CIT 103 & CIT 104

Object Oriented Programming

By

Md. Palash Uddin

Lecturer

Dept. of CIT

Hajee Mohammad Danesh Science and Technology University, Dinajpur.

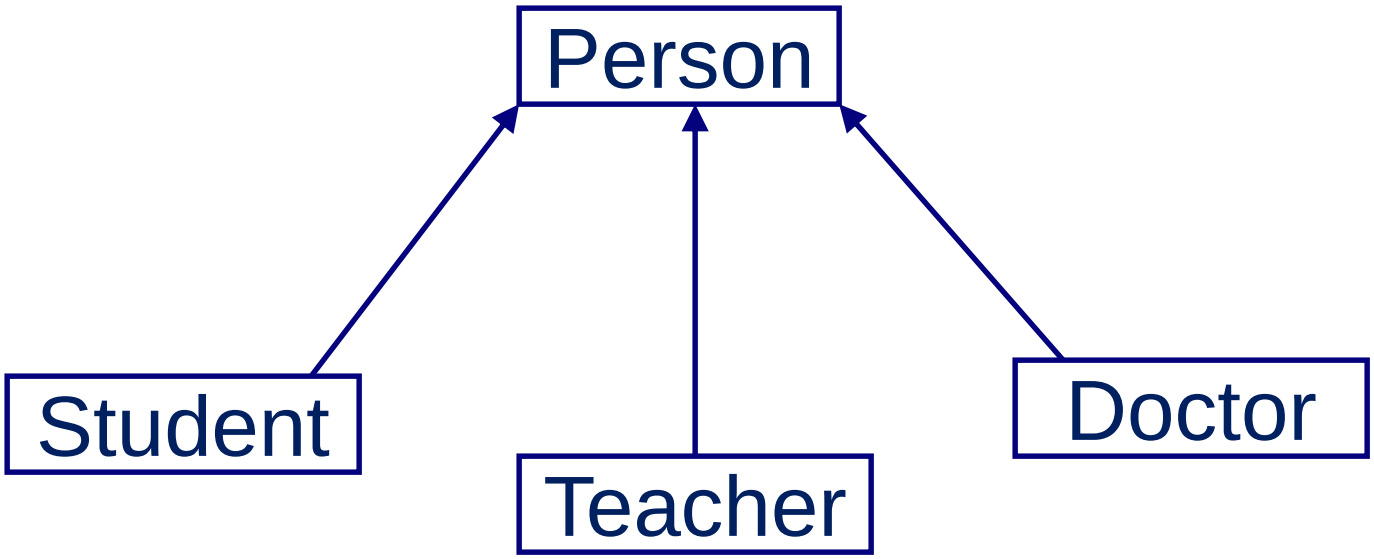
Inheritance

- **Inheritance:** Ability to **derive new objects** from old ones
  - permits objects of a more specific class to inherit the properties (data) and behaviors (functions) of a more general/base class
  - ability to define a hierarchical relationship between objects
- A child inherits characteristics of its parents
- Besides inherited characteristics, a child may have its own unique characteristics

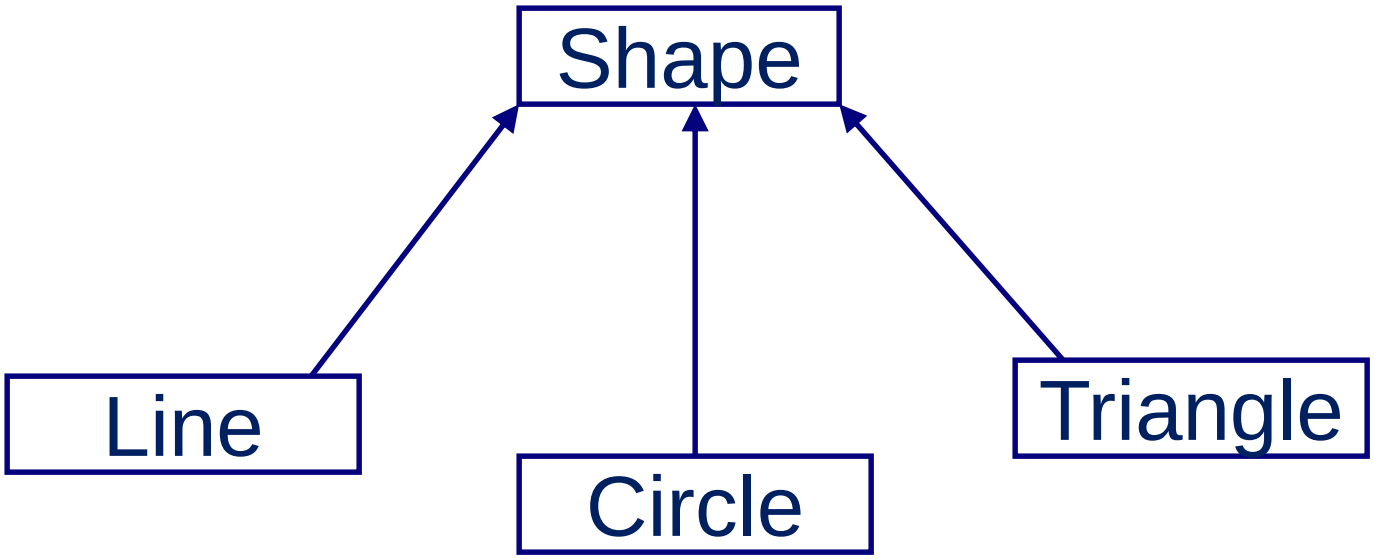
Inheritance in Classes

- Objects of one class acquire the properties of another class
- If a **class B inherits from class A** then it contains all the characteristics (information structure and behavior) of class A
- The parent class (Class A) is called ***base* class** and the child class (Class B) is called ***derived* class**
- Besides inherited characteristics, derived class may have its own unique characteristics

Example – Inheritance



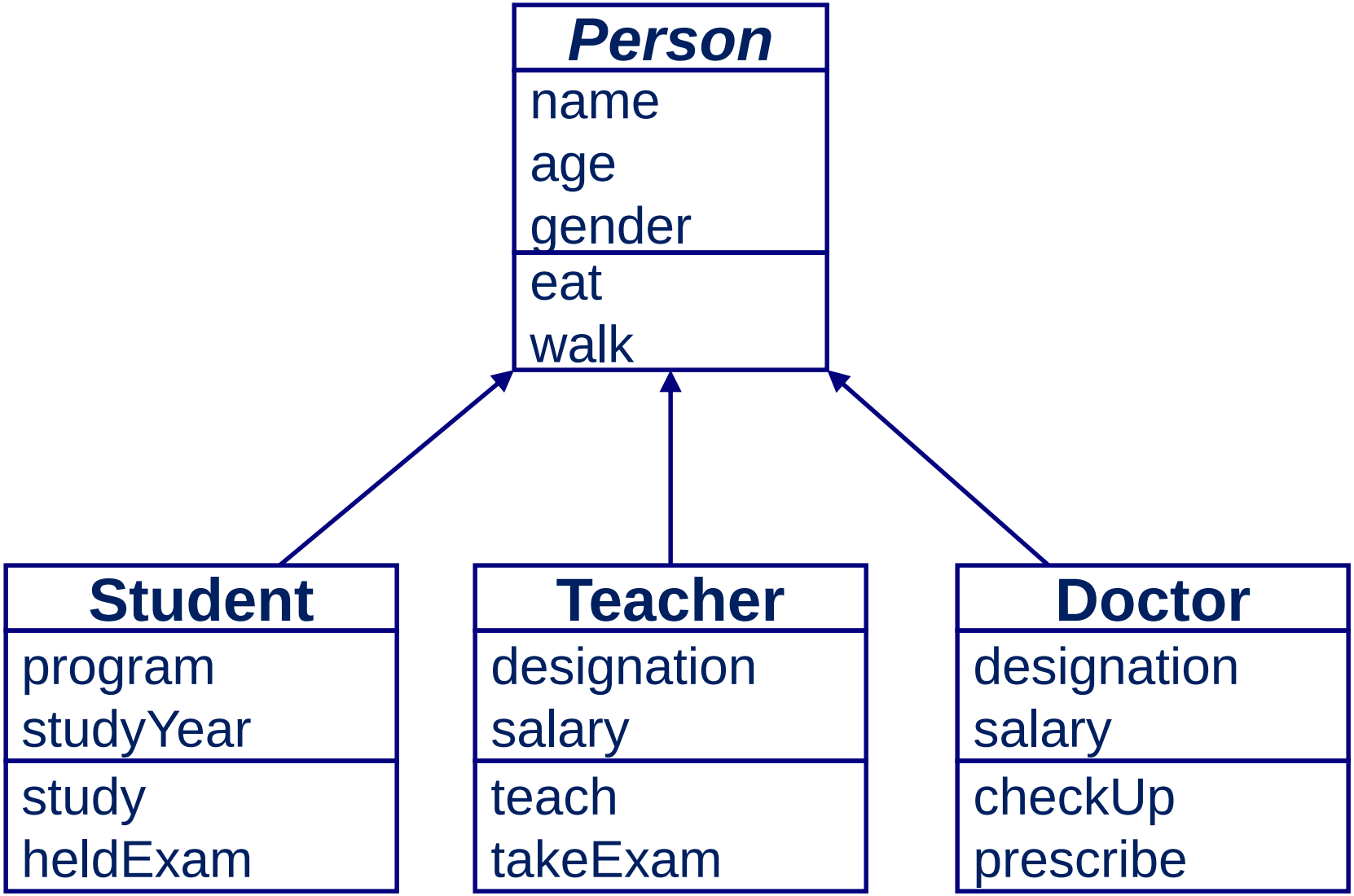
Example – Inheritance



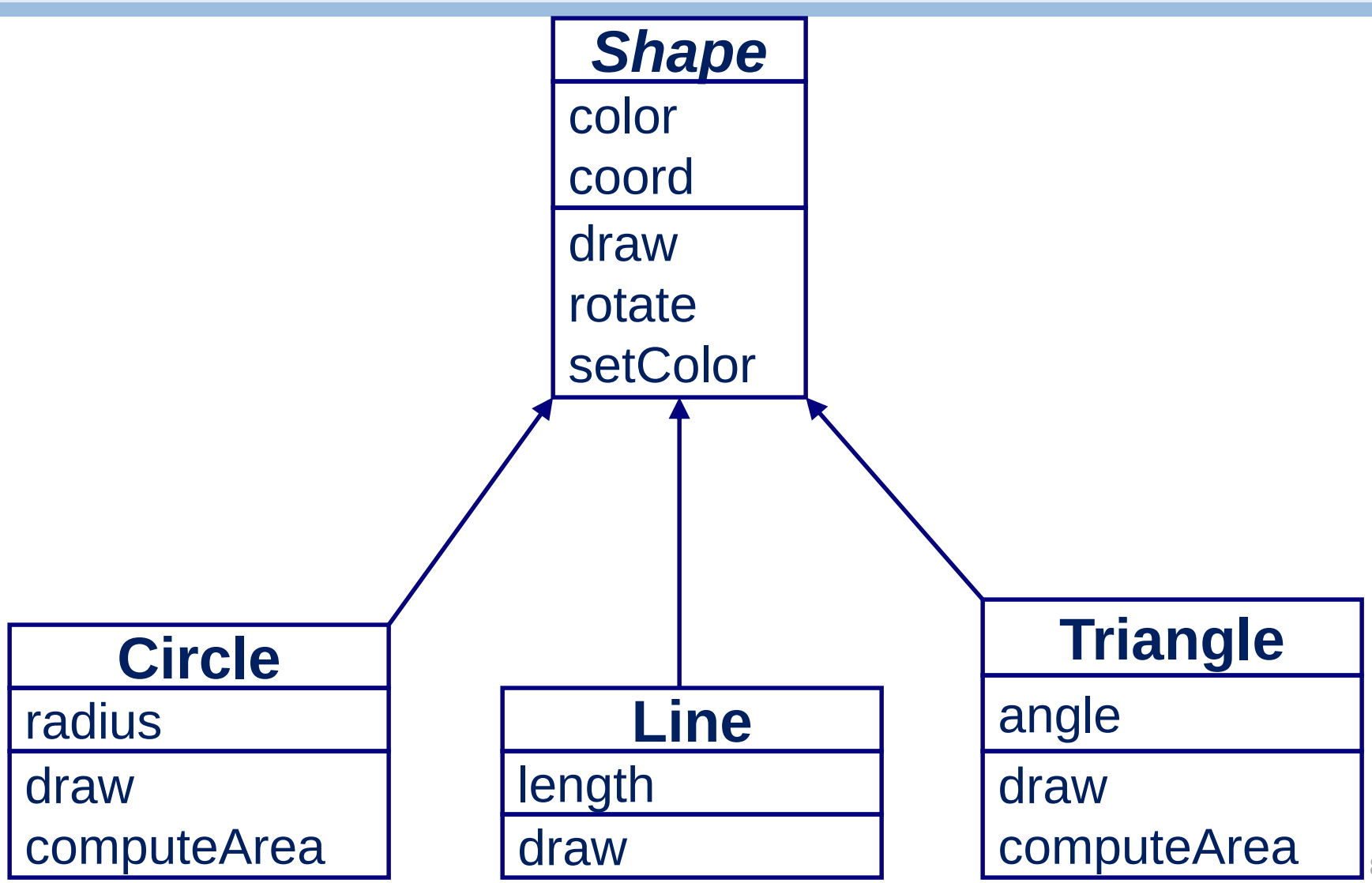
Inheritance – “IS A” or “IS A KIND OF” Relationship

- Each derived class **is a** special kind of its base class

Example – "IS A" Relationship



Example – "IS A" Relationship



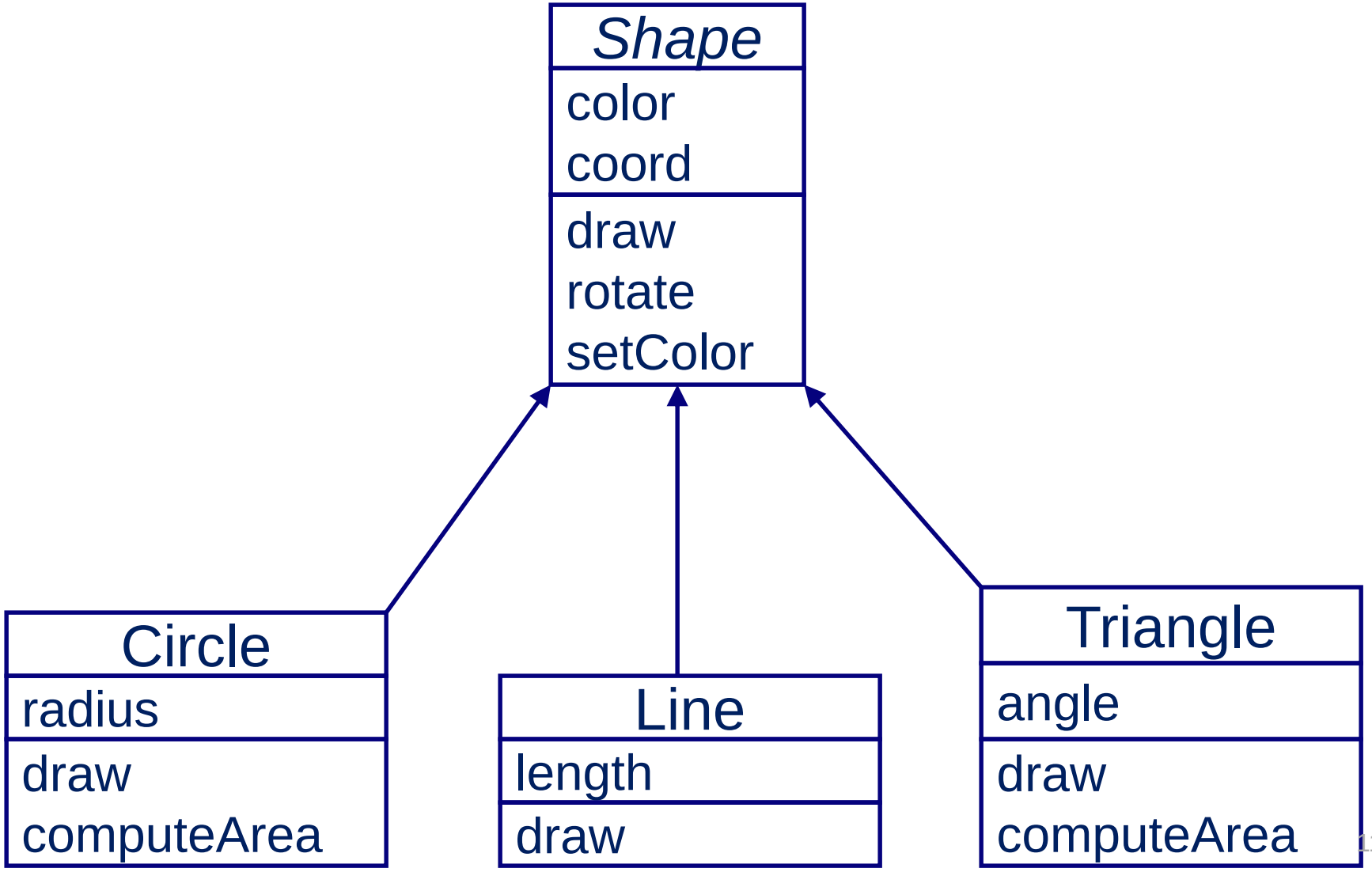
Inheritance – Advantages

- **Reuse**
- Less redundancy
- Increased maintainability

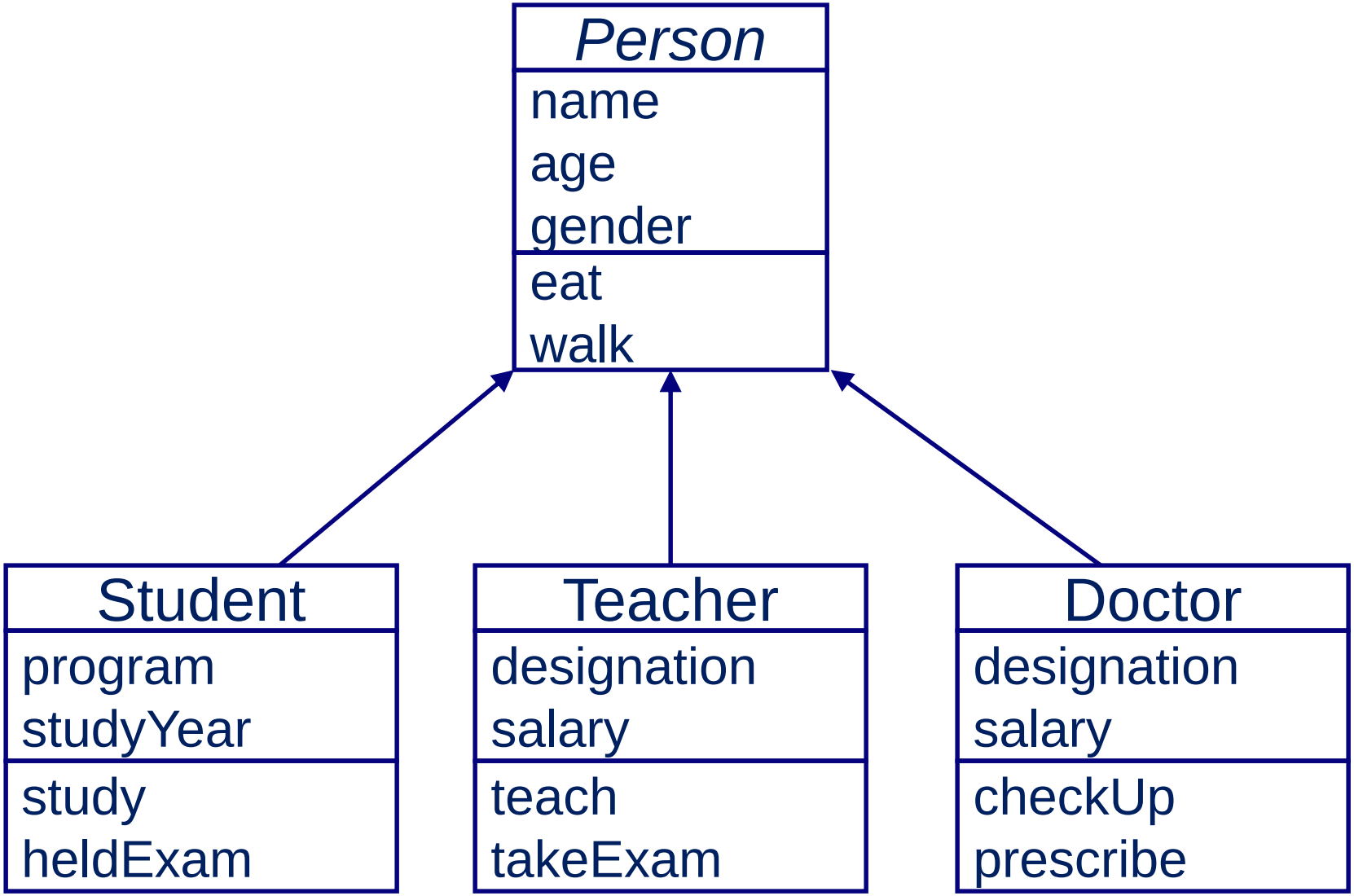
Reuse with Inheritance

- Main purpose of inheritance is **reuse**
- We can easily add new classes by inheriting from existing classes
  - Select an existing class closer to the desired functionality
  - Create a new class and inherit it from the selected class
  - Add to and/or modify the inherited functionality

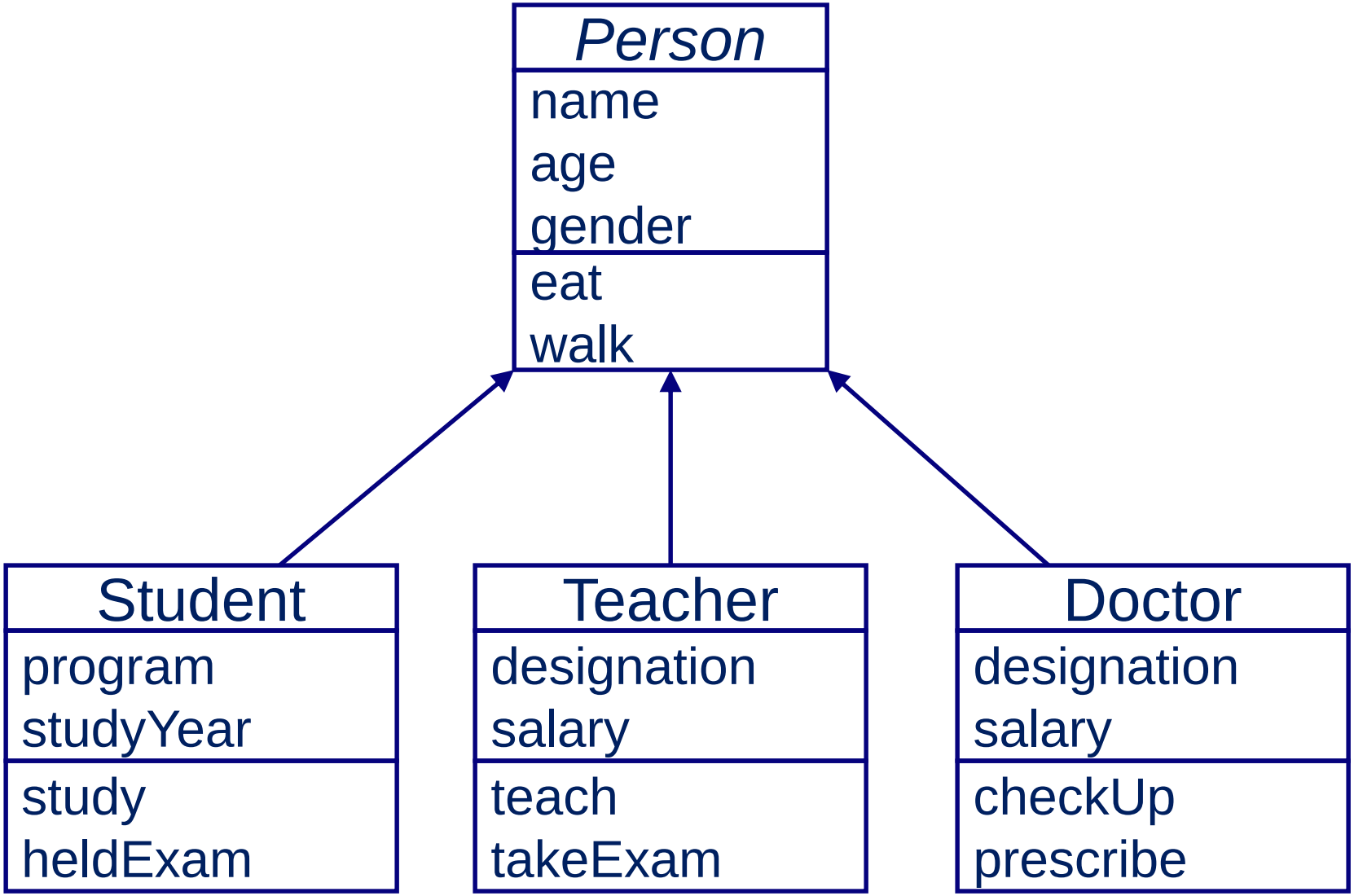
Example Reuse



Example Reuse



Example Reuse



Recap – Inheritance

- Derived class inherits all the characteristics of the base class
- Besides inherited characteristics, derived class may have its own unique characteristics
- Major benefit of inheritance is **reuse**

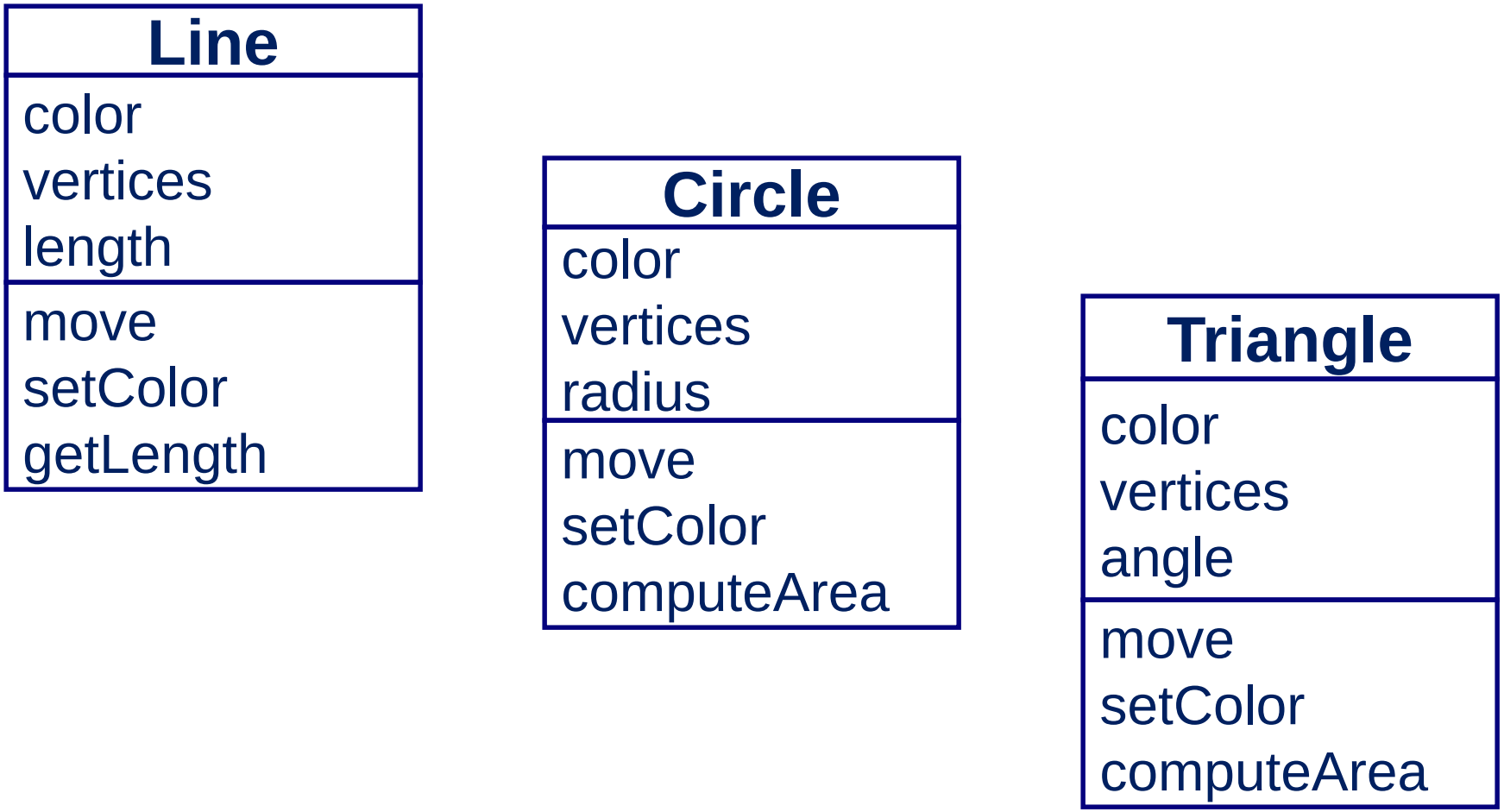
Concepts Related with Inheritance

- Generalization
- Subtyping (extension)
- Specialization (restriction)

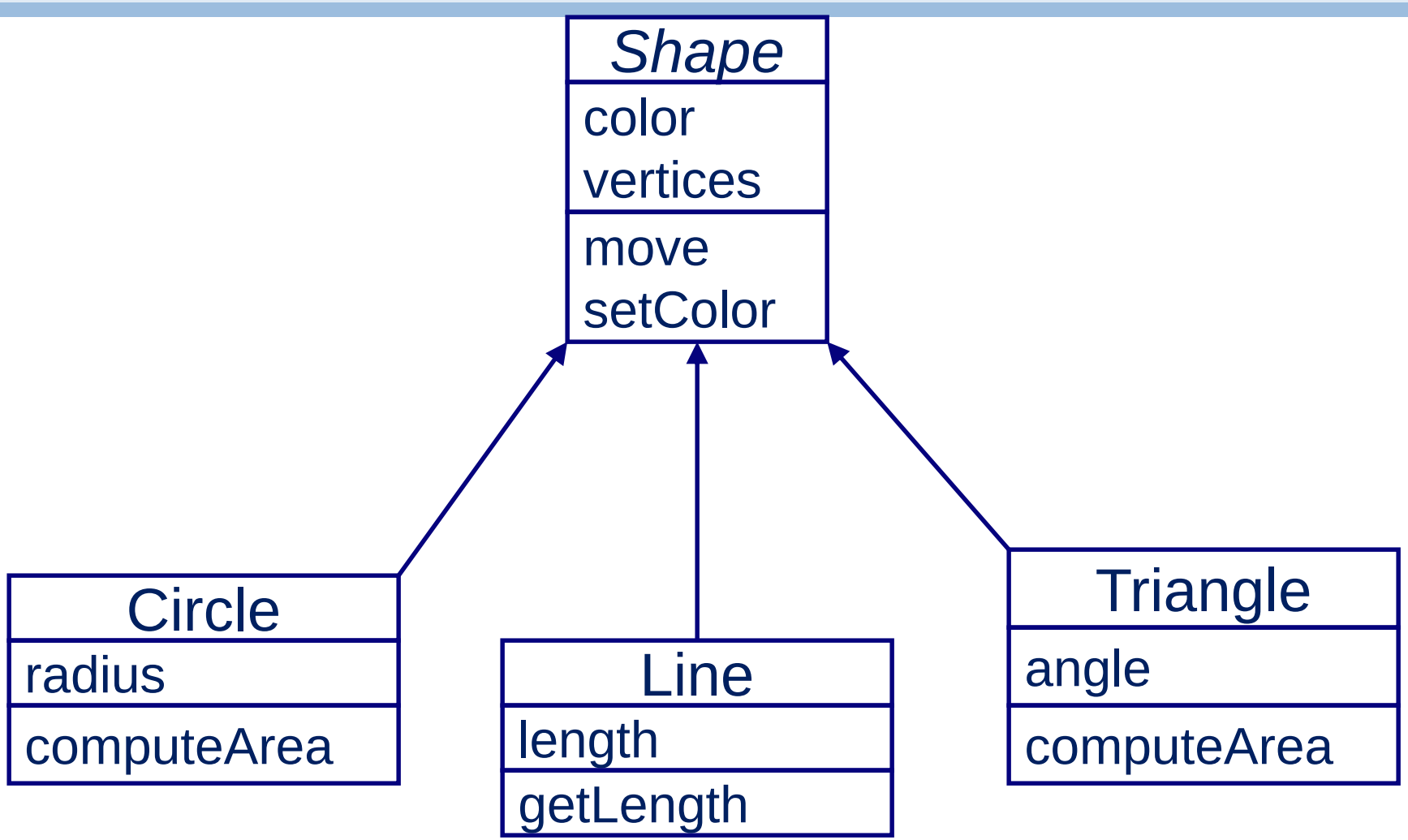
Generalization

- In OO models, some classes may have common characteristics
- We extract these features into a new class and inherit original classes from this new class
- This concept is known as Generalization

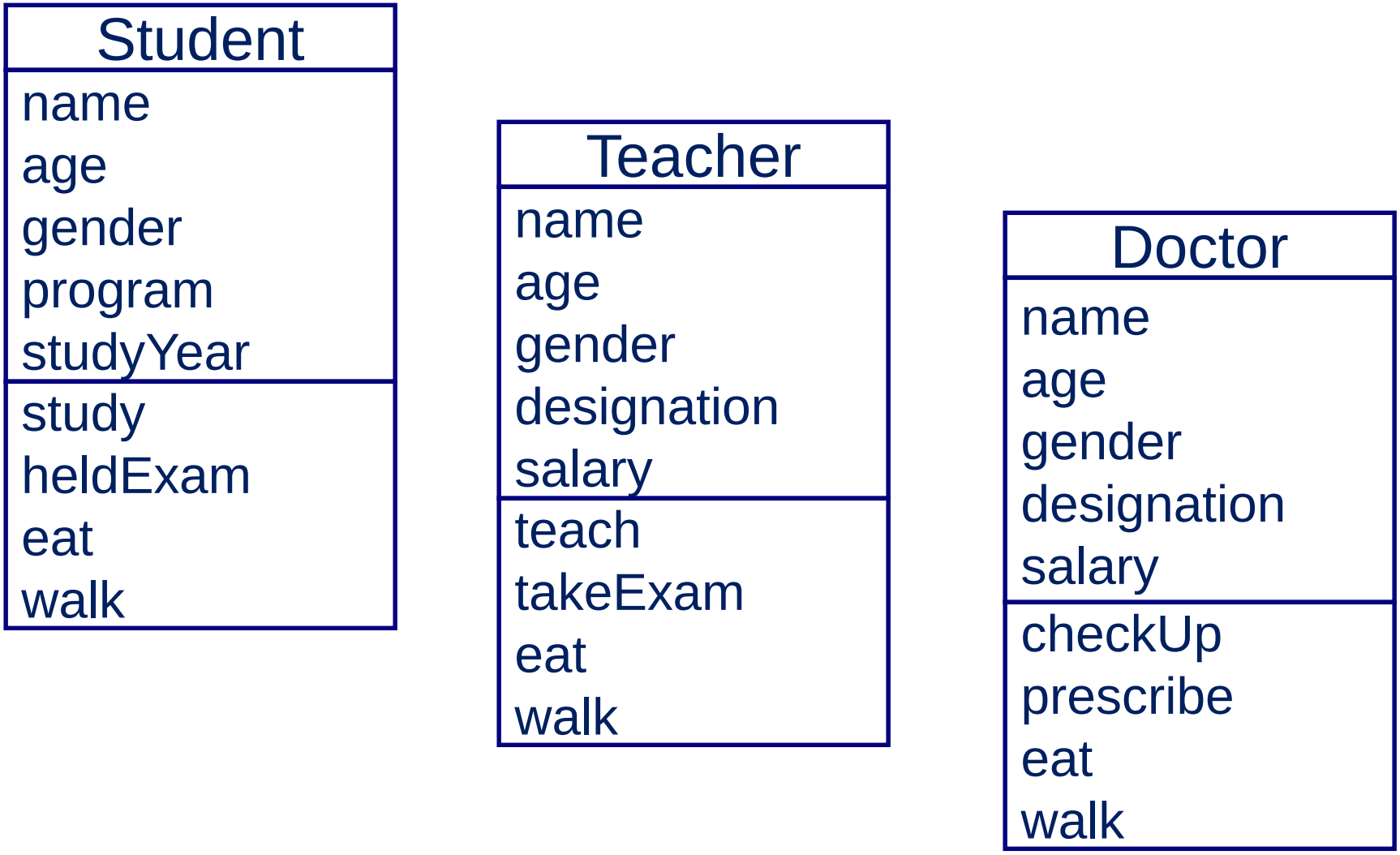
Example – Generalization



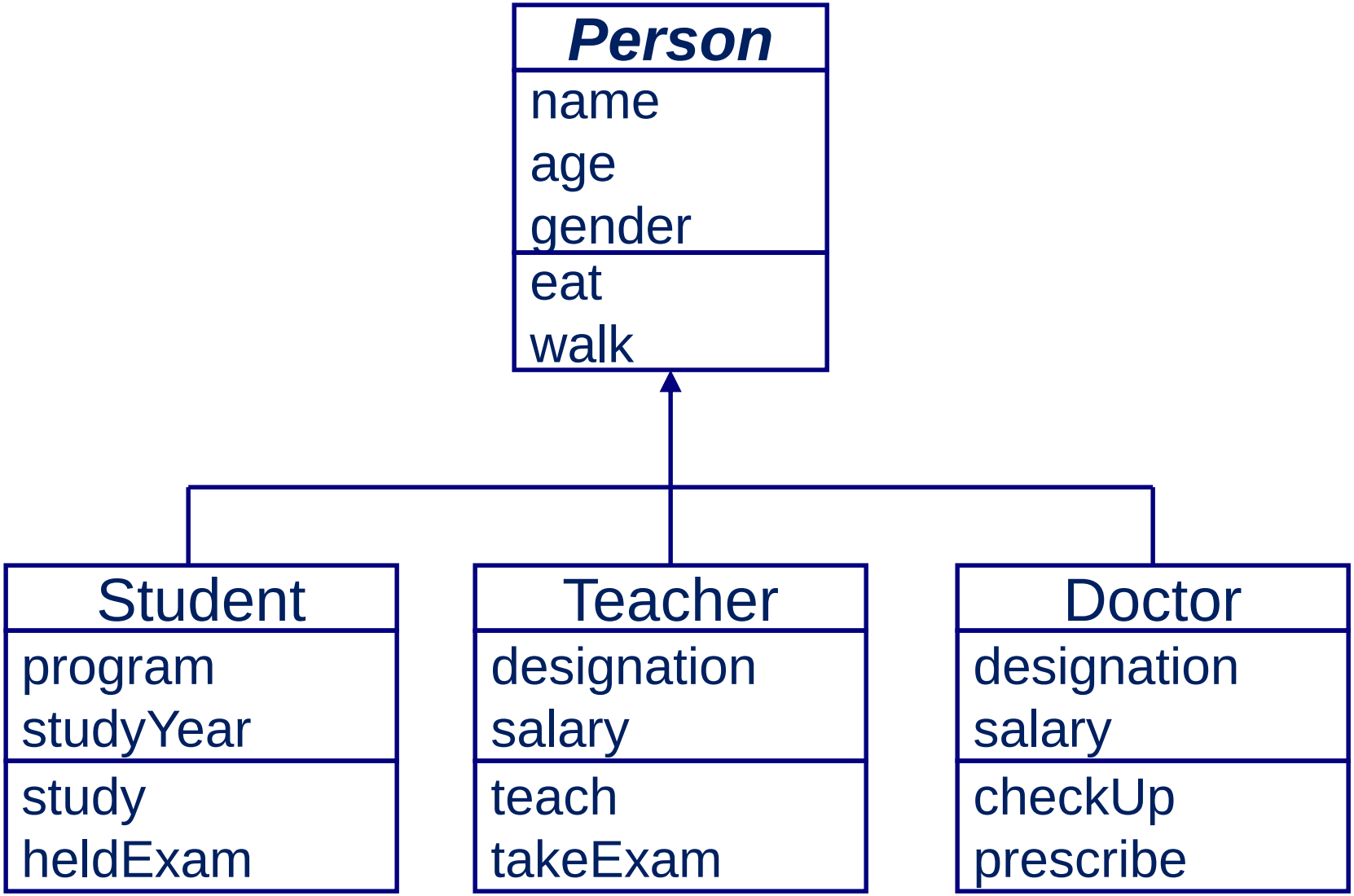
Example – Generalization



Example – Generalization



Example – Generalization



Sub-typing & Specialization

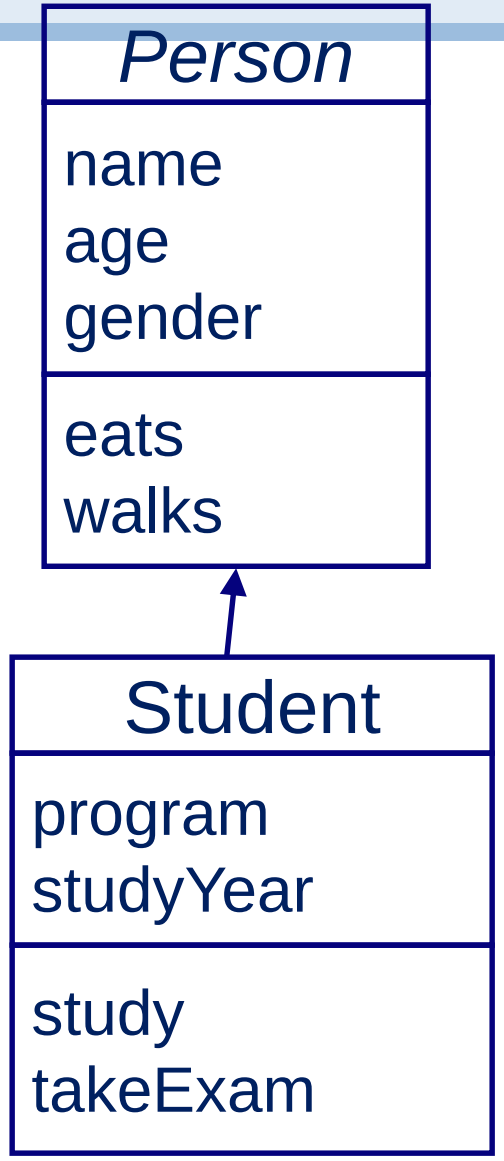
- We want to add a new class to an existing model
- Find an existing class that already implements some of the desired state and behavior
- Inherit the new class from this class and add unique behaviour to the new class

Sub-typing (Extension)

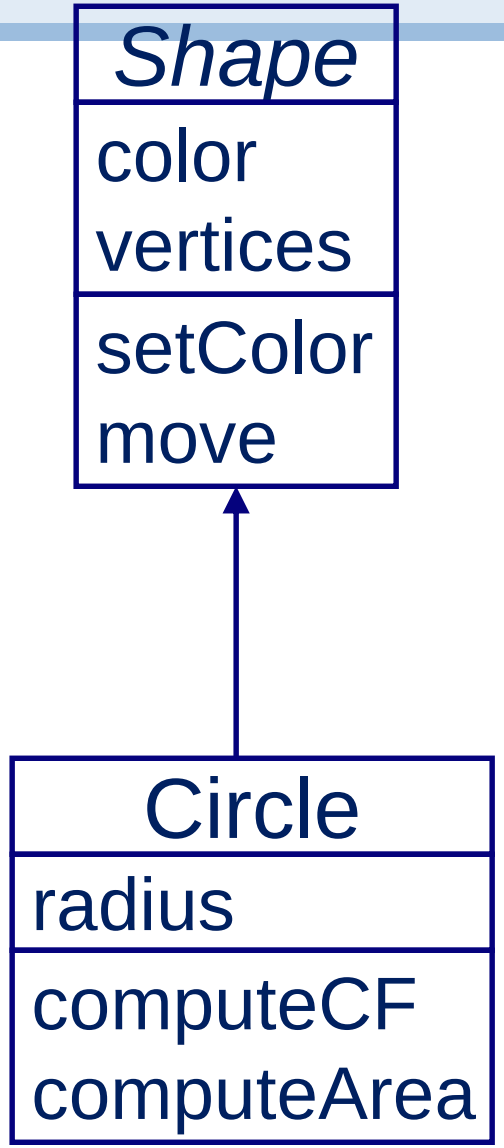
- Sub-typing means that derived class is behaviorally compatible with the base class

- Behaviorally compatible means that base class can be replaced by the derived class

Example –Sub-typing (Extension)



Example – Sub-typing (Extension)

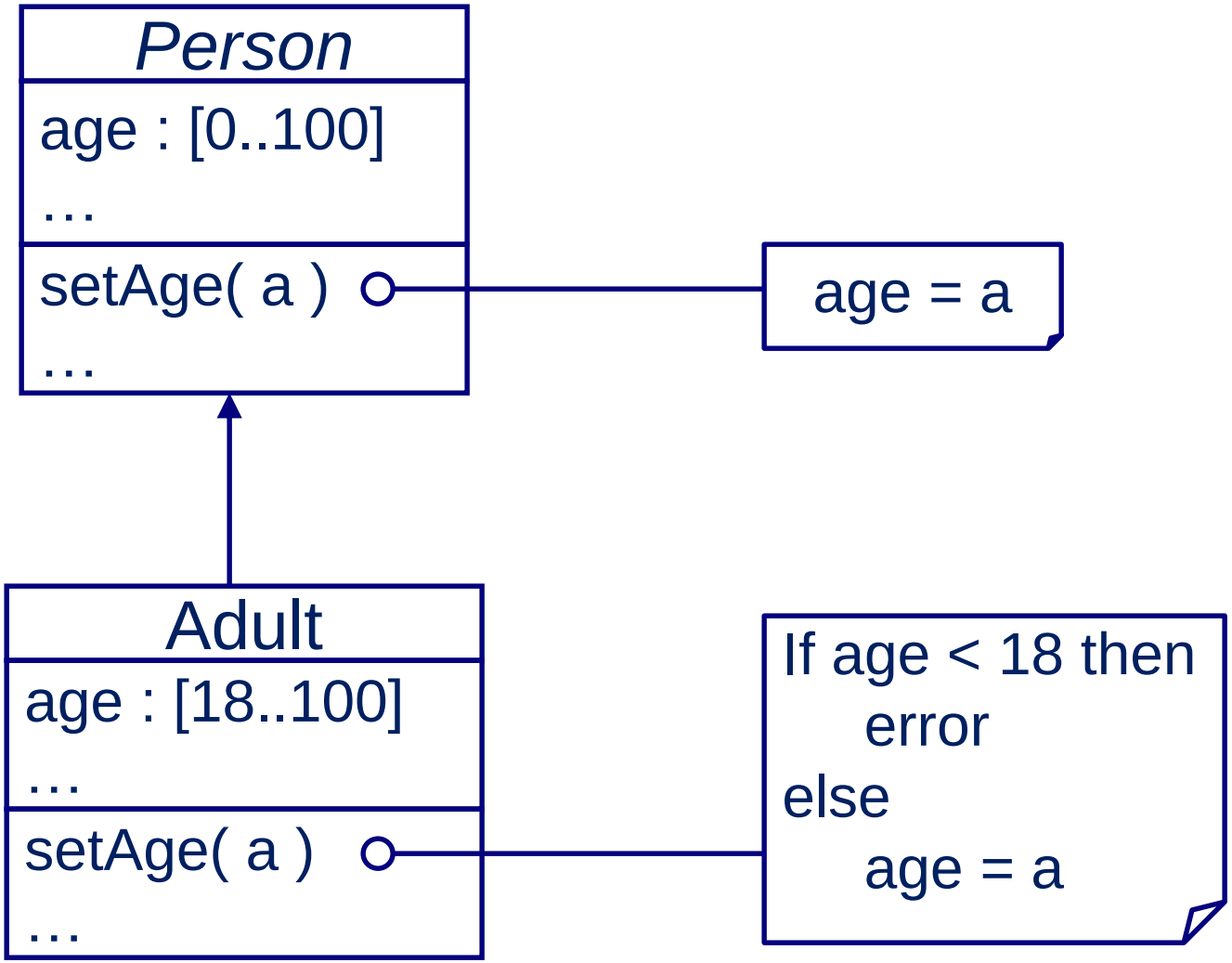


Specialization (Restriction)

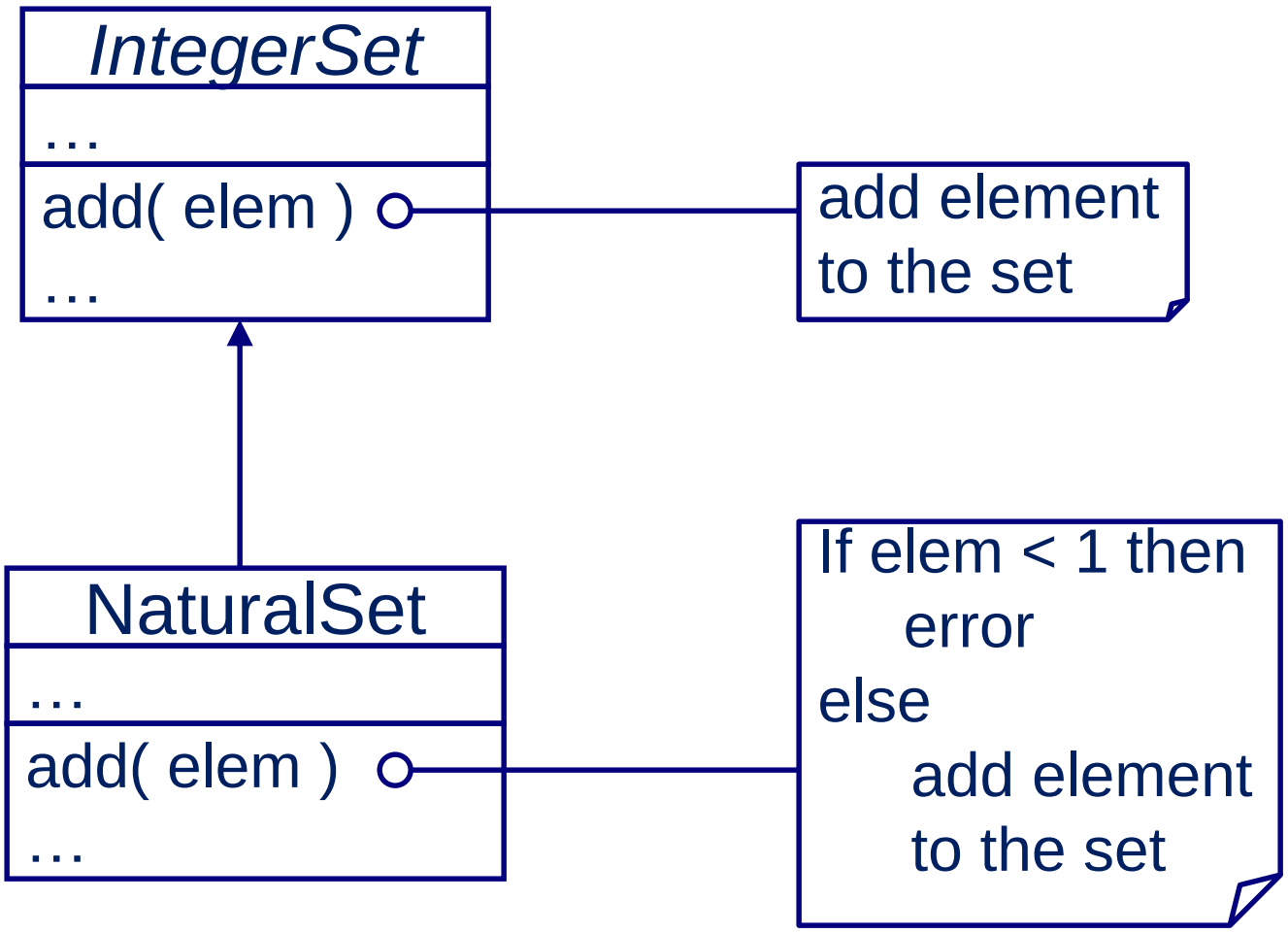
- Specialization means that derived class is behaviorally incompatible with the base class

- Behaviorally incompatible means that base class can’t always be replaced by the derived class

Example – Specialization (Restriction)



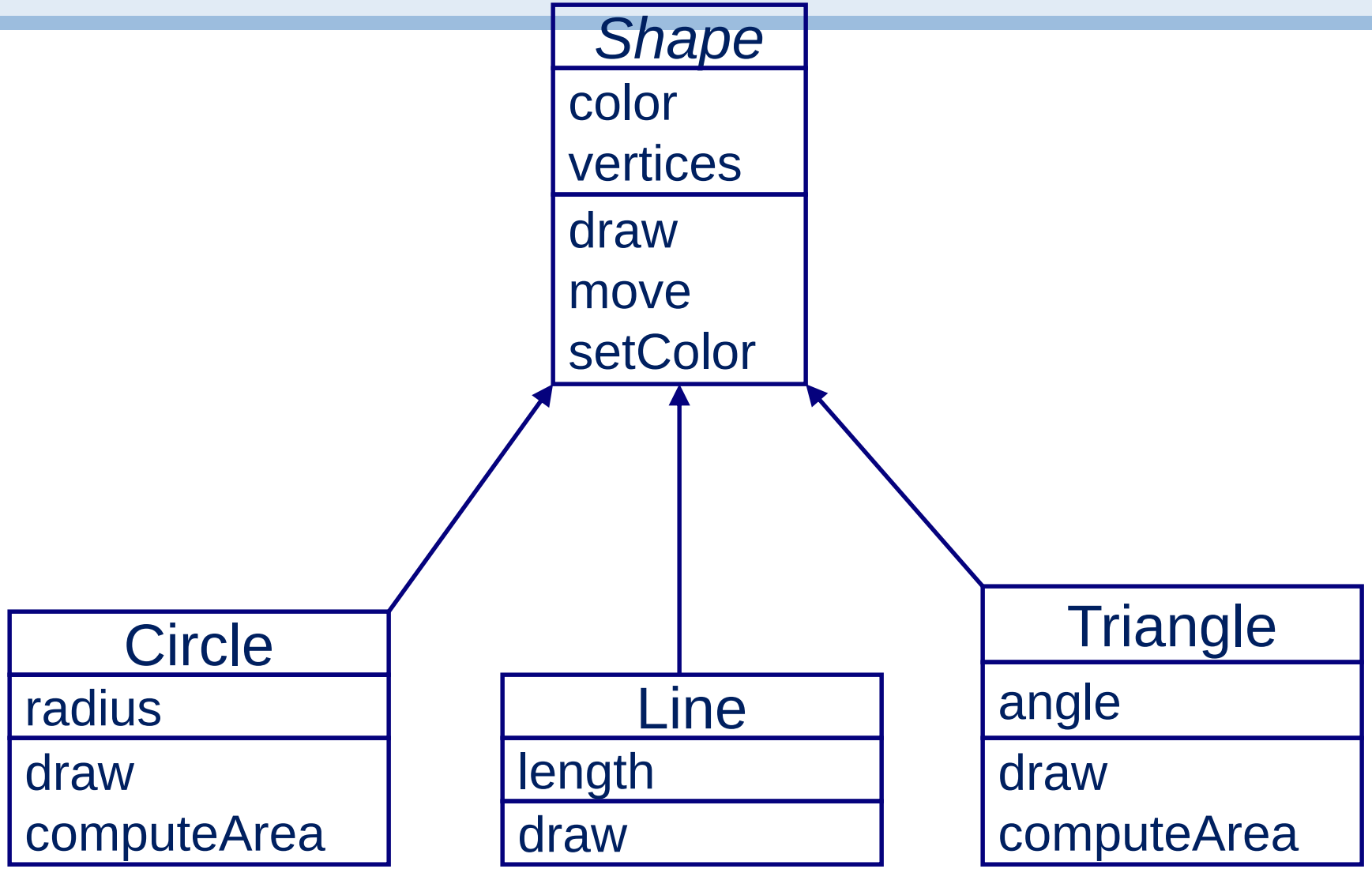
Example – Specialization (Restriction)



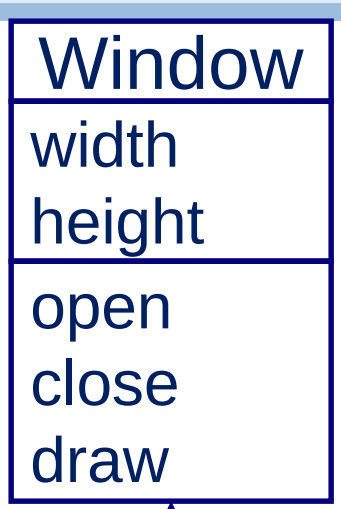
Overriding

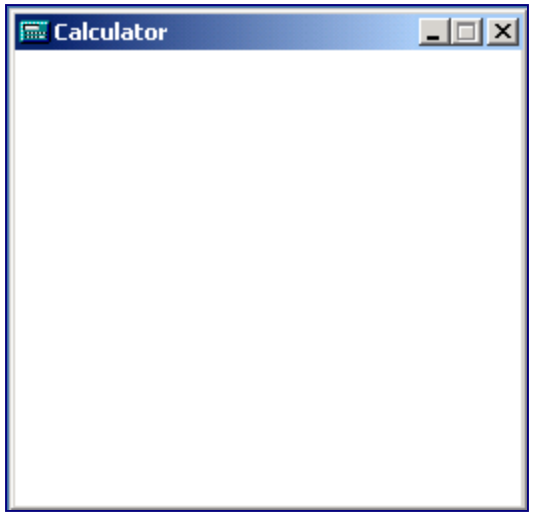
- A class may need to override the default behavior provided by its base class
- Reasons for overriding
  - Provide behavior specific to a derived class
  - Extend the default behavior
  - Restrict the default behavior
  - Improve performance

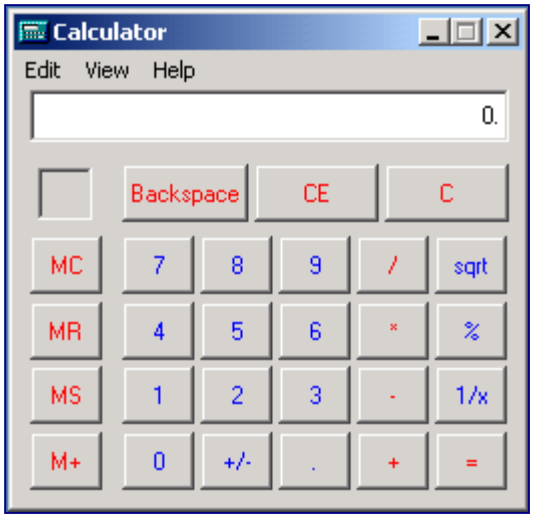
Example – Specific Behaviour

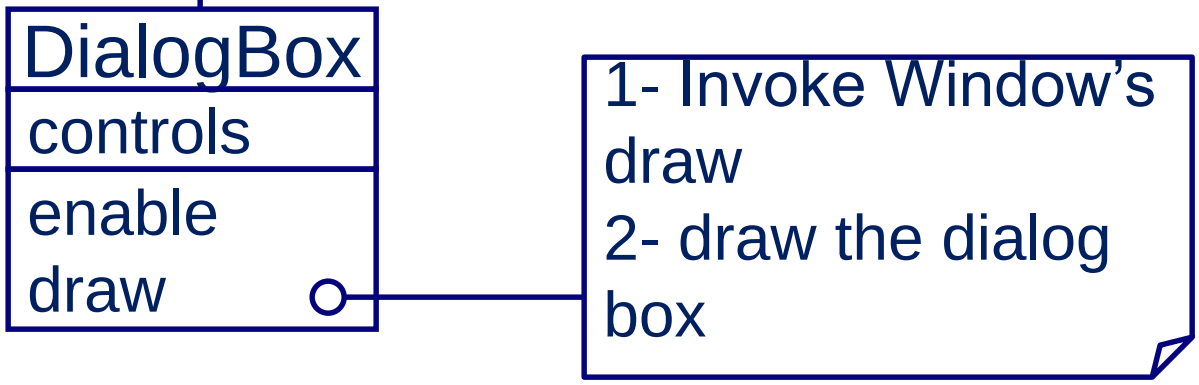


Example – Extension

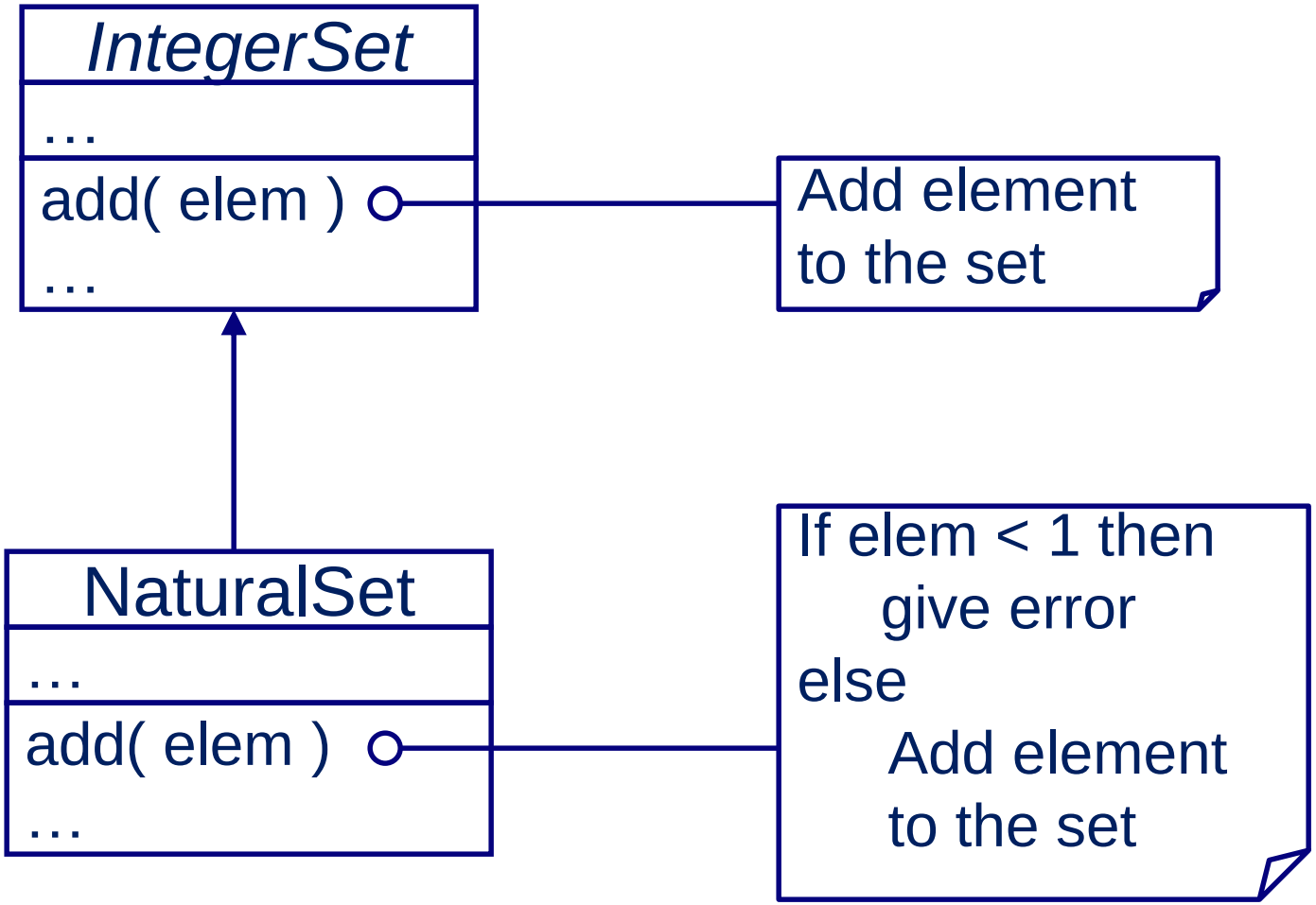






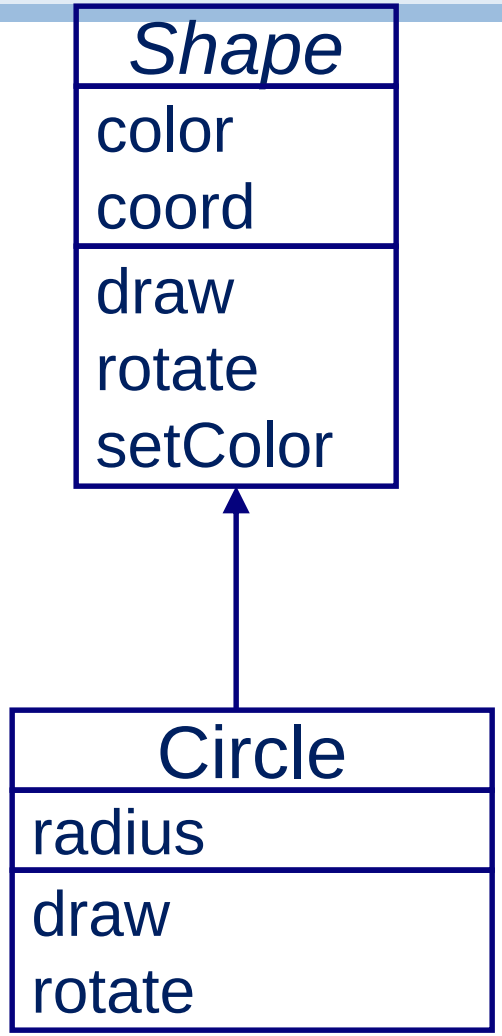


Example – Restriction



Example – Improve Performance

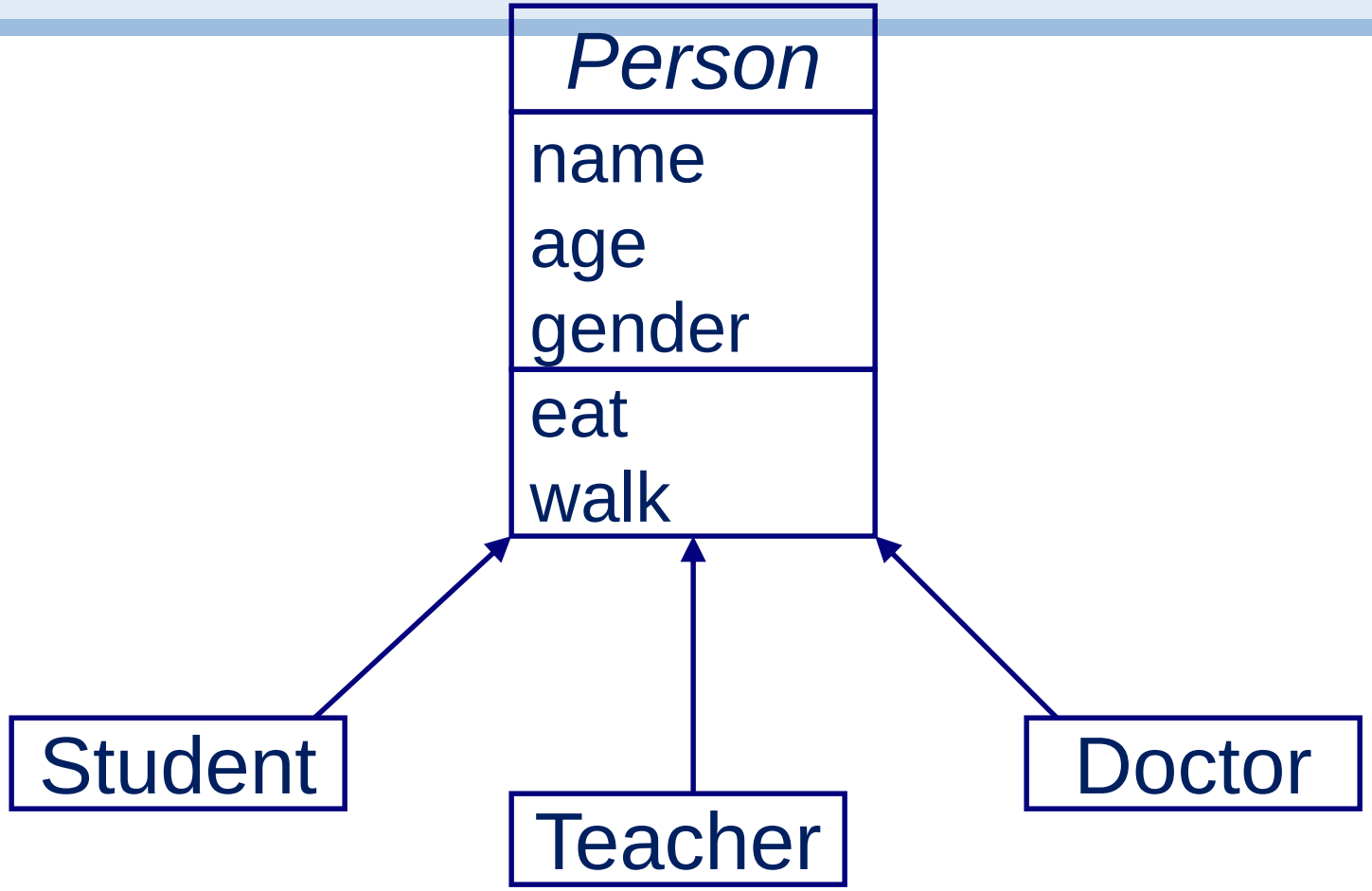
- Class Circle overrides *rotate* operation of class Shape with a Null operation.



Abstract Classes

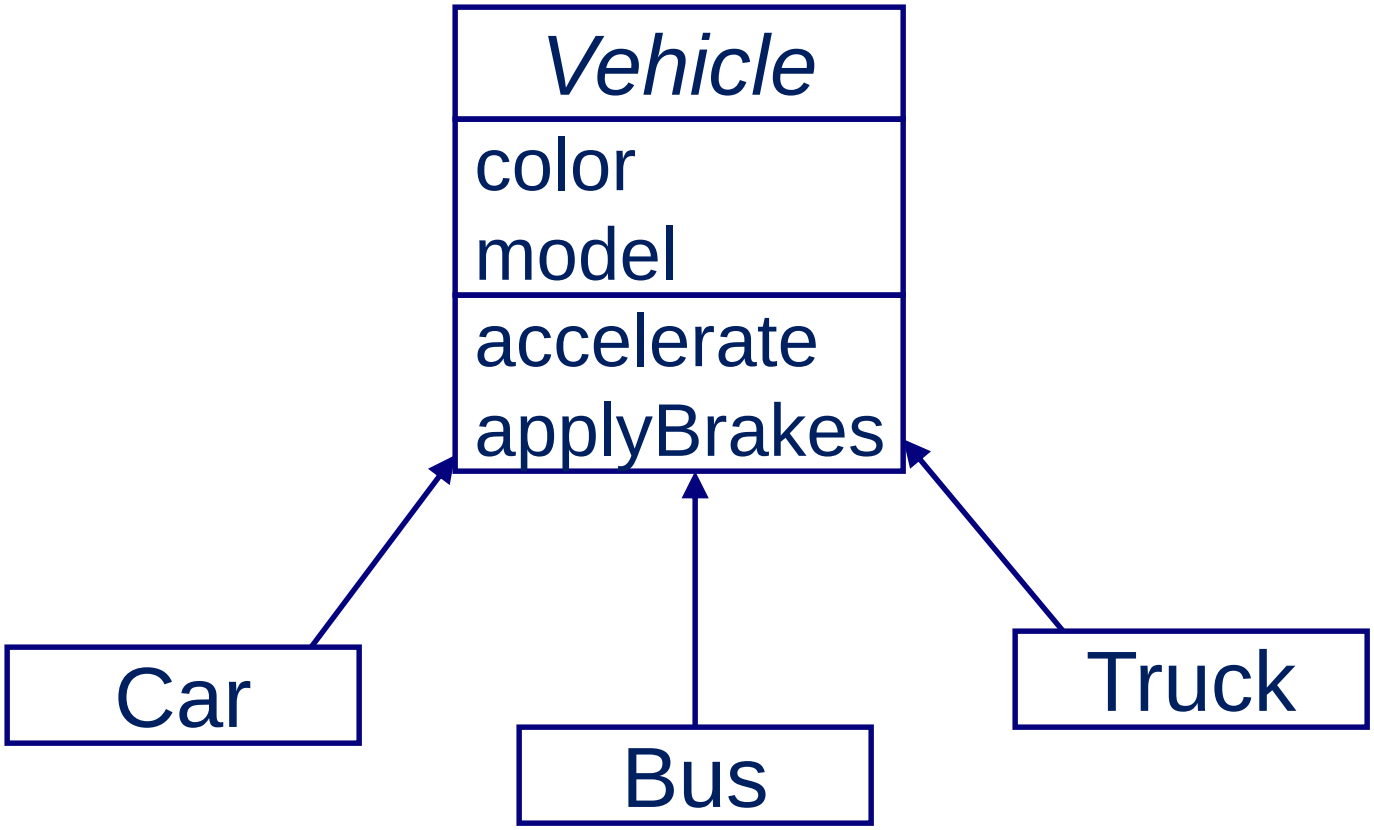
- An abstract class implements an abstract concept
- Main purpose is to be inherited by other classes
- Can’t be instantiated
- Promotes **reuse**

Example – Abstract Classes



- Here, Person is an abstract class

Example – Abstract Classes

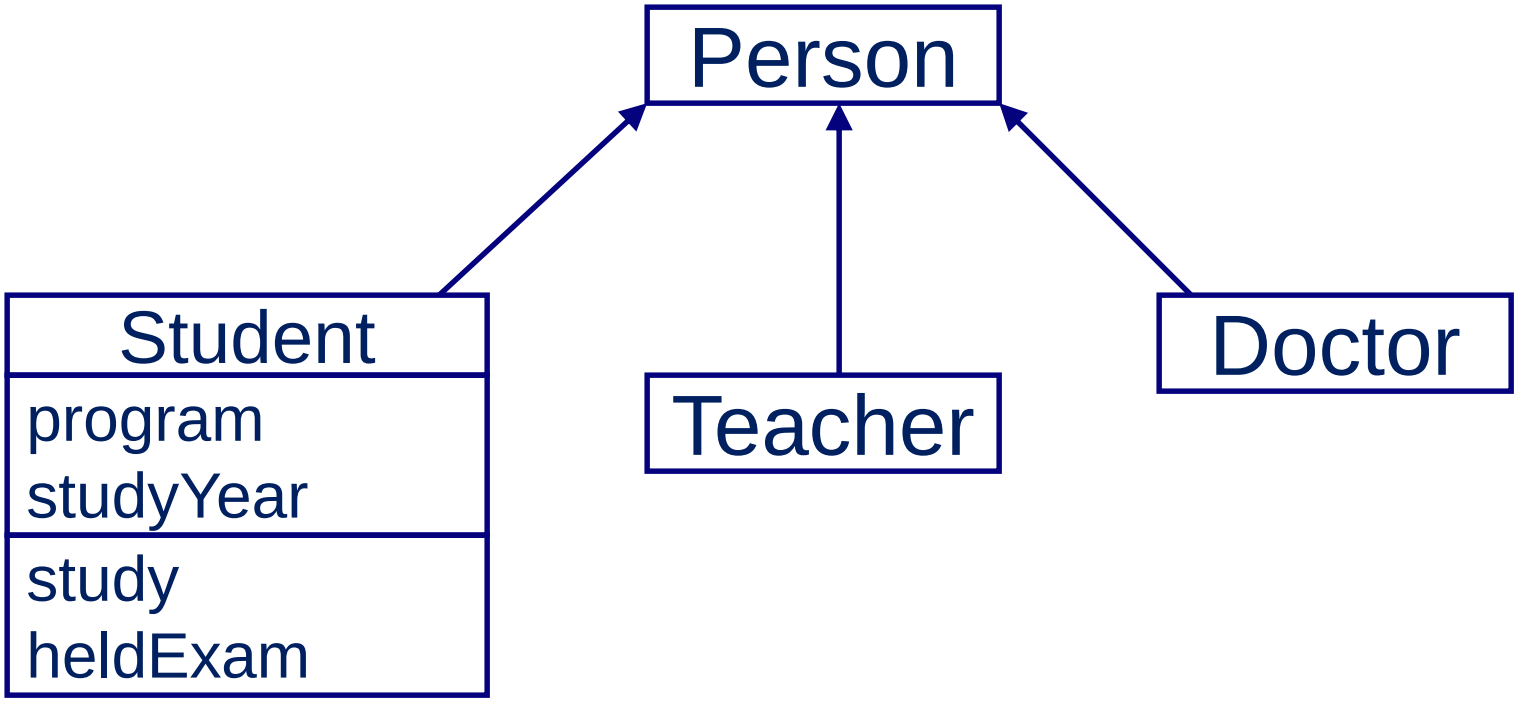


- Here, Vehicle is an abstract class

Concrete Classes

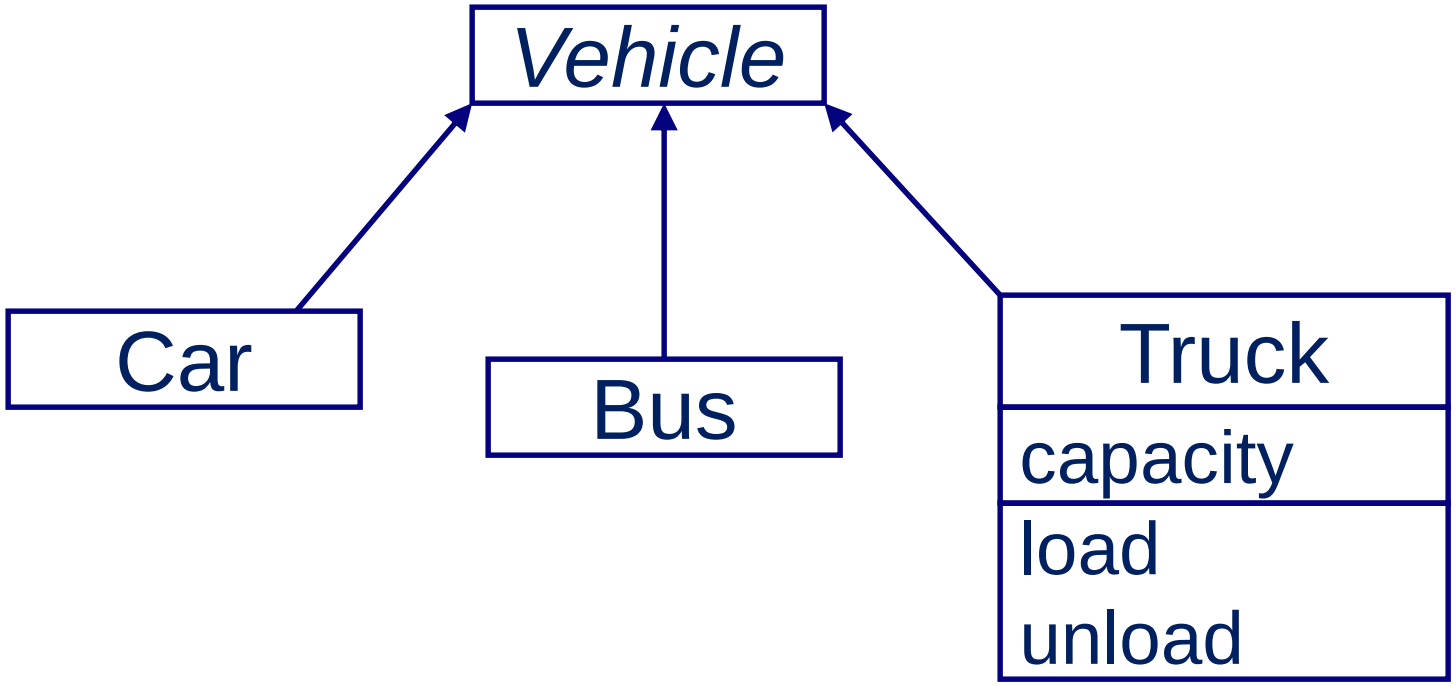
- A concrete class implements a concrete concept
- Main purpose is to be instantiated
- Provides implementation details specific to the domain context

Example – Concrete Classes



> Here, Student, Teacher and Doctor are concrete classes

Example – Concrete Classes



- Here, Car, Bus and Truck are concrete classes