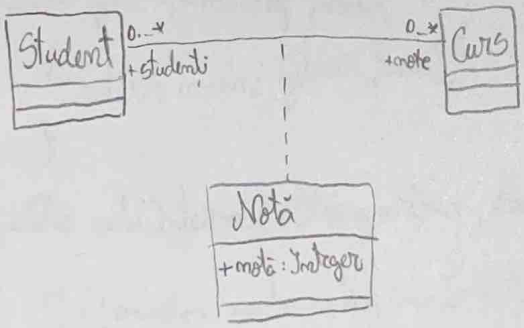
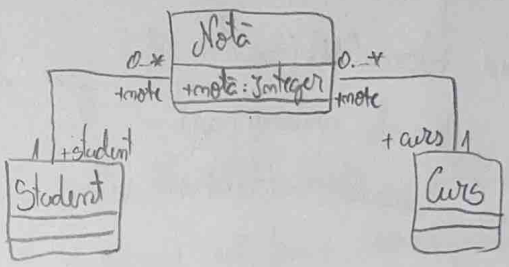
BILET 6

A. 1. Enumerați activitățile tehnice ale dezv. de soft orientate obiect.

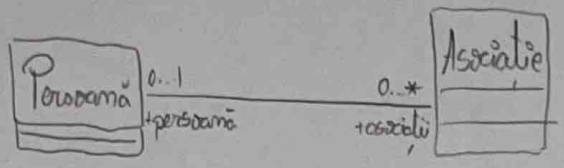
- colectarea cerințelor: diagrama cazurilor de utilizare
- analiza cerințelor: diagrame de clase și de interacțiune
  - ↓ modelul conceptual
  - ↓ modelul dinamic
- proiectarea de sistem: obiective de proiectare (ex definirea unor interfețe), arhitectura sistemului (tehnologii, platforma hardware/software, persistența datelor)
- proiectarea obiectuală: soluții concrete de implementare, se rafinează diagrama de clase cu cele descoperite la pasul anterior
- implementare: transformarea în cod a conceptelor definite până în acest punct modelul obiectual de proiectare → cod sursă
- testare: * unitară (compară modelul obiectual cu fiecare sistem, subsistem, obiect)
  * integrare (compară modelul proiectării de sistem cu diferite subsisteme)
  * de sistem (la colectarea și analiza cerințelor: comportamentul sistemului în ansamblu vs modelul cerințelor)

2. Ilustrați, folosind un ex., modalitatea de transformare a unei clase de asociere în clase și asocieri obișnuite





3. Dați un exemplu de asociere one-to-many bidirecțională și ilustrați modalitatea de supr. a acesteia în cadul sursă.




```
public class Persoană {

    public Set<Asociații> asociații;
    public Persoană() {
        this.asociații = new HashSet();}

    public void add(Asociație a) {
        if(!asociații.contains(a))
        { asociații.add(a);
          a.internalSet(this);
        }
    }
public void remove(Asociație a){
        if (asociații.contains(a))
      { asociații.remove(a);
        a.internal_Set(null);
      }
}
void internalAdd(Asociație a) {
   if(!asociații.contains(a))
       asociații.add(a);
}
void internalRemove(Asociație a) {
{ if(asociații.contain(a))
       asociații.remove(a);
}
public Set<Asociație> getAll()
{ return Collections.unmodifiable_Set(asociații)
}
}
```

```
public class Asociație{

    public Persoană persoană;
    public Asociație() { }
    public void setPersoană(Persoană p){
        Persoană old = this.persoană;
        Persoană new = p;
        if(old != null)
            old.internalRemove(this);
        if(new != null)
            new.internalAdd(this);
        this.persoană = new;
    }
    void internalSet(Persoană p) {
        this.persoană = p;
    }
    public Persoană getPersoană()
    {   return persoană;
    }
}
```

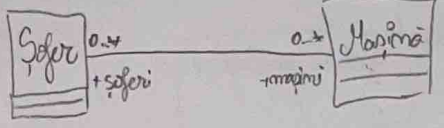
1. În ce constă testarea bazată pe echivalențe?

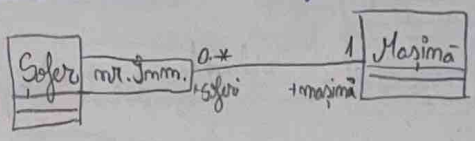
- tehnică de testare blackbox care minimizează nr. de cazuri de test, prin partiționarea intrărilor posibile în clase de echivalență și selectarea unui caz de test pt. fiecare astfel de clasă.

ex: testare validitate nume_utilizator

set1: între 6-12 caractere alfanumerice
set2: < 6 caractere (prea scurt)
set3: > 12 caractere (prea lung)

2. Ilustrați modalitatea de transf. în cod a asocierilor calificate.



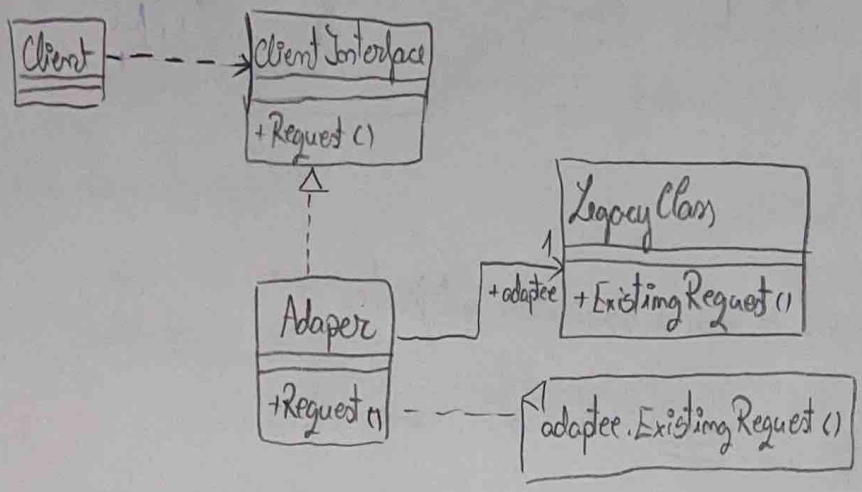


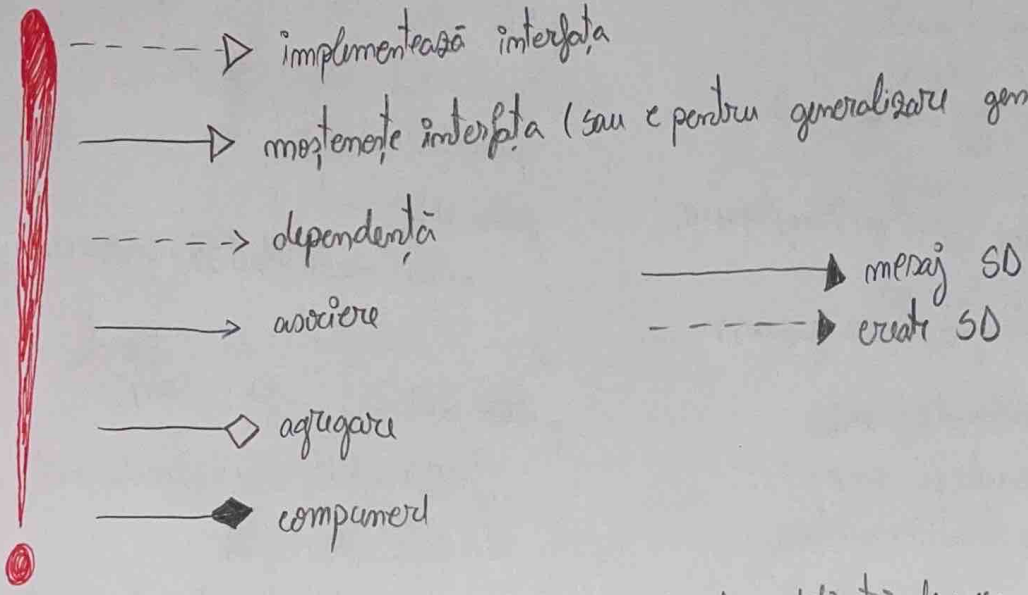

```
public class Șofer {
    public HashMap<String, Mașină> mașini;
    public Șofer() {
        mașini = new HashMap<String, Mașină>();
    }
    public void add(String nr, Mașină mașină) {
        if (!mașini.containsKey(nr))
        {
            mașini.put(nr, mașină);
            mașină.addInternal(this);
        }
    public getMașină(String nrÎnm)
    {
        return mașini.get(nr.Înm);
    }
public addInternal(String nrÎnm, Mașină m)
{ if (!mașini.containsKey(nrÎnm))
        mașini.put(nrÎnm, m);
}
public HashMap<Mașină> getAll()
{ return Collections.UnmodifiableMap(mașini); }
```

```
public class Mașină {
    public Set<Șofer> șoferi;
    public String nrÎnm;
    public Mașină {
        șoferi = new HashSet<Șofer>();
    }
    public void add(Șofer șofer) {
        if (șofer.getMașină(nrÎnm)!=null)
        { șoferi.add(șofer);
            șofer.addInternal(nrÎnm, mașină);
        }
    }
public HashSet<Șoferi> getAll()
{ return Collections.UnmodifiableSet
                (șoferi);
}
```

3. Descrieți șablonul Adapter.

Convertește interfața într-o clasă cu metoda așteptată de client. Adapter ajută clasele să lucreze împreună, lucru inițial imposibil din cauza incompatibilităților.





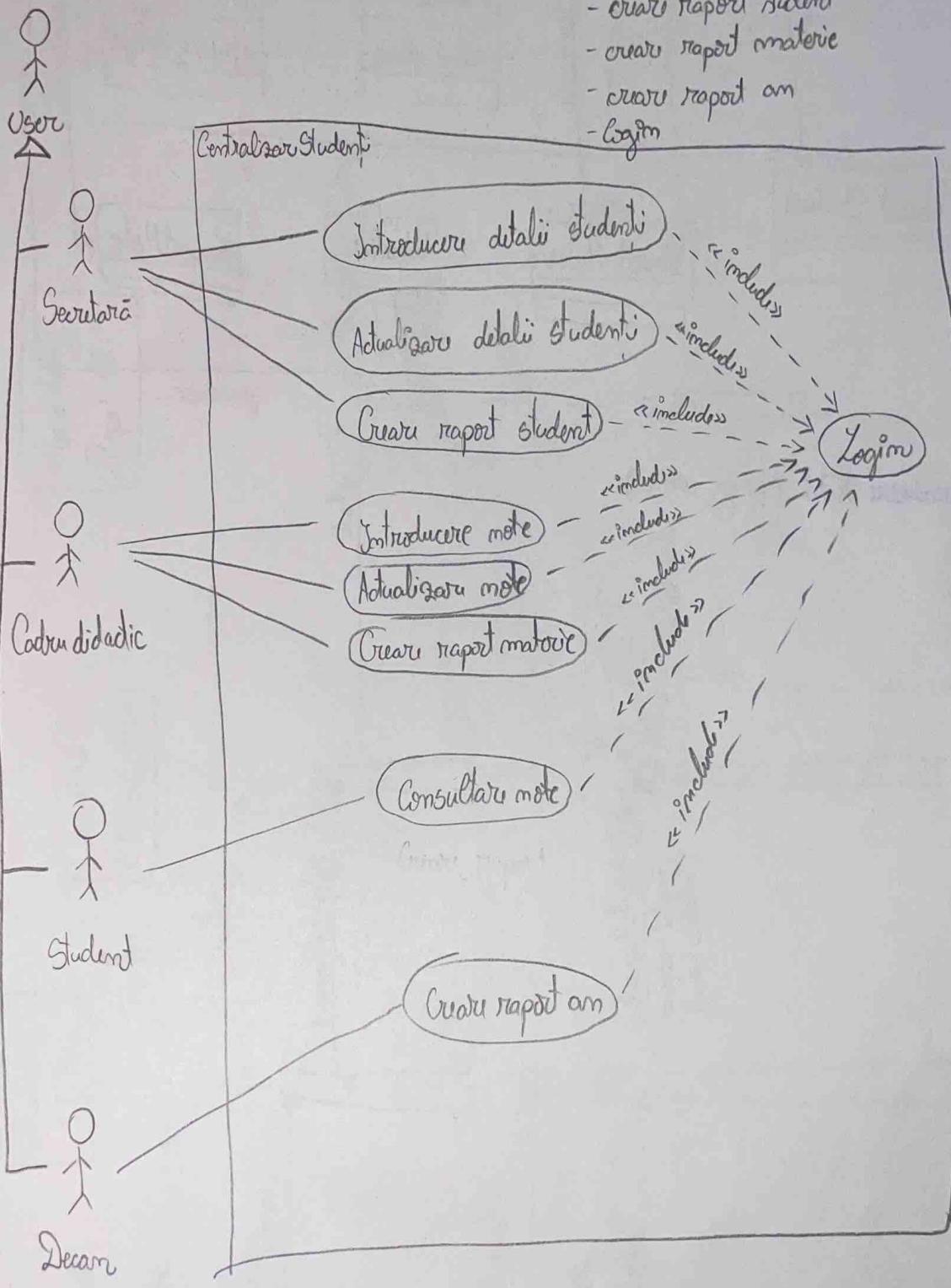
Din varii motive, LegacyClass nu poate fi modificată, de aceea avem nevoie de un adaptor.

B 1. actori, CU și diagrama CU

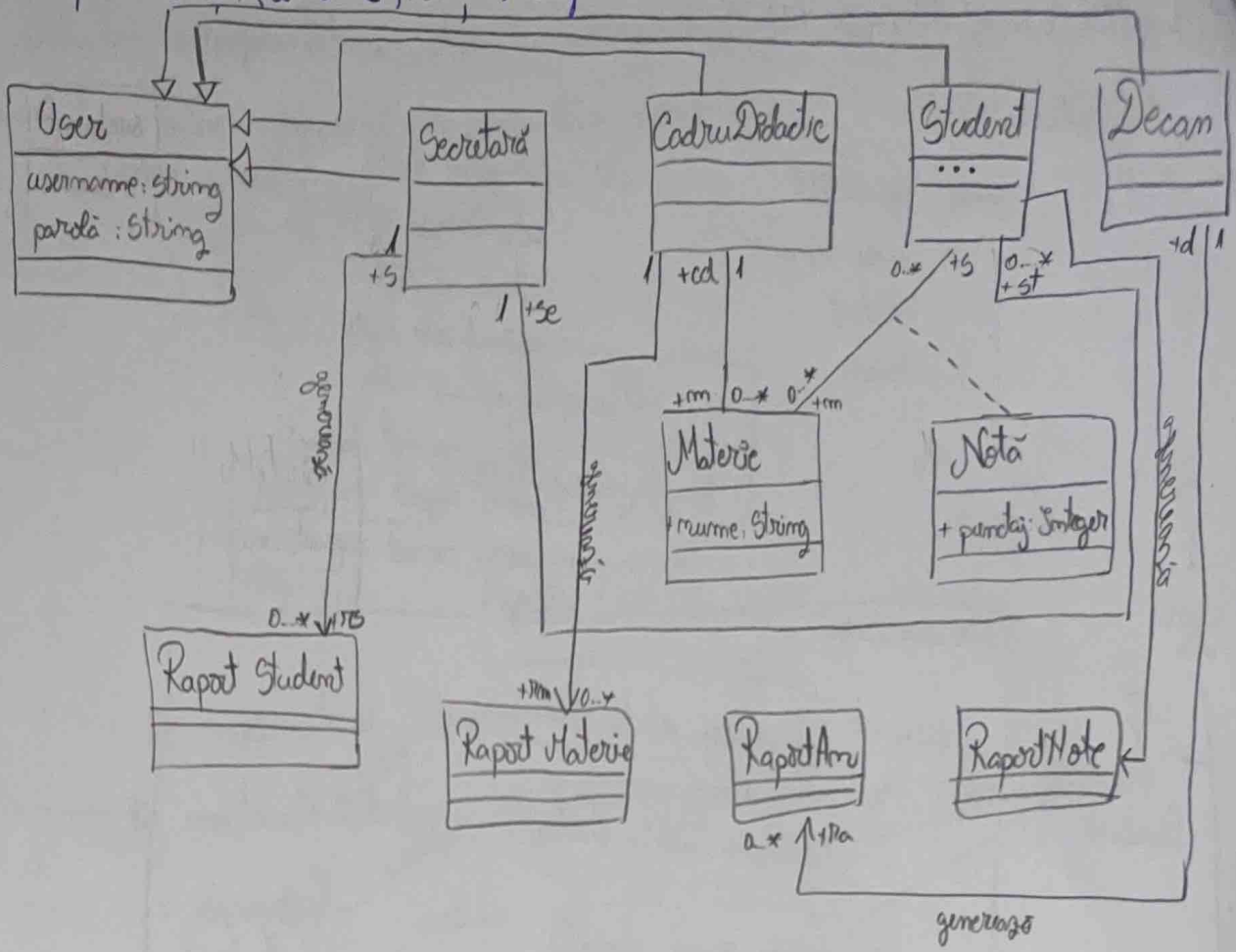
Centralizare Situație:

actori: secretară
cadru didactic
student
decan

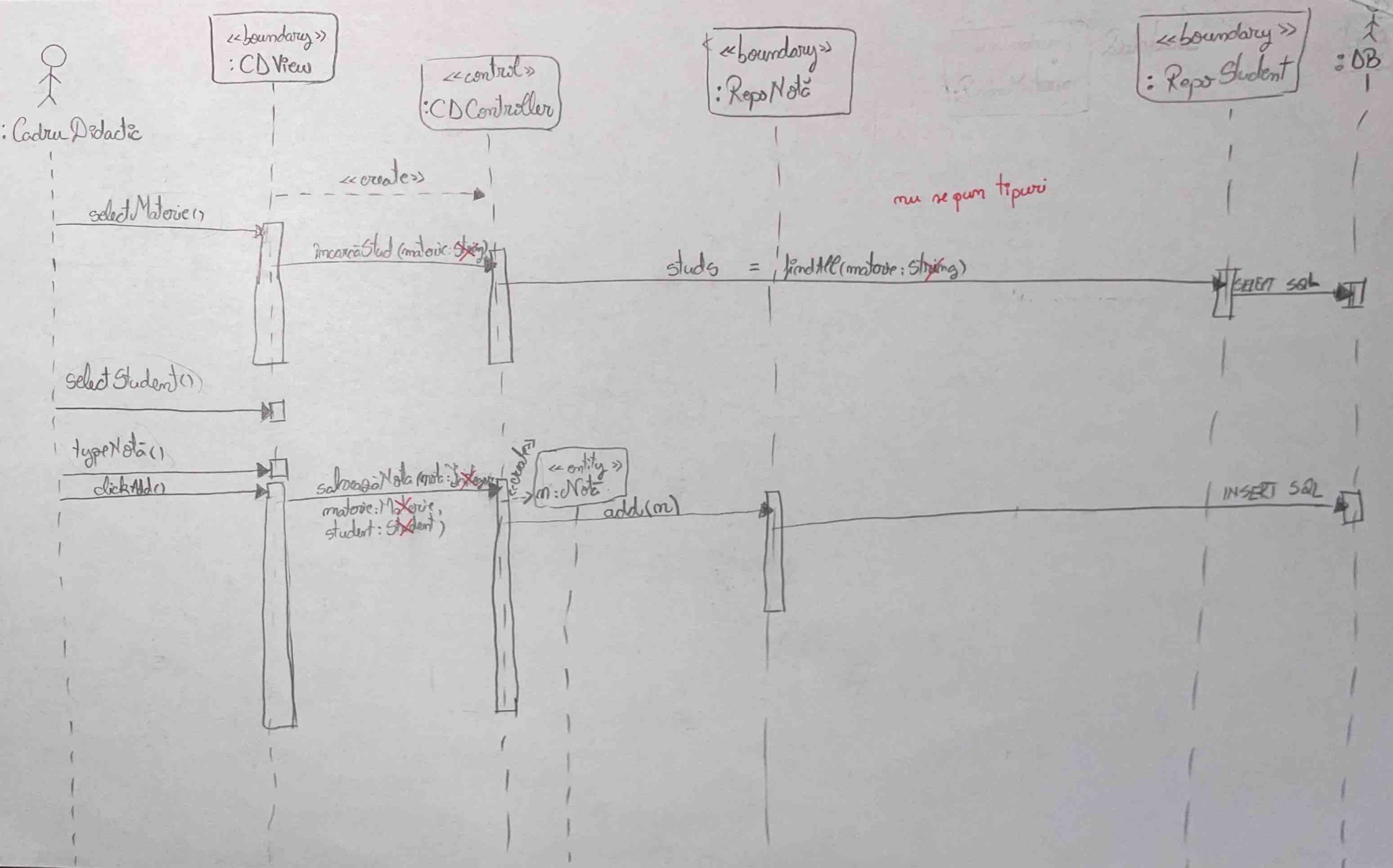
CU: - introducere detalii } despre studenți
- actualizează detalii
- introducă note
- actualizare note
- consultare note
- creare raport student
- creare raport materie
- creare raport an
- Logim



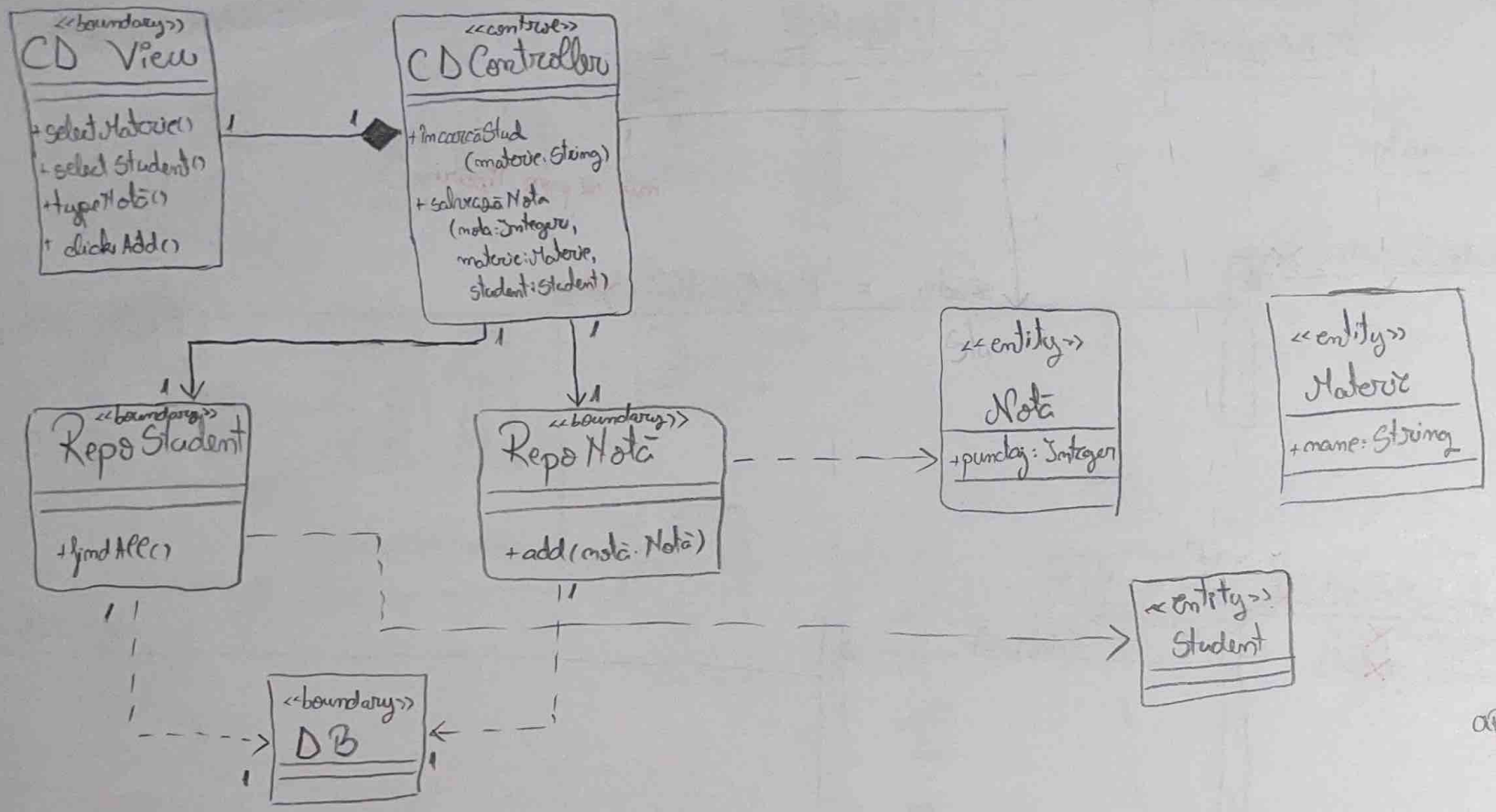
2. concepte (entități), atribute, relații, diagrama de clase



D. 1. SD Introducere Notă (pe pg urm)


: Cadru Didactic
<<boundary>>
: CDView
<<control>>
: CDController
<<boundary>>
: RepoNota
<<boundary>>
: RepoStudent
: DB
<<create>>
nu scrie tipuri
selectMaterie()
incarcaStud(materie: String)
studs = findAll(materie: String)
SELECT SQL
selectStudent()
typeNota()
clickAdd()
salveazaNota(nota: double, materie: Materie, student: Student)
<<create>>
<<entity>>
n: Nota
add(n)
INSERT SQL

2. diagrama detaliată de clase



după restul laptelor din cealaltă diagramă