

INFO 5100: Application Engineering and Development

How to engineer the user into the application?



The problem

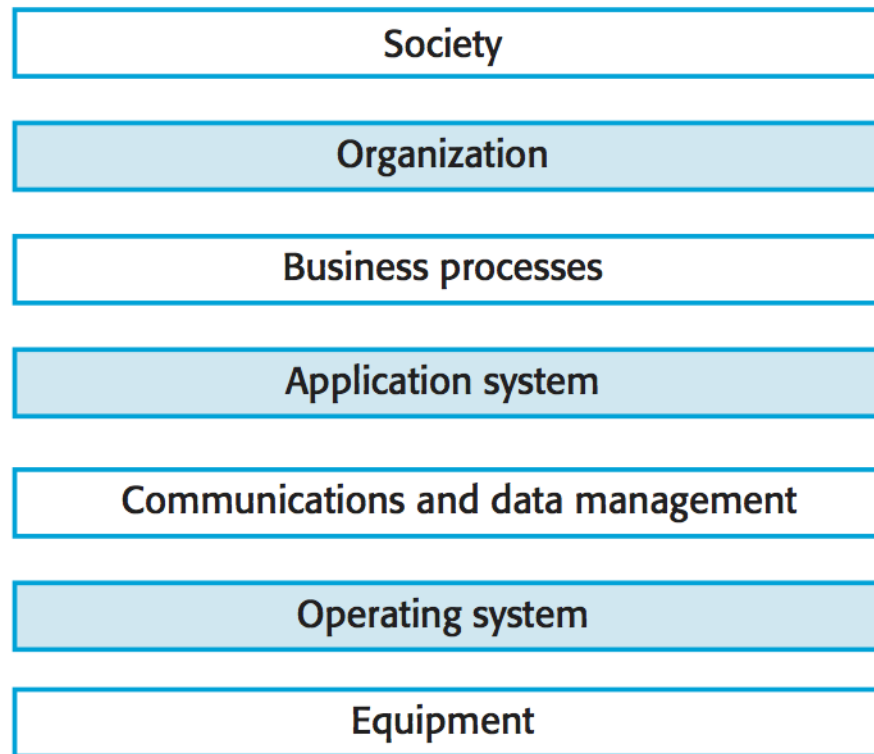
- Applications are designed for users. The applications must identify the users
- We need to know who did what, when, how, etc ..
- How to prove that you are who you say you are.

Who are the players?

Users are people



We will extend software engineering to include the social structure as well

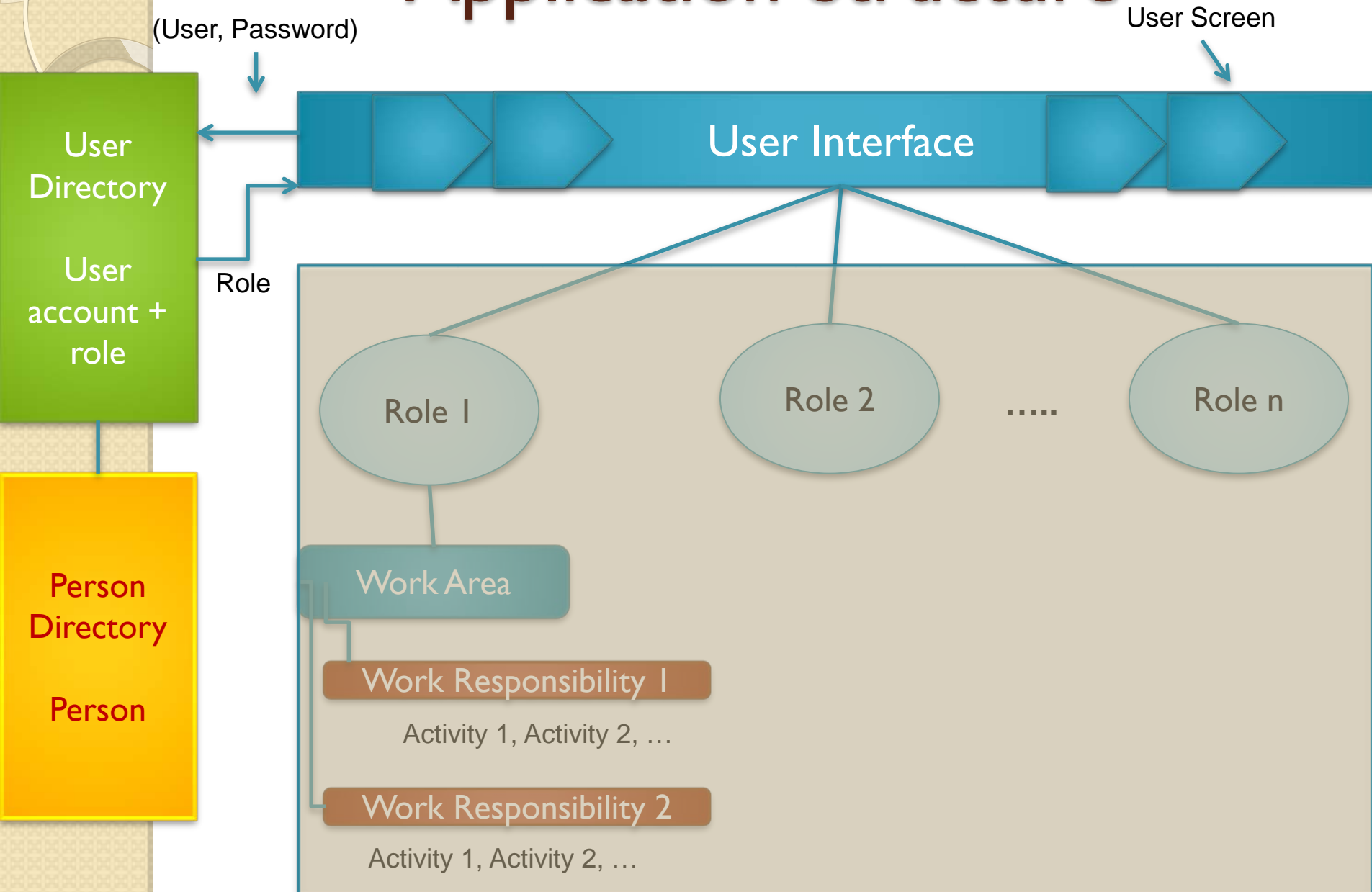


People as subject of user account access



User account access

Application Structure



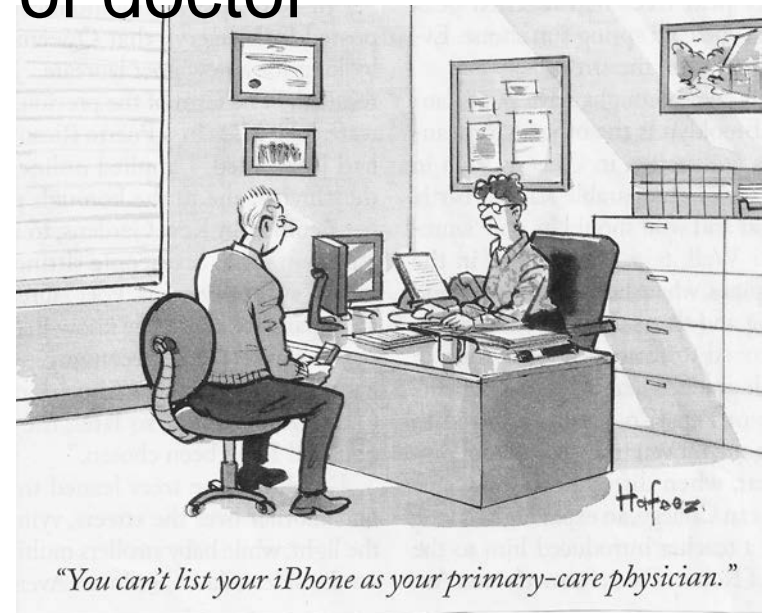
Persons as subjects of user actions

The doctor as a user uses an app to access patient record – we say patient (person) is subject of doctor action



patient


Doctor as user



The Business Model



User



First name,
Last name,
Social security number
DOB
Address,
Userid
Password
Account type
Status (active, disabled)
Creation Date
[Organization \(affiliation\)](#)
Etc

Example: Multiple user accounts means duplication of information

Joe,
Smith,
290-29-2974
2/2/1986
36 Huntington Ave,
JoeB
xxxxxx
Student
Status (active, disabled)
Creation Date
School of Business
Etc

Joe,
Smith,
290-29-2974
2/2/1996
100 Main Street,
JoeCoe
xxxxxxs
Student
Status (active, disabled)
Creation Date
College of Engineering
Etc

Joe,
Smithy,
200-29-2974
2/2/1886
360 Huntington Ave,
JoeNeu
xxxxxx
Student
Status (active, disabled)
Creation Date
Northeastern University
Etc

What if we split the info into:

First name,
Last name,
Social security number
DOB
Address

Userid
Password
Account type
Status (active, disabled)
Creation Date
Organization (affiliation)
Etc



What if we split the info into:

First name,
Last name,
Social security number
DOB
Address

Personal information

Userid
Password
Account type
Status (active, disabled)
Creation Date
[Organization \(affiliation\)](#)
Etc

Affiliation specific

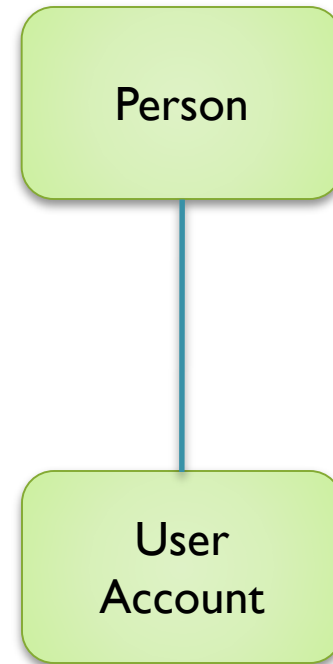
For example if you are a double major and have two accounts with two different colleges

Person has an account with organization (user account)

What if we split the info into:

First name,
Last name,
Social security number
DOB
Address

Userid
Password
Account type
Organization (affiliation)
Etc



Compare this to previous example

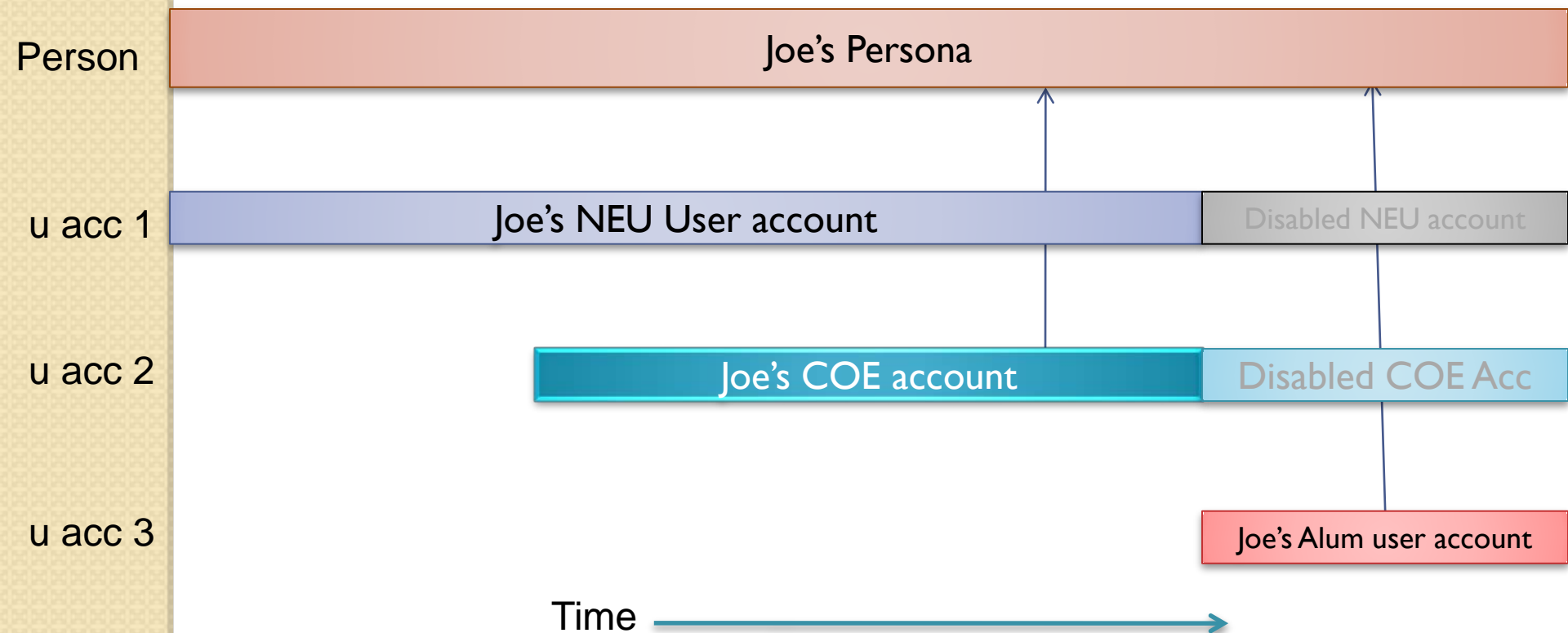
Joe,
Smithy,
200-29-2974
2/2/1886
360 Huntington Ave,

JoeB
xxxxxx
Student
School of Business
Etc

JoeCoe
xxxxxxs
Student
College of Engineering
Etc

JoeNeu
xxxxxx
Student
Northeastern University
Etc

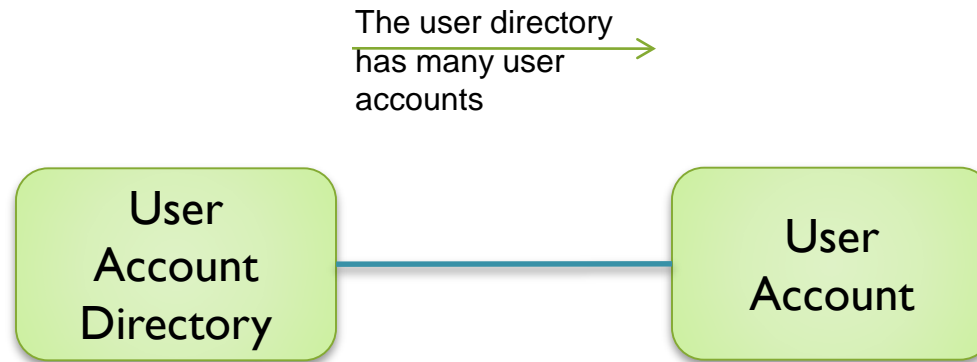
One person multiple user accounts over time: All link to Joe the person



What if we split the info into:



Group user accounts to a directory

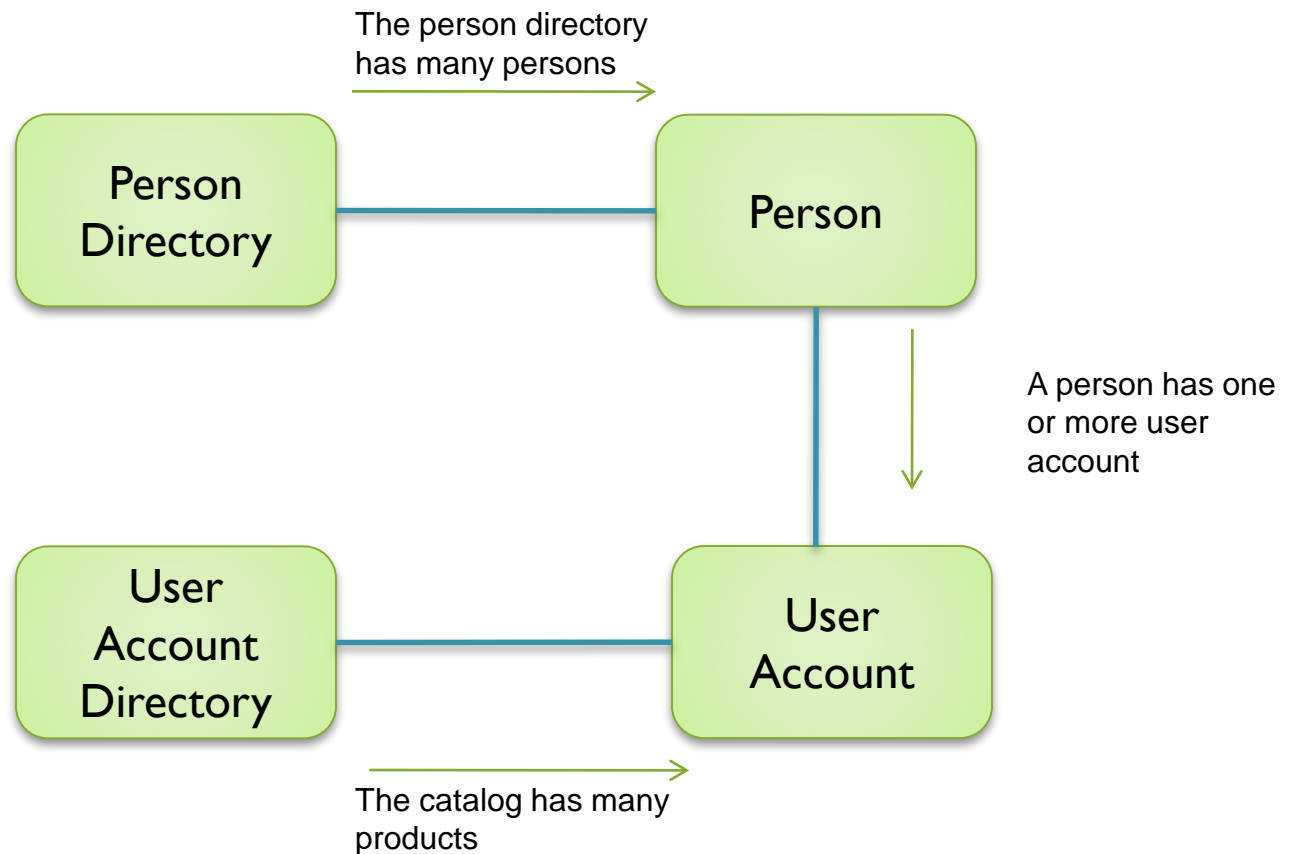


Group user accounts to a directory

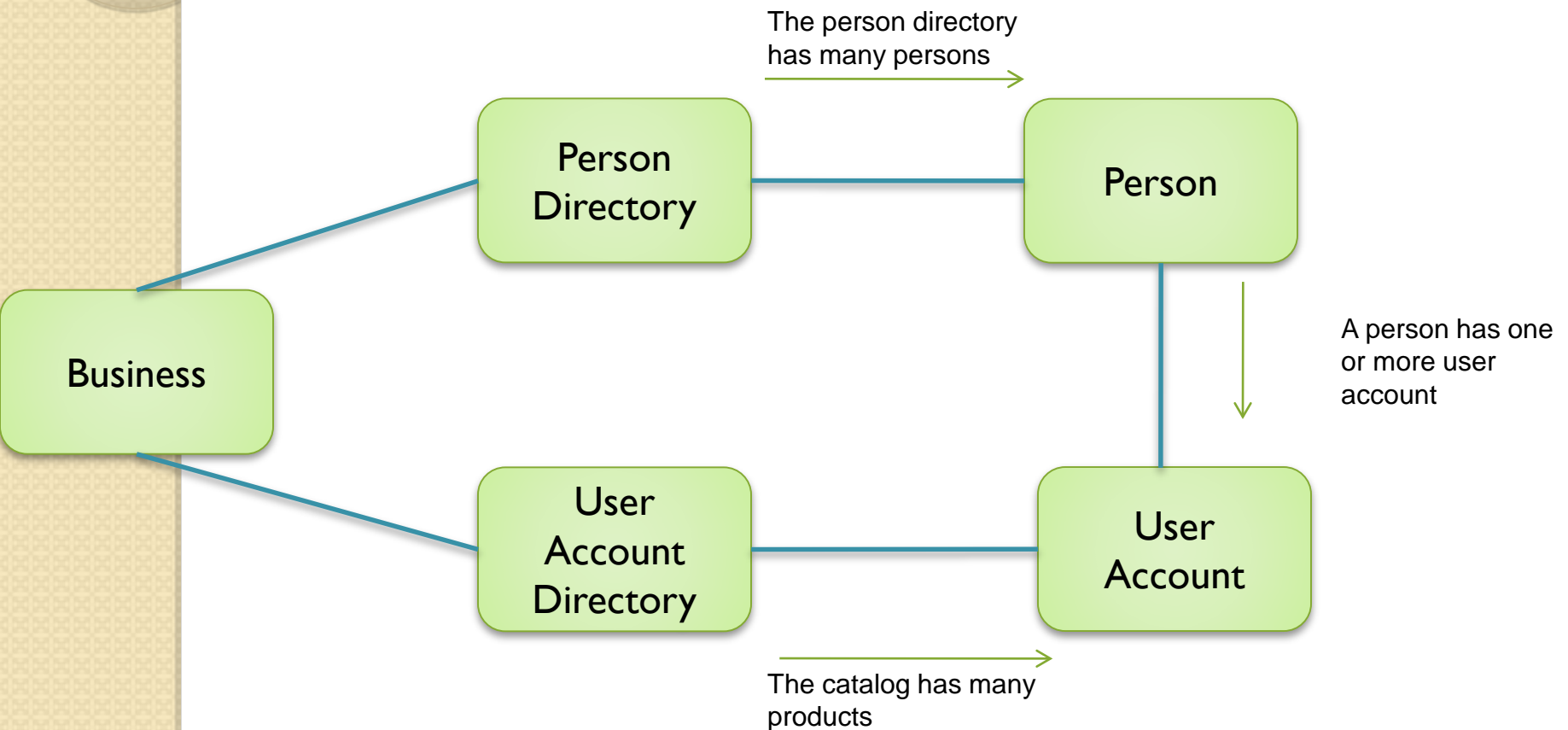
The person directory
has many persons →



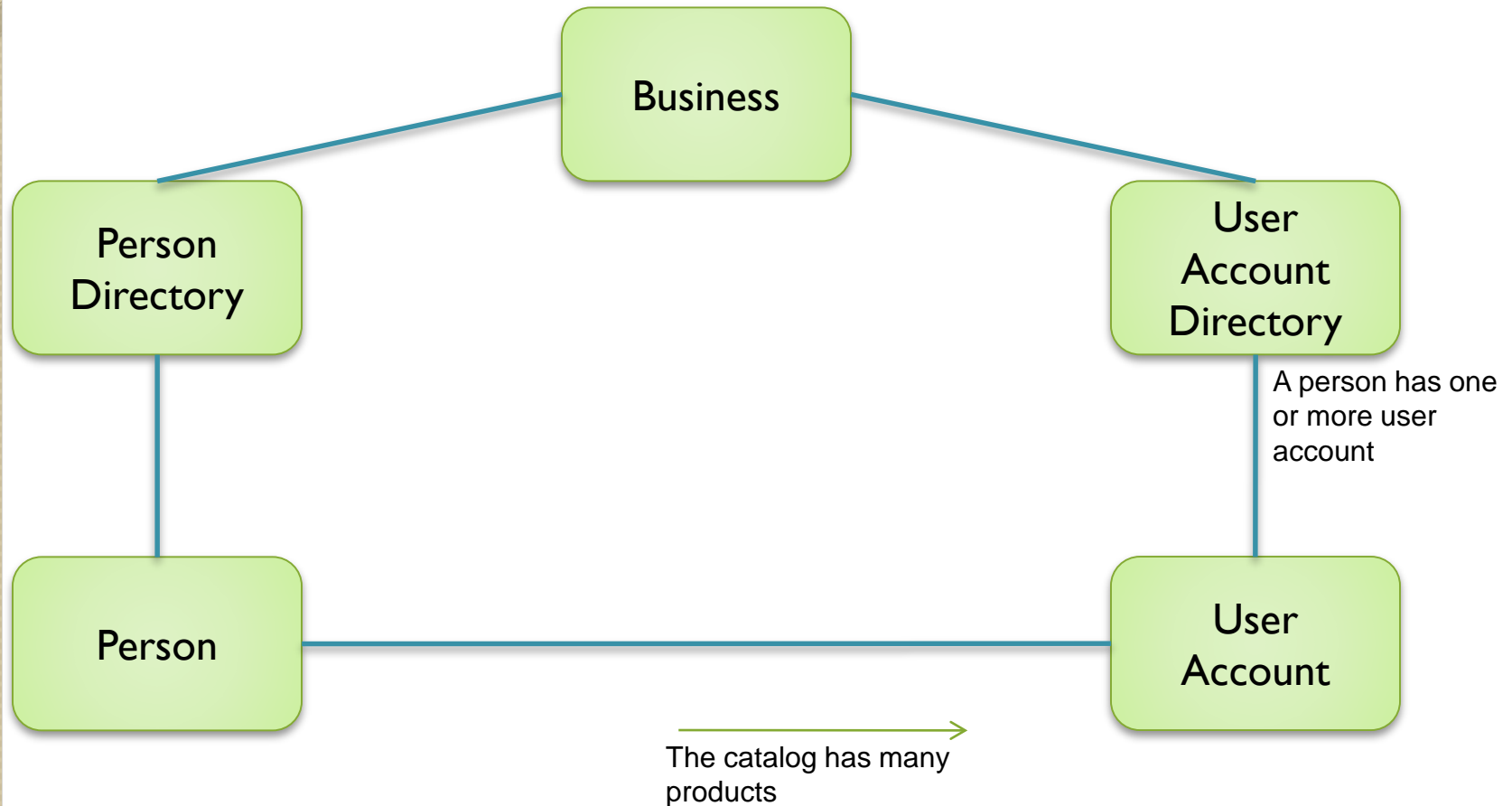
Combine the two:



Business concept is the organizational Structure that unifies these personnel and their accounts



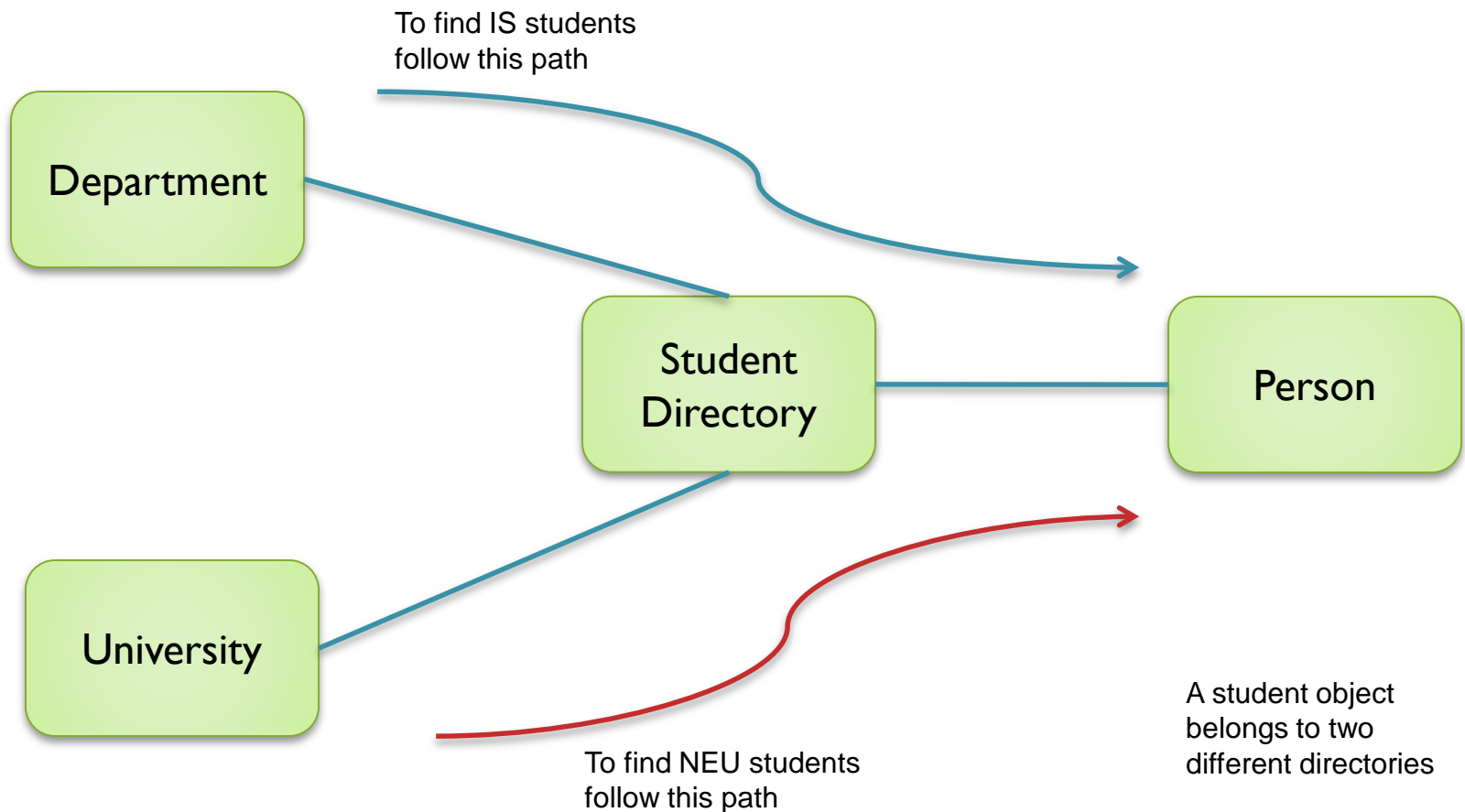
Combine the two:



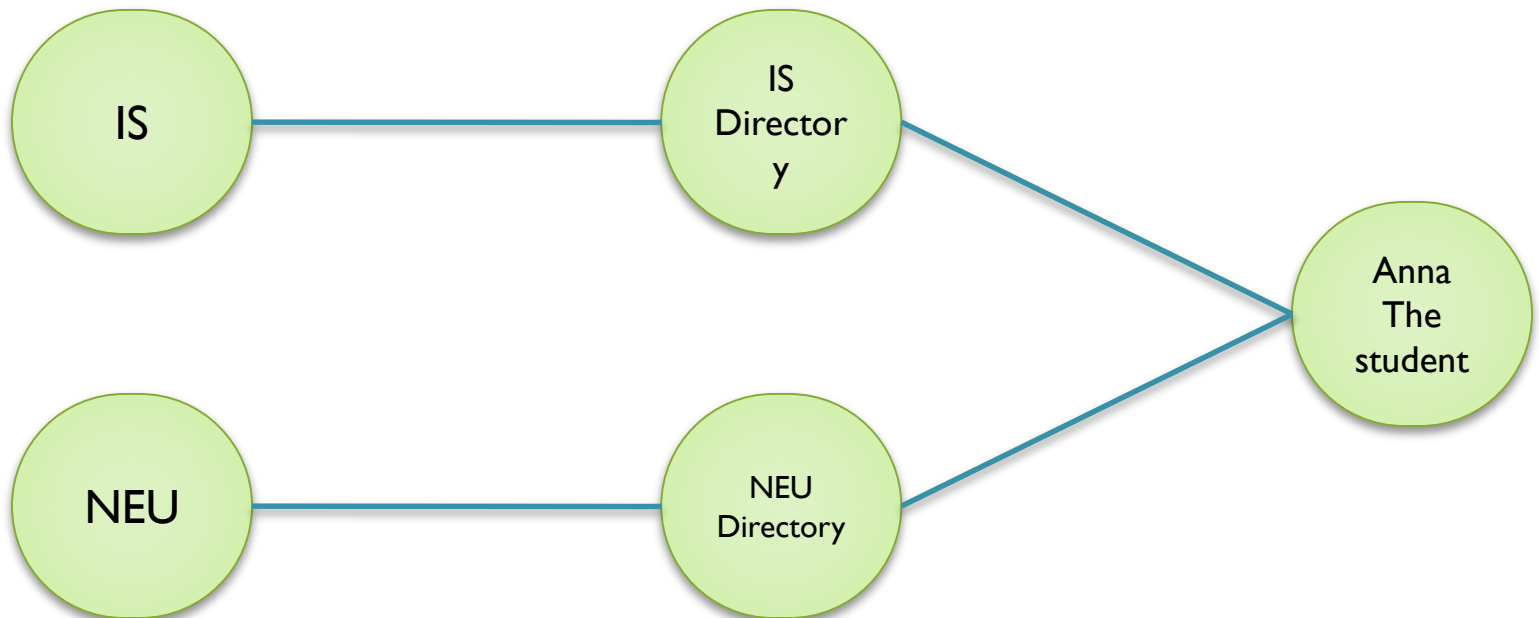
Other patterns



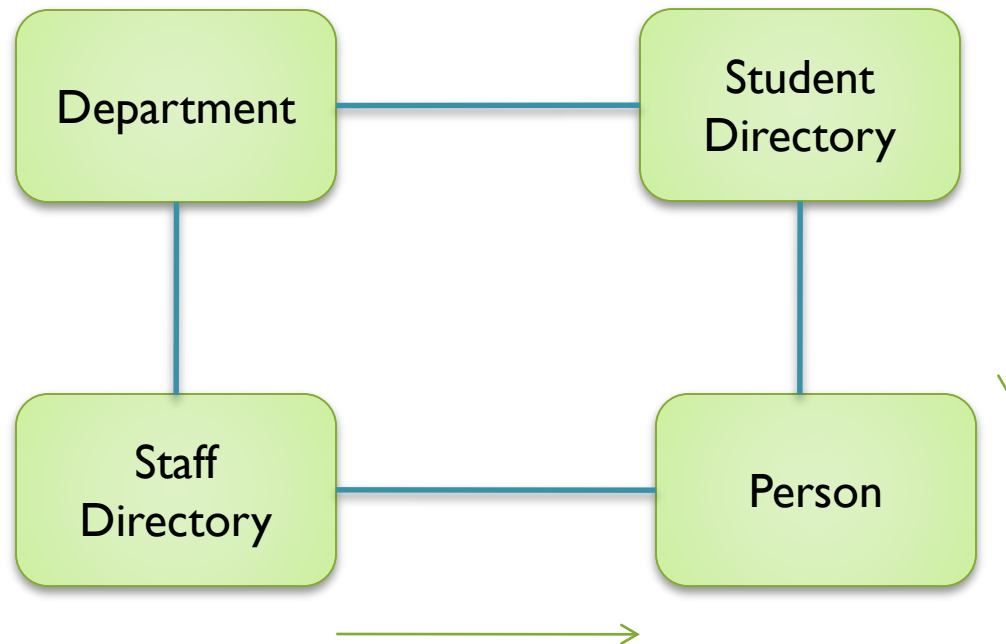
The same group of students but classified differently



Model Instances Example



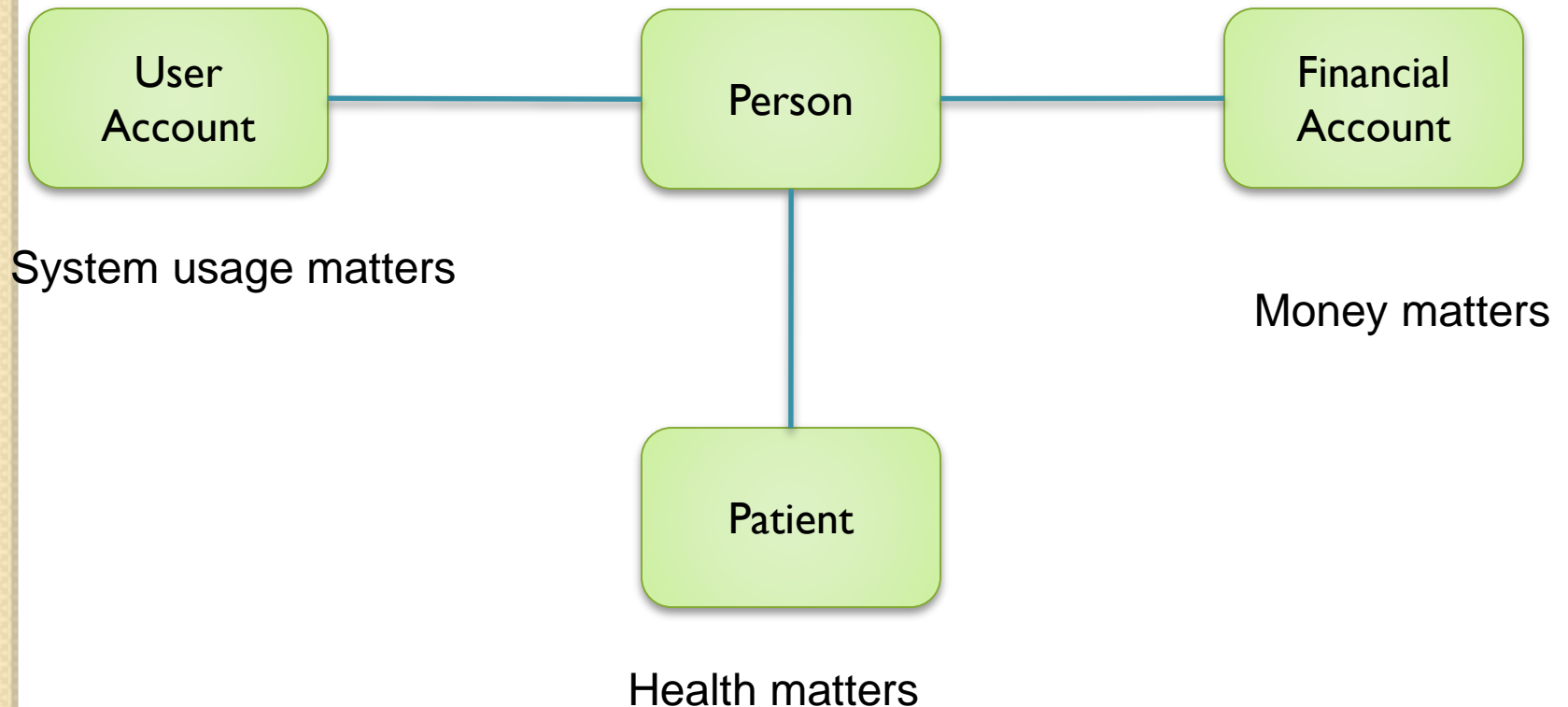
Directory is a way to group persons into categories or roles



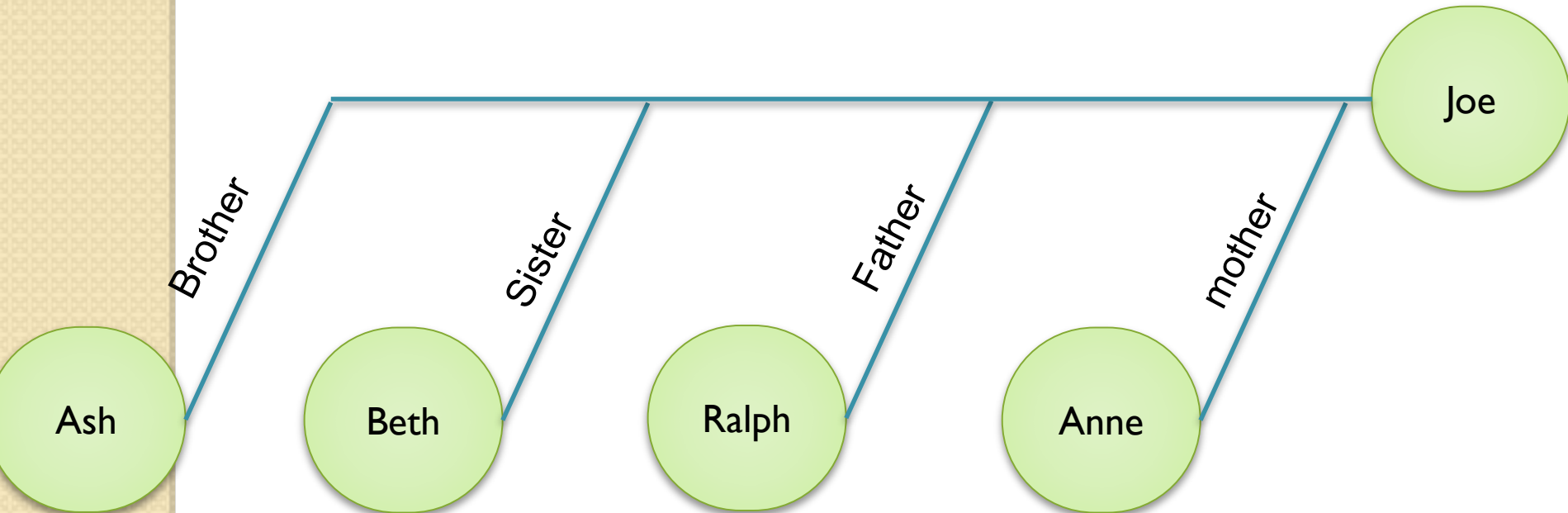
Other patterns

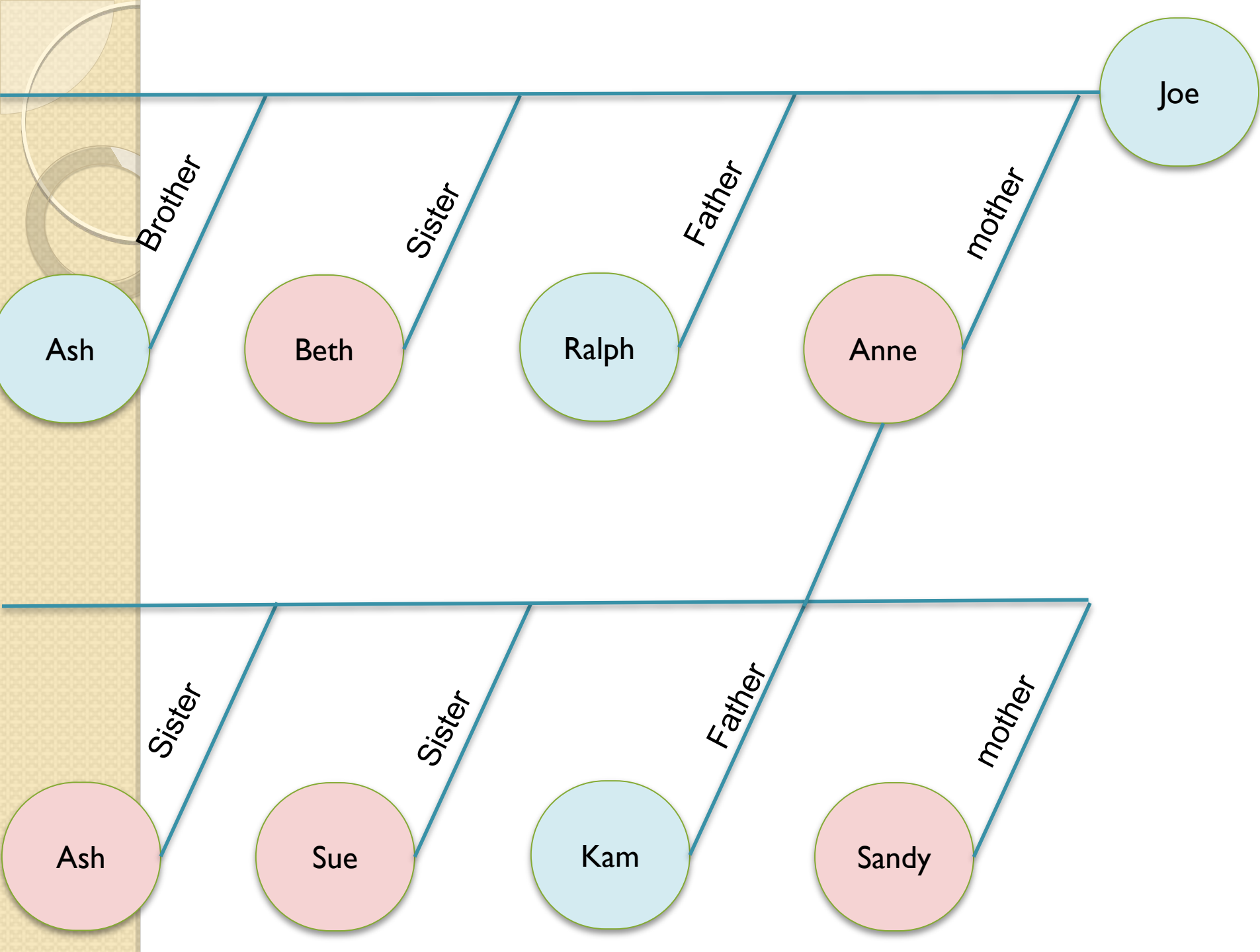


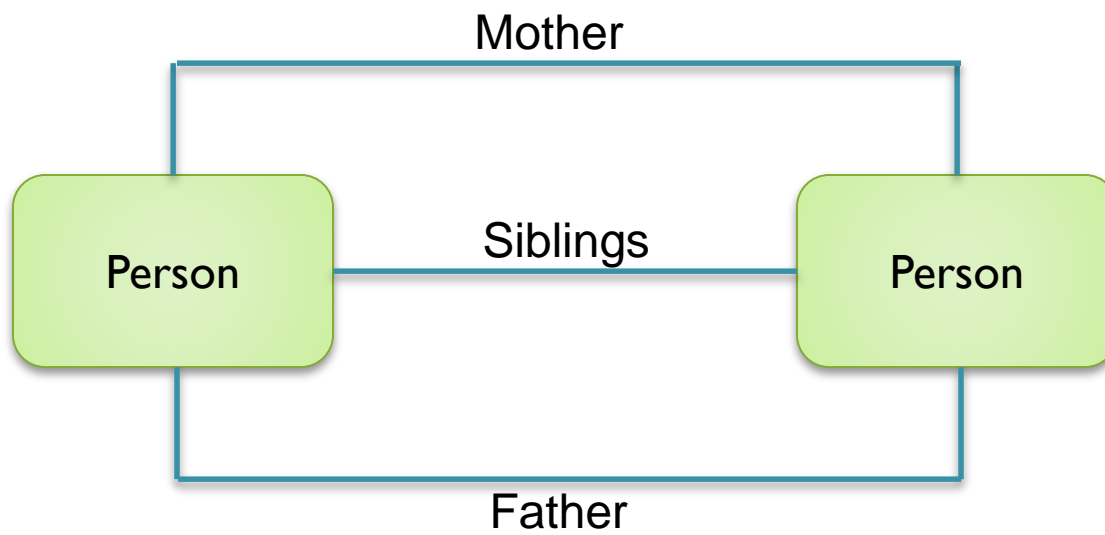
Decouple information (modularity)



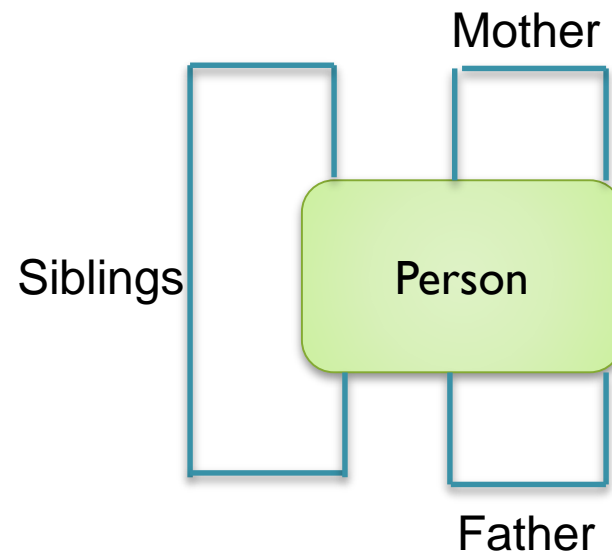
Person



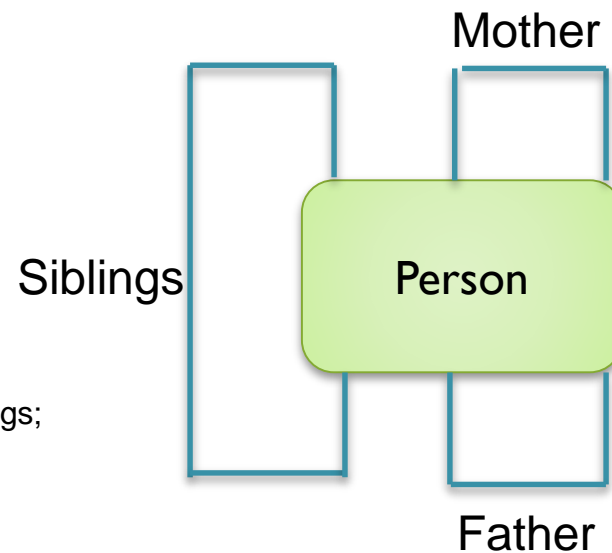




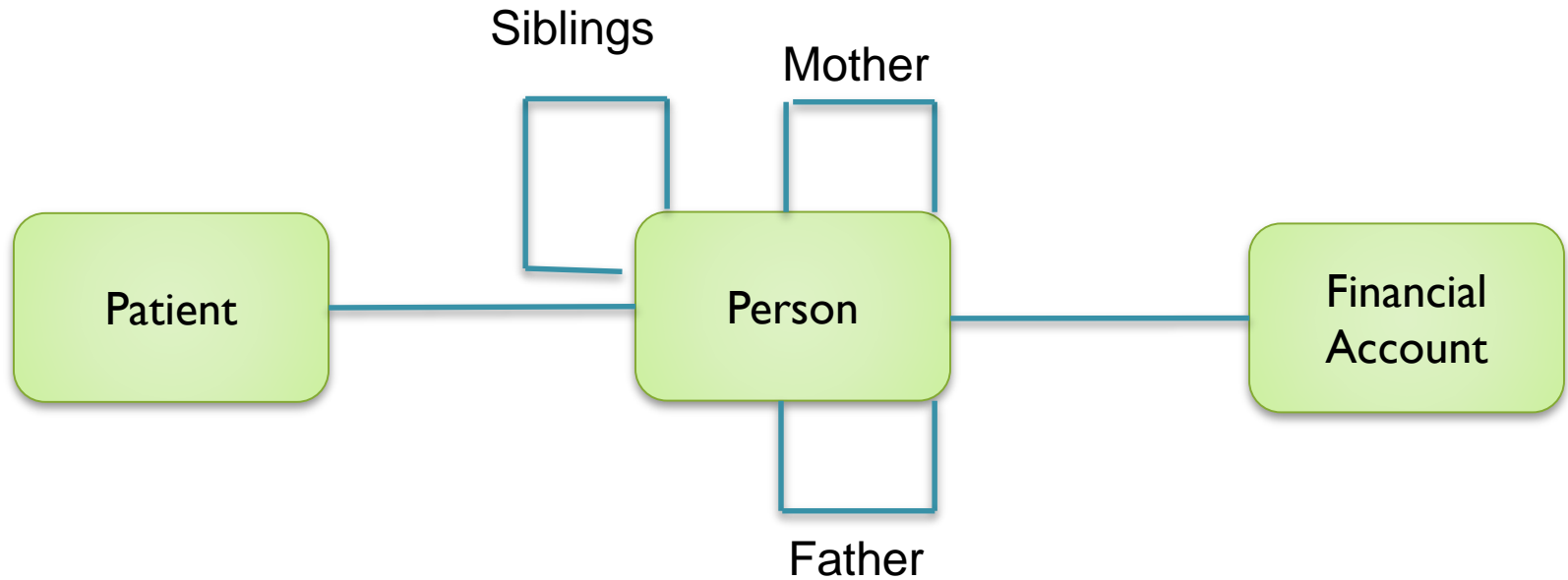
Recursive Representation



Recursive Representation



```
Class Person {  
  
    ArrayList <Person> siblings;  
    Person mother ;  
    Person father;  
  
:  
    Public void addSibling (Person p){}  
    Public void setFather (Person p){}  
    Public void setMother(Person p){}  
}
```

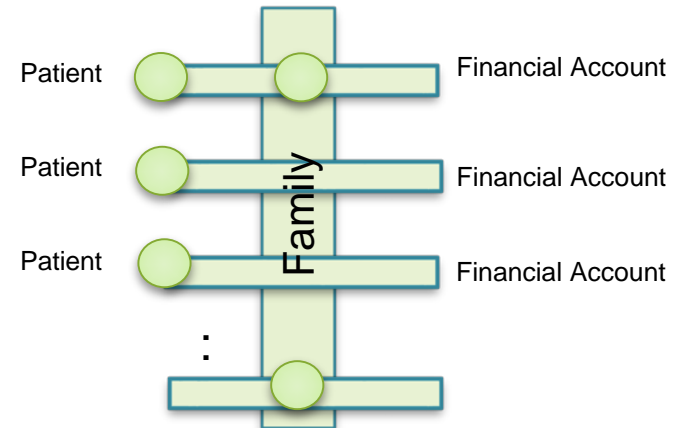


```

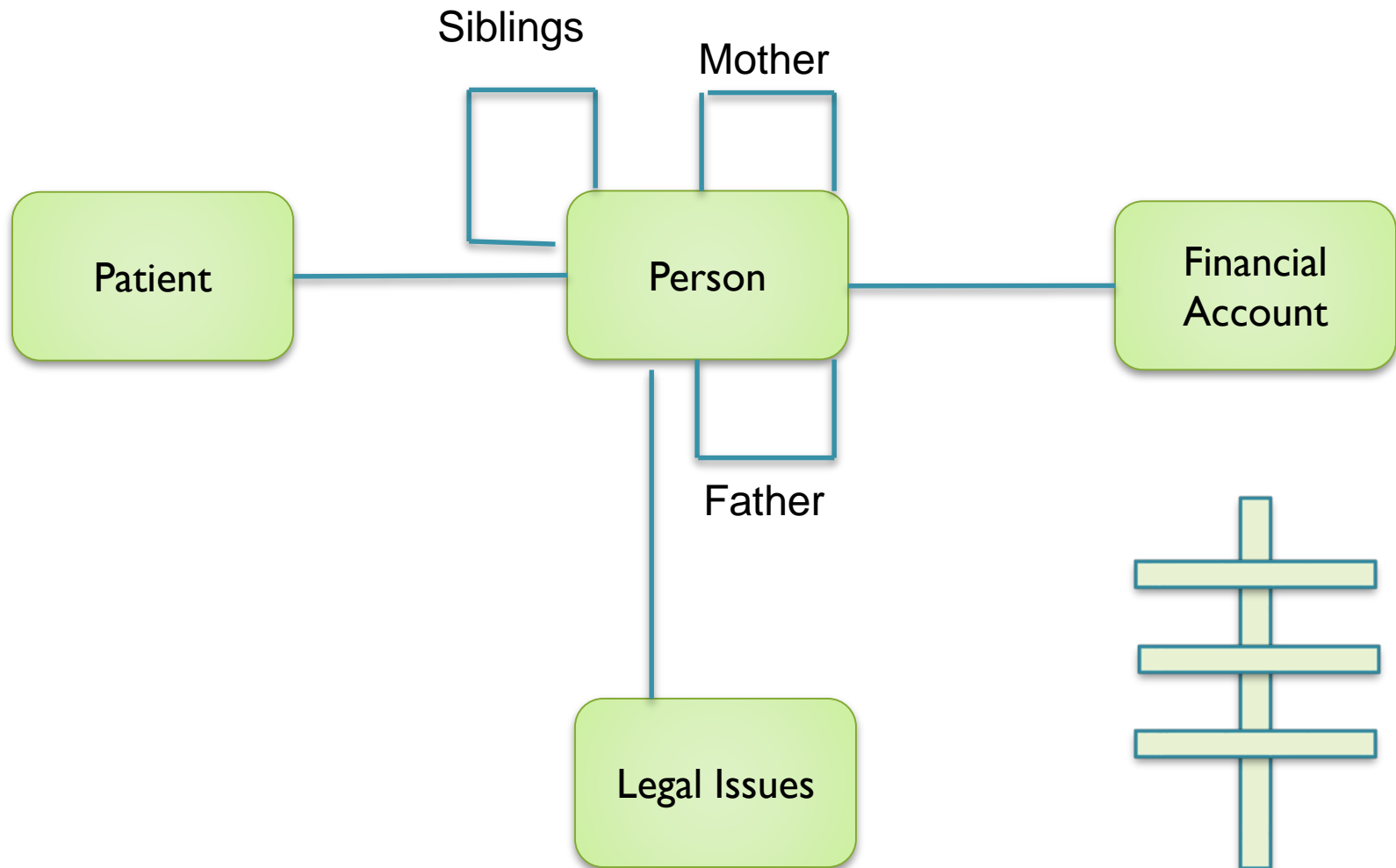
Class Person {
    ArrayList <Person> siblings;
    Person mother ;
    Person father;

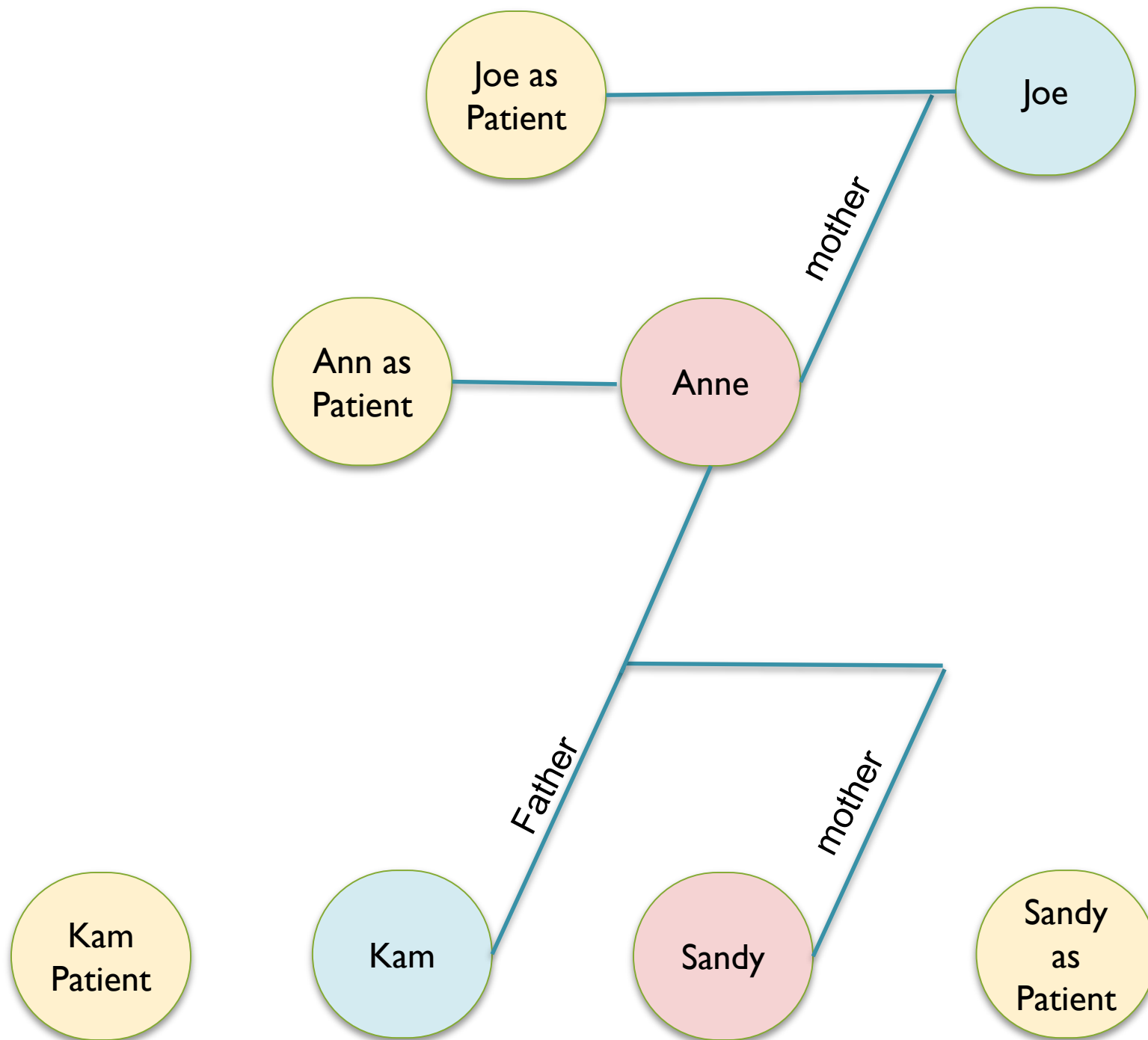
    FinancialAccount fn;
    Patient patient;

    :
}
  
```

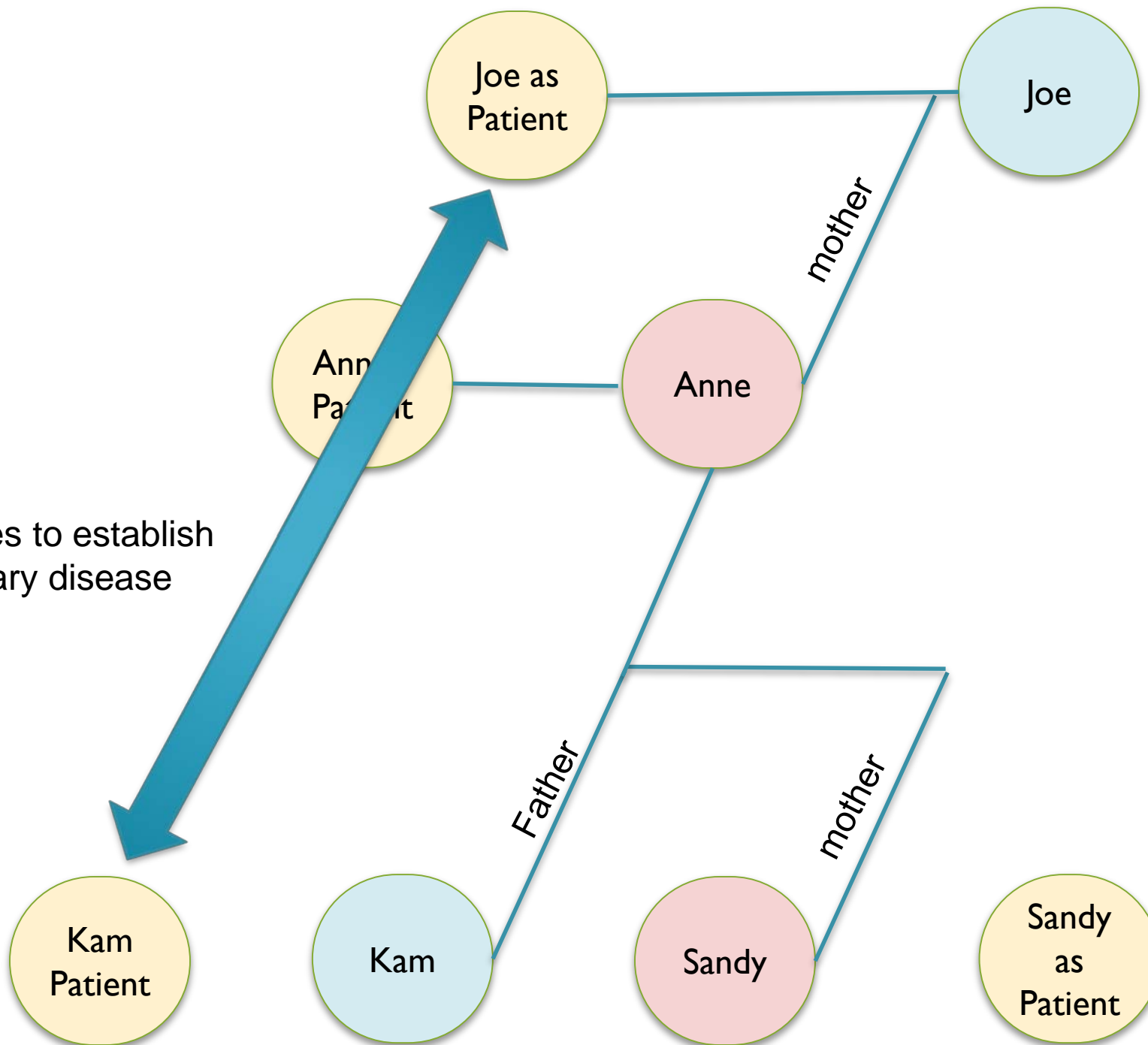


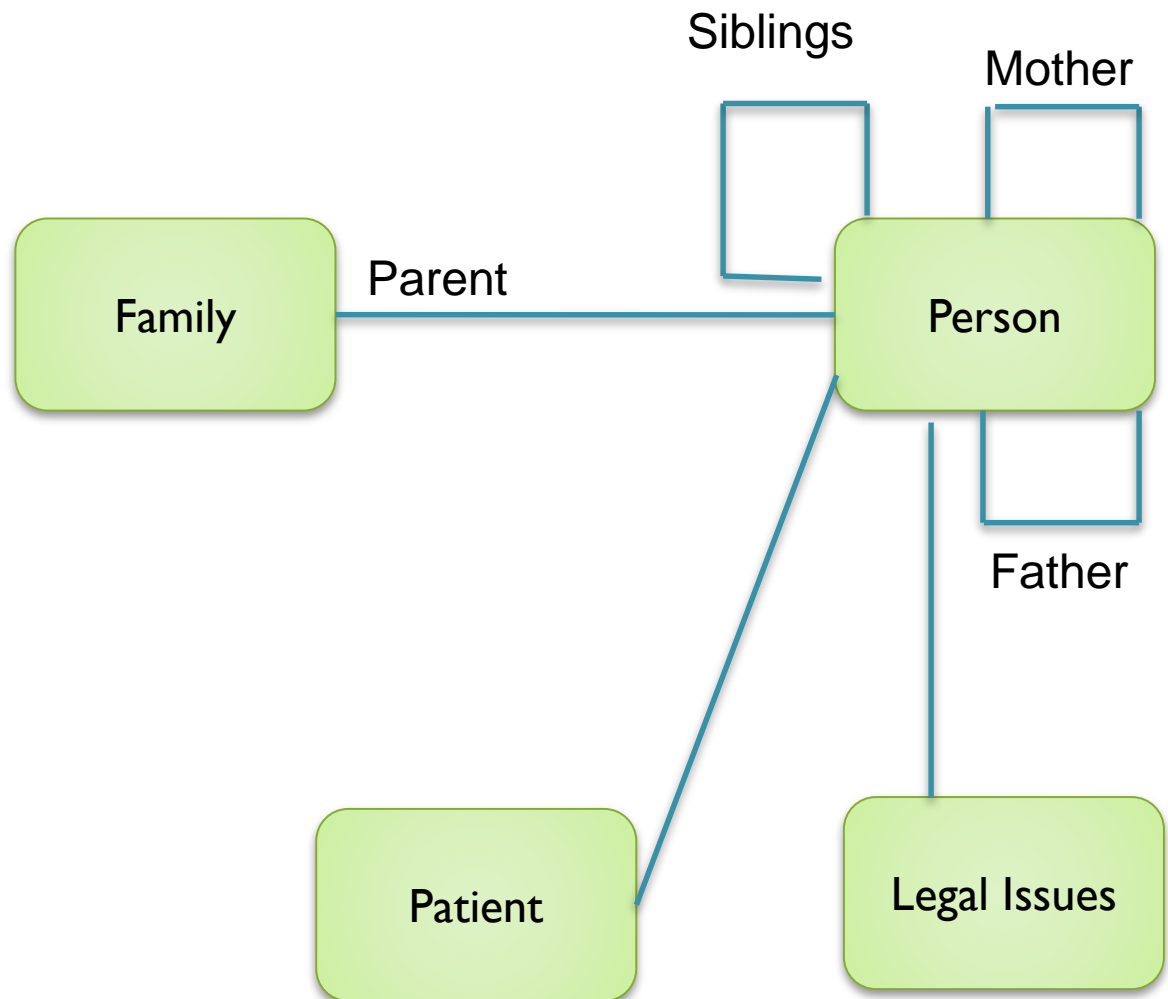
A way to establish connection between financial, legal, and health in poor communities – all as they relate to family problems

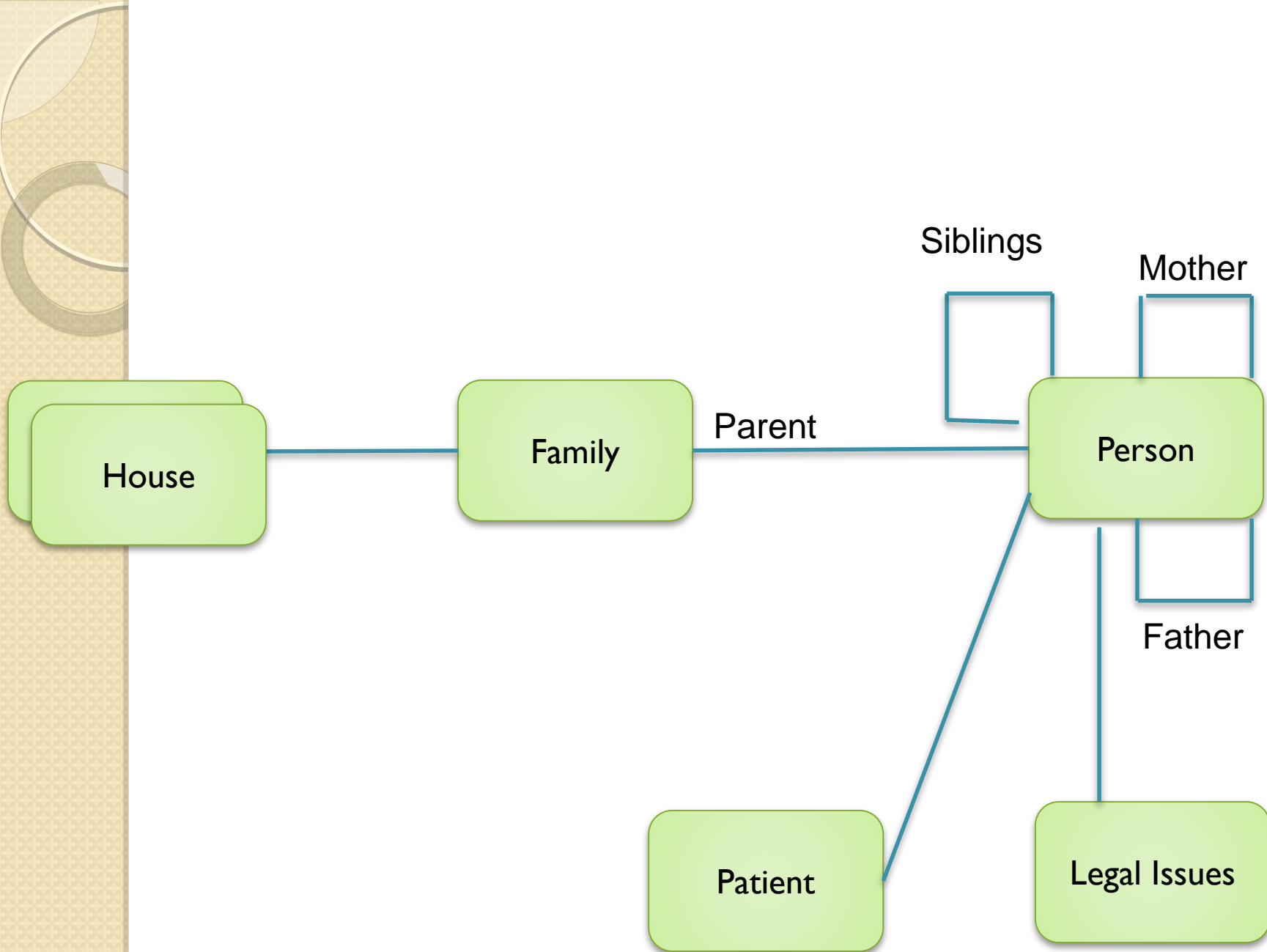


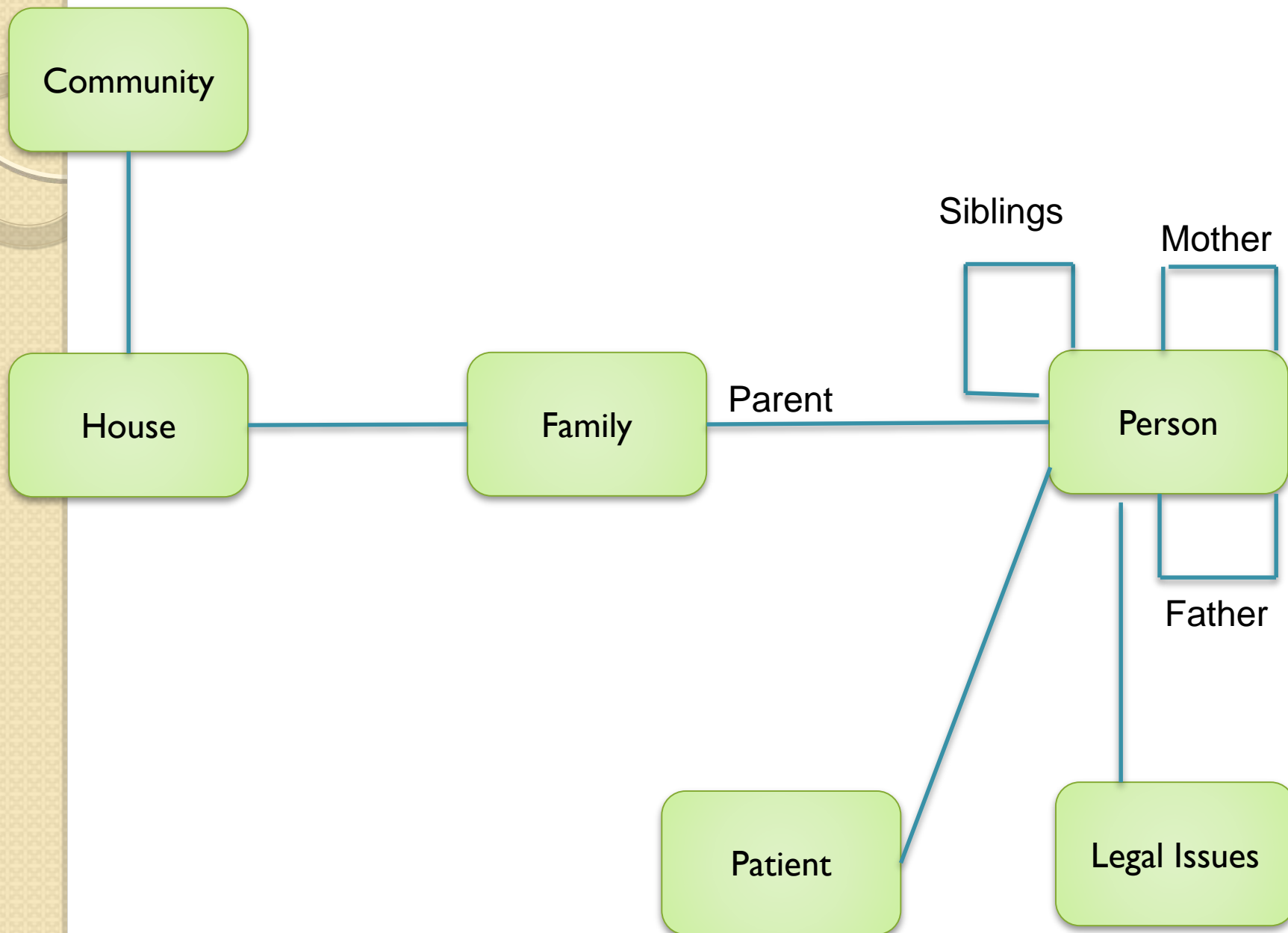


Linkages to establish
hereditary disease









City

Community

House

Family

Person





Java Implementation



The User-cases

1. Manage Person database

- Add a person
- Update a person
- Remove/Disable a person

2. Manage User Account Directory

- Add UserAccount
- Remove UserAccount
- Update UserAccount

Approach

- Define business, person and user directory classes
- Define user screens
 - Determine the input requirements for each user screen (is it business? person directory? person? Etc.
 - Remember: You must navigate from the root object, which is business in this case. For example `business.getPersonDirectory().findPerson(Id);`

Define Business Class

under the Business package

```
class Business {  
    Private String name;  
    Private PersonDirectory persondirectory; // a  
        reference variable that keeps track  
    Private UserAccountDirectory  
        useraccountsdirectory;
```

```
    public Business (String n) {  
  
        name = n; //the this operator means this business object.  
        persondirectory = new PersonDirectory ( ) ;  
        useraccountsdirectory = new  
            UserAccountDirectory( )
```

Define PersonDirectory Class

under the Business.HumanResources.PersonDirectory package

```
class PersonDirectory{
```

```
    private ArrayList<Person> personlist; // a reference  
        variable that keeps track
```

```
        // of the persons in one
```

```
        place
```

```
    public PersonDirectory() {
```

```
        personlist= new ArrayList();
```

```
    }
```

```
}
```

Define UserAccount Directory Class

under the Business.SystemAdministration package

```
class UserAccountDirectory{  
    private ArrayList<UserAccount> useraccountlist;  
        // a reference variable that keeps all
```

```
        // suppliers in one place
```

```
    public UserAccountDirectory() {
```

```
        useraccountlist = new ArrayList(); // make sure to  
        create the arraylist and assign
```

```
        variable
```

```
        // it to the reference
```

```
        created immediately as
```

```
        // the arraylist will be
```

```
        useraccount directory
```

```
        // you create the
```

Define MainJFrame Class

under the userinterface package

Create a global variable of type Business

```
class MainJFrame {  
    private Business business; // Global Variable
```

```
    public MainJFrame() {
```

```
        initComponents();
```

```
        business =
```

```
        ConfigureABusiness.Initialize("General  
Motors"); //notes we did not create an
```

```
instance of the class. //We executed a method  
on the class. That is the idea of //"static  
methods on classes"
```

Define Class called ConfigureABusiness

under the Business.BusinessConfiguration package

Create a global variable of type Business

```
class ConfigureABusiness{

    public static Business Initialize (String n) {
        // returns a business object

        <create a business>
        <add some persons>
        <add some user accounts>
    }
}
```

ConfigureABusiness

```
public static Business Initialize (String n) { // returns a business
object
```

```
Business b = new Business(n);
```

```
PersonDirectory pd = b.getPersonDirectory();
```

```
Person p = pd.newPerson(); //create person object
p.setFirstName("Ann");
p.setLastName("Wells");
```

```
:
```

```
p = pd.newPerson(); // create a second person object
p.setFirstName("John");
p.setLastName("Brown");
:
```

```
UserAccountDirectory uad= b.getUserAccountDirectory(); // prepare to
create user accounts
```

```
Person p2 = pd.findPersonByLastName("Brown");
```

ConfigureABusiness

```
public static Business Initialize (String n) { // returns a business object

    Business b = new Business(n);

    PersonDirectory pd = b.getPersonDirectory();

    Person p = pd.newPerson(); //create person object
    p.setFirstName("Ann");
    p.setLastName("Wells");

    :

    Person p = pd.newPerson(); // create a second person object
    p.setFirstName("John");
    p.setLastName("Adam");
    :

    UserAccountDirectory uad= b.getUserAccountDirectory(); // prepare to create user
    accounts

    Person  p2 = pd.findPersonByLastName("Brown");

    If (p2!=null){

    UserAccount ua = uad.newUserAccount();

    ua.setPerson(p2);           //link user account to the Mr. Brown
    ua.setUserId("jadam");
    ua.setPassWard("pw"); ua.setRole("System Admin")
```


Things you will need to do

- The person class must have a method toString as follows

```
Public String toString(){  
    return getFirstName() + "  
        "+getLastName();  
}
```

This will return first name followed by space and followed by last name

Things you will need to do

- The User account class must have a method toString as follows

```
public String toString(){  
    return getUserId() + "  
}
```

If you insert a user account object into a swing component, java swing will call the toString method on the object in order to display meaningful information to the user

UserAccountDirectory class

- You will need an isValidUser method on this class
- This method will take a user name and password and determine if the user exists in the directory database. If found it returns the user account if not it returns null

UserAccountDirectory class (contd.)

- `public UserAccount isValidUser (String userid, String pwd){`
- For each useraccount in the user directory do the following step
- If (useraccount id matches userid and useraccount password matches pwd) then return the useraccount object
- Otherwise continue until no more to check
- Return null if not found
- `}`

How to enter data into a combobox?

- Get the person directory
- Get the list of persons as an arraylist
- For each person in list of persons
 - Insert person into the combobox
 - Use `comboBox.addItem(person);`
- The `toString` method on the person class will display the name of the person in the combobox

How to determine the selected person in the combobox

- When there is an action performed even to create a user account, for example, retrieve the selected person object from the combobox using:
 - `Comboxname.getSelectedItem()`.
- Make sure to force the selected item to be a person because the combobox does not remember the object type.
- `selectedperson = (Person)comboxname.getSelectedItem();`