Constructors in Inheritance and Method Overriding

Shirley B. Chu

June 22, 2020

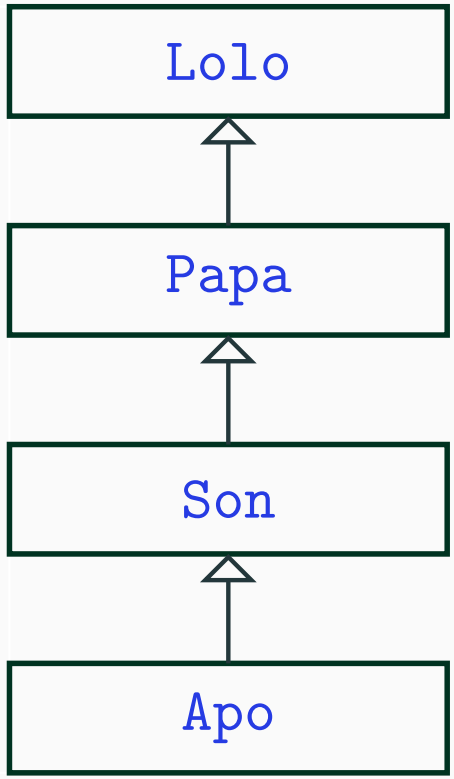
De La Salle University
College of Computer Studies

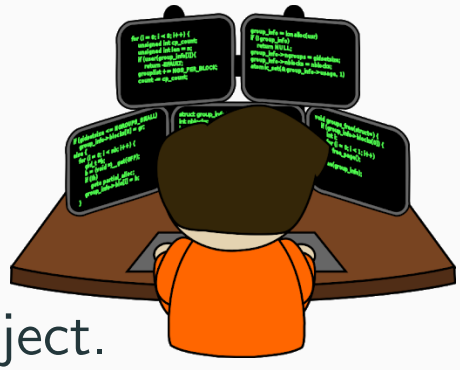
Recall: Constructors

A *constructor* is used to construct and initialize all the member variables.

Code the following

1. For each class, code only a no parameter constructor. Inside that constructor, simply display which class is being created.





2. Create a `Driver` class, and instantiate an `Apo` object.

Constructors in Inheritance

- Constructor of the superclass is invoked by the subclass.

Constructors in Inheritance

- Constructor of the superclass is invoked by the subclass.
- If the constructor did not have an explicit call the constructor of its superclass, the compiler automatically adds a call to the constructor of its superclass.

Constructors in Inheritance

- Constructor of the superclass is invoked by the subclass.
- If the constructor did not have an explicit call the constructor of its superclass, the compiler automatically adds a call to the constructor of its superclass.
  - It calls the no parameter constructor of the superclass.
  - It is added as the **first statement** of the constructor.

super

- `super` is used to refer to the members of its superclass (parent).

`super`

- `super` is used to refer to the members of its superclass (parent).
- `super();` with the appropriate parameters explicitly invokes the corresponding constructor of the superclass,

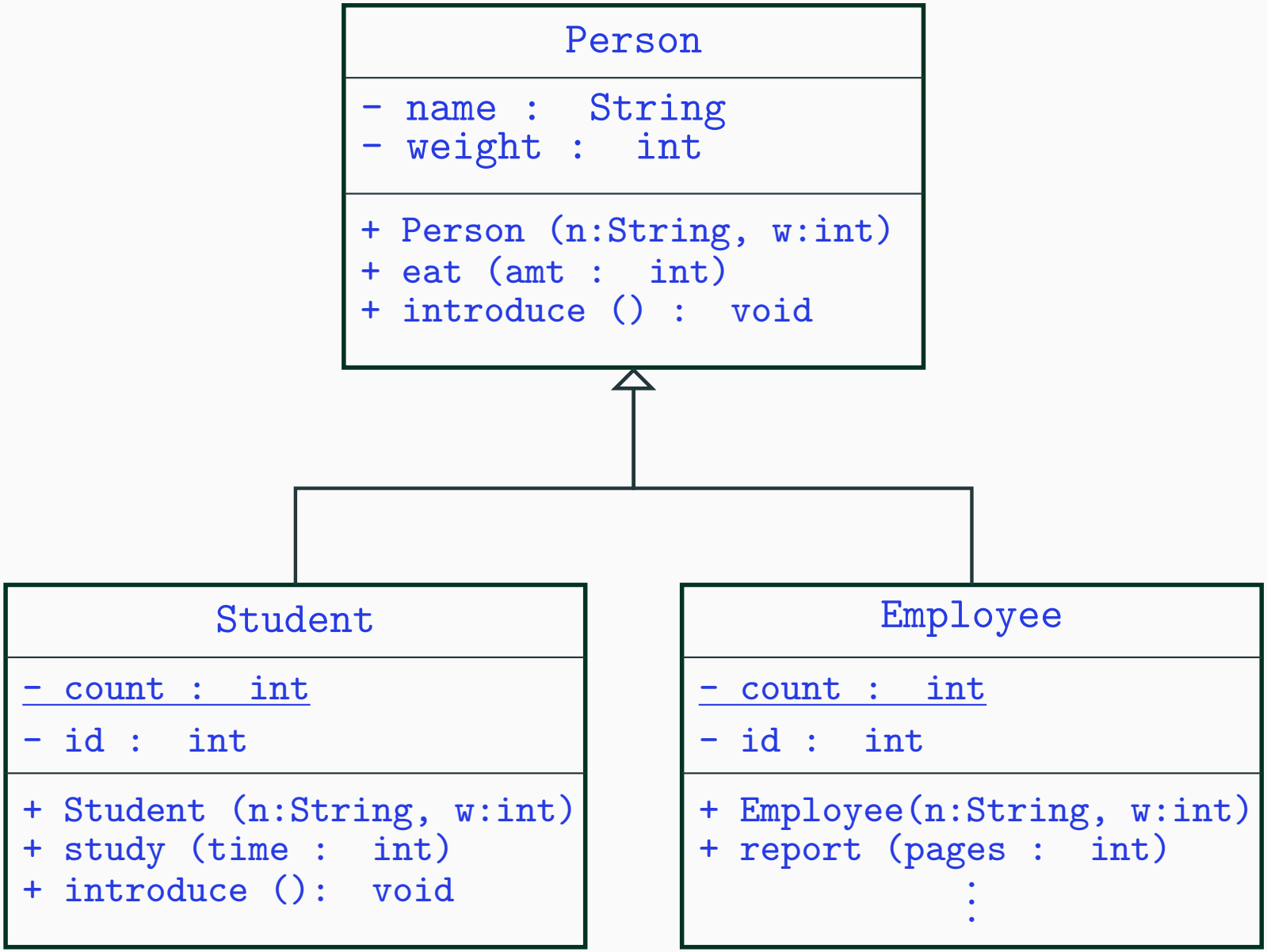
Method Overriding

- Two method sharing the same signature (name, parameters, return type). One is found in the superclass, the other is in the subclass.

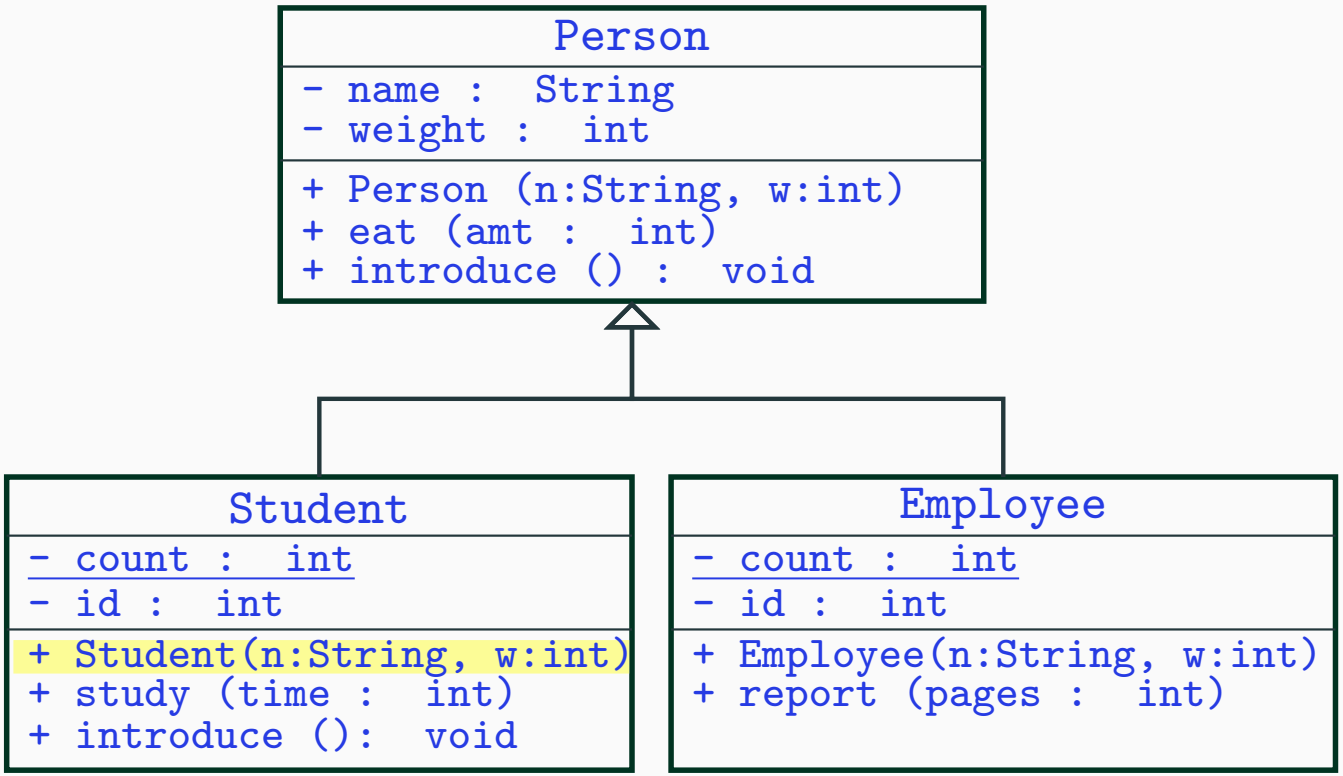
Method Overriding

- Two method sharing the same signature (name, parameters, return type). One is found in the superclass, the other is in the subclass.
- *Method overriding* means that the implementation of the **inherited method** will be modified. (e.g. `toString()` and `equals()`)

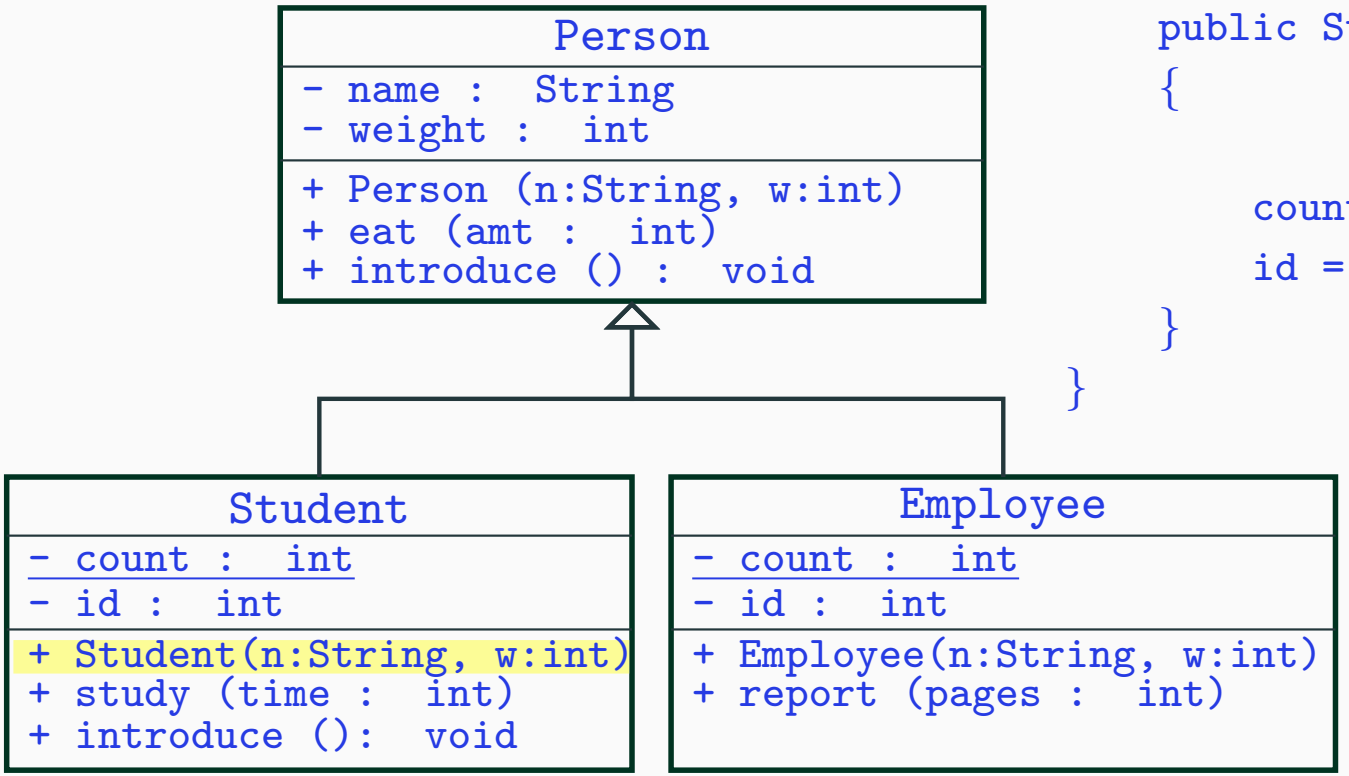
Example


Person
- name : String
- weight : int
+ Person (n:String, w:int)
+ eat (amt : int)
+ introduce () : void
Student
- count : int
- id : int
+ Student (n:String, w:int)
+ study (time : int)
+ introduce (): void
Employee
- count : int
- id : int
+ Employee(n:String, w:int)
+ report (pages : int)
:

Student's constructor

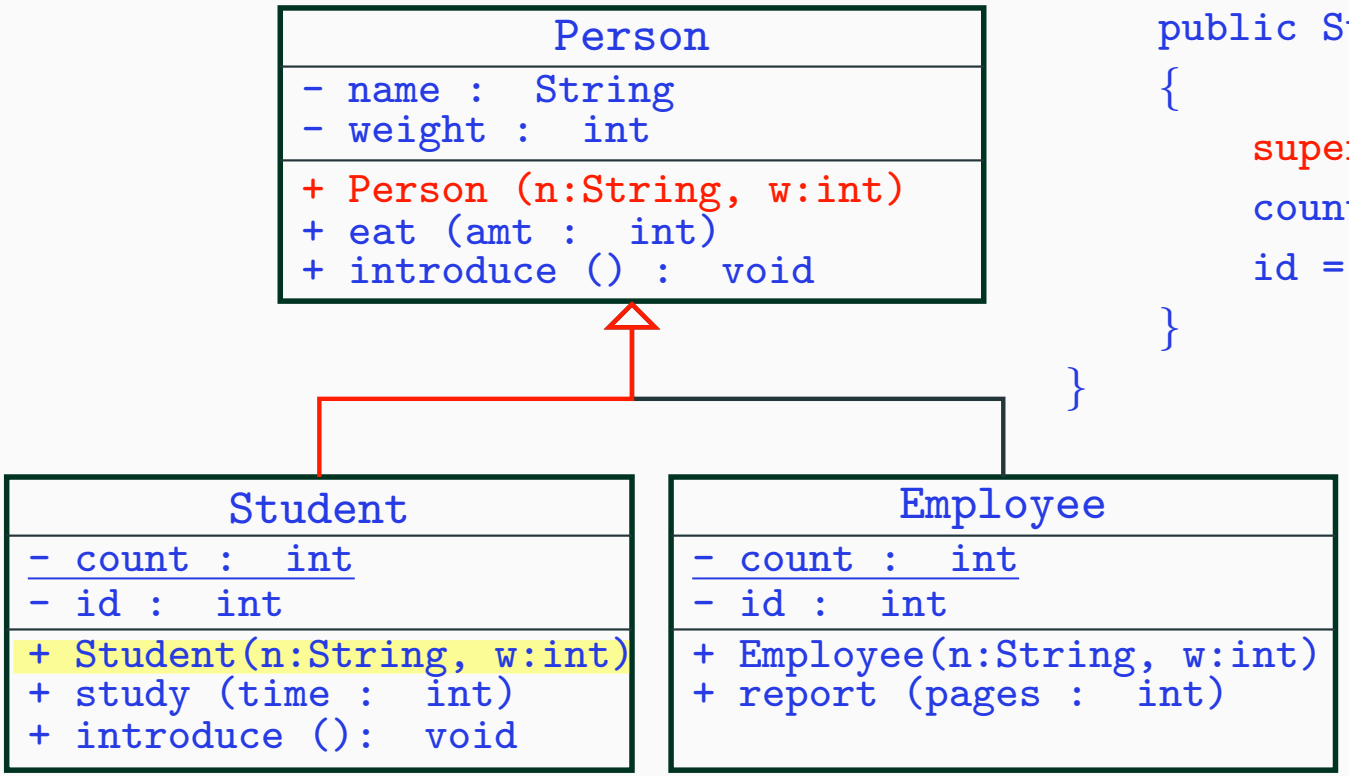


Student's constructor



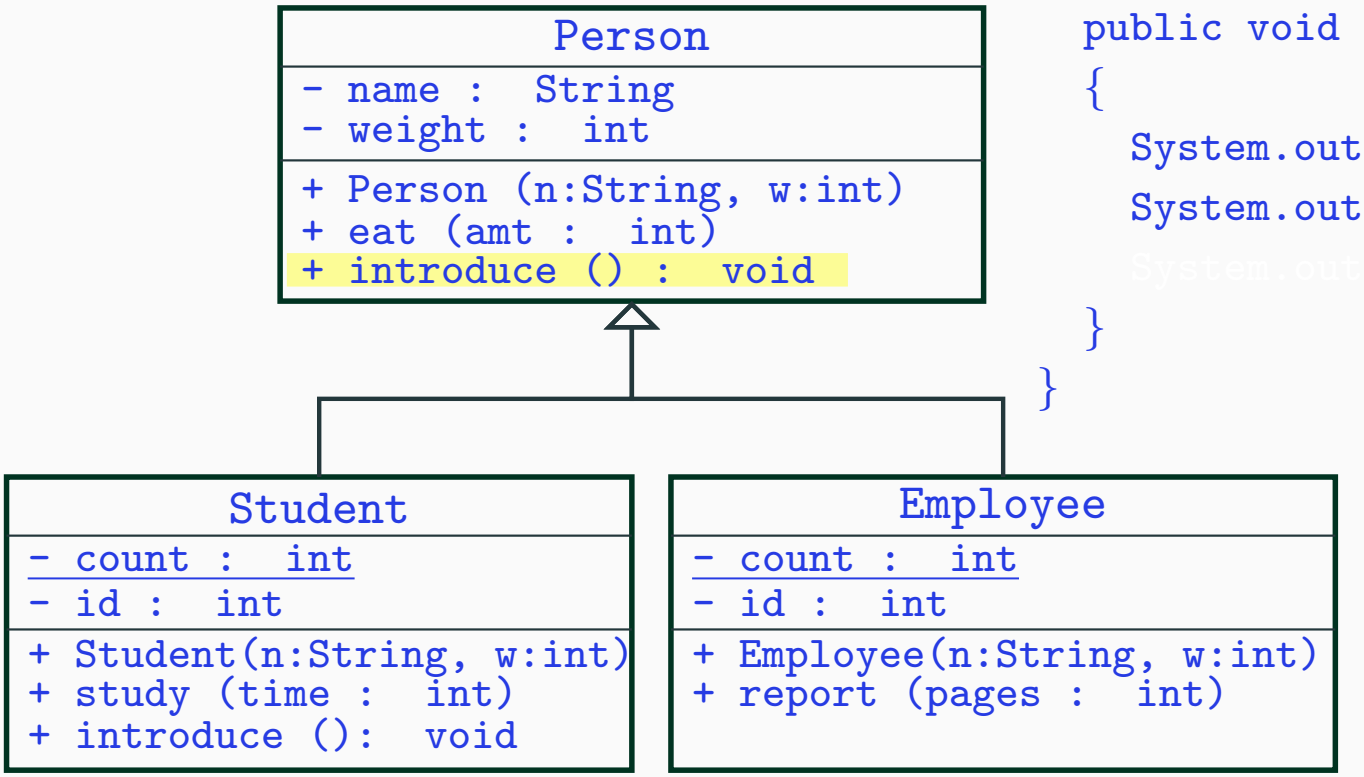

```
public class Student extends Person
{
    public Student (String n, int w)
    {

        count++;
        id = 2020 * 1000 + count;
    }
}
```

Student's constructor



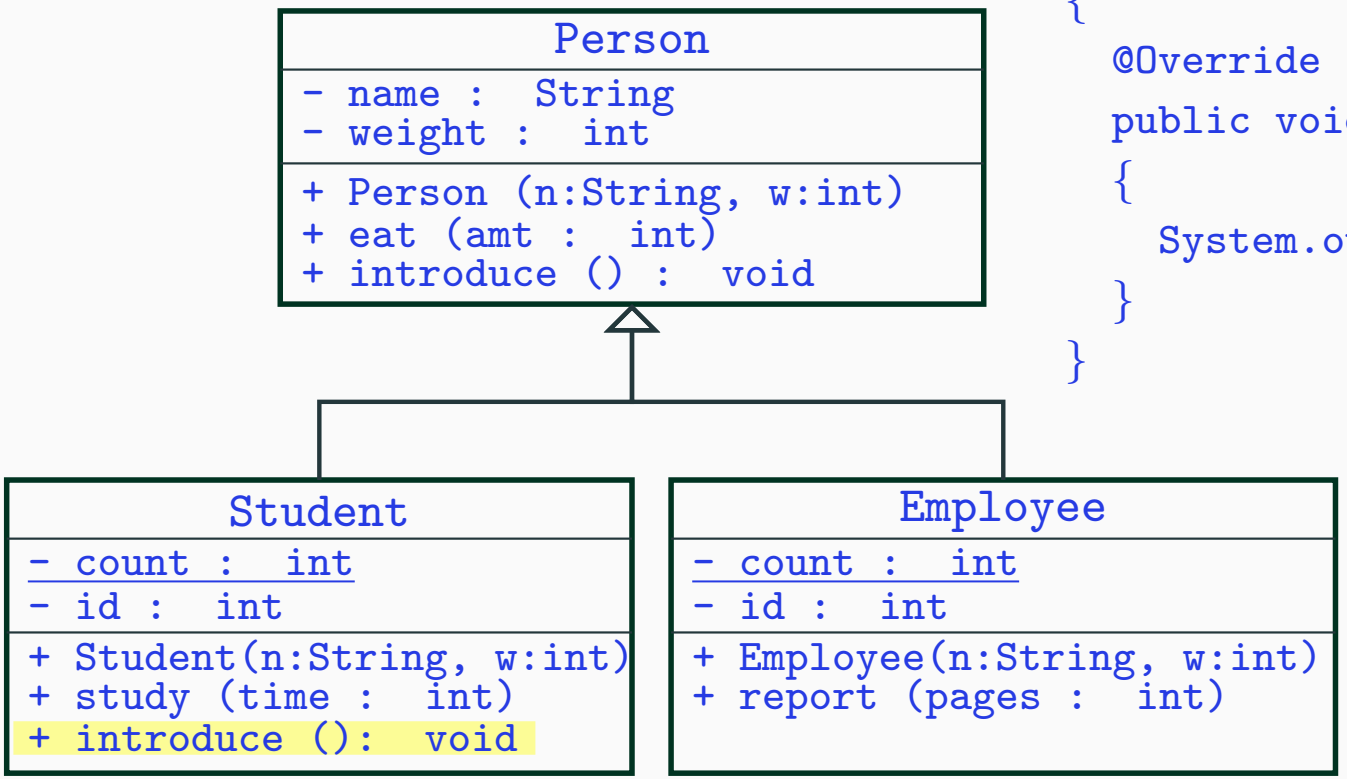

```
public class Student extends Person
{
    public Student (String n, int w)
    {
        super (n, w);
        count++;
        id = 2020 * 1000 + count;
    }
}
```

introduce() of Person



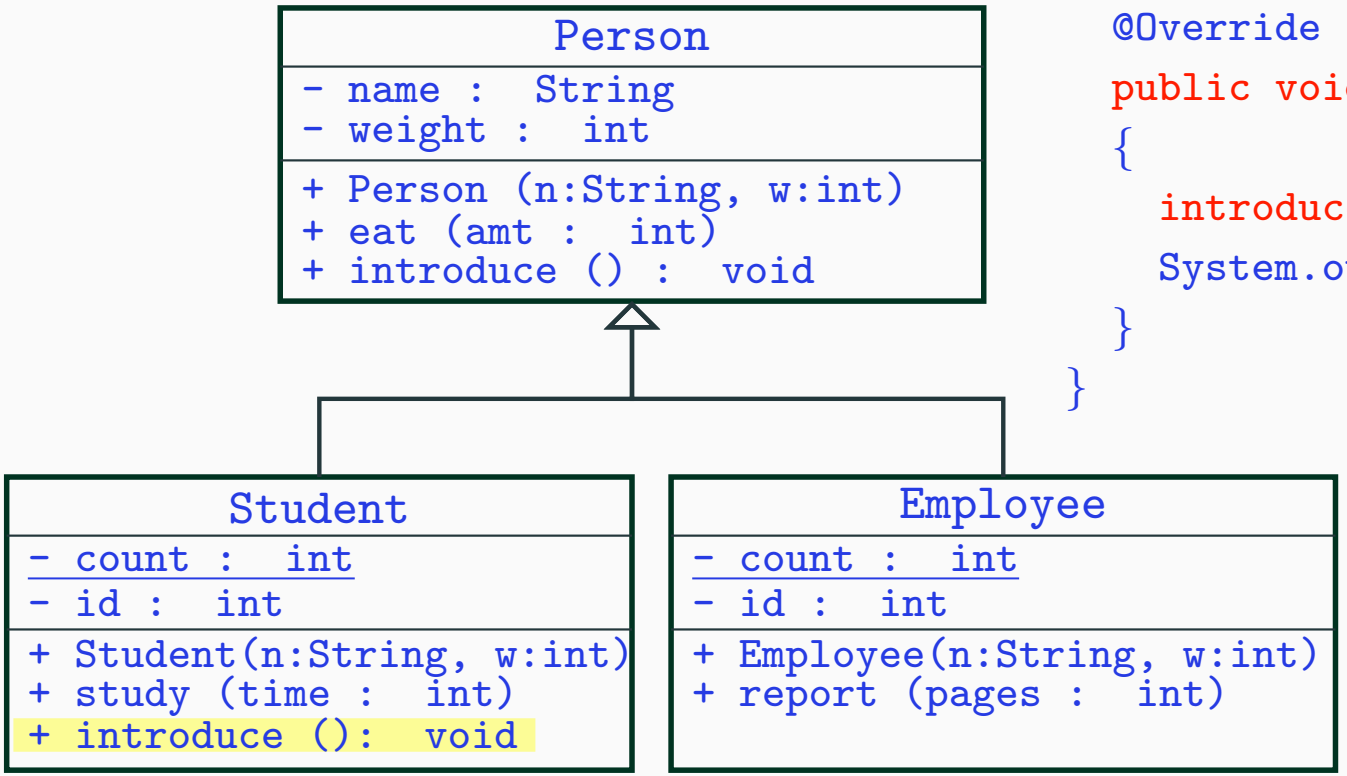

```
public class Person
{
  public void introduce ()
  {
    System.out.print ("I'm " + name);
    System.out.println (", " + weight
    System.out.println ( + " lbs");
  }
}
```

introduce() of Student



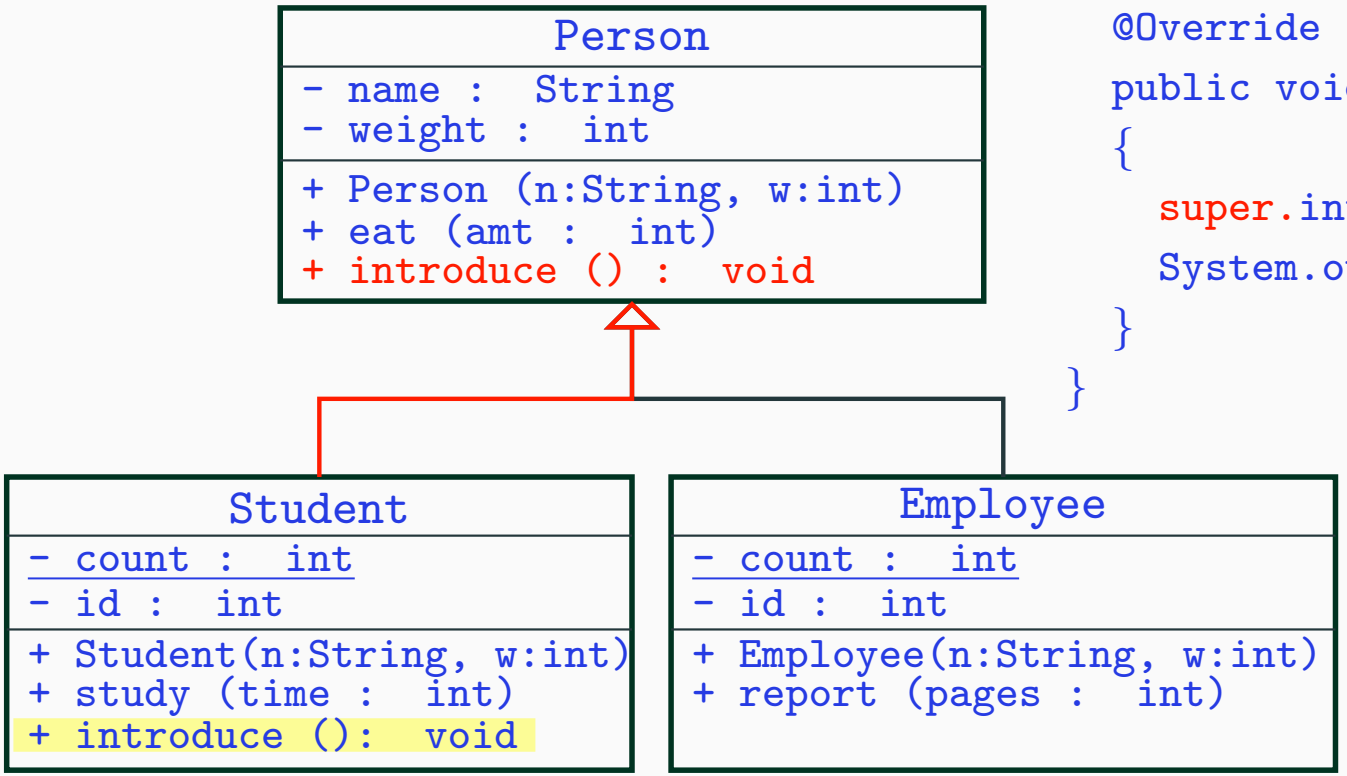

```
public class Student extends Person
{
  @Override
  public void introduce ()
  {
    System.out.println ("ID " + id);
  }
}
```

introduce() of Student



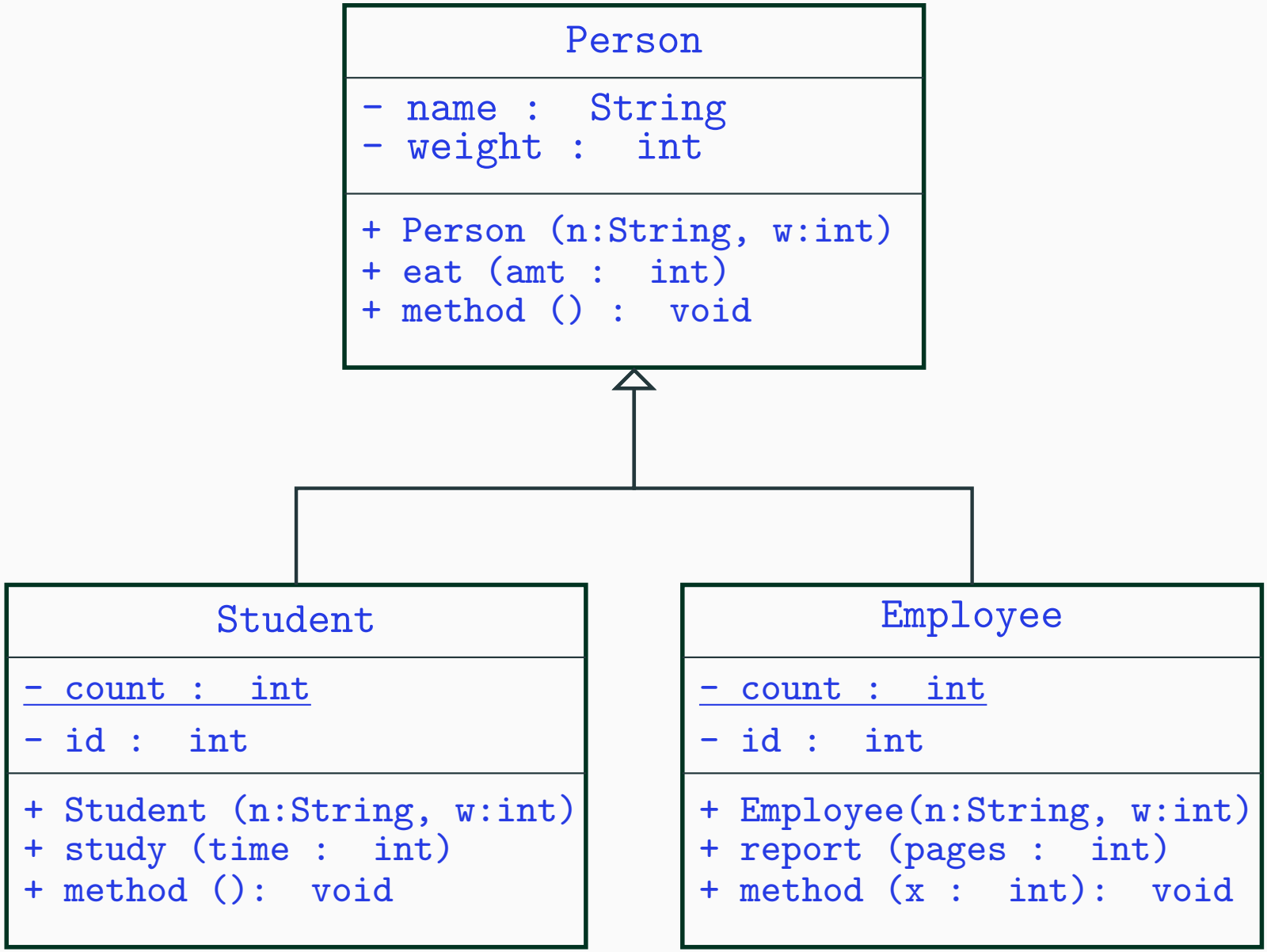

```
public class Student extends Person
{
  @Override
  public void introduce ()
  {
    introduce ();
    System.out.println ("ID " + id);
  }
}
```

introduce() of Student

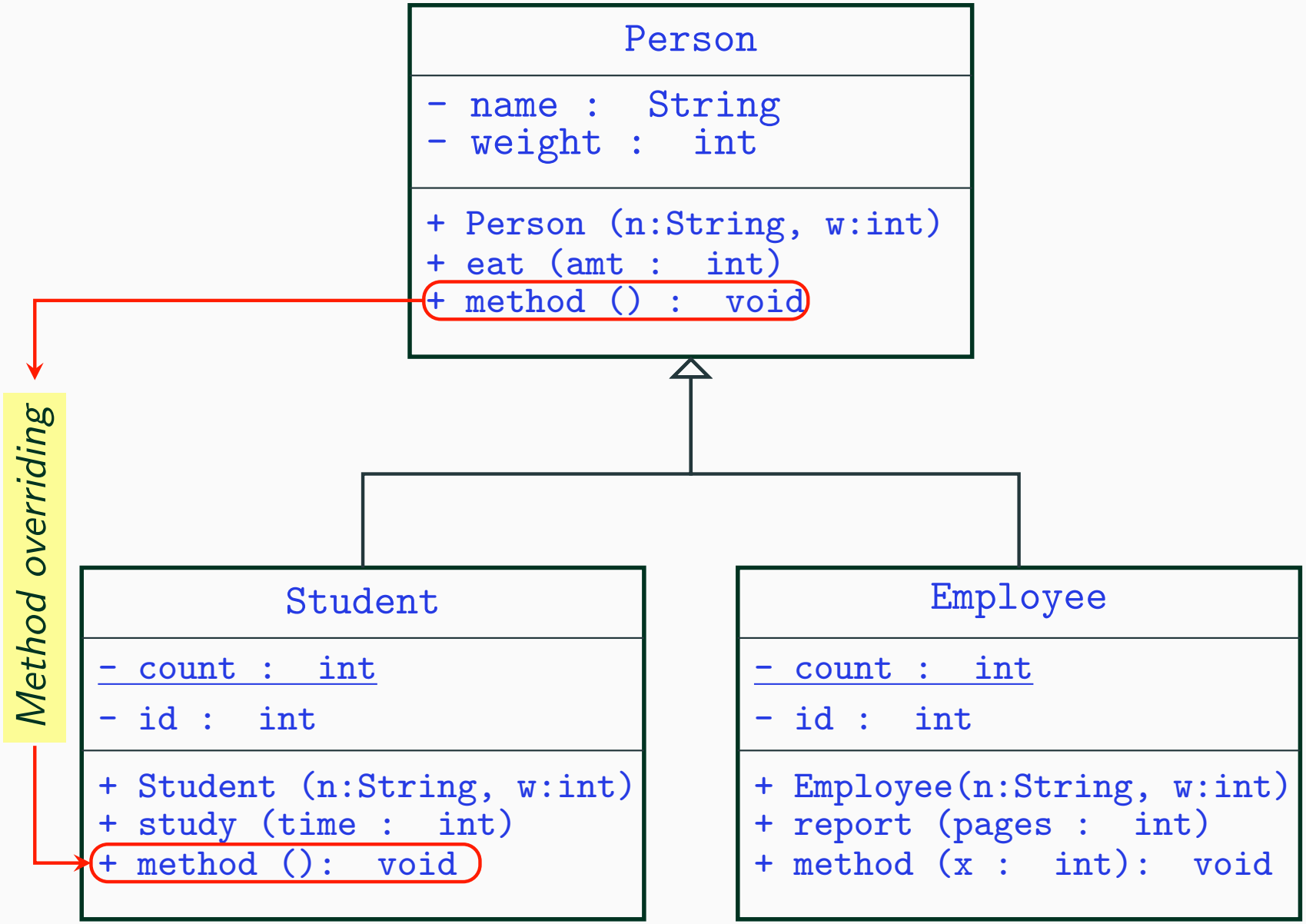



```
public class Student extends Person
{
  @Override
  public void introduce ()
  {
    super.introduce ();
    System.out.println ("ID " + id);
  }
}
```

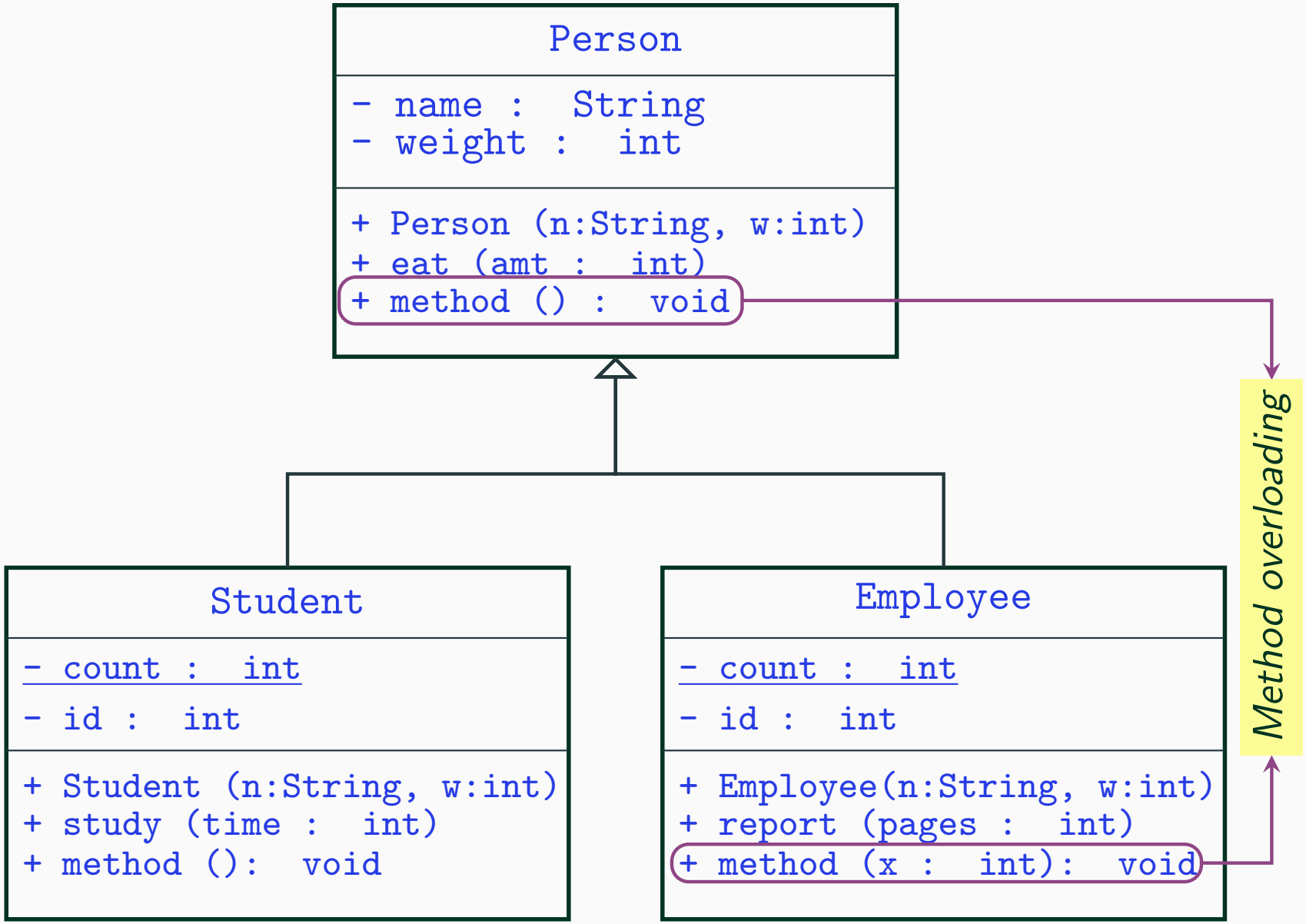
Overloaded or Overridden method



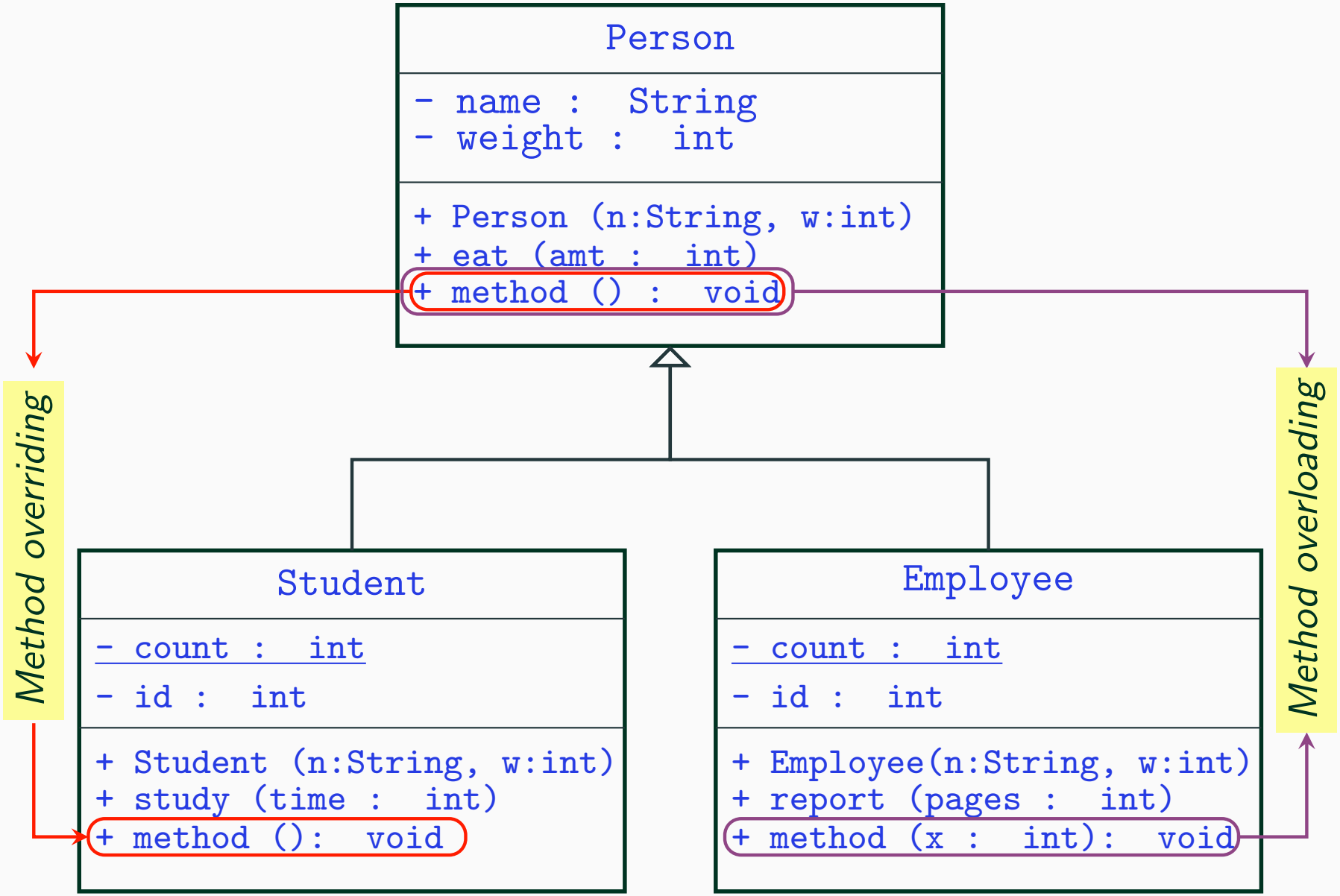
Overloaded or Overridden method



Overloaded or Overridden method



Overloaded or Overridden method



☺ **Thank you!** ☺