UNIK-4710 FINAL PRESENTATION

José María Moreno Retamero

FIRST OF ALL

At the beginning I started to implement a web application with html,php and jsp but I realized that it was hard for me because all the programming languages that involves: Java,html,php,jsp,OWL API libraries…

So i decided to drop out that project, I can’t afford that because I’m not really a good programmer, and start a new one using just Java.

FIRST OF ALL

So I created a window based Java application which is based on an ontology created on Protégé, then we connect the ontology to my Java programming interface (Eclipse) to work directly with the ontology. We can see, edit and remove classes, individuals and properties.

BEFORE TO START

I missed some time on my previous web application, I struggled with Protégé 4.3, programming in Java and took me a long time the “how to” of the entire new project; but finally I got so much knowledge and now I really understand what the ontology's are made for and how to work with them.

Despite of this my application doesn’t have all the functionalities that I would like to implement but have the basic ones to understand and work with ontology's

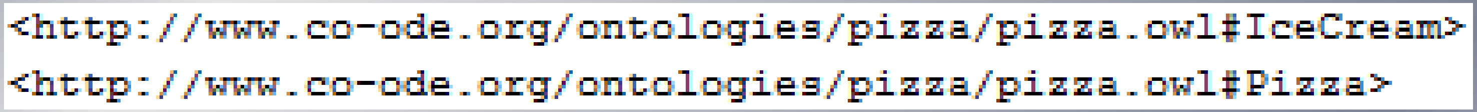
PROGRAMING WITH A ONTOLOGY

I would like to talk about the **vital** importance of the reasoner when you are programming with an ontology, because it takes me a bit time of realized of that fact.

THE REASONER

THE REASONER

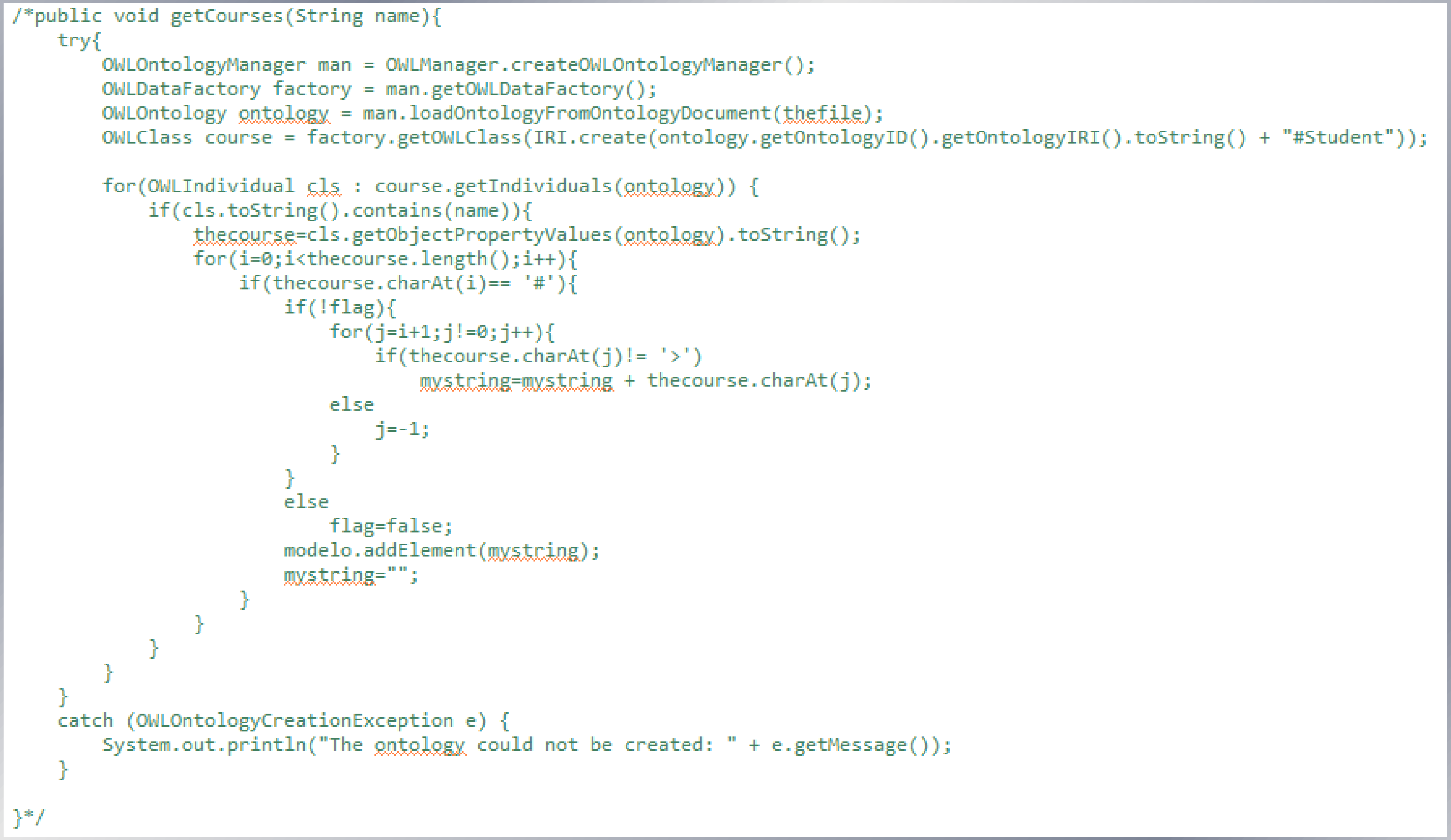
When I started to implement my application I use some basic examples from the OWL API help without using reasoner. At this point I started to view things coming out of my ontology and I get excited. Showing individuals or classes like that:



```
<http://www.co-ode.org/ontologies/pizza/pizza.owl#IceCream>
<http://www.co-ode.org/ontologies/pizza/pizza.owl#Pizza>
```

THE REASONER

What the hell is that long string(IRI)? I don’t need it, let’s start to cut it.



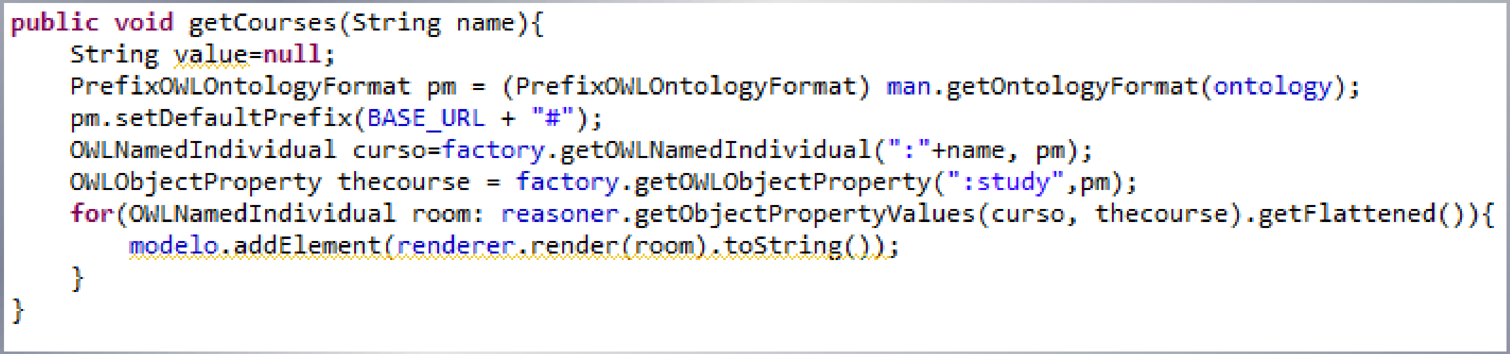
```
/*public void getCourses(String name){
    try{
        OWLOntologyManager man = OWLManager.createOWLOntologyManager();
        OWLDataFactory factory = man.getOWLDataFactory();
        OWLOntology ontology = man.loadOntologyFromOntologyDocument(thefile);
        OWLClass course = factory.getOWLClass(IRI.create(ontology.getOntologyID().getOntologyIRI().toString() + "#Student"));

        for(OWLIndividual cls : course.getIndividuals(ontology)) {
            if(cls.toString().contains(name)){
                thecourse=cls.getObjectPropertyValues(ontology).toString();
                for(i=0;i<thecourse.length();i++){
                    if(thecourse.charAt(i)== '#'){
                        if(!flag){
                            for(j=i+1;j!=0;j++){
                                if(thecourse.charAt(j)!= '>')
                                    mystring=mystring + thecourse.charAt(j);
                            else
                                j=-1;
                            }
                        }
                        else
                            flag=false;
                        modelo.addElement(mystring);
                        mystring="";
                    }
                }
            }
        }
    }
    catch (OWLOntologyCreationException e) {
        System.out.println("The ontology could not be created: " + e.getMessage());
    }

}*/
```

THE REASONER

As my project progressed my code was longer and complicated and with more problems; I went crazy and I’m started to think that was not the right way to connect with the ontology. We can say at this point that I began to “reason”

THE REASONER

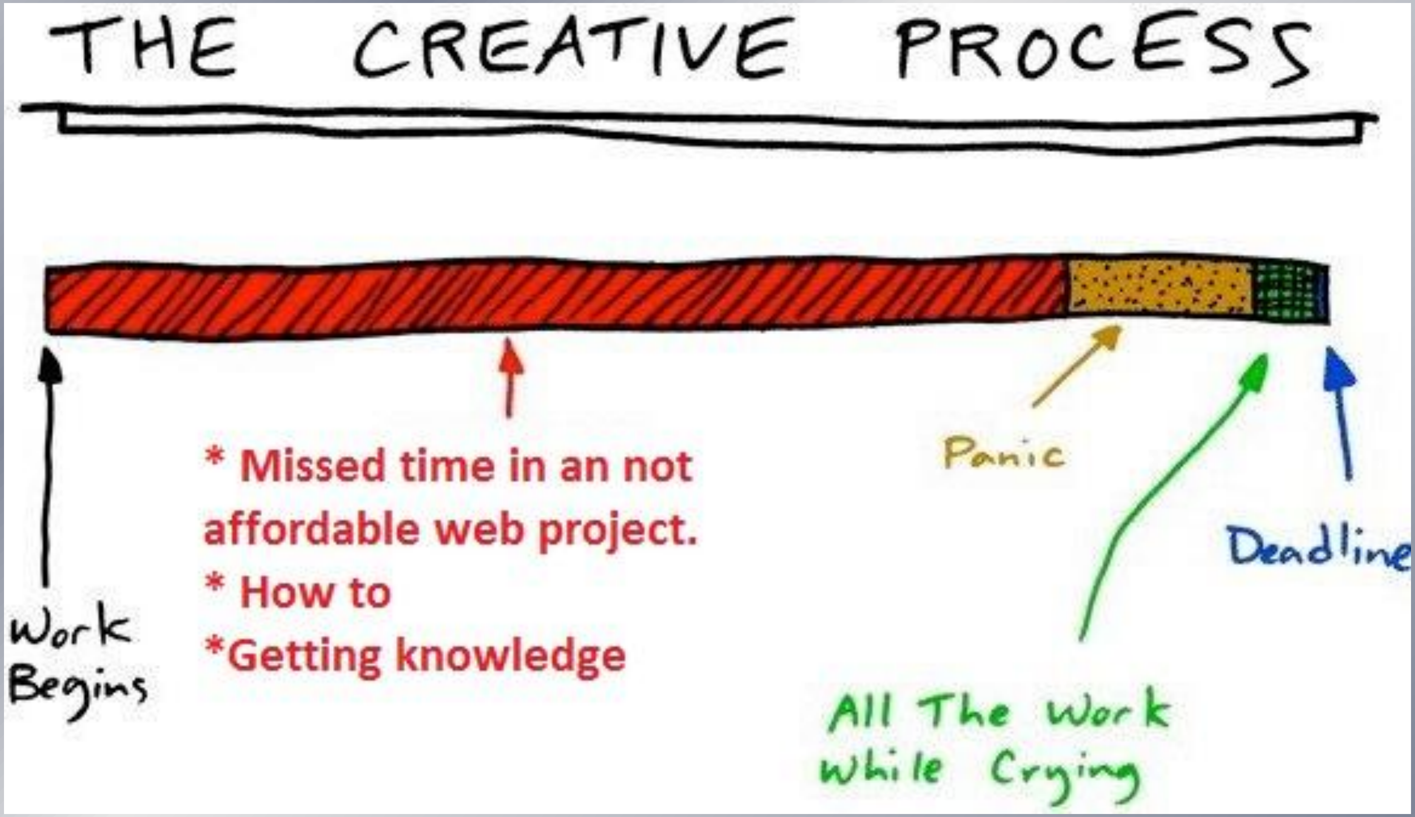
Using the reasoner we can see how much simplifies the code for the same function



```
public void getCourses(String name){
    String value=null;
    PrefixOWLOntologyFormat pm = (PrefixOWLOntologyFormat) man.getOntologyFormat(ontology);
    pm.setDefaultPrefix(BASE_URL + "#");
    OWLNamedIndividual curso=factory.getOWLNamedIndividual(":"+name, pm);
    OWLObjectProperty thecourse = factory.getOWLObjectProperty(":study",pm);
    for(OWLNamedIndividual room: reasoner.getObjectPropertyValues(curso, thecourse).getFlattened()){
        modelo.addElement(renderer.render(room).toString());
    }
}
```

And it gets just the name!!The long string is gone!!

STORY OF MY PROJECT IN A PICTURE


THE CREATIVE PROCESS
Work Begins
* Missed time in an not affordable web project.
* How to
*Getting knowledge
Panic
All The Work while Crying
Deadline

BEFORE TO START

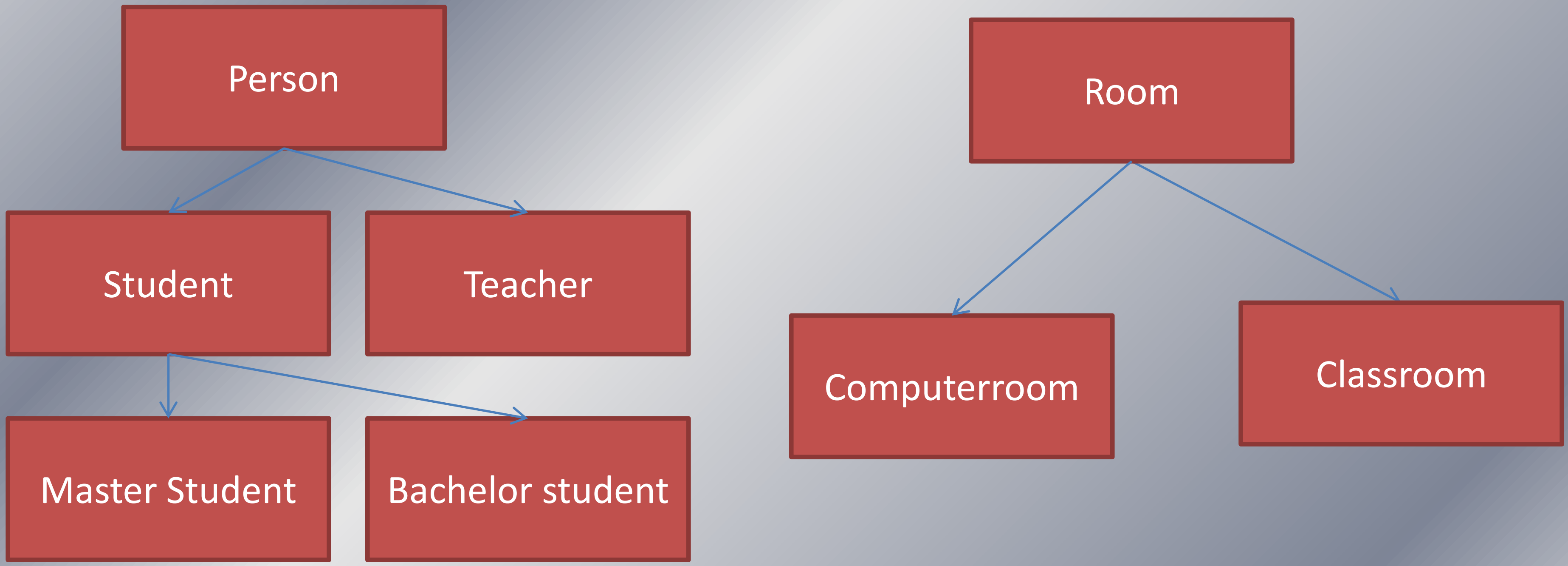
I would like to thank my classmates:

Martin who teach me some tips on Protégé and how to start with my application.

Javier who told me how to use the OWL API libraries properly

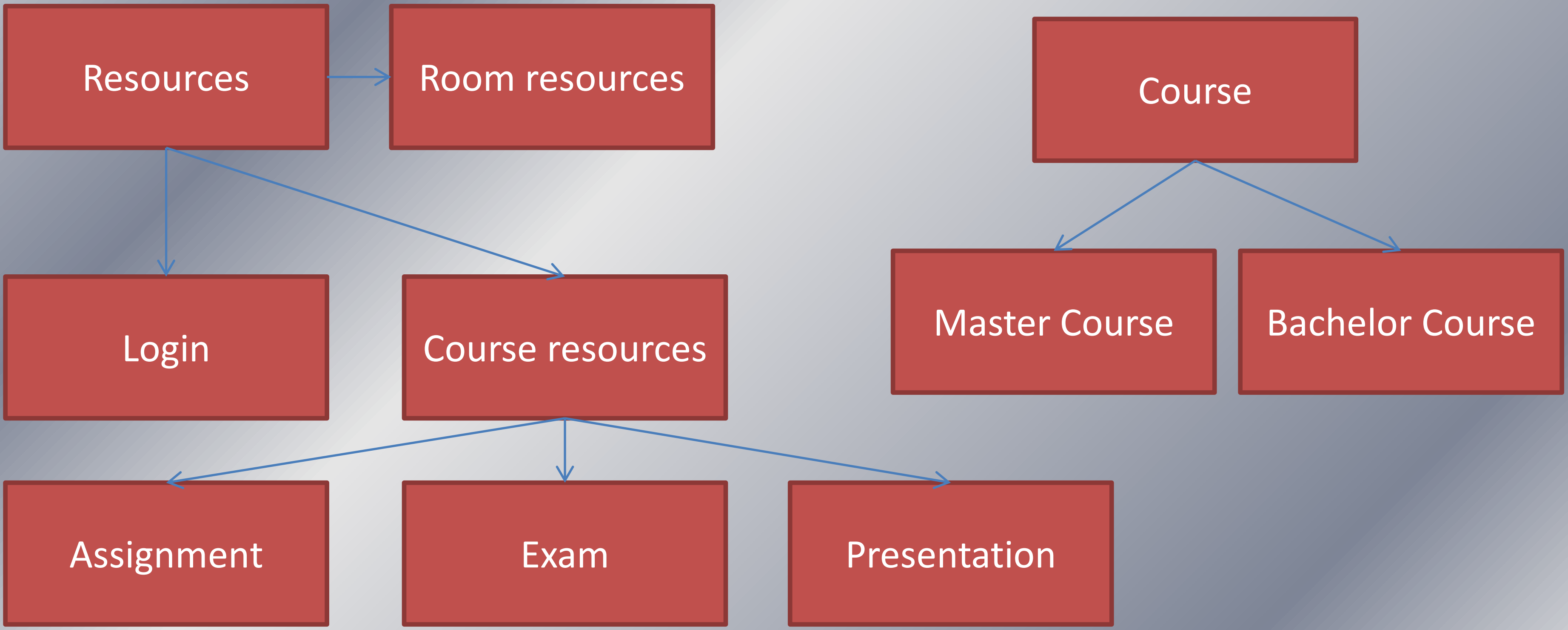
CLASSES

The ontology is about an university and the classes and subclasses seem like that



CLASSES

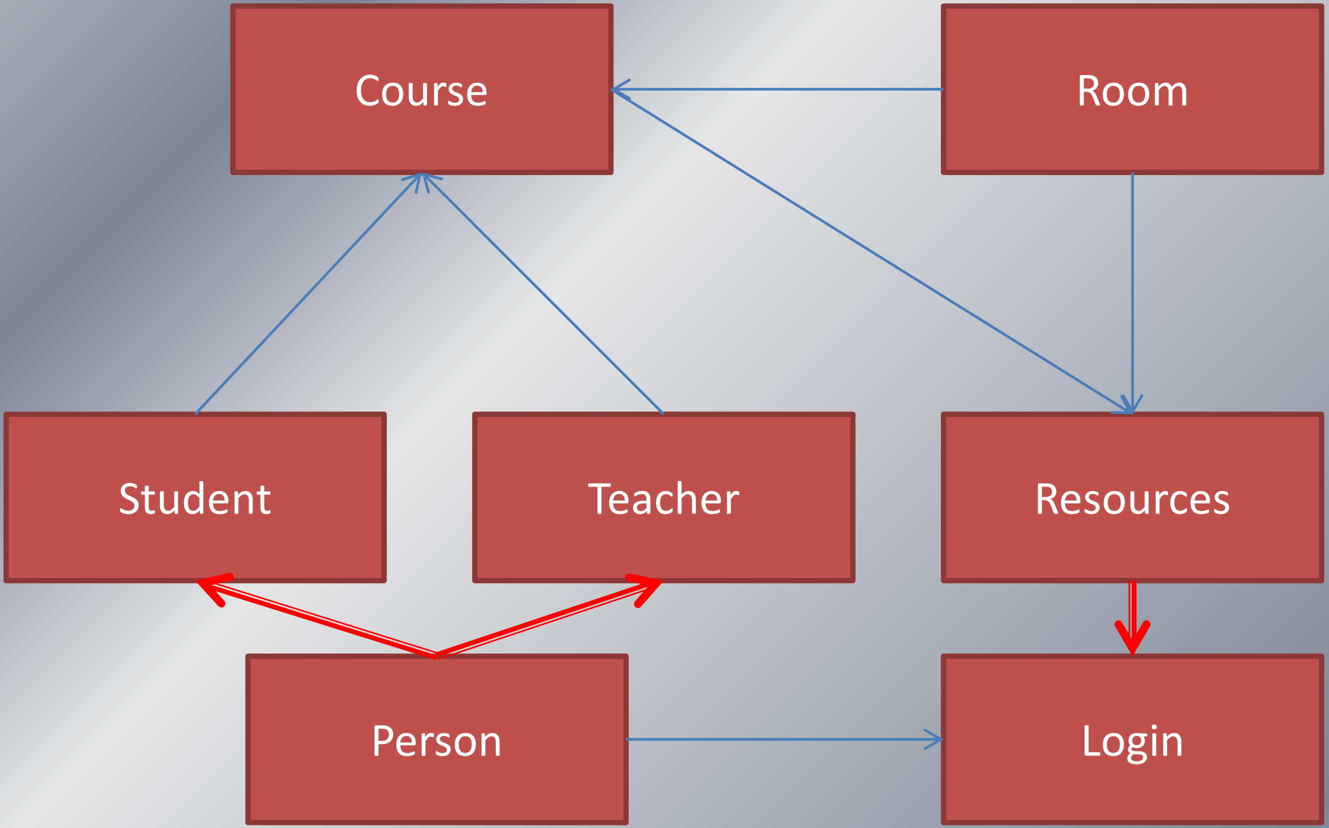
The ontology is about an university and the classes and subclasses seem like that



CONNECTION OF THE CLASSES

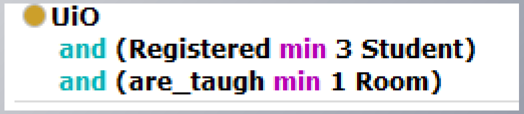
Red arrows are subclasses

Blue arrows are connections between classes



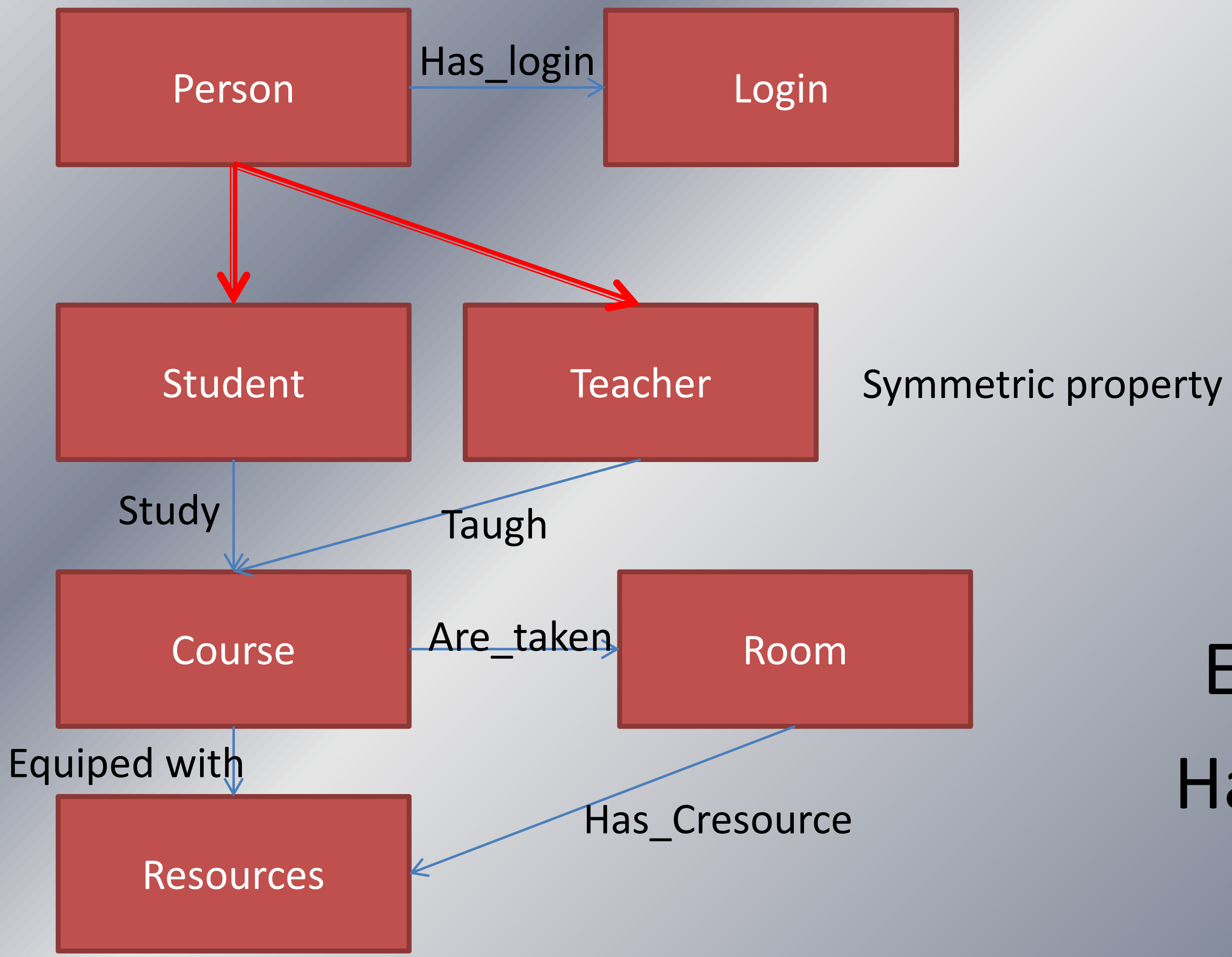
RESTRICTIONS

- On the courses are min 3 Students
- Courses are taugh in min 1 Room
- Person study/teach min 1 Course
- Person have just 1 Login
- Course could have min 1 C_Resources
- Room could have min 1 R_Resources



```
● UiO
   and (Registered min 3 Student)
   and (are_taugh min 1 Room)
```

GOING INSIDE OF ONTOLOGY



Properties:

Has_login

Study

Taugh

Are_taken

Equiped with

Has_Cresource

WHAT THE APPLICATION CAN AND CANNOT DO

IMPLEMENTED

•Students and teachers login/logout

•A student can see their registered courses, what teacher teaching the course, in which room will taught and room resources

•A teacher can see their registered courses, students registered in that course, in which room will taught and room resources

•Admin can create and delete students and teachers.

NOT IMPLEMENTED

•Alerts

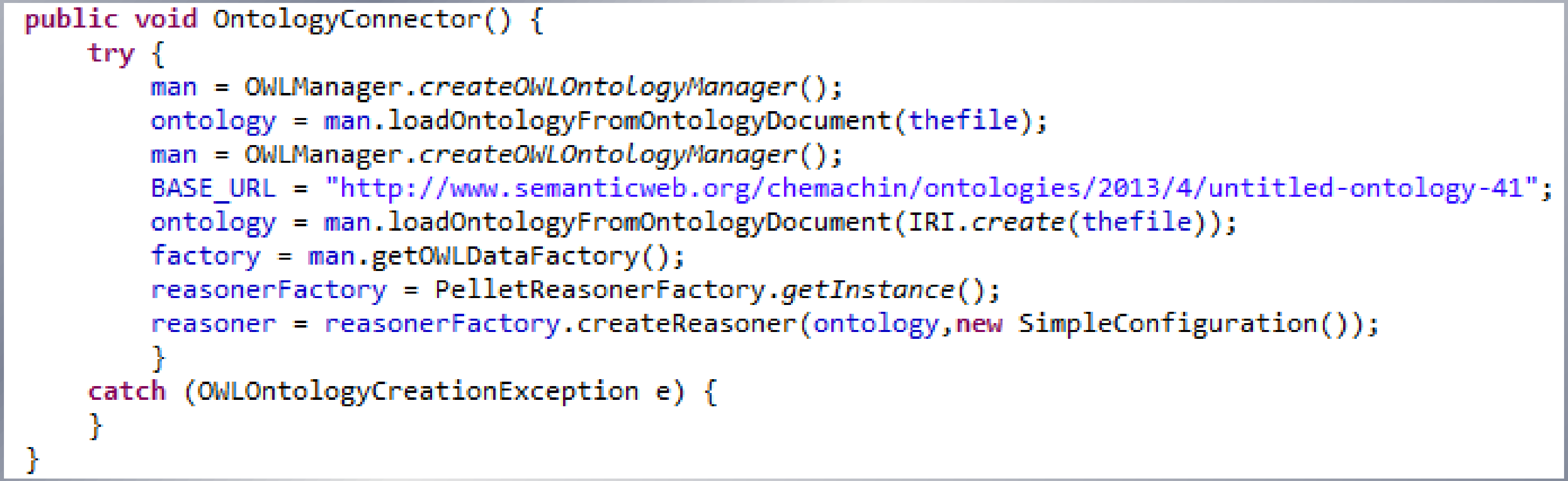
•Create rooms, resources and courses.

GOING INSIDE THE APPLICATION

Now, we are going to see what functionalities the application implement and what not, through windows captures.

CONNECTION WITH THE ONTOLOGY

For connect with the ontology I created the function OntologyConnector:

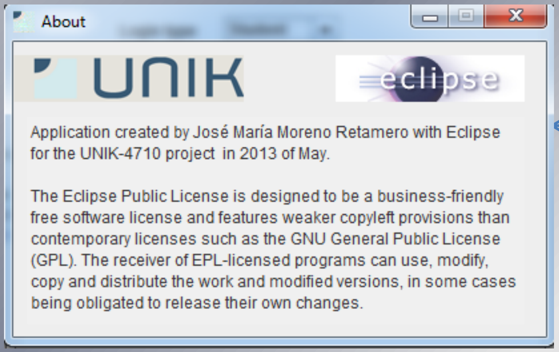


```
public void OntologyConnector() {
    try {
        man = OWLManager.createOWLOntologyManager();
        ontology = man.loadOntologyFromOntologyDocument(thefile);
        man = OWLManager.createOWLOntologyManager();
        BASE_URL = "http://www.semanticweb.org/chemachin/ontologies/2013/4/untitled-ontology-41";
        ontology = man.loadOntologyFromOntologyDocument(IRI.create(thefile));
        factory = man.getOWLDataFactory();
        reasonerFactory = PelletReasonerFactory.getInstance();
        reasoner = reasonerFactory.createReasoner(ontology,new SimpleConfiguration());
        }
    catch (OWLOntologyCreationException e) {
    }
}
```

Don’t forget to include the OWL API libraries, Pellet libraries, reasoner and of course, copy your ontology to the root folder of your application

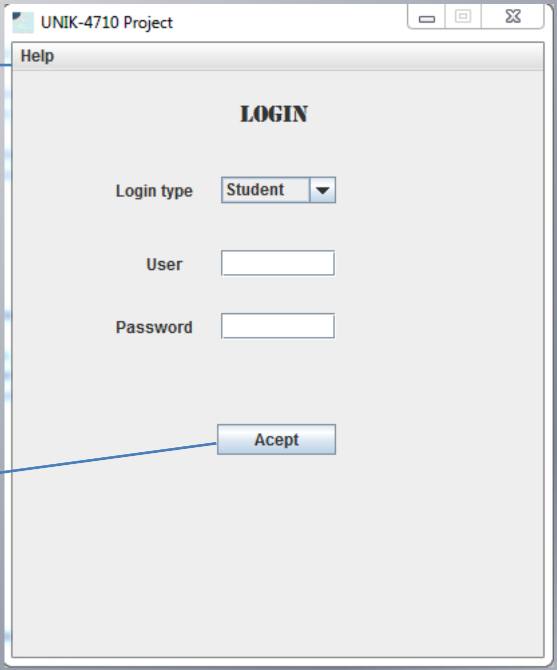
LOGIN

The application implements Login of students, teachers and one administrator and check that the user exists and if the password is correct.


About

Application created by José María Moreno Retamero with Eclipse for the UNIK-4710 project in 2013 of May.

The Eclipse Public License is designed to be a business-friendly free software license and features weaker copyleft provisions than contemporary licenses such as the GNU General Public License (GPL). The receiver of EPL-licensed programs can use, modify, copy and distribute the work and modified versions, in some cases being obligated to release their own changes.


UNIK-4710 Project

Help

LOGIN

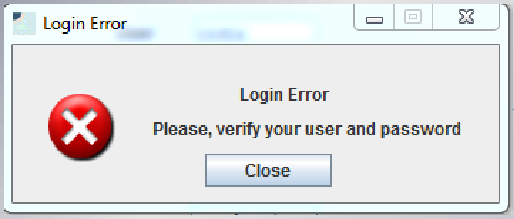
Login type: Student

User

Password

Acept

Login failed

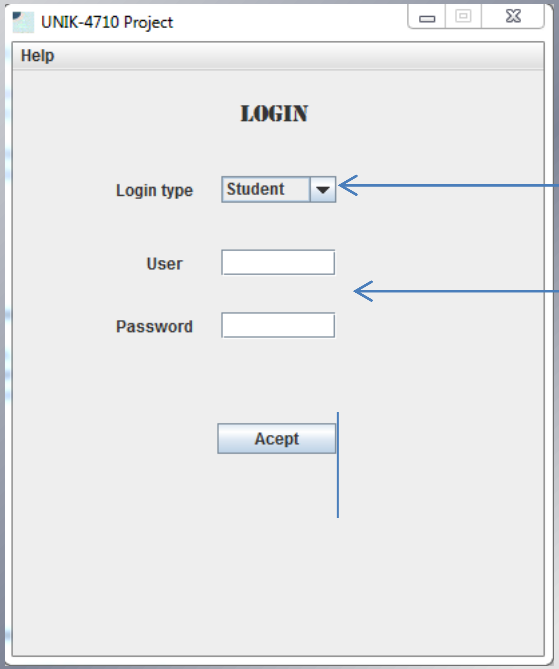

Login Error

Login Error

Please, verify your user and password

Close

LOGIN



Subclasses of class person

Comparing individuals of selected person class
And individuals of class Login

Login function

```
private boolean Login(String type){
    boolean flag=false;
    PrefixOWLOntologyFormat pm = (PrefixOWLOntologyFormat) man.getOntologyFormat(ontology);
    pm.setDefaultPrefix(BASE_URL + "#");
    OWLClass genderClass = factory.getOWLClass(":"+type, pm);
    for (OWLNamedIndividual userlog : reasoner.getInstances(genderClass,false).getFlattened()) {
        if(renderer.render(userlog).toString().equals(user.getText())){
            if(PasswordCheck(user.getText())){
                flag=true;
                break;
            }
        }
    }
    return flag;
}
```

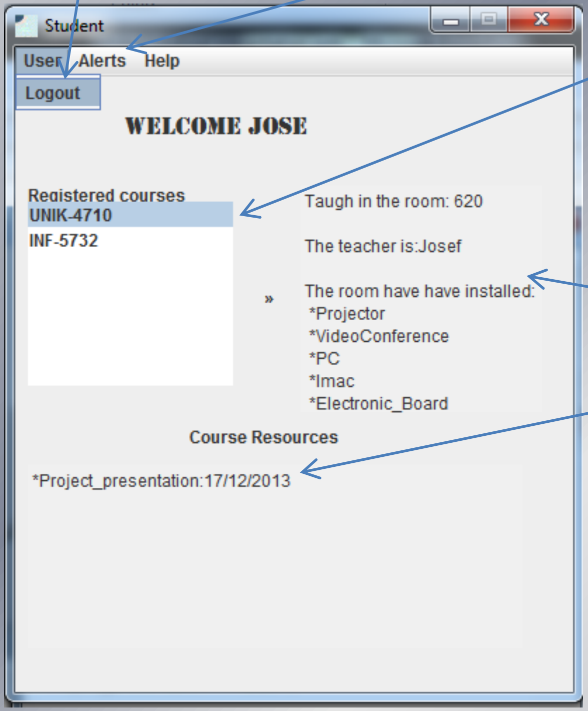
STUDENT

Logout

Alerts are not implemented

Clicking on the registered courses of the student will show information about the course

The information showed



WELCOME JOSE

Registered courses
UNIK-4710
INF-5732

Taugh in the room: 620

The teacher is:Josef

The room have have installed:
*Projector
*VideoConference
*PC
*Imac
*Electronic_Board

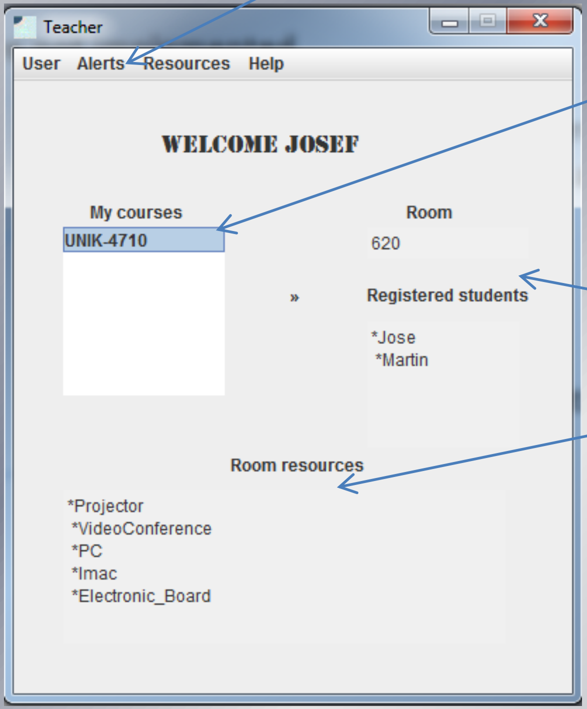
Course Resources

*Project_presentation:17/12/2013

TEACHER

Alerts and resources are not implemented

Clicking on the registered courses of the teacher will show information about the course



UNIK-4710

620

*Jose
*Martin

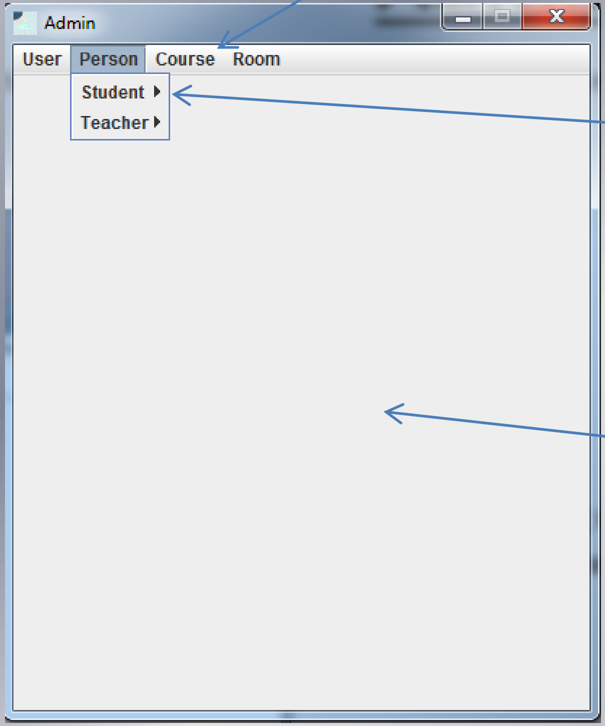
*Projector
*VideoConference
*PC
*Imac
*Electronic_Board

The information showed

ADMIN

Admin user can login as a teacher and have his own user and password

Create or delete courses and rooms are not available

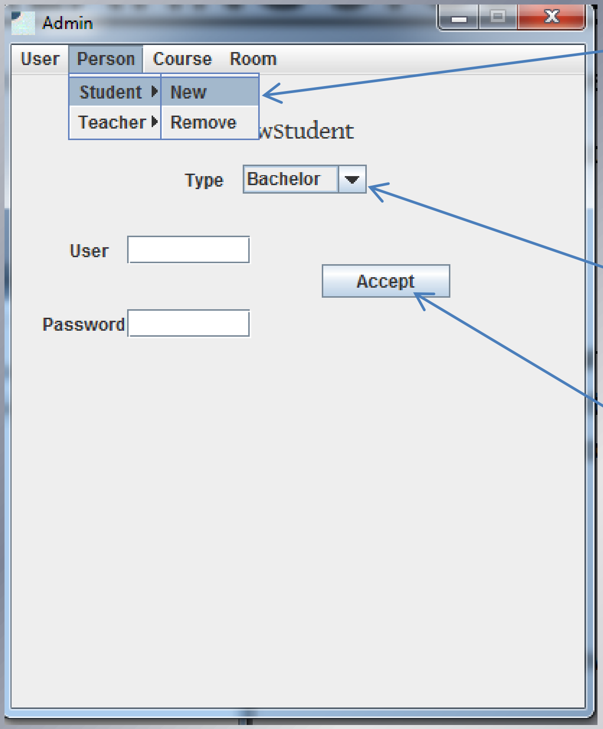


We can create and remove students and teachers from the ontology. The visual interface is similar for both two.

Main window of user admin appears in blank

CREATING STUDENT/TEACHER

Admin user can login as a teacher and have his own user and password



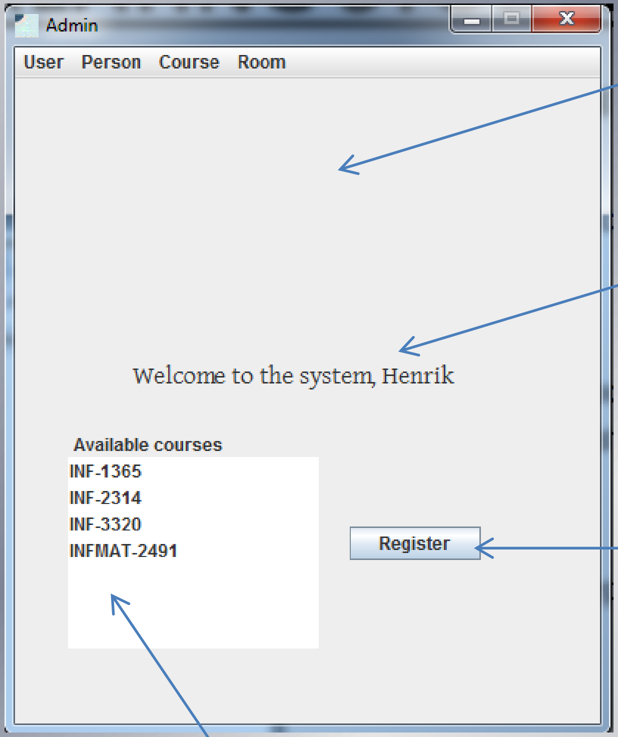
Menu for choose our option

Combo box for choose type of student (this is deactivated
When you are registering a new teacher)

Clicking accept will create a new individual

CREATING STUDENT/TEACHER

After create a user we need to assign him/her the courses



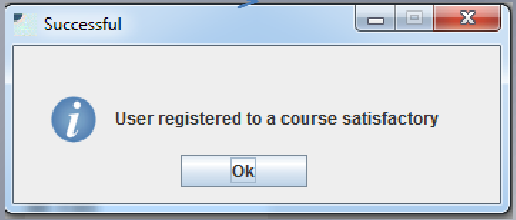
Welcome to the system, Henrik

The other controls disappear

Welcome message

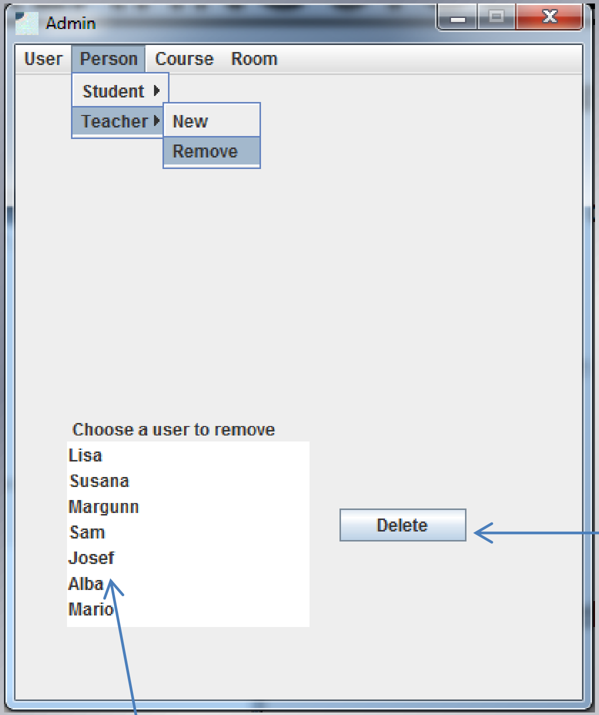
Register to the selected course

Available courses



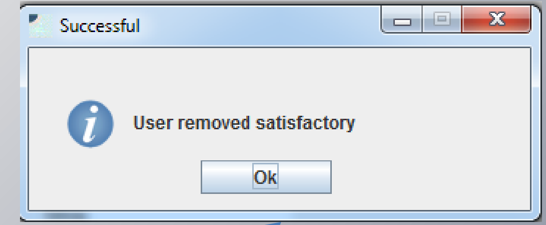
User registered to a course satisfactory

REMOVING STUDENT/TEACHER



Choose a user to remove

Lisa
Susana
Margunn
Sam
Josef
Alba
Mario



User removed satisfactory

Delete the selected teacher

Existing teachers
Note that the admin user doesn't appear

THE END

Thanks for your attention