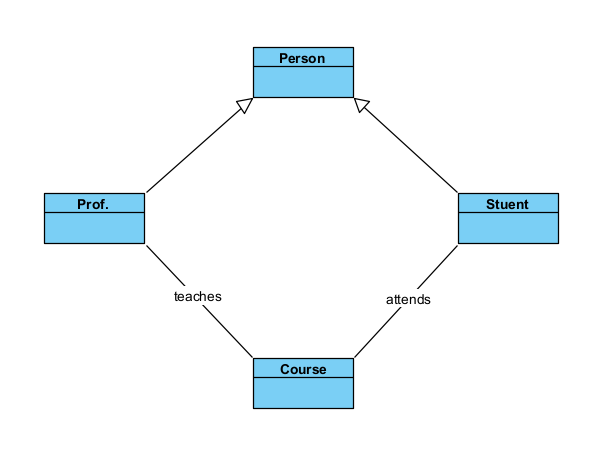
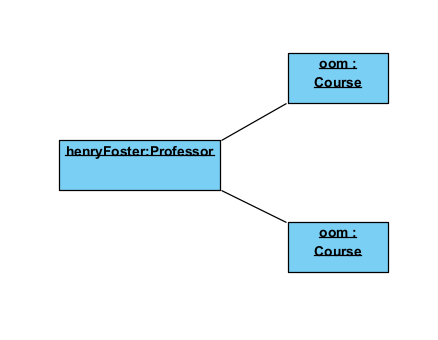
Lab 1 – IS - Breaz Bogdan

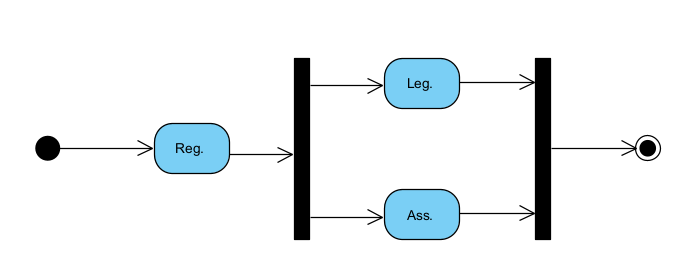
**4)** **Class Diagram**



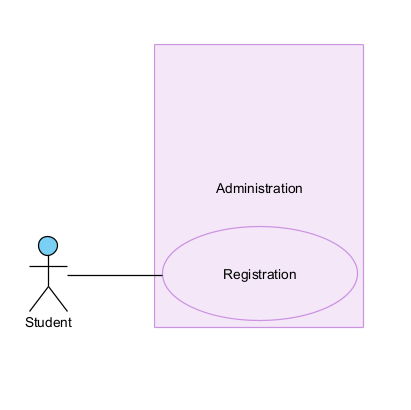
**Object Diagram**



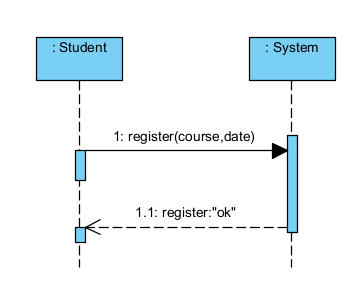
**Activity Diagram**

****

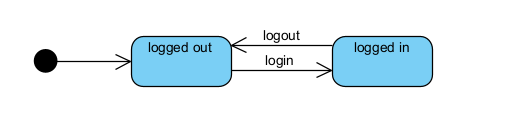
**Use case Diagram**

****

**Sequence Diagram**

****

**State Machine Diagram**

****

**1)**

UML (Unified Modeling Language) este un limbaj de modelare folosit pentru a reprezenta sisteme software, ajută la vizualizarea, specificarea, construirea și documentarea componentelor unui sistem, este utilizat în analiza și proiectarea software, oferă un limbaj comun pentru dezvoltatori, arhitecți și alte părți implicate, ofera documentarea sistemului pentru mentenanță și dezvoltare ulterioară, permite generarea automată de cod în unele cazuri.

Principalele tipuri de diagrame UML sunt:

* Diagrama de cazuri de utilizare – arată interacțiunea dintre utilizatori și sistem,
* Diagrama de clase – descrie structura statică a sistemului, relațiile dintre clase și atributele acestora,
* Diagrama de secvență – ilustrează ordinea în care obiectele comunică între ele,
* Diagrama de activitate – reprezintă fluxul de lucru și procesele dintr-un sistem,
* Diagrama de stare – modelează stările prin care trece un obiect pe parcursul ciclului său de viață.

**2)**

Modelele sunt reprezentări abstracte ale unui sistem real, utilizate pentru a înțelege, analiza și comunica structura și comportamentul acestuia. Ele ajută la reducerea complexității prin evidențierea caracteristicilor esențiale și ignorarea detaliilor nesemnificative.

Utilitatea modelelor:

Clarificarea cerințelor – ajută la definirea și înțelegerea cerințelor sistemului înainte de implementare,

Comunicare eficientă – oferă un limbaj comun între părțile implicate (dezvoltatori, designeri, utilizatori, manageri),

Identificarea și rezolvarea problemelor – permite detectarea și corectarea problemelor înainte de implementare,

Documentație și mentenanță – facilitează actualizările și modificările ulterioare ale sistemului,

Simulare și testare – permite verificarea comportamentului sistemului înainte de implementare.

**5)**

Diagrame de structură (aspecte statice ale sistemului):

Clase, Obiecte, Pachete, Componente, Structură compozită, Implementare, Profil.

Diagrame de comportament (aspecte dinamice ale sistemului):

Cazuri de utilizare, Stare, Activitate, Secvență, Comunicare, Timp, Prezentare interacțiuni.

Asemănări și deosebiri

Asemănări: Toate modelele UML descriu sistemul software, clarifică cerințele și îmbunătățesc comunicarea.

Deosebiri:

Structura - evidențiază relațiile statice.

Comportamentul - descrie interacțiunile și procesele.

Interacțiunea - se axează pe schimbul de mesaje.

Implementarea - arată distribuția pe hardware.

**6)**

**Diagrama de clase**

Modelează structura statică a unui sistem, incluzând clase, atribute, metode și relații.

Utilizează conceptele de generalizare, asociere și agregare pentru a descrie legăturile dintre clase​.

**Diagrama de obiecte**

Reprezintă o instanță concretă a unui sistem la un anumit moment de execuție.

Se bazează pe diagrama de clase, dar arată obiectele reale și valorile lor​.

**Diagrama de pachete**

Grupa elemente UML (clase, diagrame, componente) pe baza unor caracteristici comune.

Utilă pentru organizarea logică a unui sistem mare​.

**Diagrama de componente**

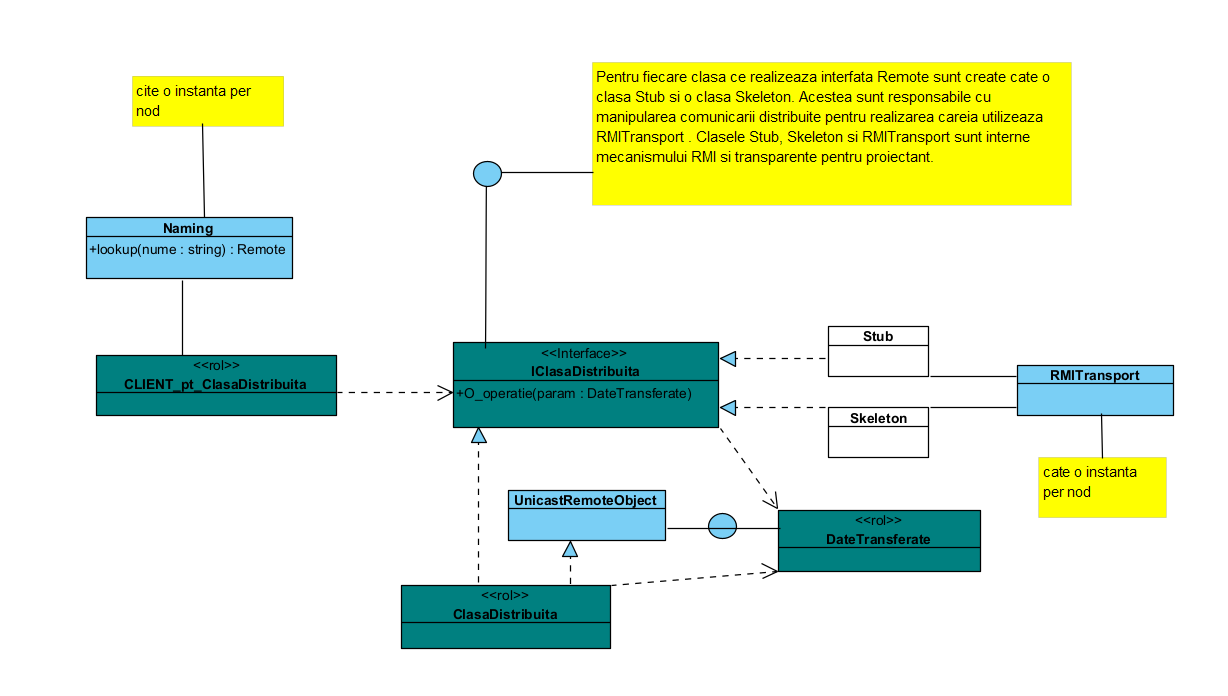
Modelează componentele software **independente** și relațiile dintre ele.

Permite vizualizarea arhitecturii sistemului, incluzând interfețele acestora​.

**Diagrama de implementare**

Reprezintă distribuția componentelor software pe echipamente hardware.

Include noduri de execuție și conexiuni de rețea pentru modelarea arhitecturii fizice a sistemului​.

**7)** ****