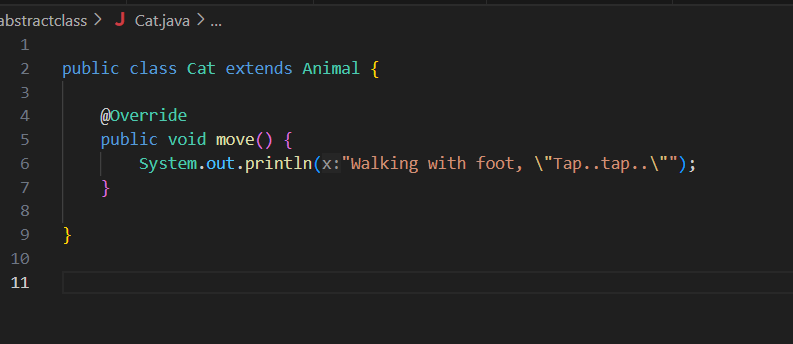
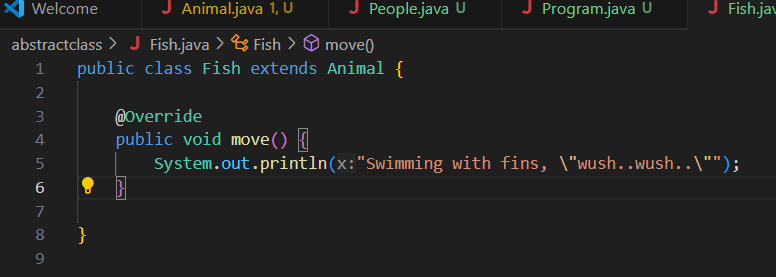
**Jobsheet 09 – Abstract Class and Interface**

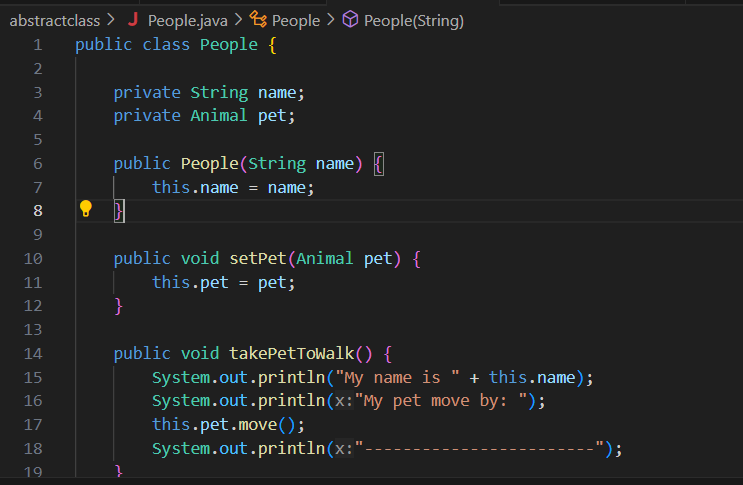
Erwan Majid 08 2i

* Experiment 1: Abstract Class

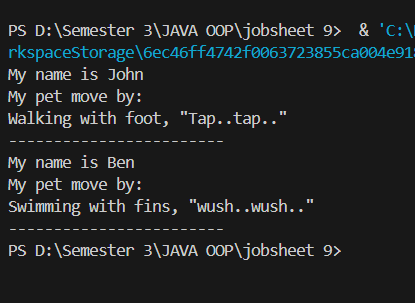








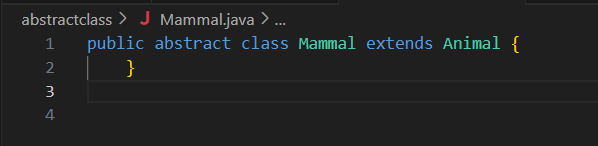




Discussion questions:

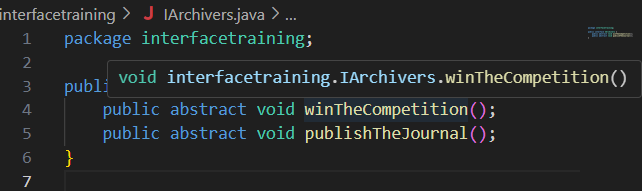
Can it be possible if a class that extends an abstract class does not implement abstract methods

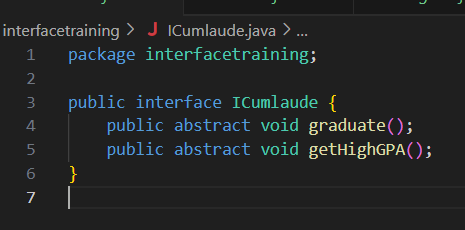
in its parent class? Prove it!

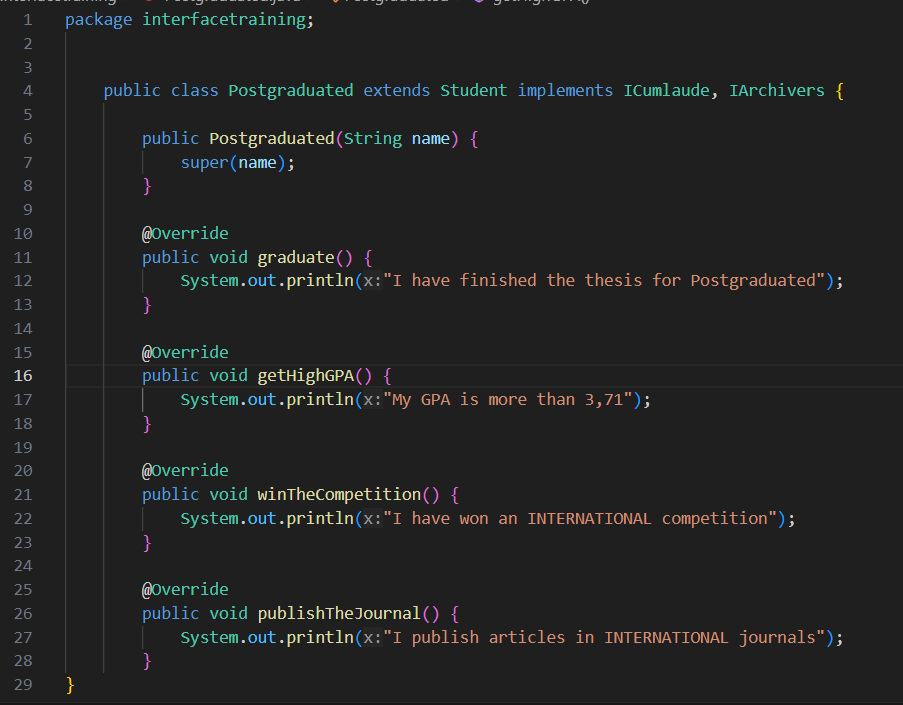


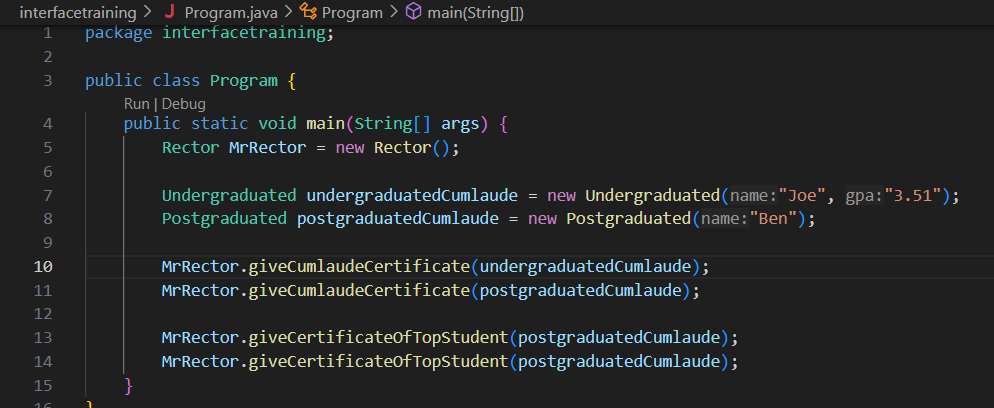
This setup proves that Mammal can extend Animal without implementing the move() method because it’s also declared abstract. Only the concrete subclass Cat is required to implement move(). This is the essence of abstract class flexibility in Java.

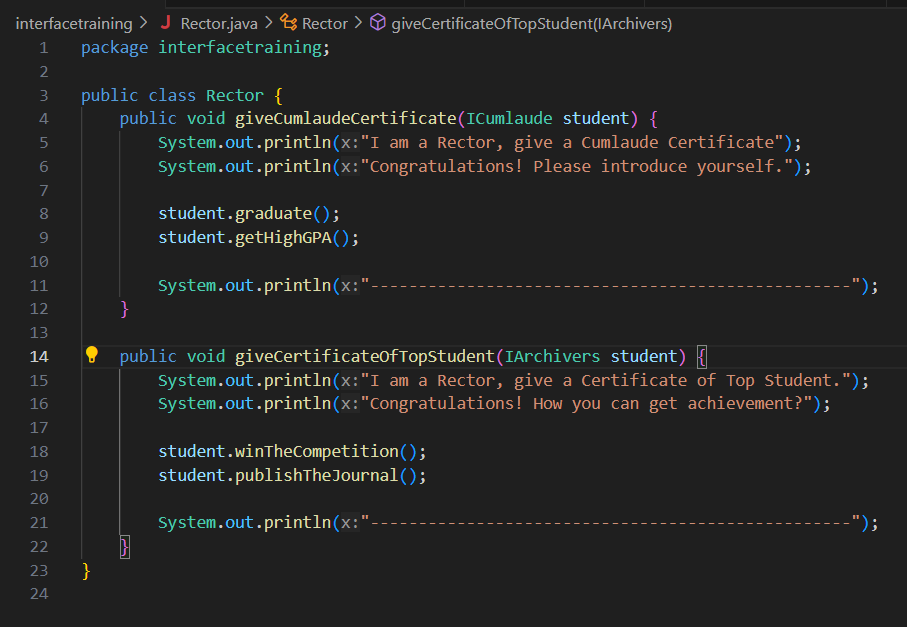
* Experiment 2: Interface

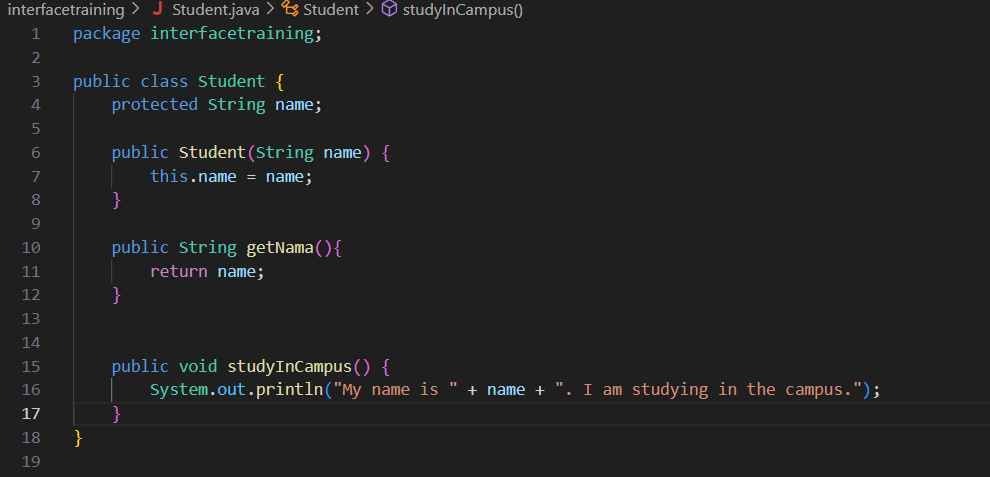


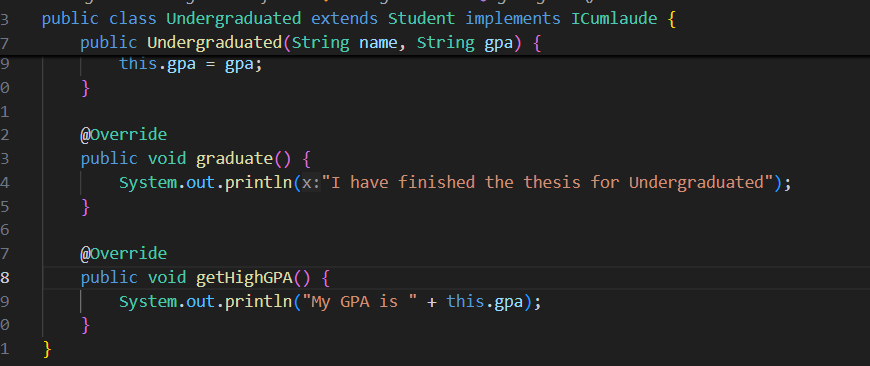












a. Whydidstep 9 cause an error? Explain!

The code causes an error because the giveCumlaudeCertificate method in the Rector

class expects an instance of ICumlaude as its parameter. However, in your main

method, you are passing ordinaryStudent, which is an instance of the Student class,

and Student does not implement the ICumlaude interface.

b. Canthe studyInCampus() method be called from the undergraduatedCumlaude

object in the Program class? Why?

Since undergraduatedCumlaude is an Undergraduated object, and Undergraduated

inherits from Student, it has access to all public and protected methods from Student,

including studyInCampus(). Therefore, calling

undergraduatedCumlaude.studyInCampus() is valid and will work without any issues.

c. Canthe studyInCampus() method be called from the student parameters in the

giveCumlaudeCertificate() method in the Rector class? Why?

Directly calling studyInCampus() on student in giveCumlaudeCertificate is not

possible because student is of type ICumlaude, which doesn’t have this method.

d. Modify the giveCumlaudeCertificate() method on the Rector class so that the

results of the Program class execution become as follows

Experiment 3  
