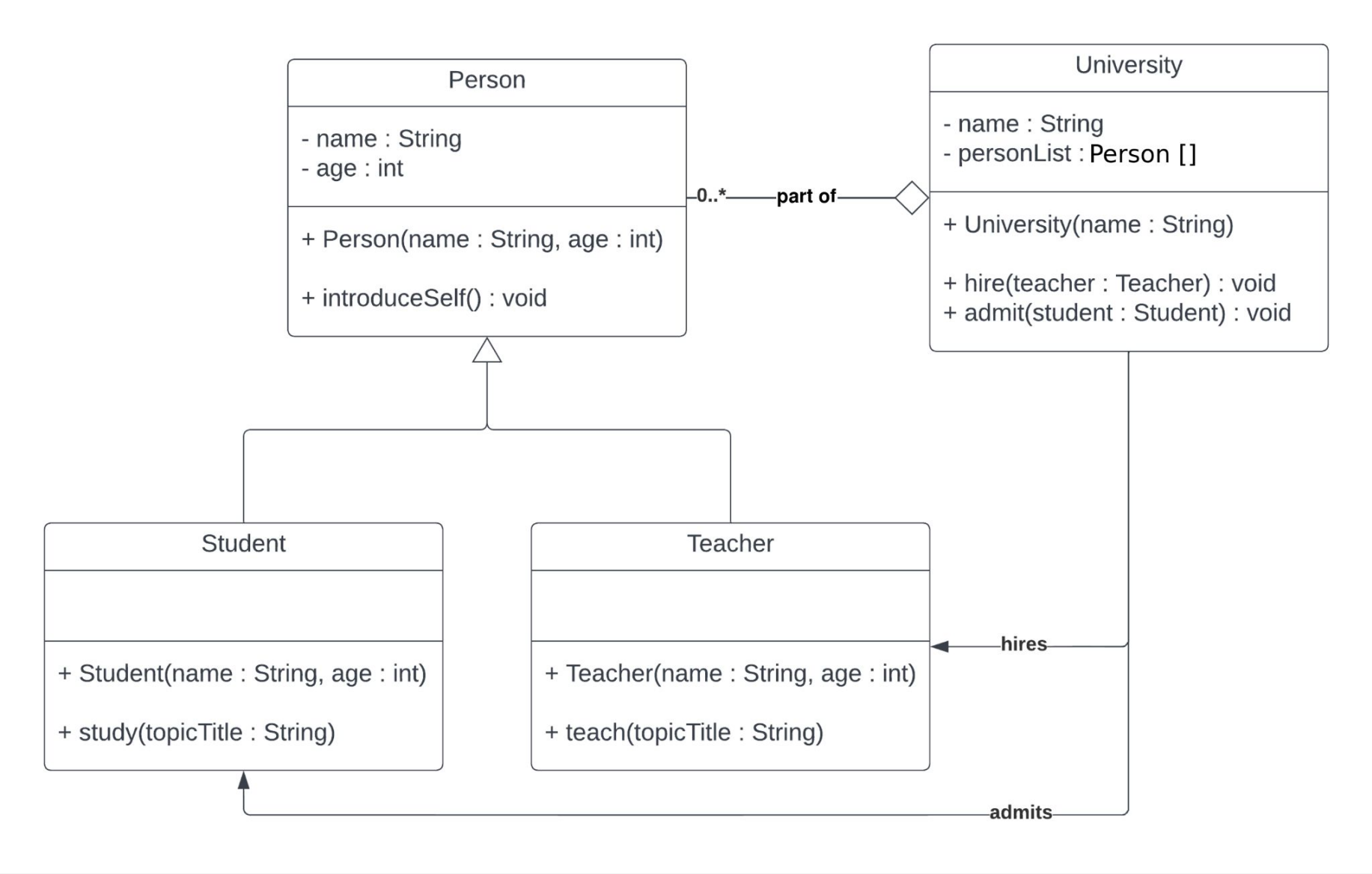


Why not use protected?

When will name and age be used? They aren't so its best to keep private.



How close did you get? ☺️

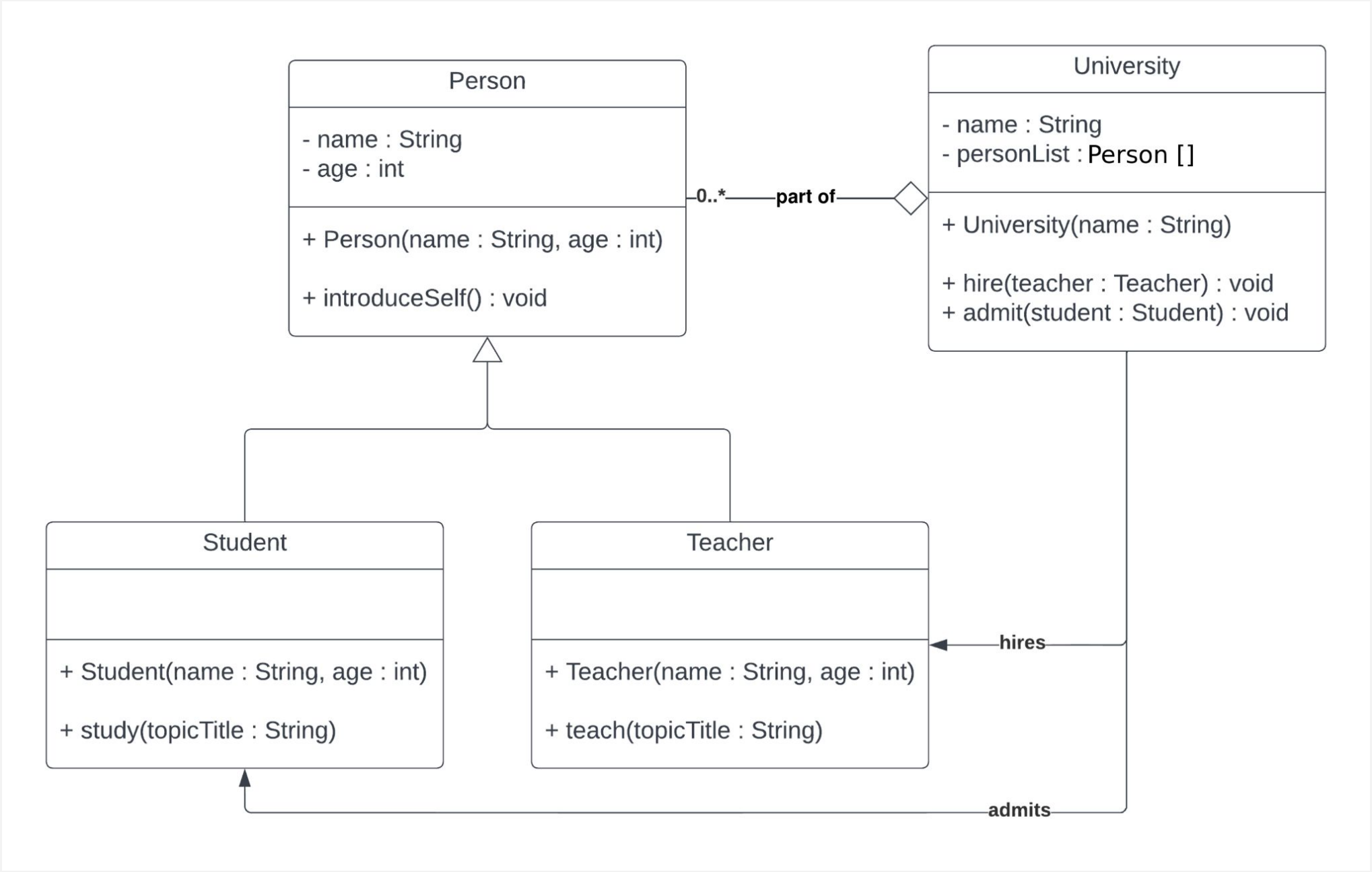


Questions? ☺️

Modified Student + Teacher + Univ Scenario

- What if the student and teachers are meant to introduce themselves in a different manner?
  - Student: Hi! I'm <name> and I'm a student.
  - Teacher: Hello. I am <name>. I work at the university.

- How would we make this work given their superclass is where the logic is located?


Person
- name : String
- age : int
+ Person(name : String, age : int)
+ introduceSelf() : void
0..*
part of
University
- name : String
- personList : Person []
+ University(name : String)
+ hire(teacher : Teacher) : void
+ admit(student : Student) : void
Student
+ Student(name : String, age : int)
+ study(topicTitle : String)
Teacher
+ Teacher(name : String, age : int)
+ teach(topicTitle : String)
hires
admits

Modified Student + Teacher + Univ Scenario

- What if the student and teachers are meant to introduce themselves in a different manner?
  - Student: Hi! I'm <name> and I'm a student.
  - Teacher: Hello. I am <name>. I work at the university.

- How would we make this work given their superclass is where the logic is located?
  - We could consider overriding!

Overriding

- Ability of subclasses to modify-from-scratch the contents of methods provided by superclasses
  - A subclass can override a method of a superclass
- Allows for the modified method to be inherited from the superclass and retain its method's name while still having different logic

Overloading

```
public void play() {
    ...
}

public void play(Song s) {
    ...
}

public void play(Playlist pl) {
    ...
}
```

Overriding

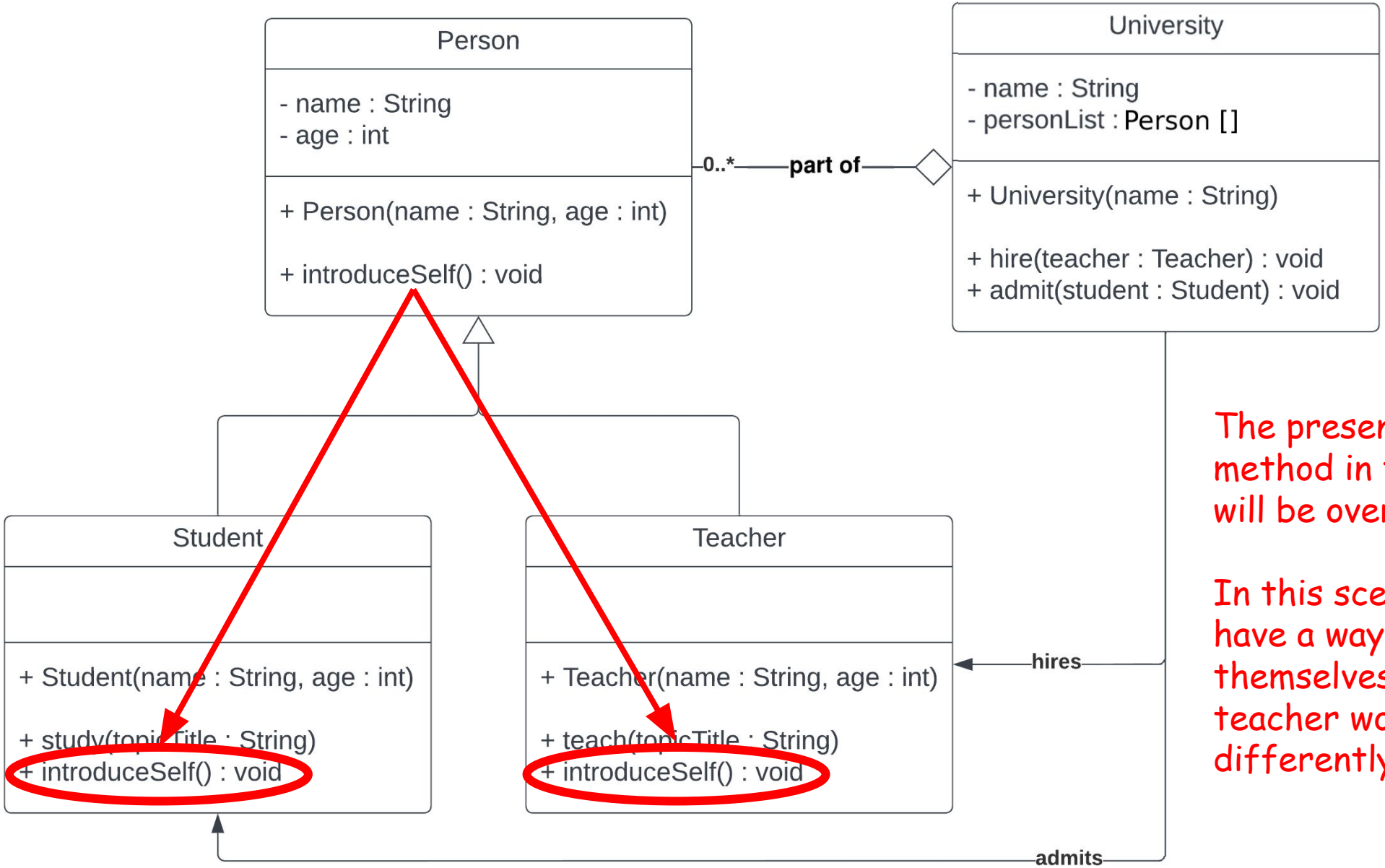
```
// In parent class
public String toString() {
    return “Name: ” + this.name;
}

// In child class
@Override
public String toString() {
    return super.toString()
        + “Additional info”;
}
```

This isn't required but encouraged

You can also complete overwrite the method from scratch

How do we show overridden method in our modified scenario?



The presence of a superclass' method in the subclass implies it will be overridden

In this scenario, Person might have a way to introduce themselves, but a student and a teacher would do the same action differently

Questions? ☺️

Disclaimer: Multiple Inheritance

- In Java, a class can only have one direct superclass
- Multiple inheritance is the concept of inheriting members from multiple superclasses
  - There are issues to this – such as tracing which members belong to which class and how memory is managed
  - There are ways to achieve this in Java, such as using Interfaces – which we'll discuss in a later session
- However, other OO languages, like C++, have some kind of support for multiple inheritance

Summary (Module 6)

- Inheritance defines the “is a” relationship
  - Is the ability of a subclass to inherit members of a superclass
- Polymorphism is an OOP concept that allows subclasses to be stored in superclass variables
- Overriding allows subclasses to replace and/or add logic in a superclass’ method
- There is a lot more to this topic, but we’ll discuss things slowly ☺️

Abstract classes
Interfaces

Practice Exercise 9

- Class Diagram of a Telephone Scenario

Independent Learning Week

- Module 5: Learn GUI and Practice MVC -> MCO2
- *Review all modules*
- *Practice Exercises are all open for you to practice on*
- Read on Module 7 (Abstract classes and Interfaces)
- Practice 9 *(Try to implement it in code)*

Keep learning…