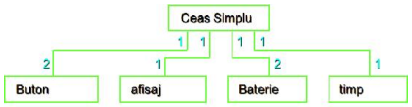
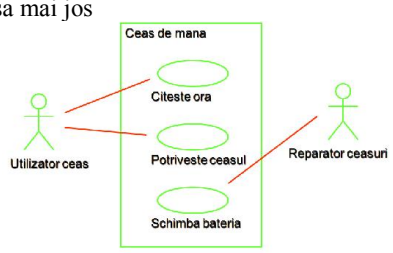
Laboratorul 4
Modelarea folosind UML

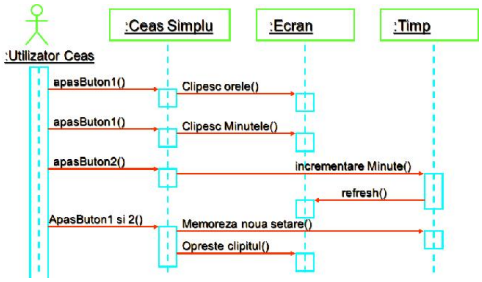
1. Sa se implementeze aplicatia ceasului discutata la curs care are
   a. urmatoarea diagram de clase:



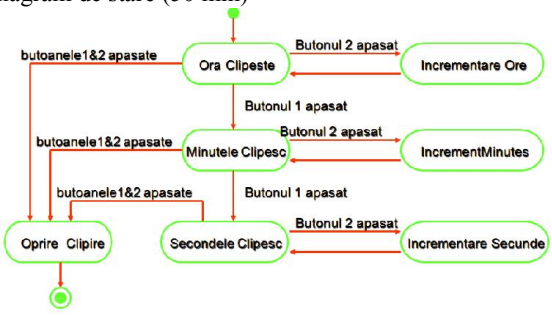
   b. functionalitatea descrisa mai jos



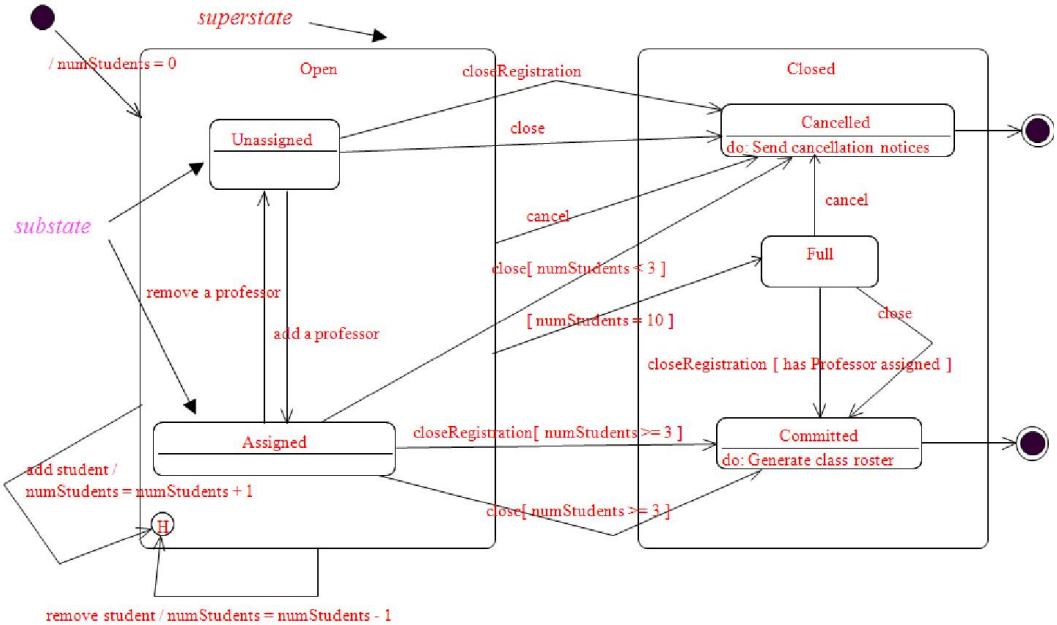
   c. urmatoarea diagrama de secventa



   d. si urmatoarea diagram de stare (50 min)



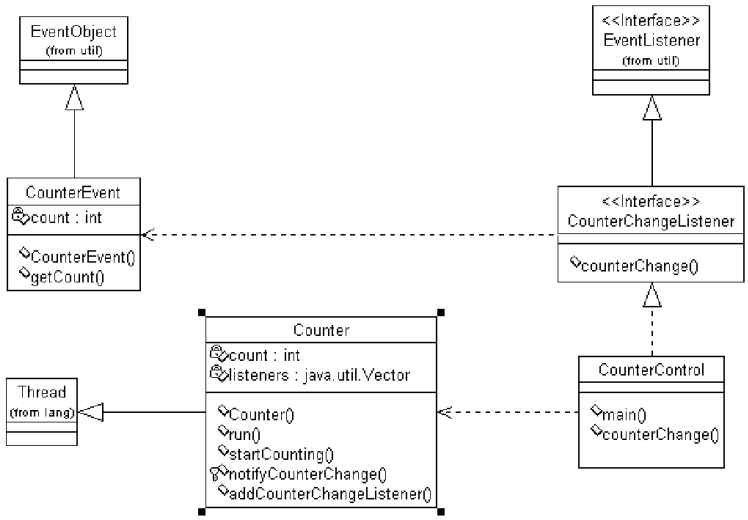
2. Pornind de la urmatoarea diagram de stare care se refera la gestiunea inscrierilor la un curs optional pentru oricare professor dintr-un department:



Sa se propuna diagramele de clase de baza, si sa se dezvolte programul aferent(50 min).

3. **Tema acasa:** Sa se scrie o aplicatie care defineste o clasa Numarator care la schimbarea valorii numaratorul genereaza un eveniment CounterEvent. Acest eveniment va fi receptat de o clasa Receptor care la aparitia evenimentelor afiseaza valoarea numaratorului.
Pentru orice eveniment nou se creaza o clasa derivata din clasa EventObject, iar interfetele pentru receptionarea evenimentelor trebuie sa extinda clasa EventListener din pachetul java.util. Clasa Counter este un numarator care se ruleaza pe un fir de executie propriu si este capabil de gestionarea obiectelor de tip listener. Aceste obiecte listener sunt stocate intr-un vector (java.util.Vector) si la modificarea numaratorului sunt anuntati. (*notifyCounterChange*).

Diagrama de clase:



Schelet orientativ pentru programul Java:

```
class CounterEvent extends EventObject
{  private int count;
   CounterEvent ( Object source, int count ){
        …}
   public int getCount(){
            …  }
}

interface CounterChangeListener extends EventListener
{  void counterChange( CounterEvent e );   }

class Counter extends Thread
{
            …
public Counter( string name, int count )
   {
     …}
   public Counter( int count ){
       …}
   public Counter(){
     … }
   public void run(){
      …      }
   public void startCounting(){
       …  }
   public notifyCounterChange( int count ){
      …}
   }
   public synchronized void addCounterChangeListener( CounterChangeListener ccl ){
      … }
   public synchronized void removeCounterChangeListener( CounterChangeListener ccl ){
      … }
}
```

```
public class CounterControl implements CounterChangeListener
{
	public CounterControl()
	{…
	}

	public static void main(String[] args)
	{
		CounterControl cc = new CounterControl();
	}

	public void counterChange(CounterEvent e)
	{
		System.out.println("Counter value has changed: " + e.getCount());
	}
}
```