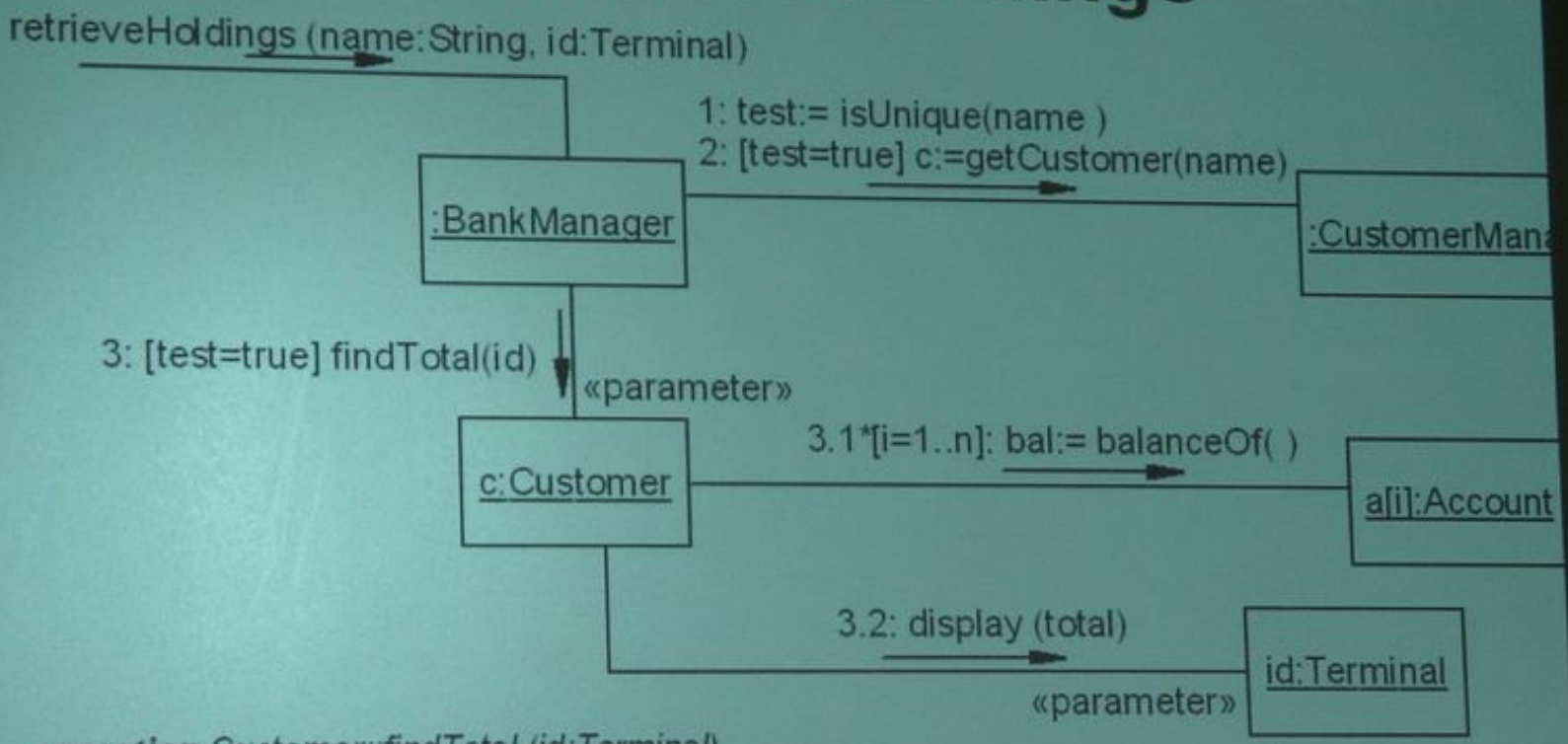
Retrieve Holdings



operation *Customer::findTotal (id:Terminal)*

foreach *account of the customer*

bal=account.balanceOf();

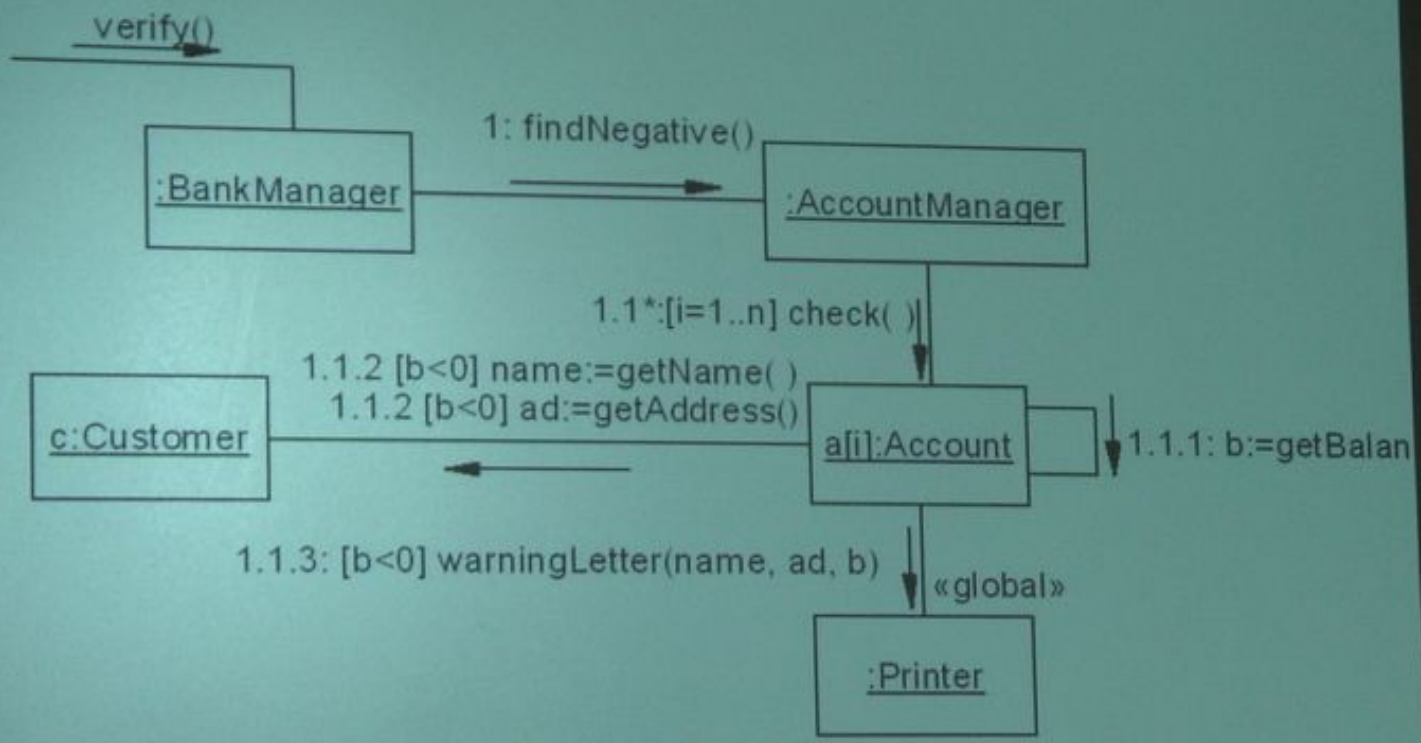
total=total+bal;

end foreach;

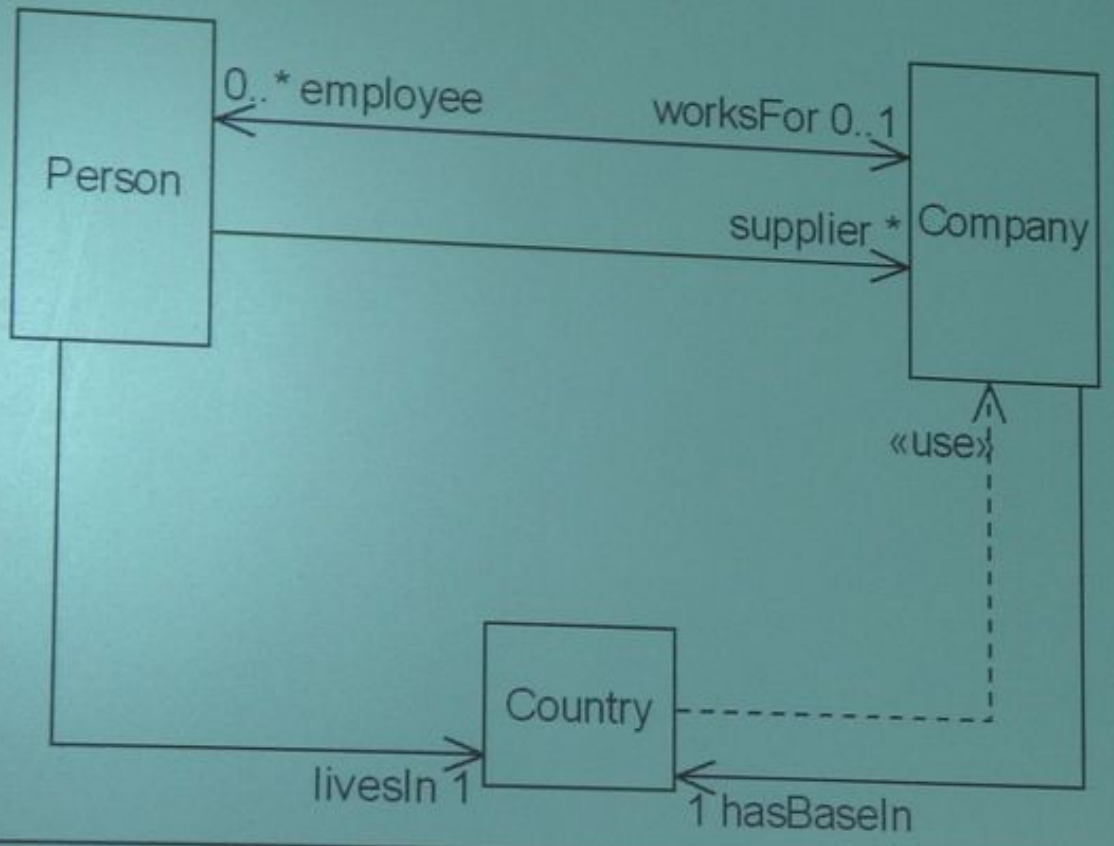
display the total holdings on the teller's terminal;

Exercise

Monthly Verification

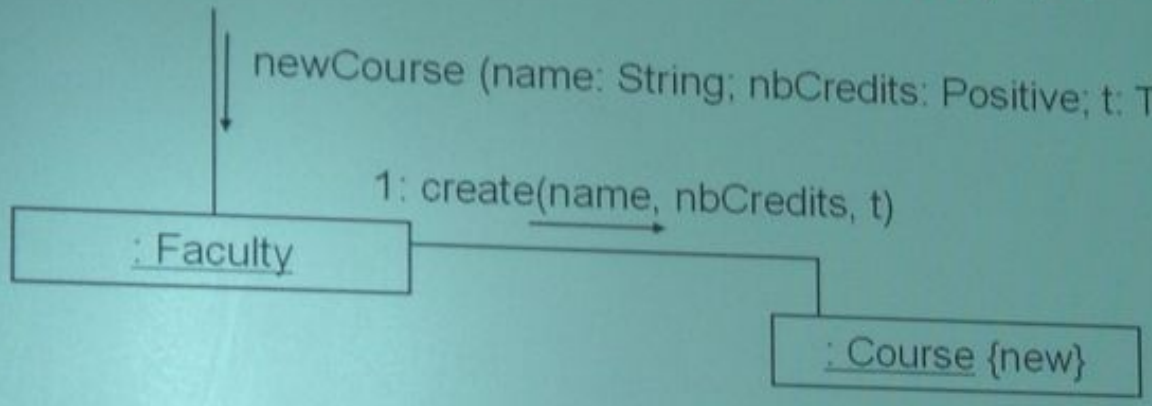


Dependency Model: Simplified



Exercise

School: newCourse, V1

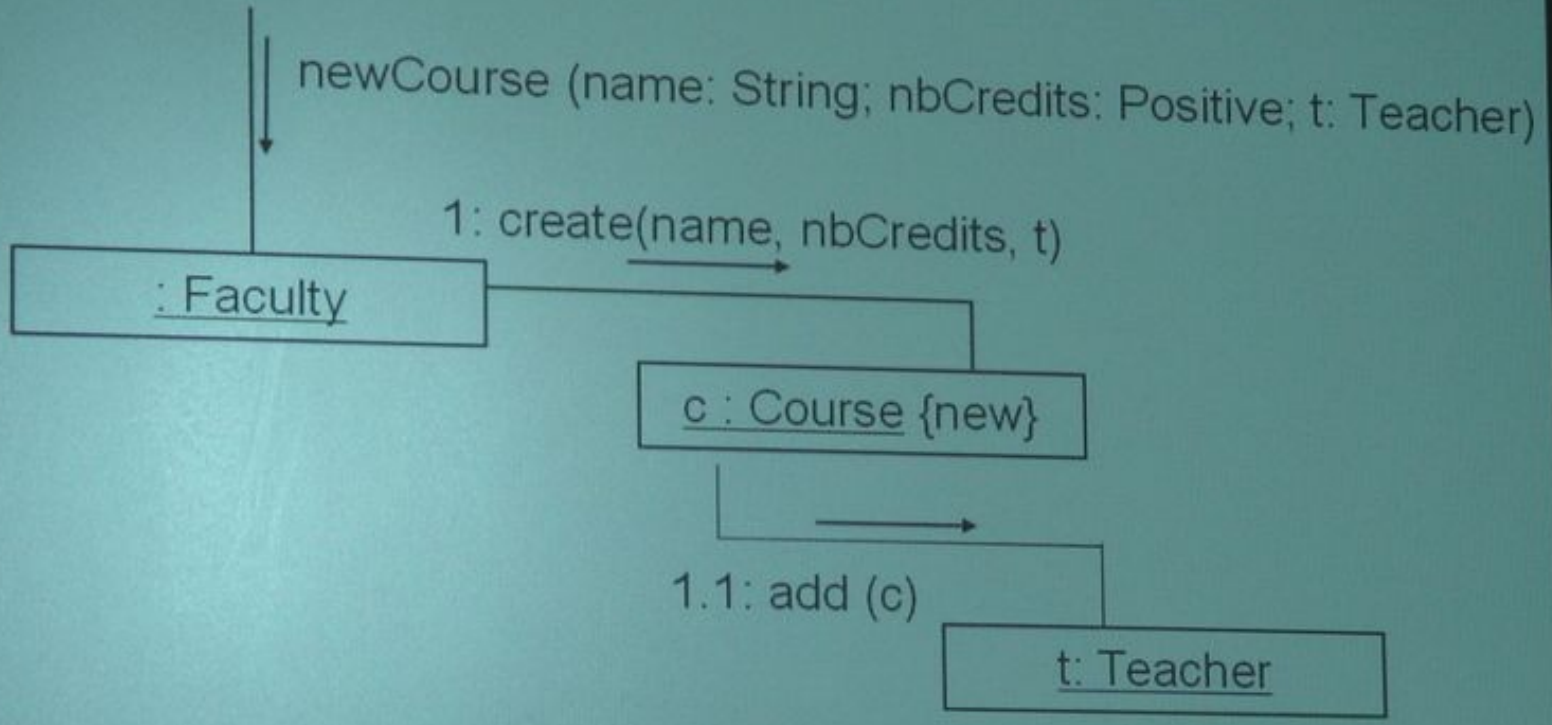


In order to decentralize responsibilities inside the school, we introduce a class Faculty.

The real problem to be solved is to decide how the link to the teacher of course is implemented. One way is to decide that a course "knows" abo only teacher.

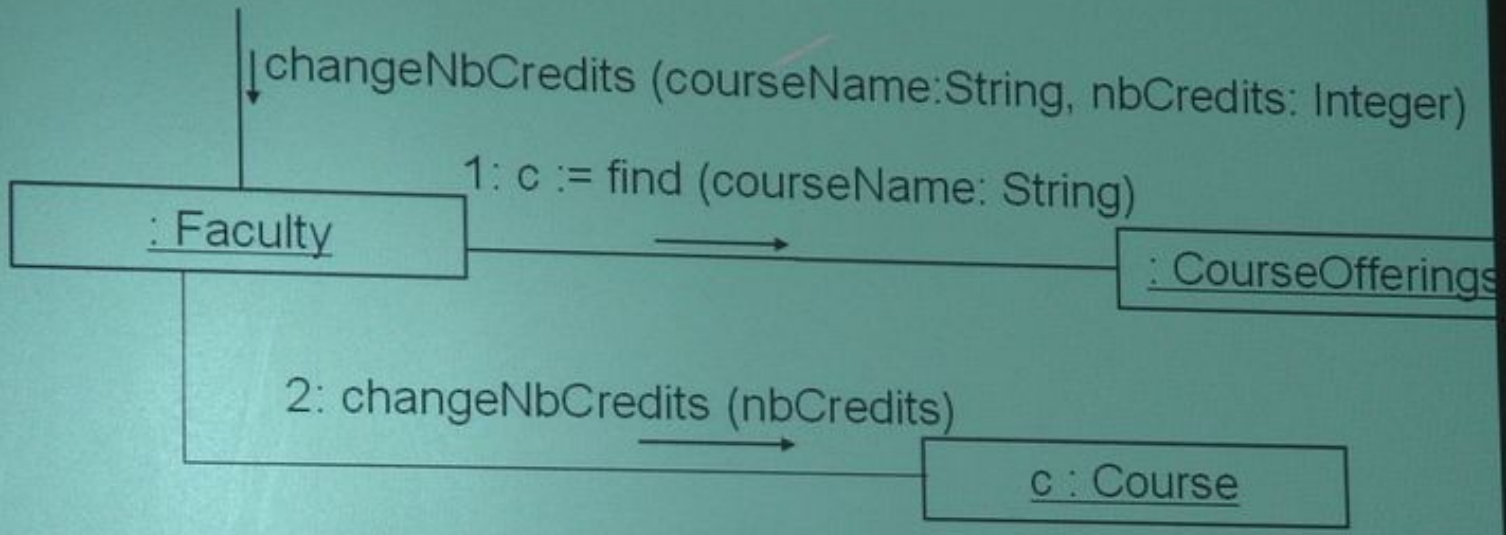
There are other problems here...

School: newCourse, V2



In this version, a teacher knows about all her/his courses.

School: changeNbCredits

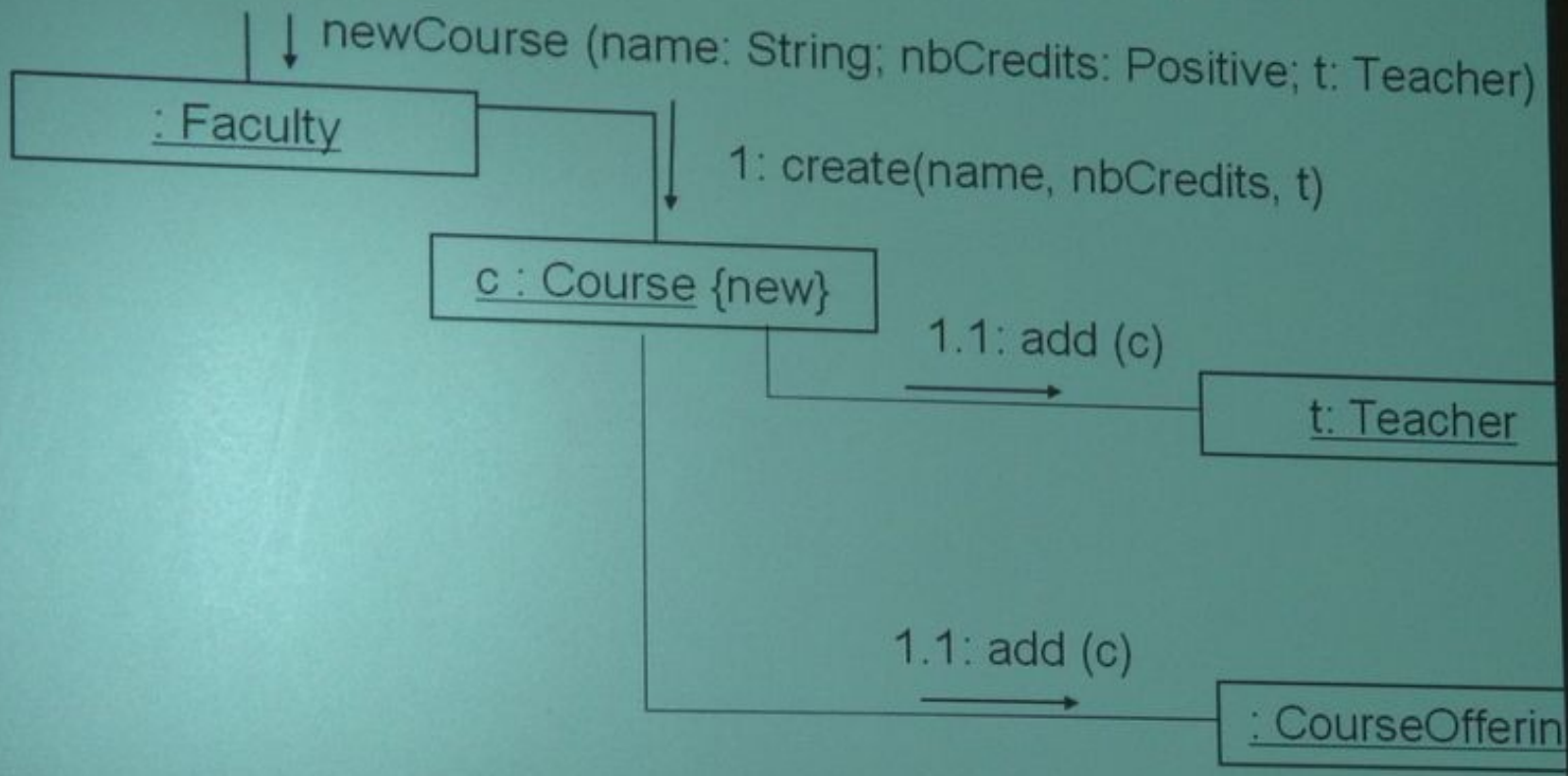


operation *Faculty::changeNbcredits* (courseName:String,*nbCredits: Integer)*

find the course c with the given name (1)

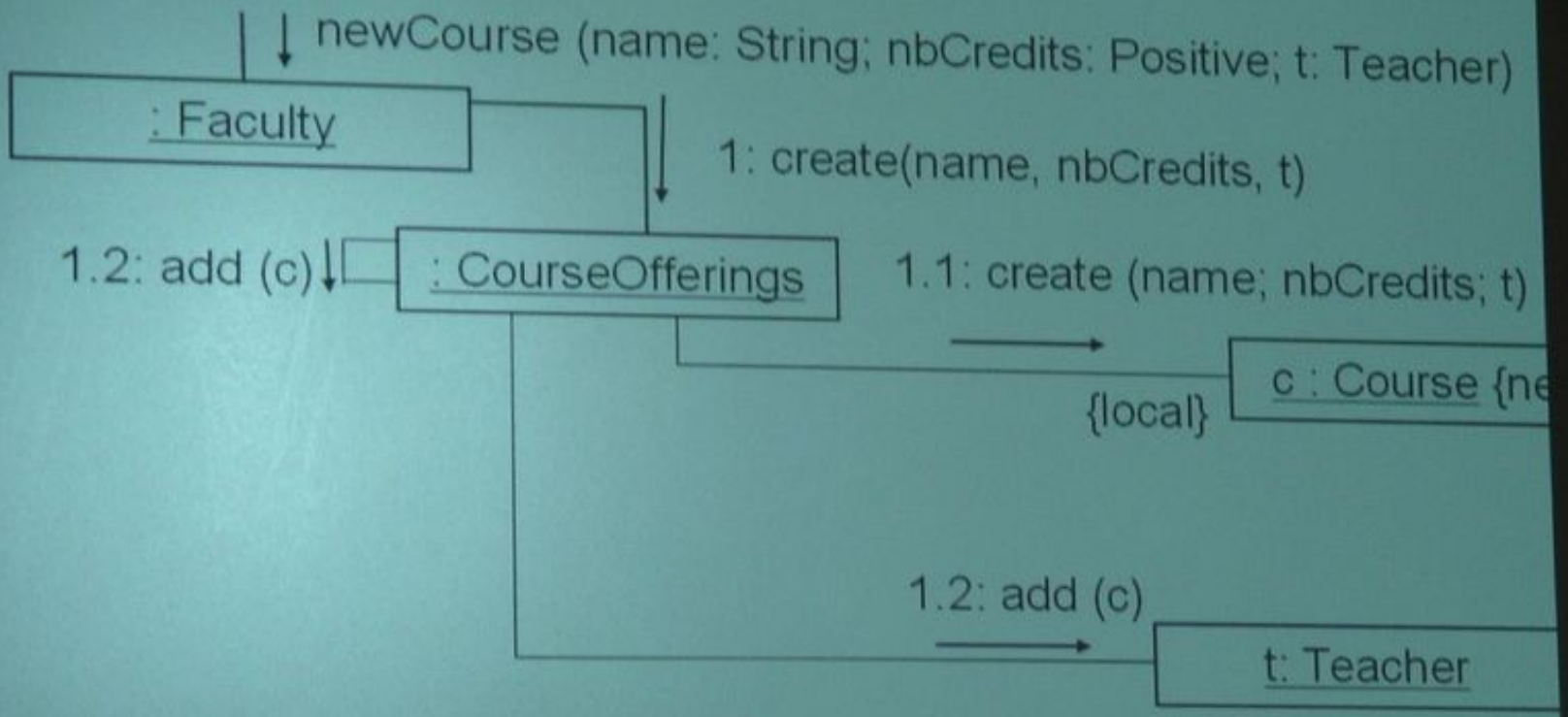
c.nbCredits=nbCredits (2)

School: newCourse, V3



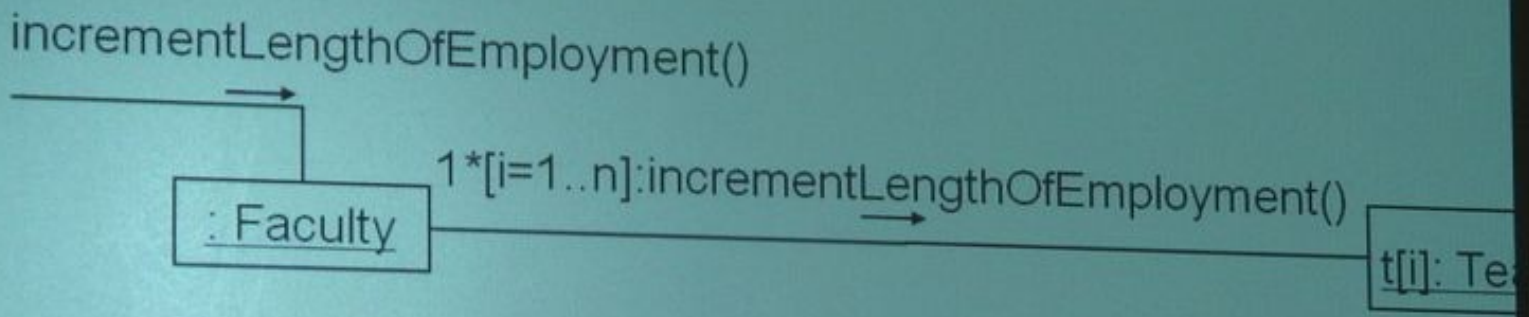
CourseOfferings knows about all courses given at the school.

School: newCourse, V4



Fulfills the same functional need as V3, but through a different design.

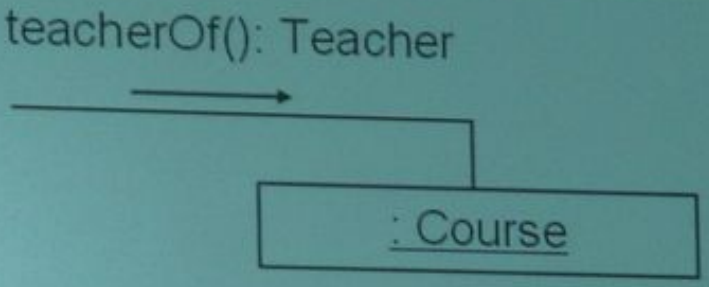
School: incrementLengthOfEmployment



Operation *Teacher::incrementLengthOfEmployment()*

this.lengthOfEmployment := this.lengthOfEmployment + 1

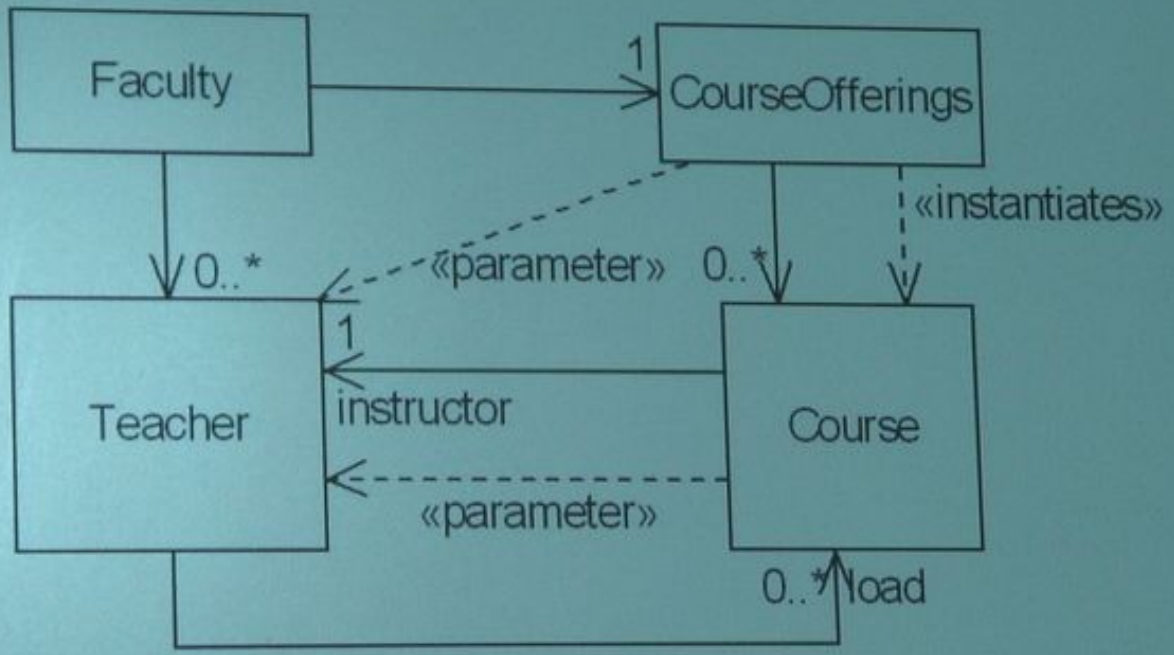
School: instructor of a course



operation *Course::teacherOf ():Teacher*

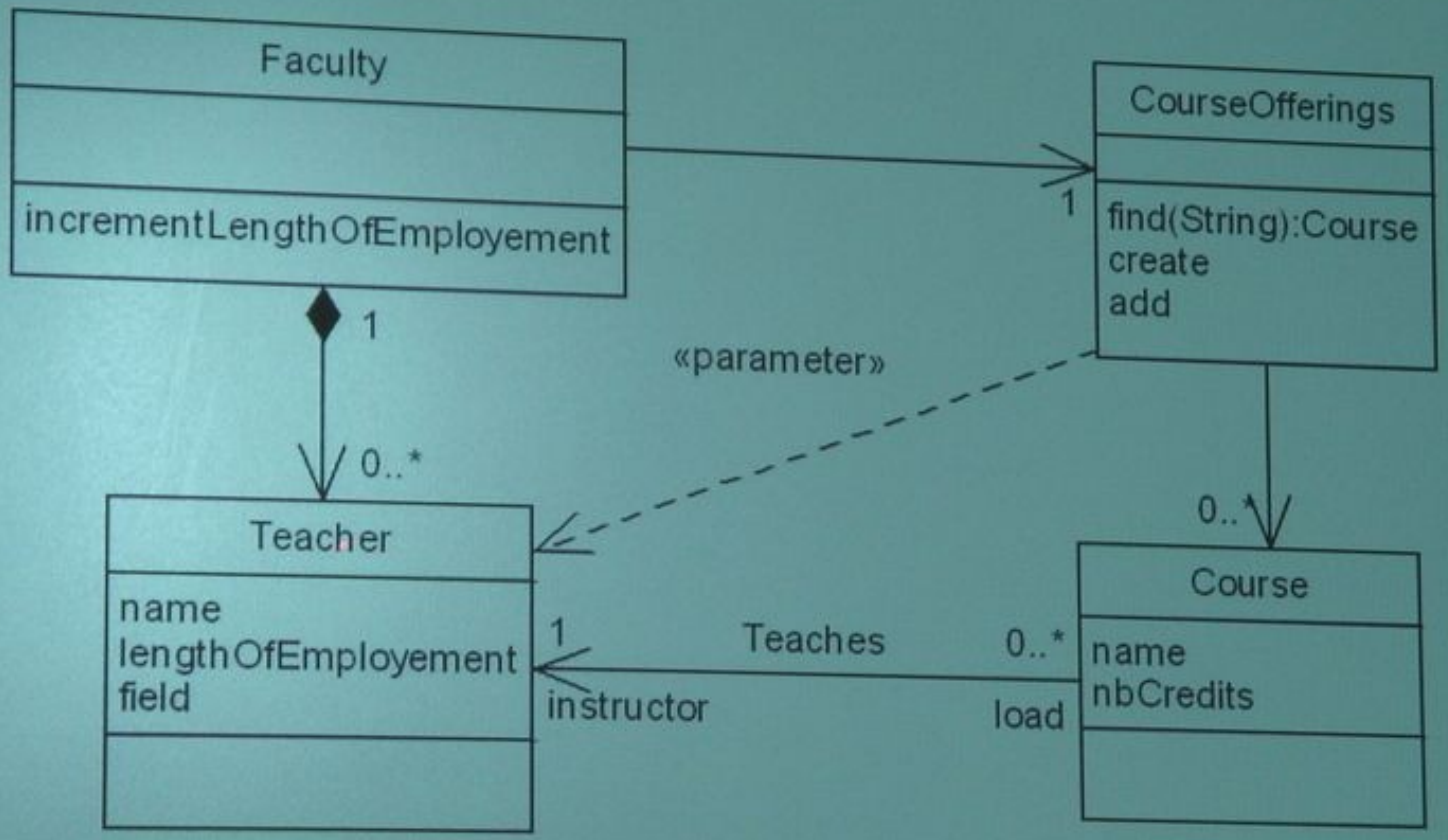
return this.teacher;

School: Dependency Model



The model can be simplified, the parameter dependency can be removed.

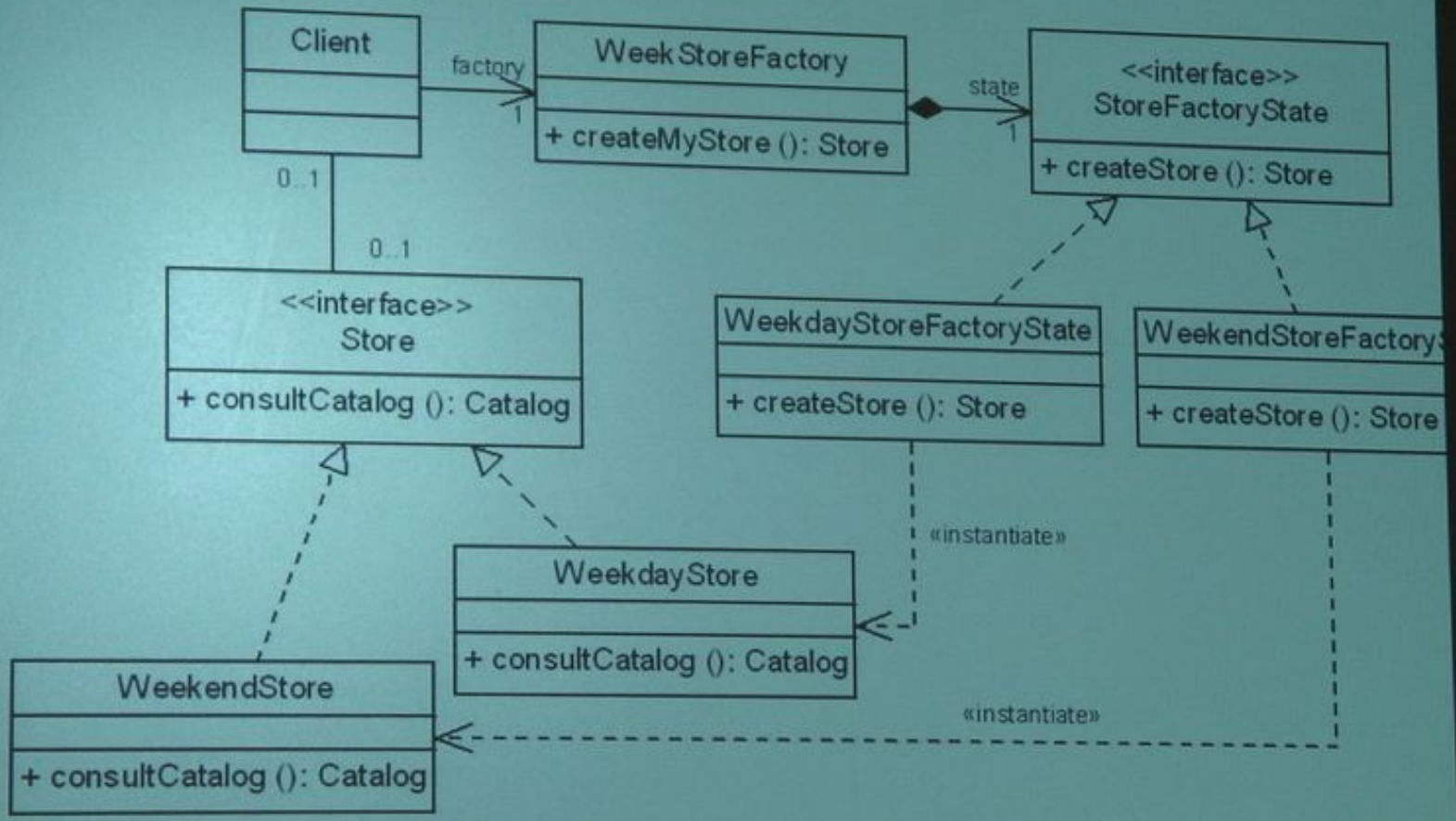
School: Design Class Model



Exercise

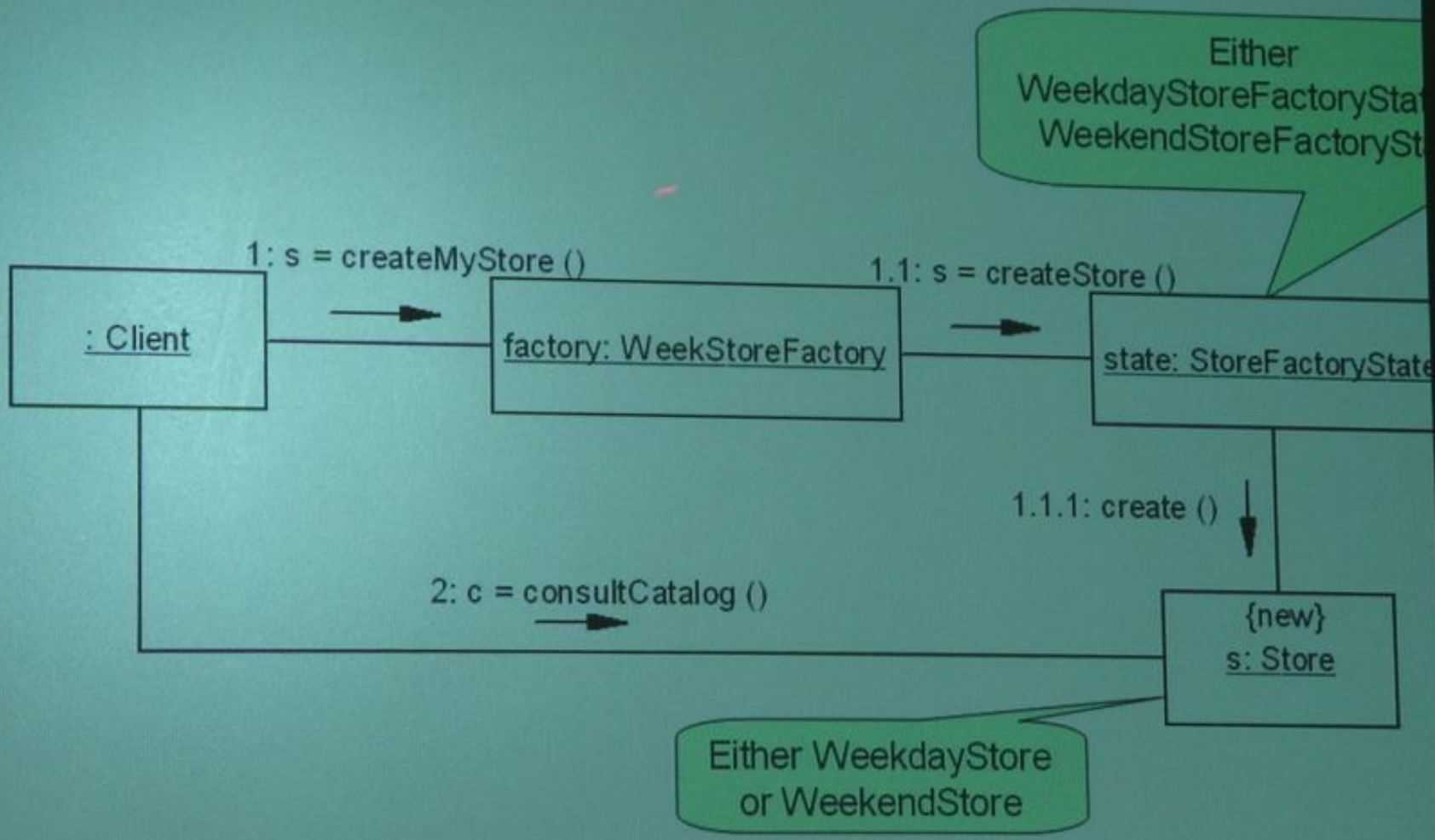
Store

Solution (applying State):



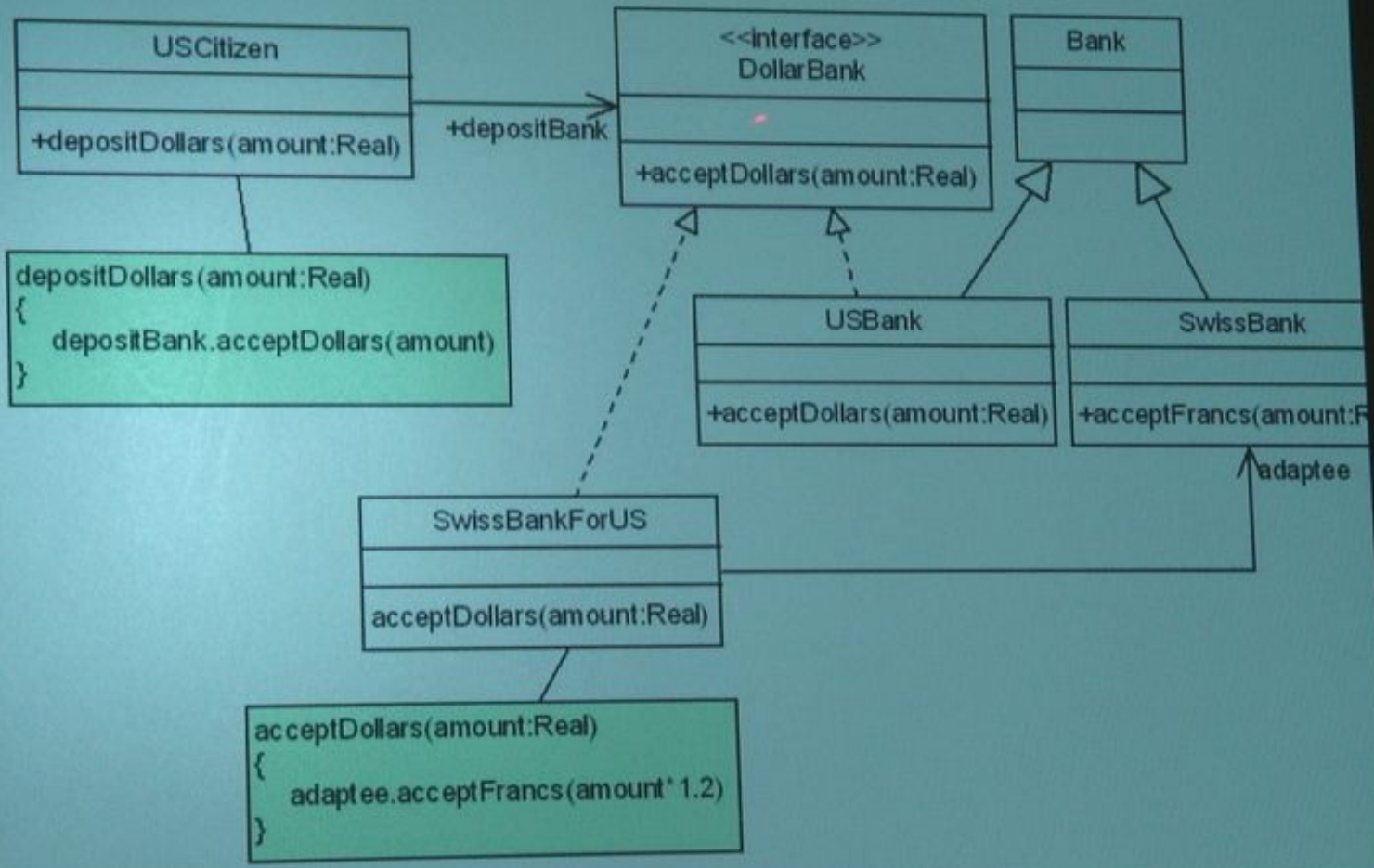
Store

Solution (applying State) (cont'd):

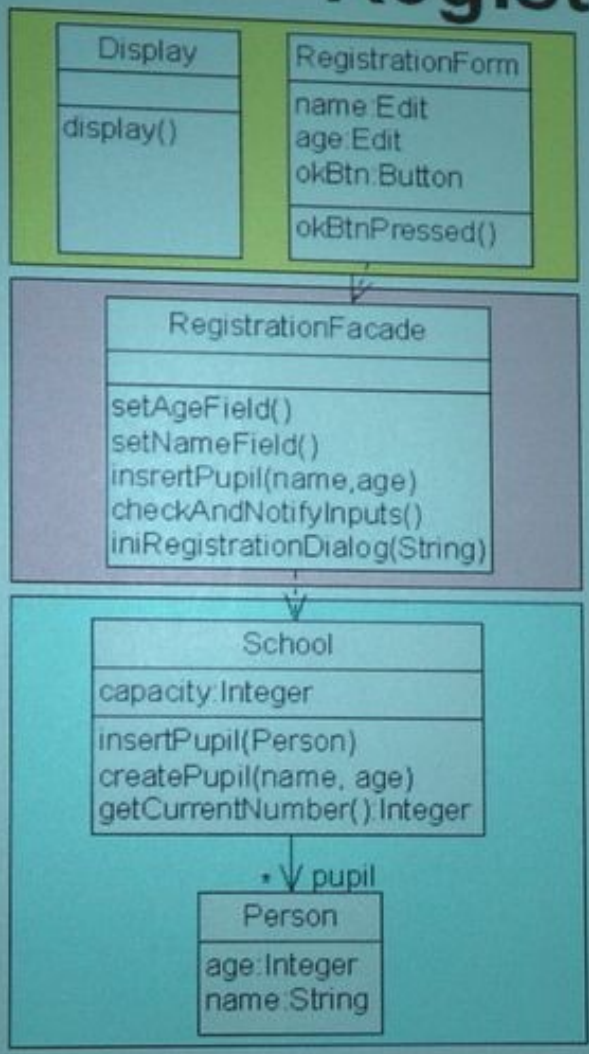


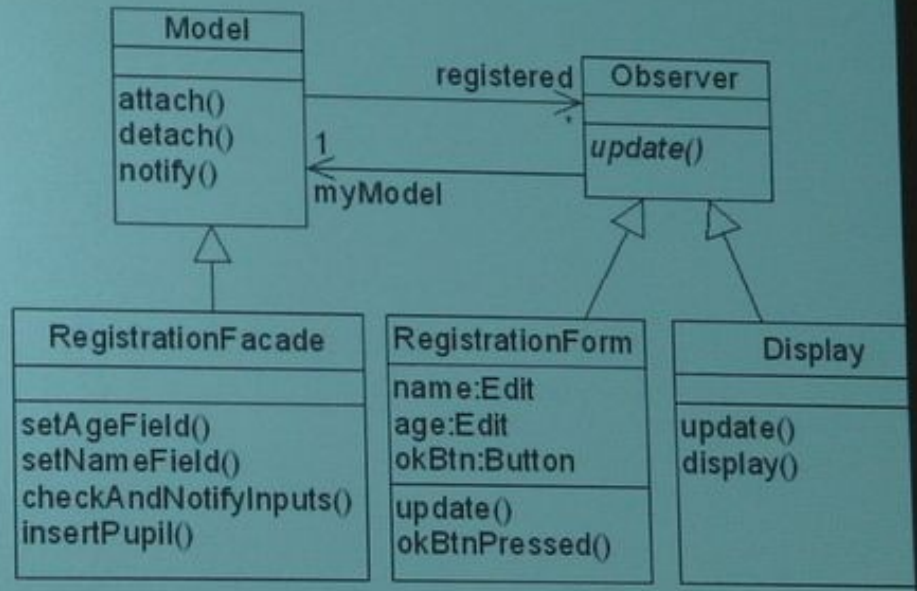
Exercise

Bank



Registration to School





Registration to School