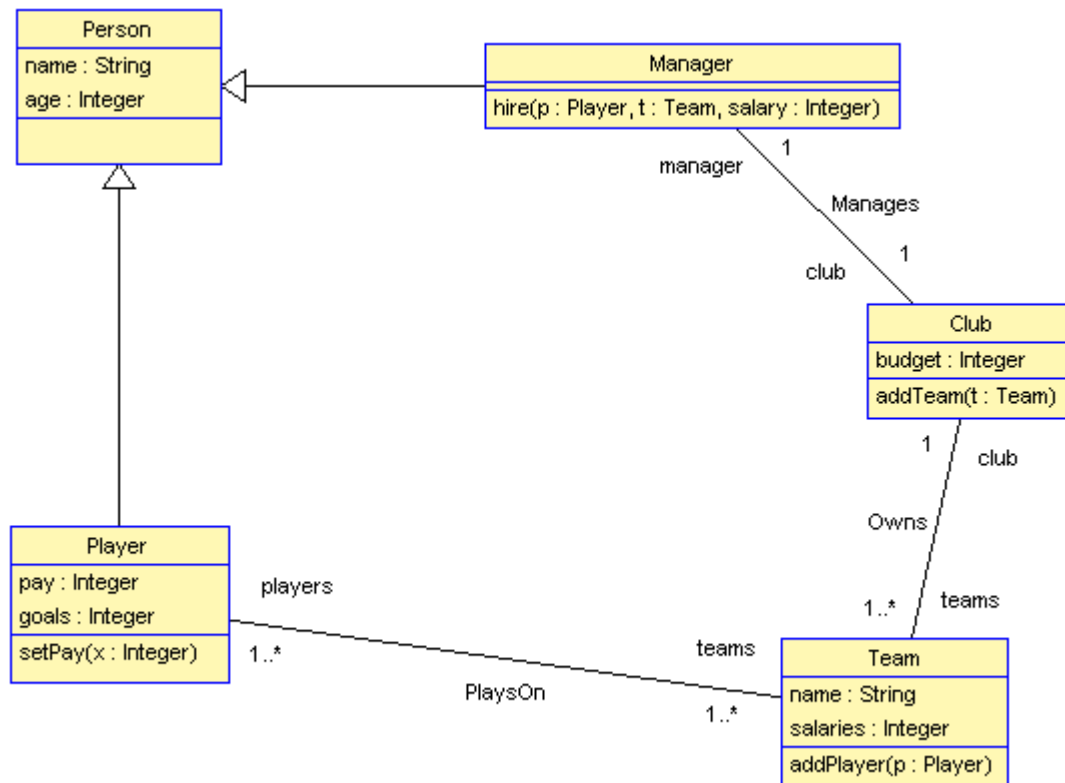


USE Lab Test for groups G1, G2, G3

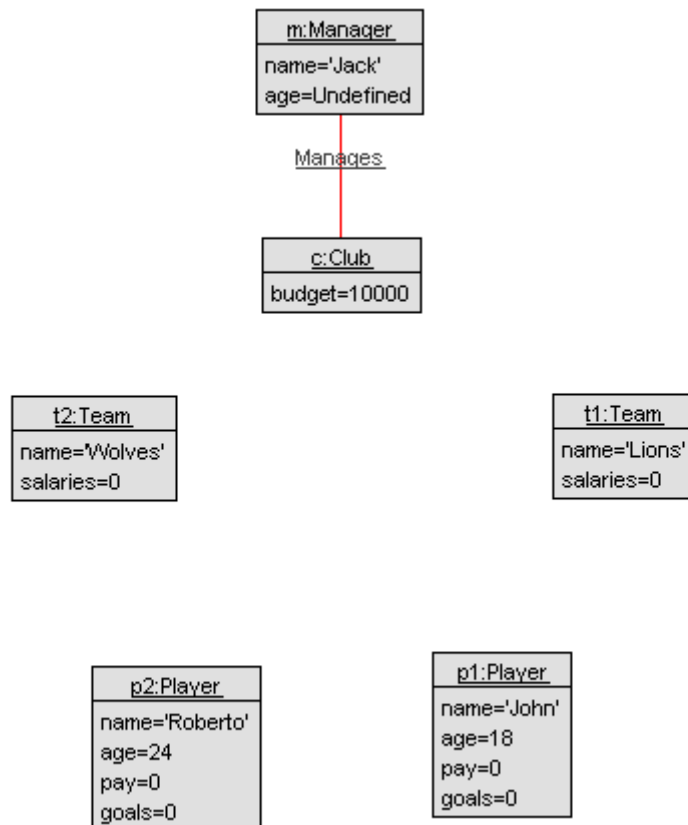
1. Using a text editor such as Notepad++, complete the partial USE model provided in [club.use](#), so that it yields the following class diagram in USE. Save you class layout too.



2. Once you get this class diagram, copy and paste the diagram into a Word document named [club.doc](#).
3. The default or initial pay for a player should be 0 and the salaries for a team should be initially 0. Add these initial values to the class definitions if you can. Otherwise, move on.

Next you are to write some operation contracts.

- First load [club.soil](#), open your object diagram, rearrange and save layout to get something like:



- Modify the manager object **m** so that it has your full name and age, copy and paste the new object diagram to the Word document [club.doc](#).

The following contracts can be put alongside the corresponding operation declaration or in the constraints section. Either way is fine.

Contract for addTeam(t : Team)

- Do **not** implement this operation in SOIL. Write an appropriate precondition and postcondition for this operation in [club.use](#) and save it.

Testing addTeam(t : Team) with !openter/!opexit

- Reload your model and use !openter/!opexit to test this contract with the objects **c** and **t1** provided in the soil file, [club.soil](#), which will create some test objects for you. Do 3 tests: get postcondition to fail, postcondition to succeed and finally the precondition to fail.
- When this is done use menu **File | Save protocol** to save your test interactions in [club.txt](#).

- Add OCL code to **club.use** to represent the following contracts:

Contract for addPlayer(p : Player)

Precondition

- player **p** should not already be a member of the team

Postconditions

- player p now a member of the team

Contract for hire($p : \text{Player}$, $t : \text{Team}$, $\text{salary} : \text{Integer}$)

Preconditions

- team t should be owned by club which is associated with the manager
- the existing salaries plus salary of new member should be less than the club budget
- if the player is younger than 21, then his maximum pay should be 500.

Postconditions

- team salaries after operation = salaries before + player's salary
- player's pay equals salary

SOIL Code

10. Implement setPay(), addPlayer() and hirePlayer() in SOIL code. One of these should call the other two. Re-save your model as [club.use](#).

Testing Contracts & SOIL code

11. Test the contracts and soil code from steps 9 and 10 using the objects provided in [club.soil](#). Save your testing protocol in [club1.txt](#). Get the 4 preconditions to fail once and everything to succeed twice. Do **not** use !openter!/opexit here.

Diagrams

12. Create a sequence diagram view in USE, copy and paste it to [club.doc](#). Also copy your object diagrams after the hiring to [club.doc](#).

13. Add this invariant to the class Club: any player on any team in club c should earn at most 6000. Save it in [club.use](#). You may need to rename [club.use](#) to [club.use.txt](#) before submitting on Webcourses.

Webcourses

Submit:

- club.use (or club.use.txt)
- club.txt
- club1.txt
- club.doc + 2 layout files.

Zippped or rar files will not be marked. **Do not compress, especially not RAR.**