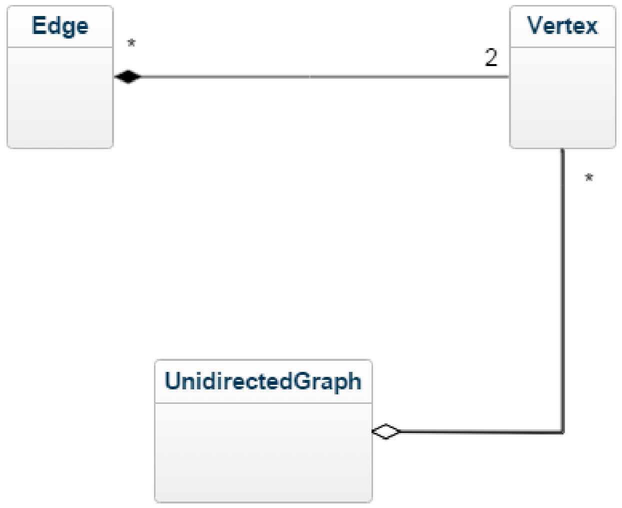
Probleme UML

Balotă Ioana-Dorina
30235

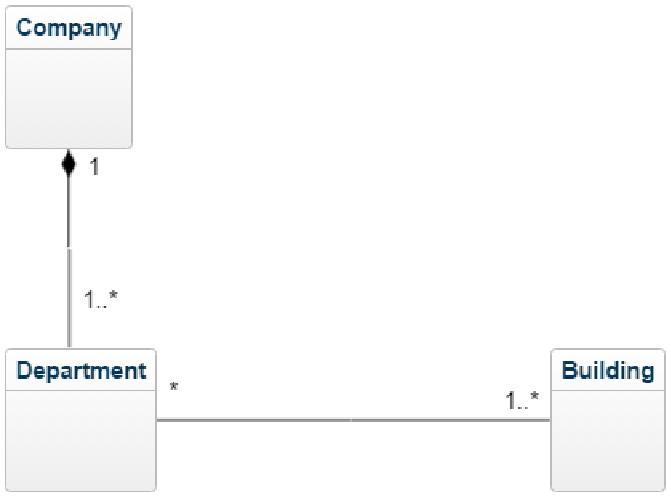
P1. Prepare a class model to describe undirected graphs. An undirected graph consists of a set of vertices and a set of edges. Edges connect pairs of vertices. Your model should capture only the structure of graphs (i.e. connectivity).

Soluție:



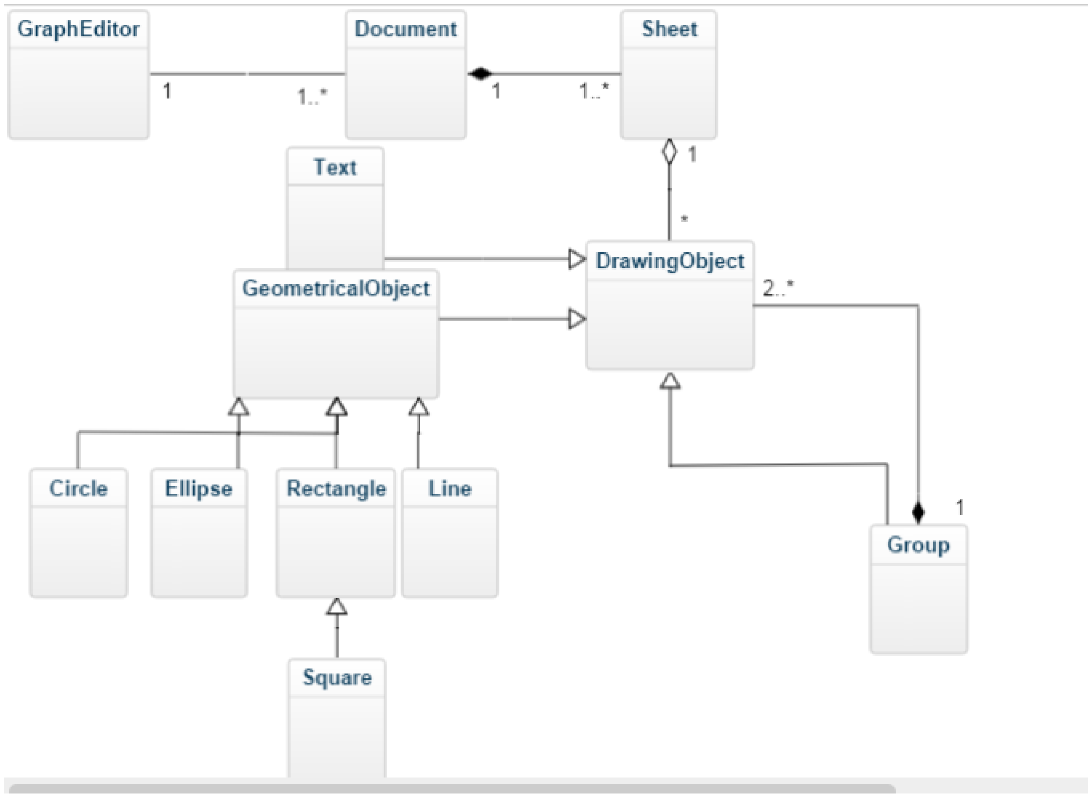
P2. Consider the following specification: "*A company consists of several departments. Each department is located in one or more buildings*. ". Draw a class diagram to model the concepts above (no attributes and operations, just classes and the relationships between them including multiplicities).

Soluție:



P3. Prepare a class diagram for a graphical document editor that supports grouping. Assume that a document consists of several sheets. Each sheet contains drawing objects, including text, geometrical objects, and groups. A group is simply a set of drawing objects, possibly including other groups. A group must contain at least two drawing objects. A drawing object can be a direct member of at most one group. Geometrical objects include circles, ellipses, rectangles, lines and squares.

Soluție:



P4. A* is an advertising company in New York. A* deals with other companies that it calls clients. A record is kept of each client company. Clients have advertising campaigns, and a record is kept of every campaign. Each campaign includes one or more adverts. A* nominates members of creative team, which work on campaigns. One member of the creative team manages each campaign. Staff may be working on more than one project at a time. When a campaign starts, the manager responsible estimates the likely cost of the client and agrees it with the client. A finish date may be set for a campaign at any time, and may be changed. When the campaign is completed, an actual completion date and the actual cost are recorded. When the client pays, the date is recorded. The manager checks the campaign budget periodically. The system should also hold the staff grades, keep a record of the contact information for each staff member, and should calculate staff salaries.

Requirements

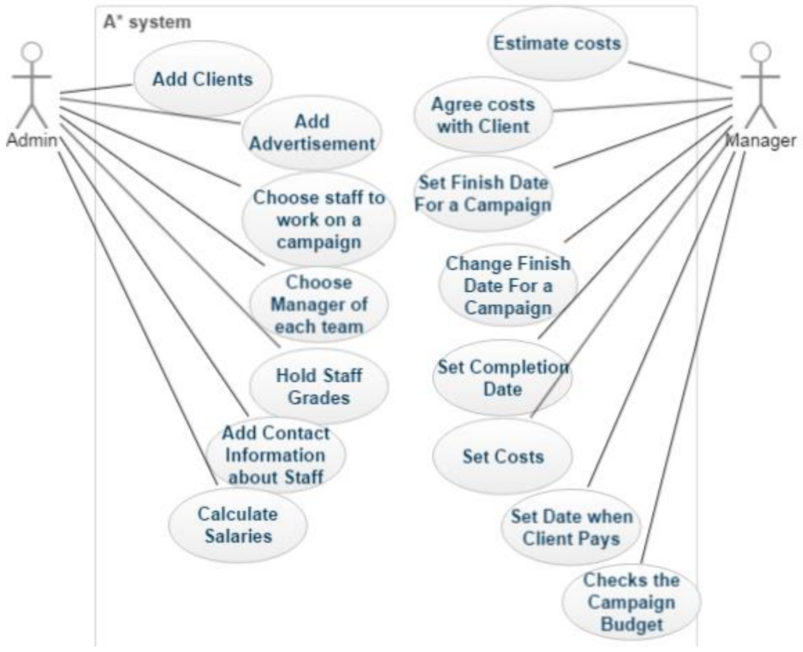
a) Draw a UML use case diagram for the A* information system.

b) Draw a UML class diagram. The class diagram should represent all of the classes, their attributes and operations, relationships between classes, multiplicity specifications.

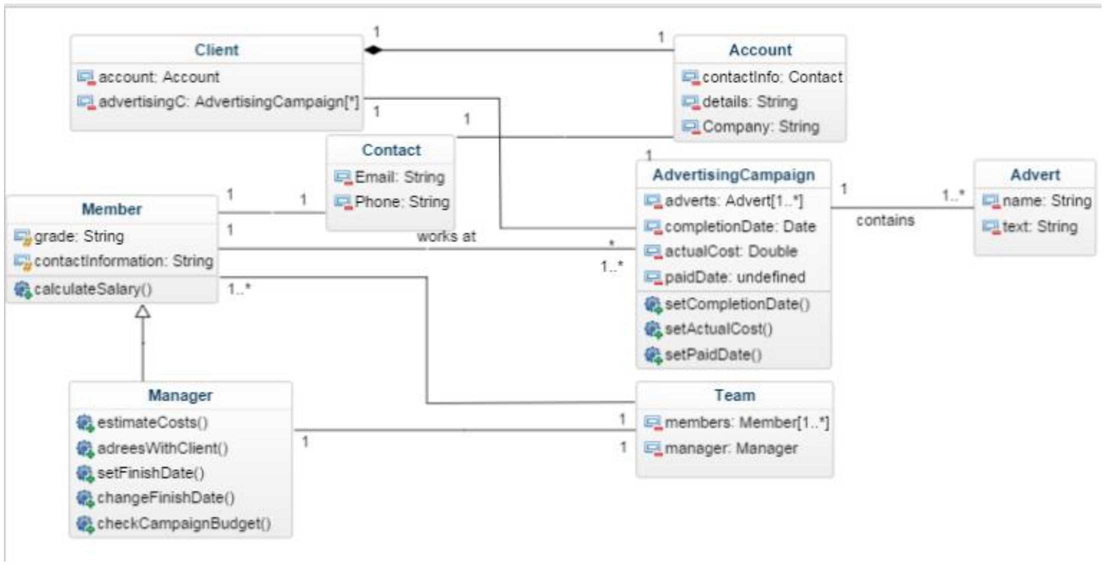
c) Draw a UML sequence diagram for the use case *Add Advertisement*.

Soluție:

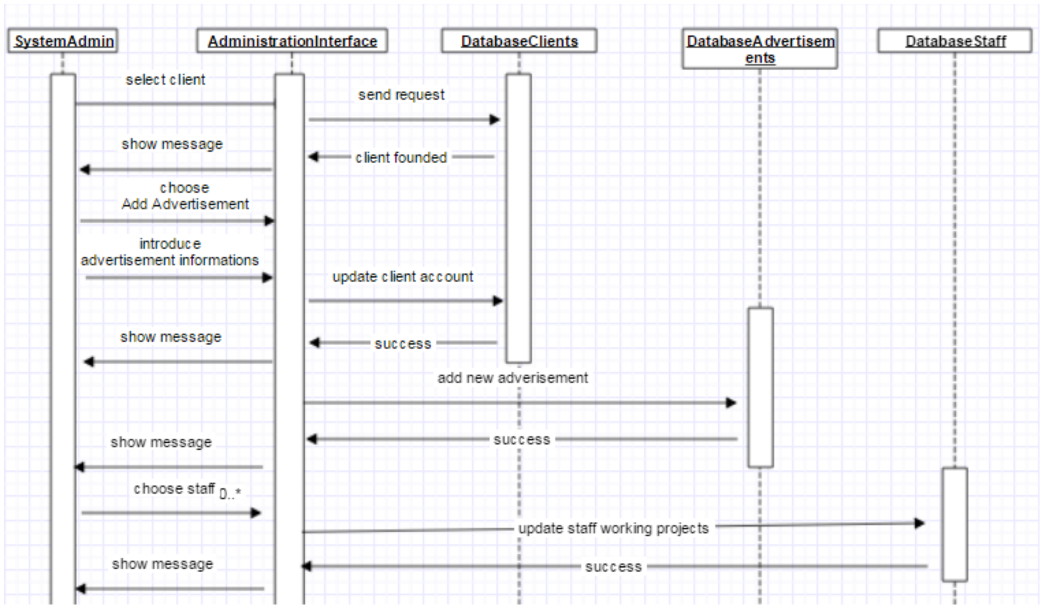
a)


A* system
Admin
Manager
Add Clients
Add Advertisement
Choose staff to work on a campaign
Choose Manager of each team
Hold Staff Grades
Add Contact Information about Staff
Calculate Salaries
Estimate costs
Agree costs with Client
Set Finish Date For a Campaign
Change Finish Date For a Campaign
Set Completion Date
Set Costs
Set Date when Client Pays
Checks the Campaign Budget

b)



c)



P5. Design a photo/video sharing Web application. The application should allow a potential new user to create an account, activate the account via an email link or view photos and videos without being logged in. Login would allow him to edit his profile, post videos and post photos either stand-alone or in an album he creates. Also, upon viewing a photo or a video, anyone, logged in or not, can flag the content for questionable content. A content advisor should review each flagged item once logged in, then decide on whether the content is acceptable for the scope of the Web application. If the content is inappropriate, the content advisor can block the owning user's account.

Requirements

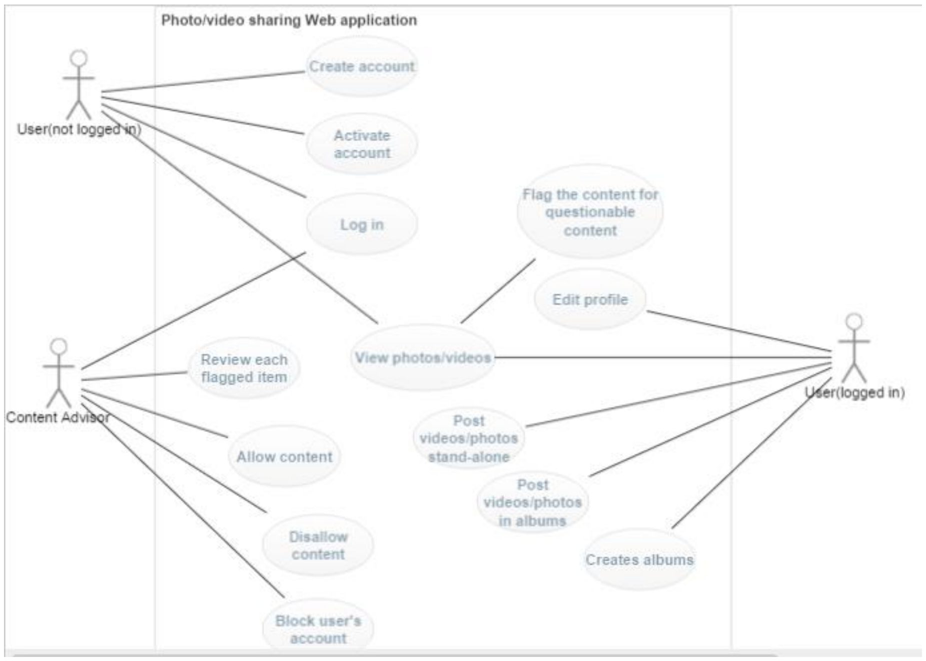
a) Draw a UML use case diagram for the photo/video sharing Web application.

b) Draw a UML class diagram. The class diagram should represent all of the classes, their attributes and operations, relationships between classes, multiplicity specifications.

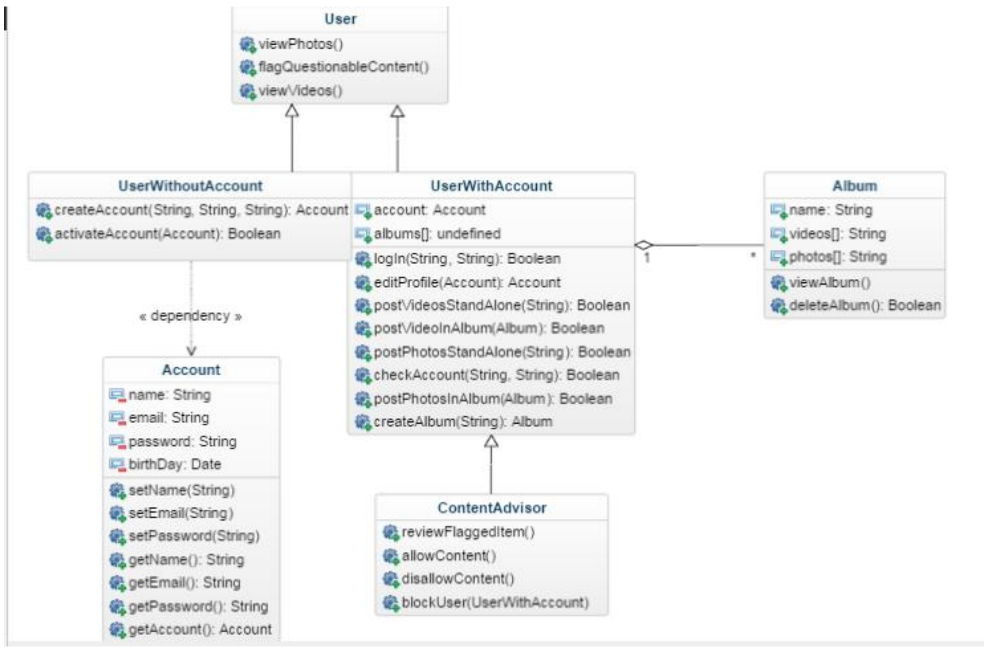
c) Draw a UML sequence diagram for the use case *Block User*.

Soluție:

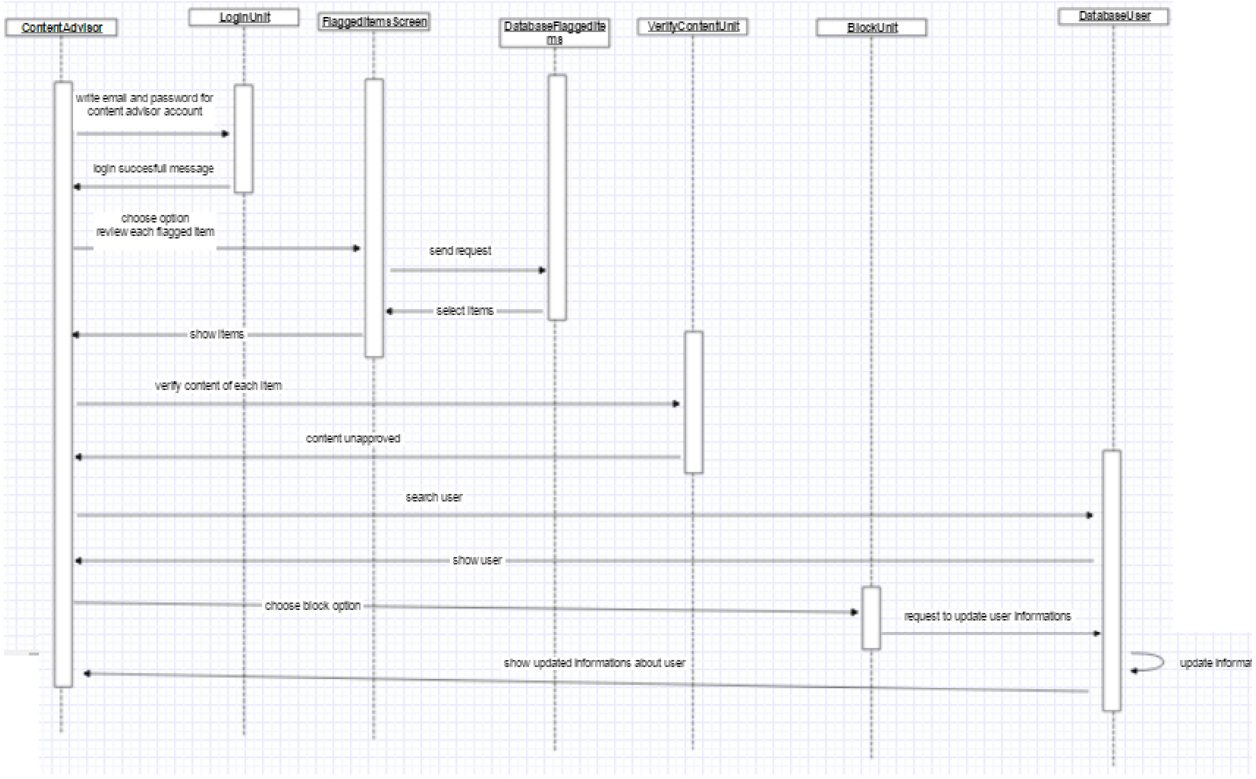
a)



b)



c)

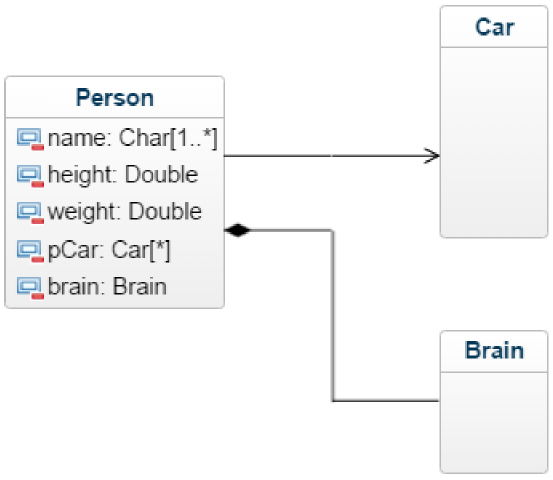


P6. Draw an UML diagram for the following C++ segment of code:

class Person

```
{
public:
        Person(const char* aName);
        void speak();
        void drive();
private:
        char* name; // Persons Name
        Brain brain; // persons brain
        Car* pCar; // Car owned by the person
        double height;
        double weight;
};
```

Soluție:

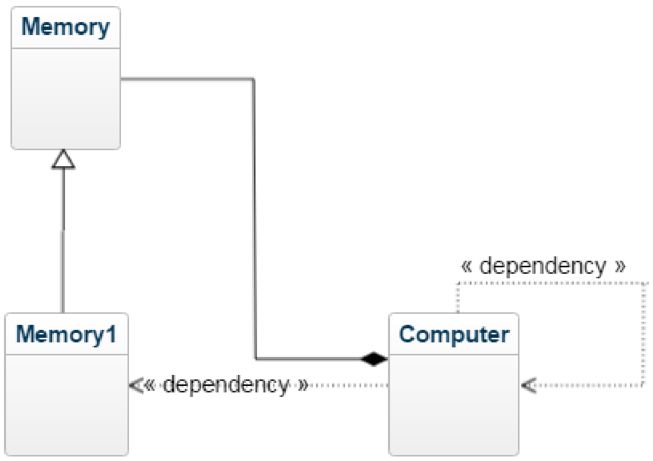


P7. Given the following code:

```
class Memory {…} // Assume a copy constructor is provided for this class.

class Memory1 extends Memory {…} // Assume a copy constructor is provided.

class Computer
{
  private Memory theMemory;
  public Computer(Computer another)
 {
     if (another.theMemory instanceof Memory1)
            theMemory = new Memory1(another.theMemory);
     else
            theMemory = new Memory(another.theMemory);
 }
}
```

Draw a UML class diagram showing the relationship between these classes.

Soluție:



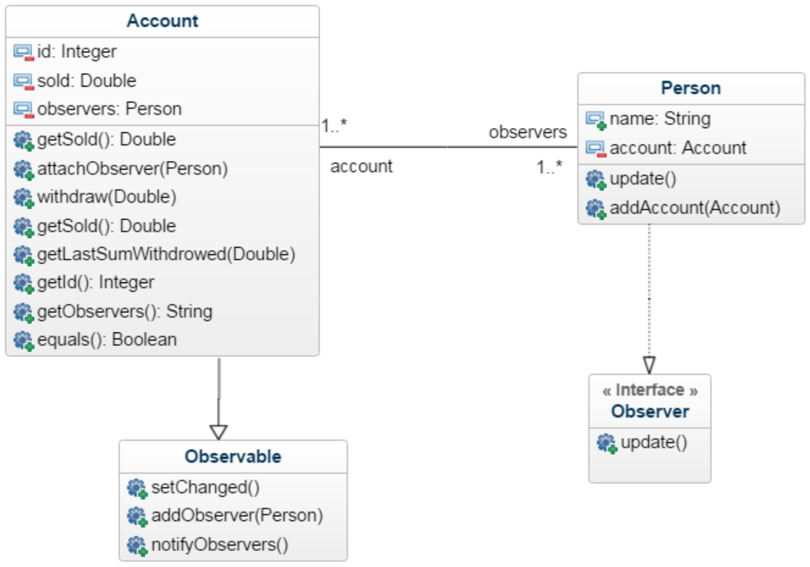
P8. Choose a design pattern studied in previous courses. Think of a problem in which it could be applied, draw the associated class diagram and write the associated source code.

Soluție:

a)Enunț

Simularea unei aplicații bancare în care o persoană poate avea oricâte conturi, iar un cont poate fi administrat de mai multe persoane. S-a exemplificat folosirea pattern-ului Observer pentru cazul de utilizare retragere bani.

b)Diagrama de clase

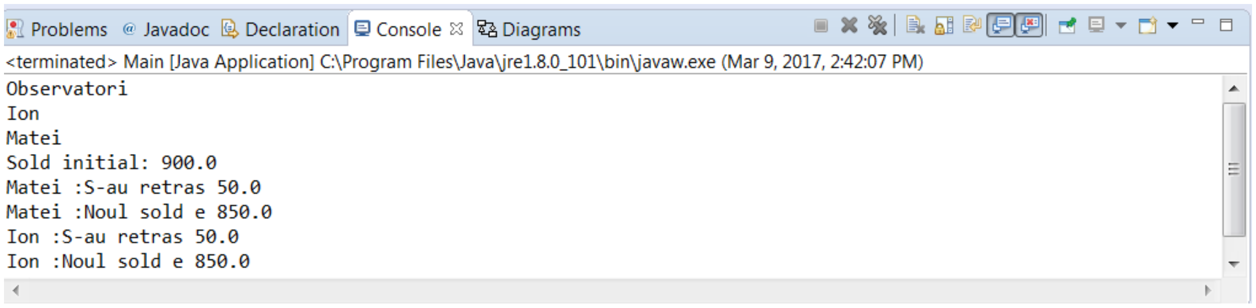


c)Cod

```java
import java.util.ArrayList;
import java.util.Observable;
public class Account extends Observable
{
        private int id;
        private double sold ;
        private double lastSumWithdrowed=0;
        private ArrayList<Person>observers;

        public Account(int id, double sold)
        {
                this.id=id;
                this.sold=sold;
                observers=new ArrayList<Person>();
        }

    public void attachObserver(Person p)
    {
        observers.add(p);
        this.addObserver(p);
    }

    public void withdraw(double sum)
    {
        if(sum<sold)
        {
                sold=sold-sum;
                lastSumWithdrowed=sum;
                setChanged();
                notifyObservers();
        }
    }

    public double getSold()
    {
        return sold;
    }

    public double getLastSumWithdrowed()
    {
        return lastSumWithdrowed;
    }
    public double getId()
    {
        return id;
    }

    public boolean equals(Account ac)
    {
        if(id==ac.getId())return true;
        else return false;

    }
    public void getObservers()
    {
        for(int i=0;i<observers.size();i++)
        {
                System.out.println(observers.get(i).name);
        }
    }
}
```

```java
import java.util.*;
import java.util.Observer;

public class Person implements Observer{

    String name;
    ArrayList< Account> account;

    public Person(String nume, Account acc)
    {
        account=new ArrayList<Account>();

        this.name=nume;
        account.add(acc);

        for(int i=0;i<account.size();i++)
        account.get(i).attachObserver(this);

    }
    public void update(Observable ac, Object obj)
    {
        for(int i=0;i<account.size();i++)
        {
                if(ac.equals(account.get(i))==true)
                {
                        System.out.println( name+" :S-au retras "+account.get(i).getLastSumWithdrowed());
                        System.out.println(name+" :Noul sold e" +account.get(i).getSold());
                }
        }
    }

    public void addAccount(Account acc)
    {
        account.add(acc);
    }
}



public class Main {

        public static void main(String[] args)
        {
                Account a=new Account(1234,900.0);

                Person p1=new Person("Ion",a);
                Person p2=new Person("Matei",a);

                System.out.println("Observatori ");
                a.getObservers();
                System.out.println("Sold initial: "+a.getSold());
                a.withdraw(50);


        }

}
```

d)Demonstrare funcționalitate



```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (Mar 9, 2017, 2:42:07 PM)
Observatori
Ion
Matei
Sold initial: 900.0
Matei :S-au retras 50.0
Matei :Noul sold e 850.0
Ion :S-au retras 50.0
Ion :Noul sold e 850.0
```