Security System for Patient DB

Final report for CSE367

Fei Gao, Dan Wang and Jin Ma
Computer Science & Engineering
The University of Connecticut
Storrs, CT 06269-3155
{fgao, dwang, jinma}@engr.uconn.edu

05/08/2001

ABSTRACT

With the development of middle technologies, people can take advantage of the resources spread through out their organizations and collaborative organizations around the world. This distributed resource environment makes the most of the available resources, at the mean time it poses the issue of security when people access these valuable resources through network because most middleware products do not have enough security. In this paper, a role-based access control security model is discussed. The main idea in this model is a resource can not be accesses by any user. An authenticated user can only access a resource based on the user's role and each role has specific privileges. This model is implemented in a CORBA-based distributed environment.

Contents

1 Introduction to CORBA

1.1 What is DRE?

A distributed resource environment (DRE) provides an infrastructure for distributed application components to interact with each other in a client/server paradigm. Distributed components (servers, databases, etc.) are defined as resources that publish services for use by clients.

1.2 What is CORBA?

Common Object Request Broker Architecture (CORBA) is the product of Object Management Group (OMG). It defines a DRE that allows distributed applications to interoperate with each other, application to application communication, regardless of whether they are written in Java, C++, or other languages, or where these applications reside.

CORBA uses an object-oriented approach for creating software components that can be reused and shared between applications. First, each service object presents each client object a well-defined interface to encapsulate its inner details through well-known IDL (Interface Definition Language), which reduces application complexity. Then, as the middleware, ORB (Object Request Broker) connects a client application with the service objects it wants to use. The client program does not need to know whether the service object, which it is in communication with, resides on the same computer or on a remote computer. The client program only needs to know the interface's name of the service object and understand how to use the interface. The ORB takes care of the details of locating the object, routing the request, and returning the result.

1.3 How CORBA Works

There are a large number of scenarios where CORBA could be applied. For example: A new printer can be connected to the network and announce its presence and capabilities. A client can then just send “print” command without having to be specially configured to use this printer. CORBA passes the request to this printer and then the printer executes the command. So how does it work? How can a client use the printer without knowing where is the printer? (Here where means the network address of the printer. Of course, the client knows the geographic location of the printer). Let’s take a look at the magic.

1.3.1 Components

Basically, in a CORBA environment, there are three kinds of players (see Figure 1).

- **Server**, which provides services, such as a printer
- **Client**, which uses services, such as printing files
- **CORBA Lookup Service**, which connects the server and the client

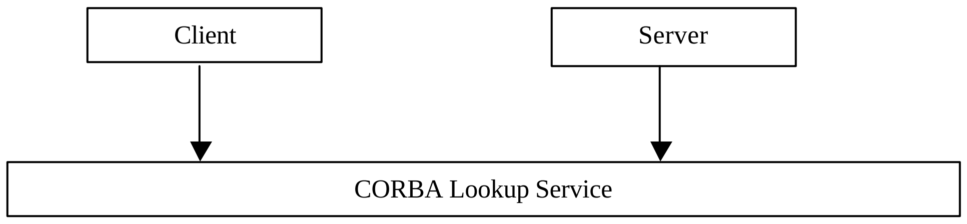


Figure 1: Components in a CORBA environment

1.3.2 Service Stub/Proxy and Skeleton

In order to know how CORBA works, first we need to know what are service stub/proxy and skeleton. They are actually two objects created by server. When the client finds the service, a stub object, not the real implementation object, is downloaded to the client; Afterwards, the client uses it to communicate back to the service skeleton, which will pass the request to the real service implementation. Finally, the running results are passed by the service skeleton to the service stub in client side. The stub is part of the service that is visible to clients, but its function is to pass method calls back to the server (see Figure 2).

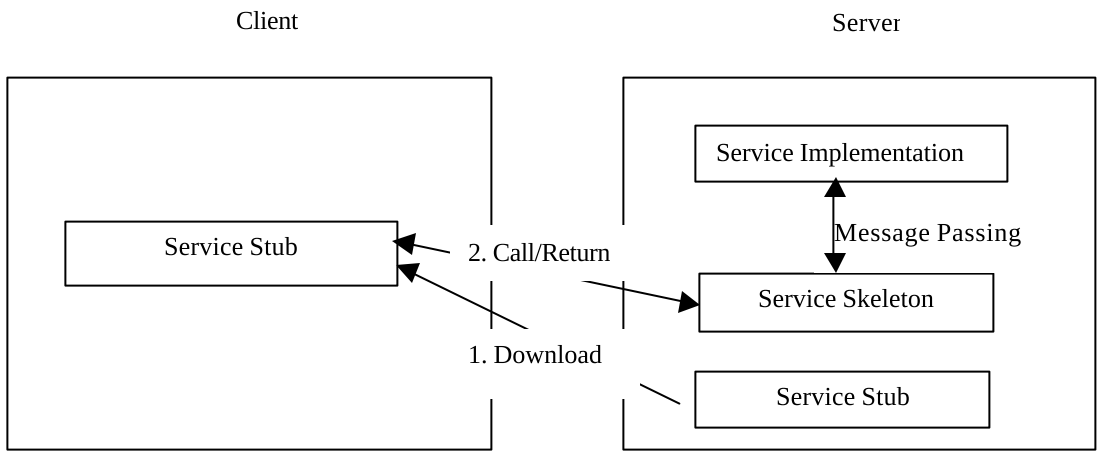


Figure 2: Service Stub/Proxy and Skeleton

1.3.3 Service Registration

Service is implemented, created and provided to the client by a server. A server can provide several services, such as printing, scanning. Client can also provide services, in such cases, client turns out to be a server. A server plays several roles as follows:

1. It implements and creates the service object, also generates service stub and skeleton.
2. It registers the service object with CORBA lookup services.
3. It executes the tasks and returns the result to the client

In order to register the service object with the CORBA lookup service, the server must first find the lookup service. This is done as follows: the server broadcasts lookup message around, if there already exists a CORBA bokup service, this CORBA lookup service responds to the server by sending its proxy object, so-called registrar, back to the server. It is through this registrar proxy that the server communicates with CORBA lookup service. After finding the lookup service, the server registers its service objects with the CORBA lookup service. This involves uploading the service proxy and storing it on the CORBA lookup service (see Figure 3).

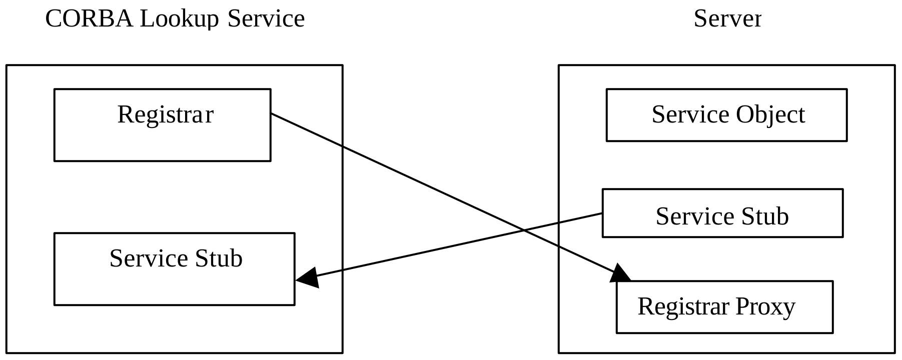


Figure 3: Service Registration

1.3.4 Client Lookup

The client on the other hand, goes through the same mechanism to get a registrar proxy from the lookup service. But this time it does something different from the server, which is to download the service stub from the CORBA lookup service and store it on the client (See Figure 4).

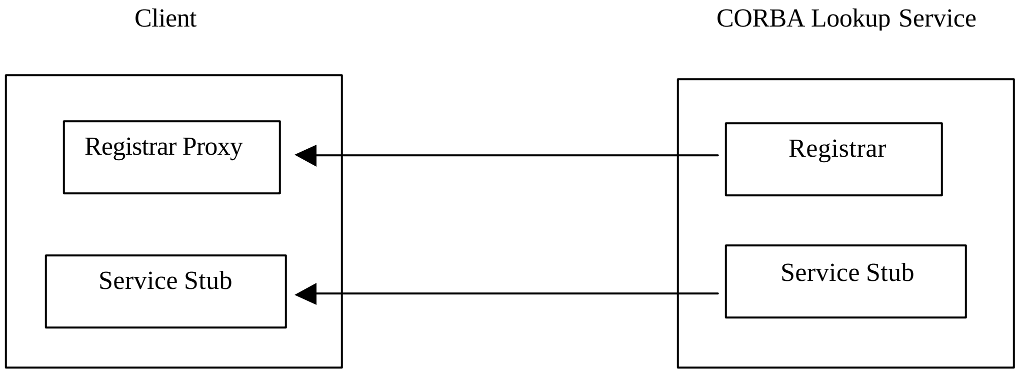


Figure 4: Client Lookup

1.3.5 Final Structure

Finally, after service registration and client lookup, the final structure is shown in Figure 5. The client gets the service stub, which it uses to invoke remote methods located on the server side. The server responds the client by sending the returning result of remote method invocation back to the client.

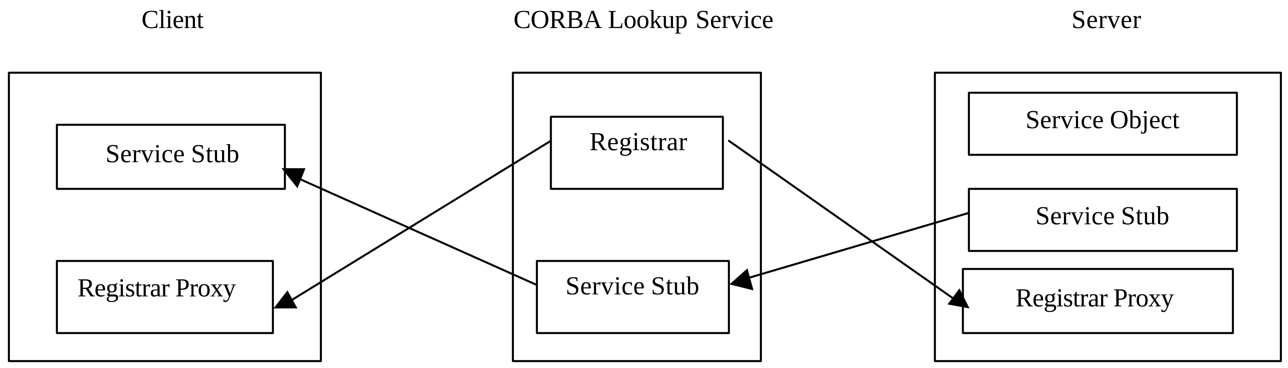


Figure 5: Final Structure

1.4 Advantages and Disadvantages

Actually, there exist some other middle wares that have similar functions as CORBA, such as JINI, RMI, DCOM, Servlet, etc. However, CORBA has its own advantages and disadvantages depicted as follows.

Advantages

- In CORBA environment, the client and the server don't need to know each other's location.
- It supports multi-language applications; the client can be written in Java, while the server in C++.
- It supports multi-platform applications; the client can be run in UNIX, while the server in NT.
- CORBA provides some other services, such as Naming Service, Event Service, etc

Disadvantages

- Its running speed is relatively slow.
- Its security mechanism is under development.

2 CORBA Naming Service

After investigating the mechanism of CORBA architecture, let's take look at one of the CORBA services – CORBA Naming Service. CORBA Naming Service is like the telephone yellow pages for objects. It relies on the CORBA main infrastructure but provides an efficient way of managing and locating objects with their names.

Names are used-defined values that identify objects. The naming service maps these use-defined names to object references. A name-to-object association is called a *name binding*. A Naming Service maintains a database of bindings between names and object references. You can reference a CORBA object using a sequence of names that form a hierarchical naming tree. In the figure, each dark node is a naming context. An object's name consists of a sequence of components that form a compound name. Only the leaf nodes are bound to implementations of objects (See Figure 6).

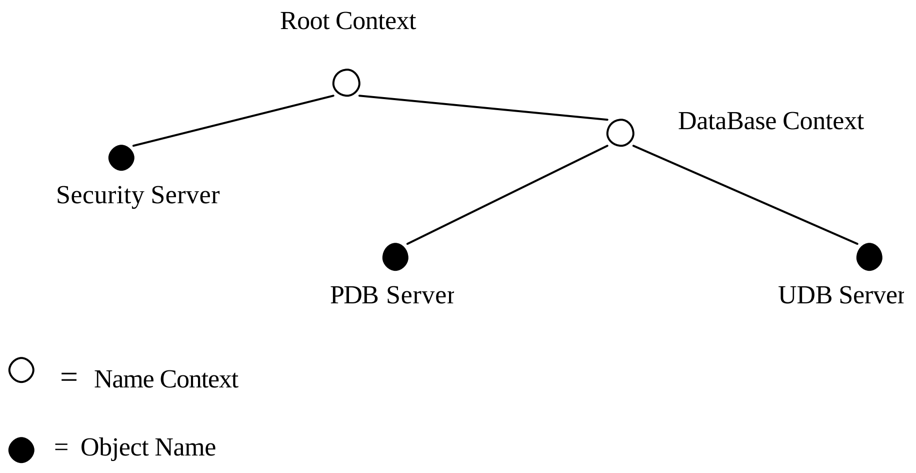


Figure 6: CORBA Objects Support Hierarchical

Each name component is a structure with two attributes: 1) *identifier* is the object's name string; 2) *kind* is a string in which you can put a descriptive attribute for your name. The Naming Service does not interpret, assign or manage these attributes in any way. They are used by higher levels of software.

Figure 7 shows a simplified view of the client/server naming interactions.

i) A server invokes bind to associate a logical name with an object reference.

ii) The Name Server adds this obj_ref/name binding to its namespace database.

iii) A client application invokes resolve to obtain an object reference with this name.

iv) The client uses the object reference to invoke methods on the target object.

So the Name Server serves both clients and servers. Servers export name/object bindings to the Name Server; clients then find these objects.

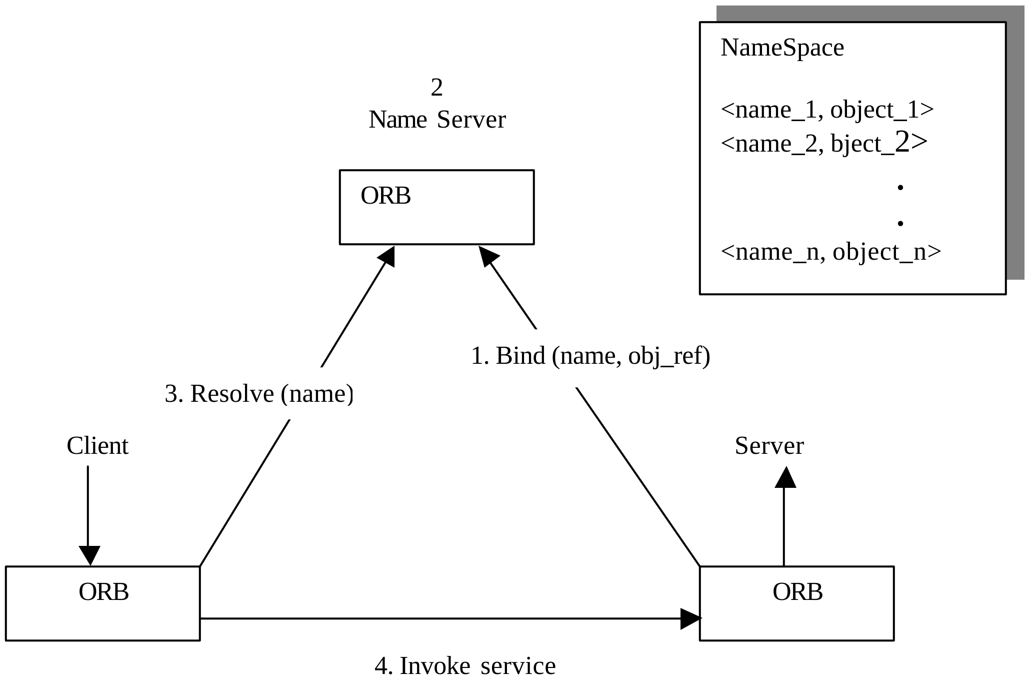


Figure 7: How Server and Clients interact with Name Server

The CORBA Naming Service will exemplify its importance when there are a large number of objects in a distributed environment because what fundamental to CORBA is how to locate and manage the objects in an effective and efficient way.

3 Security in DRE

Middleware is software that enables seamless client/server interactions in a distributed environment. It uses an object-oriented approach for creating software components that can be reused and shared among applications. With the development of middleware, people can take advantage of the distributed resources in local or global network area. The most commonly used resources are Databases, application servers. These distributed resources provide people with a cost-effective way to share information and service, at the meantime, they pose the problem of security. In our point of view, The main issue of using a distributed resource is who can use it, when the user can use it and where the user can use it and to what extent a user can use it.

In order to address the issue of security, a security system is a must for any distributed resource. The purpose of the security system is to:

Protect resources. Only trusted user can use the resource within some limit and the user can only access the resource from trusted IP addresses. Security system needs to authenticate the user before the user can access resource. When an authenticated user accesses the resource, the security system should also have control over the user's access. A user can only access the resource that the user is authorized to access. Figure 8 shows the basic structure of a client/server application with a security system.

Protect user information. As user's ID and password flow through network, they are vulnerable to tampering, the security system needs to provide ways to protect user information.

Protect data privacy and integrity. As data is sent back and forth from and to the resource server through network, it is under potential risk of being tampered or being altered. The security system needs to provide a safer way to transfer important data through network.

In this paper, a role-based security system is discussed. It focuses on the protection of resource. The main idea in the model is only the authenticated user can use a resource. The user can only use the resource based on the user's role. This model is implemented in a CORBA-based distributed environment.

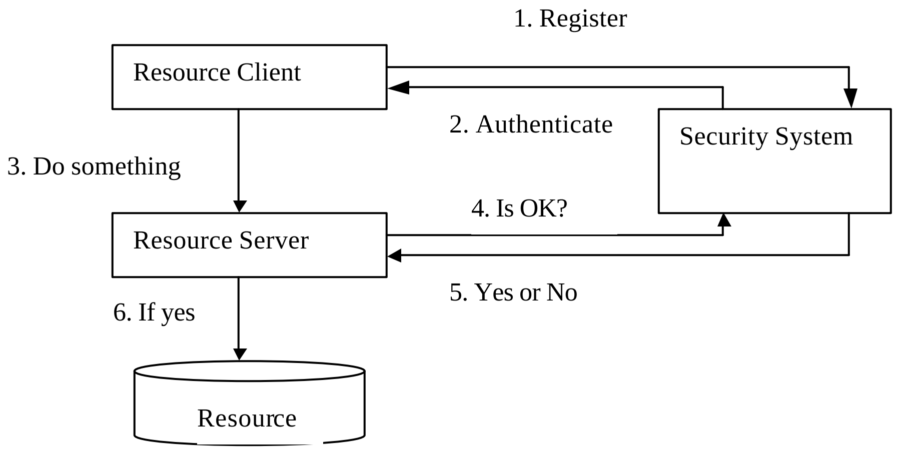


Figure 8: Client/server application with security system. The resource here can be a Database. If the client wants to access resource, he/she has to be authenticated by the security system, then sends request to resource server. The resource server will check with security system to get authorization then access resource on behalf of client's request. Here resource, resource client, resource server and security system are all distributed in the same or different networks.

4 Overview of the system

4.1 Components

This project consists of four major components: CORBA Lookup Service, Security System, Resource, and Resource Client. CORBA Lookup Service is just like a bridge, through which Security System, Resource and Client interact with one another over the network.

4.1.1 CORBA Lookup Service

CORBA Lookup Service connects all the other components so that they can communicate with each other over the network.

4.1.2 Security System

Security system consists of security server, security clients (including policy client and enforcement client) and security database.

(a) Security Server: Provides security service to other components. It has an attached security database that stores roles, users, etc.

(b) Security Database: Stores all the roles, users and other security data. Only Security Server can update security database.

(c) Policy Client: Creates roles, grants resources to roles and designates IP to roles.

(d) Enforcement Client: Creates users and grants roles to users.

4.1.3 Resource ---Patient DB

The resource provides services for use by the client. In our project, Patient DB Server and Patient Database provide a resource called “patient DB” to the client.

(a) Patient DB Server: Publishes its methods for client to invoke remotely. These methods are about writing prescription, querying patients, and so on.

(b) Patient Database: Stores all the patient information data. Only Patient DB Server can update Patient Database.

4.1.4 Resource Client---Patient DB Client

Patient DB GUI Client is the client, which provides a GUI for users to register, drop, and query courses.

4.2 Overall Structure

This project combines CORBA with role-based access control to build a security system. The overall structure is shown in Figure 7.

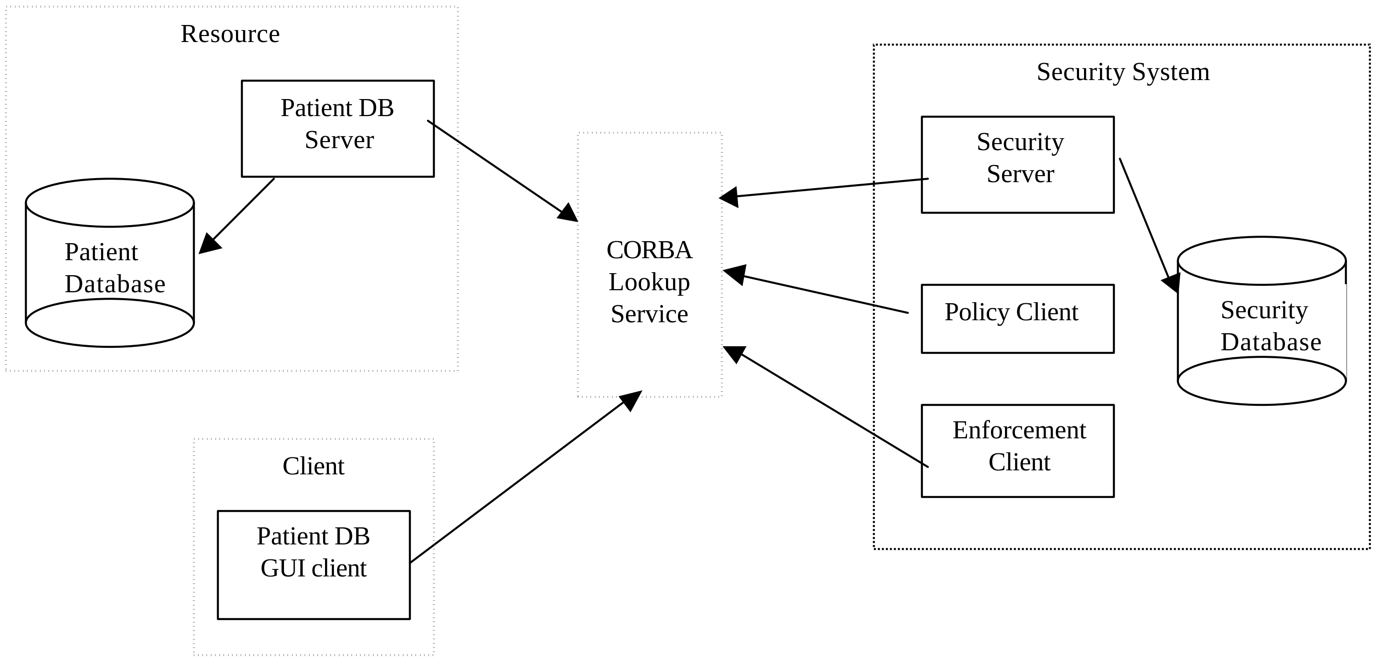


Figure 7: Overall Structure

5 System Functions

5.1 Basic Functions --- Interaction between components

Our security system for Patient DB is based on the original JINI security system for University DB, therefore it has implemented the basic functions, such as negative privileges, IP constraint (only supports * for now), time constraint---Token, 180-second time constraint for each login and permission check. The basic functions is described as follows.

5.1.1 Security Clients Access Security Server

Figure 8 shows how security clients, Policy Client and Enforcement Client, access Security Server.

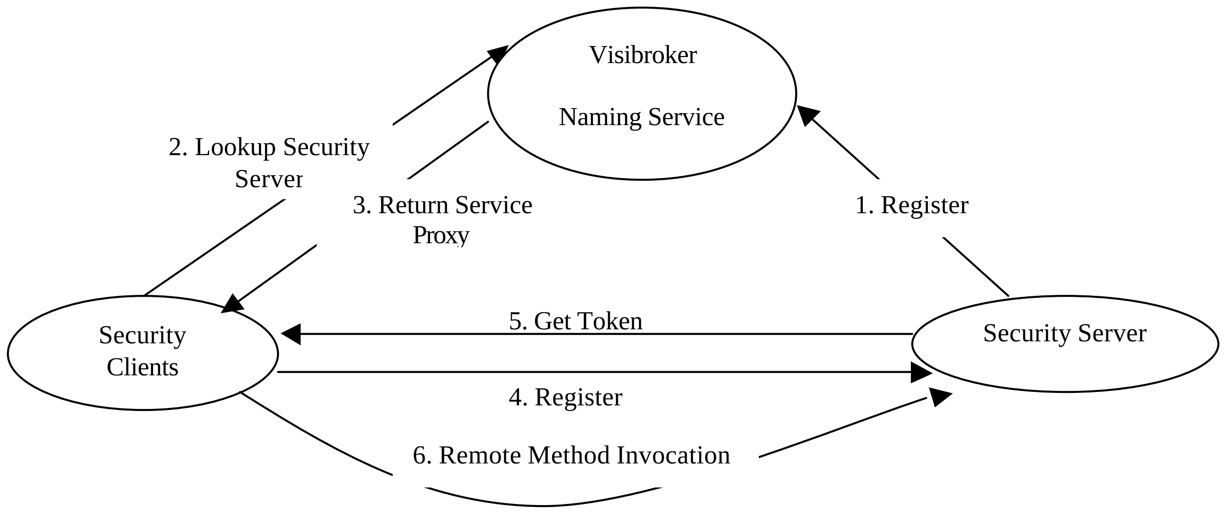
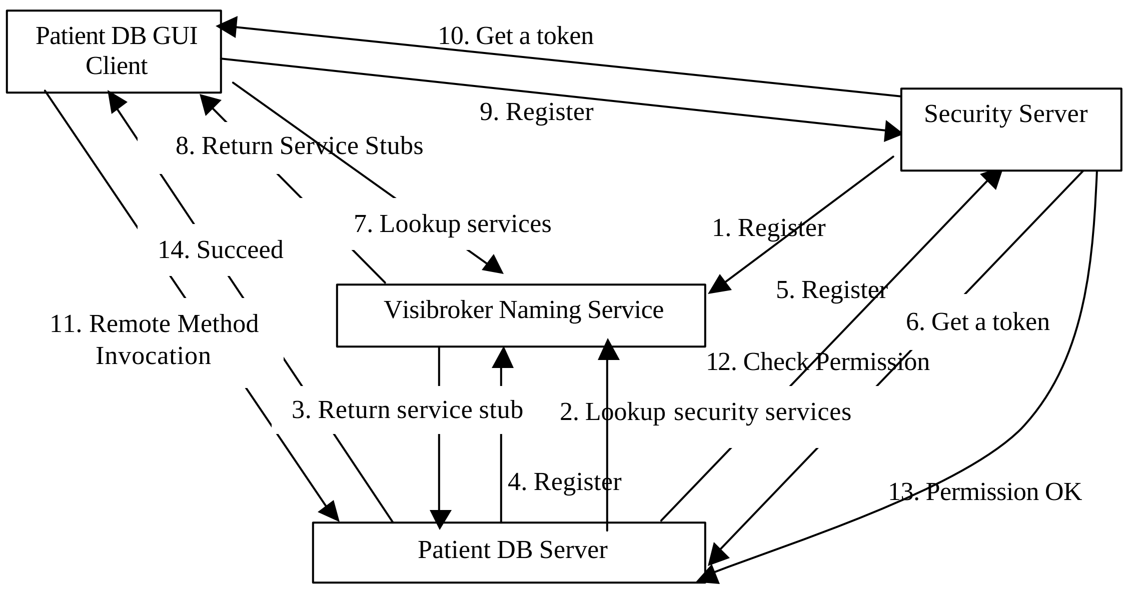


Figure 8: Security Clients Access Security Server

1. Security Server registers with Visibroker Naming Service to publish its services.
2. Security clients (Policy Client and Enforcement Client) look for security services in Visibroker Naming Service.
3. Visibroker Naming Service finds security services and returns the service stub to security clients.
4. Security clients register with Security Server by inputting user ID and password.
5. Security Server generates a token for each security client.
6. Holding a valid token, security clients remotely invoke the methods provided by security server to create roles, etc.
7. Security Server responds the invocations of each security client by modifying the security database and returning the result to each security client.

5.1.2 Patient DB Server and GUI Client Access Security Server

Figure 9 shows how Patient DB Server and GUI Client access Security Server.



1. Figure 9: Patient DB Server and GUI Client Access Security System rvices.
2. Patient DB Server looks for security services in Visibroker Naming Service.
3. Visibroker Naming Service finds security services and returns the service stub to Patient DB Server.
4. Patient DB Server registers with Visibroker Naming Service to publish its services.
5. Patient DB Server registers itself and its methods with Security Server as a resource by inputting user ID and password.
6. Security Server generates a token for Patient DB Server.
7. Patient DB GUI Client looks for security services and Patient DB services in Visibroker Naming Service.
8. Visibroker Naming Service finds services and returns the service stubs to Patient DB GUI Client.
9. Patient DB GUI Client registers with Security Server by inputting user ID and password.
10. Security Server generates a token for Patient DB GUI Client.

11. Holding a token, Patient DB GUI Client invokes remotely the methods provided by Patient DB Server.
12. Holding its own token and current client's token, Patient DB Server checks with Security Server if current client has permission to invoke current method.
13. Security Server responds Patient DB Server with the result of permission check.
14. Patient DB Server responds the invocations of current client by modifying the Patient database and returning the result to current client.

5.2 New Functions of Security Server

In order to make GUI friendlier, we added some query functions as follows:

```
/**
 *  Query all available resources from availres table
 */
public java.lang.String[] queryAvailResources ();

/**
 *  Query IDs of all available methods from availmethod table
 */
public java.lang.String[] queryAvailMethodIDs (java.lang.String arg0);

/**
 *  Query names of all available methods from availmethod table
 */
public java.lang.String[] queryAvailMethodNames (java.lang.String arg0);

/**
 *  Query descriptions of all available methods from availmethod table
 */
public java.lang.String[] queryAvailMethodDescs (java.lang.String arg0);

/**
 *  Query all resources from res table
 */
public java.lang.String[] queryAllResources ();

/**
 *  Query IDs of all methods from method table
 */
public java.lang.String[] queryAllMethodIDs (java.lang.String arg0);

/**
 *  Query names of all methods from method table
 */
public java.lang.String[] queryAllMethodNames (java.lang.String arg0);

/**
 *  Query descriptions of all methods from method table
 */
public java.lang.String[] queryAllMethodDescs (java.lang.String arg0);

/**
 *  Query all roles from role table
 */
public java.lang.String[] queryAllRoles ();
/**
 *  Query all users from users table
 */
public java.lang.String[] queryAllUsers ();

/**
 *  Query all tokens from token table
 */
public java.lang.String[] queryAllTokens ();
```

5.3 New functions of Security Clients

- We changed the appearances of all the panels, including layout, borders and so on.
- Drop down menus are added for selection of methods, resources, roles and users

5.4 New functions of Patient DB Server

We designed the architecture of Patient DB and methods depicted as follows.

5.4.1 Patient DB

```
table patient(
        patientID       varchar2(10)   not null,
        patientName     varchar2(30)   not null,
        primary key(patientID));
```

```
table medicalHistory(
        histID          integer             not null,
        patientID       varchar2(10)   not null,
        userID          varchar2(10)        not null,
        diagnosis       varchar2(100)       not null,
        time            date                not null,
        primary key(histID));

table prescription(
        presID          varchar2(10)        not null,
        patientID               varchar2(10)            not null,
        userID          varchar2(10)        not null,
        description     varchar2(100)       not null,
        time            date                not null,
        primary key(presID));

table payment(
        paymentID       varchar2(10)        not null,
        patientID       varchar2(10)        not null,
        userID          varchar2(10)        not null,
        description     varchar2(100)       not null,
        time            date                not null,
        primary key(paymentID));
```

5.4.2 Methods of Patient DB Server

```
/**
 * ifHasRight(...) tells if a user has right for a method based on the methods' id
 */
public boolean ifHasRight (int token,  int methodID);

/**
 * method 0,  getPatientMedicalHistory(...)
 * query PDB medicalHistory table
 * returns the medical history of the queried patient
 * only doctors and nurses have the right to do so.
 */
public java.lang.String[][] getMedicalHistory (int token,
                                        java.lang.String patientID);

/**
 * Method 1, getDiagnosis(...)
 * Querys PDB medicalHistory table
 * Returns all the diagnosis of the queried patient
 * Only doctors and nurses have the right to do so.
 */
public java.lang.String[][] getDiagnosis (int token,
                                java.lang.String patientID);

/**
 * Method 2,  getPrescription(...)
 * Querys PDB prescription table
 * Returns the prescription history of the queried patient
 * Only doctors and nurses have the right to do so.
 */
public java.lang.String[][] getPrescription (int token,
                                  java.lang.String patientID);

/**
 * Method 3, getPaymentMode(...)
 * Querys PDB payment table
 * Returns the payment history of the queried patient
 * Only accountants have right to do so
 */
public java.lang.String[][] getPaymentMode (int token,
                                  java.lang.String patientID);

/**
 * method 4 getPatientList(...)
 * get the patient list, which include the patient ID and patient name
 * returns the list of patients
 * only accountants have right
 */
public java.lang.String[][] getPatientList (int token,
                                  java.lang.String patientID);

/**
 * Method 5 writeDiagnosis(...)
 * Writes diagnosis for a patient into PDB medicalHistory table
 * Returns true if successfully done, false if faild
 * Only doctors have the right.
 */
public boolean writeDiagnosis (int token,
                        java.lang.String userID,
                        java.lang.String patientID,
                        java.lang.String diagnosis,
                        java.lang.String time);

/**
 * Method 6 writePrescription(...)
 * Writes prescription for a patient into PDB prescription table
 * Returns true if successfully done, false if faild
 * Only doctors have the right.
 */
public boolean writePrescription (int token,
                        java.lang.String userID,
                        java.lang.String patientID,
                        java.lang.String description,
                        java.lang.String time);

/**
* Method 7 setPaymentMode(...)
 * Writes payment of a patient into PDB payment table
 * Returns true if successfully done, false if faild
 * Only accountants have the right.
 */
public boolean setPaymentMode (int token,
                      java.lang.String userID,
                      java.lang.String patientID,
                      java.lang.String mode,
                      java.lang.String time);
/**
 * Method 8, addPatient(...)
 * Adds a patient into PDB patient table
 * Returns true if successfully done, false if faild
 * Only accountants have the right
 */
public boolean addPatient (int token,
                   java.lang.String patientID,
                   java.lang.String patientName);


/**
 * Method 9 removePatient(...)
 * Deletes a patient from PDB patient table
 * Returns true if successfully done, false if failed
 * Only accountants have the right.
 */
public boolean removePatient (int token,
                              java.lang.String patientID,
                               java.lang.String patientName);
```

5.5 New functions of Patient DB Client

- We changed the appearances of all the panels, including layout, borders and so on.

- Menu items become disabled if current user doesn't have right to perform the corresponding operations.

- Get roles of current user based on user ID and password when authenticating.
- Change role/user without restarting the system

5.6 Implementation of Component Functionality

In this project, we use Visibroker for Java 4.5 and its Naming Service as CORBA Lookup service. And we use Oracle 8.1.7 as our database server, because they are both free. We installed LINUX Mandrake 2.2.17-21 in a 2-processor DELL desktop and installed Visibroker and Oracle in LINUX. Our JAVA-written server and client applications can run anywhere. Basically, we run it under NT 4.0. The functionality of each components is described as follows.

5.6.1 Security Policy Client

- Register/unregistered Patient Database server and its methods, such as getPatientMedicalHistory (token,patient_id), getPatientPrescription(token,patient_id), AddPaitent (token,patient_id).
- Add time constrain on resource.
- Create/erase role, such as create doctor, nurse and accountant roles.
- Grant/revoke resource and methods to roles, such as granting doctor the privilege of writePrescription (token, patient_id).
- Designate/revoke IP address to/from each role, so users can only access the resource from certain IP addresses.

5.6.2 Security Enforcement Client

- Create/erase users
- Add time constraint to each user

- Grant/revoke roles to/from each user.
- Grant/revoke negative resources/methods to/from users.

5.6.3 Security Server

- Verify the identity of a user.
- Check if Patient Database is still active.
- Check client's request based on the client's role and privileges, which includes the time constrain check, IP address check, role-based resource check (Does the client's role has right to use the resource?), role-based method check (Does the client's role has right to use the method?)

5.6.4 Patient Database Server

- Listens to the client's request such as query patient Database for medical history, patient lists, and writes prescription for a patient into patient Database.
- Sends client's role, IP address, request time, requested resource ID and method ID to security server for permission check.
- Listens the check result from security server.
- Fulfills the client's request if the server response is positive, it will query or update the patient Database upon client's request and sends result to client.
- Refuses the client's request if the server response is negative.

5.6.5 Patient Database Client

- Provides the GUI to users to access Patient Database Server.
- Change user or role without exiting the system.
- Whether menu items are enabled depends on the privileges of current user.

6 Conclusion and Future Work

Above all, we have presented what we have learned and done this semester. Our project, which incorporates the security using the role-based access control approach into CORBA, has proved that we can use CORBA to realize the role-based security.

6.1 Problems encountered

- Java class java.util.Vector is hard to transfer over network by Visibroker, so it is changed to java.lang.String[] or java.lang.String[][].
- Use rebind() instead of bind() to bind Visibroker Naming Service

6.2 Recommendations for improvements

- Change Change User/Role menu item to two menu items: Change role and Relogin
- Remove User ID item from PDB Client Update Panel
- Change Time to Date in PDB Client Update Panel
- Change Date to drop down menu instead of typing in.
- Remove method number from drop down menu.

6.3 Suggestions for future work

- Use CORBA to realize current JINI version, since CORBA is much faster than JINI. However, we only have 60-Day trial version Visibroker for Java 4.5.
- Try to use only one security server for PDB and UDB.
- Try to add service to (resource, method) pair. Currently version, the service is ignored.
- Try to incorporate JINI and CORBA.

- Try to use a common resource to test cooperation of JINI and CORBA.

Appendix A: User Manual

1 Instruction on how to setup the system

1.1 Software Requirements

- OS: Linux, NT
- CORBA: Visibroker for JAVA 4.5
- JAVA 1.2.2 or higher
- Oracle 8.1.7

1.2 Hardware Requirements

- Pentium II or higher
- 64M RAM or higher

1.3 CORBA Installation

- Download Visibroker for JAVA 4.5 from http://www.borland.com
- Install Visibroker for JAVA 4.5 in Linux

1.4 JAVA Installation

- Download Java 1.3 from http://java.sun.com
- Install Java 1.3 in Linux and NT

1.5 Oracle Installation

- Download Oracle 8.1.7 from http://www.oracle.com

- Install Oracle 8.1.7 in Linux
- Download Oracle driver for JAVA: classes12.zip

1.6 Security System Installation

- Security Server: classes/corbass/sserver/*.class and lib/classes12.zip
- Policy Client: classes/corbass/policy/*.class
- Enforcement Client: classes/corbass/policy/*.class
- Patient DB Server: classes/corbass/pdbserver/*.class and lib/classes12.zip
- Patient DB Client: classes/corbass/pdbclient/*.class

Note: all of above need classes/corbass/common/*.class

2 Instruction on how to run the system

2.1 Directory and File Specifications

- classes: all the class files
  - ❑ corbass.common
  - ❑ corbass.enforce
  - ❑ corbass.pdbclient
  - ❑ corbass.pdbserver
  - ❑ corbass.ssever
  - ❑ batch files for running the system
    - i) rp.bat: for running policy client
    - ii) re.bat: for running enforcement client
    - iii) ss.bat: for running security server

iv) pdbc.bat: for running pdb client

v) pdbs.bat: for running pdb server

- lib: <u>classes12.zip</u> --- Oracle driver for JAVA
- src: Source Code
  - ❑ corbass.common
  - ❑ corbass.enforce
  - ❑ corbass.oracle: SQL files for creating security and patient database
  - ❑ corbass.pdbclient
  - ❑ corbass.pdbserver
  - ❑ corbass.ssever

2.2 Running Steps

- o Start Oracle
- o Create Patient DB and Security DB
- o Start Visibroker OSAgent
- o Start Visibroker Naming Service
- o Start Security Server
- o Start Patient DB Server
- o Start all the other clients

2.3 Commands of running the system

2.3.1 Starting Oracle

- o Login dachshund.engr.uconn.edu as root
- o <u>Run /etc/init.d/dbora</u>

- Run lsnrctl start

2.3.2 Creating Database

- Login dachshund.engr.uconn.edu as a normal user
- Run sqlplus
- Input id and password
- Run start security.sql
- Run start pdb.sql
- exit

2.3.3 Starting the Visibroker

- Login dachshund.engr.uconn.edu as a normal user
- Start OSAgent: osagent &
- Start Naming Service: vbj –VBJclasspath /home/local/vbroker/lib/vbjorb.jar com.inprise.vbroker.naming.ExtFactory NameService &

2.3.4 Starting Security Server

In NT:

- ss

  OR
- start vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService -VBJclasspath ..\lib\classes12.zip corbass.sserver.SecurityServer

In Unix:

- vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService -VBJclasspath ..\lib\classes12.zip corbass.sserver.SecurityServer &

2.3.5 Starting Security Patient DB Server

In NT:

- pdbs

  OR

- start vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService -VBJclasspath ..\lib\classes12.zip corbass.pdbserver.PDBServer

In Unix:

- vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService -VBJclasspath ..\lib\classes12.zip corbass.pdbserver.PDBServer &

2.3.6 Starting Policy Client

In NT:

- rp

  OR

- start vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService corbass.policy.PolicyClient

In Unix:

- vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService corbass.policy.PolicyClient &

2.3.7 Starting Enforcement Client

In NT:

- re

  OR

- start vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService corbass.enforce.EnforceClient

In Unix:

- vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService corbass.enforce.EnforceClient &

2.3.8 Starting Patient DB Client

In NT:

- pdbc

OR

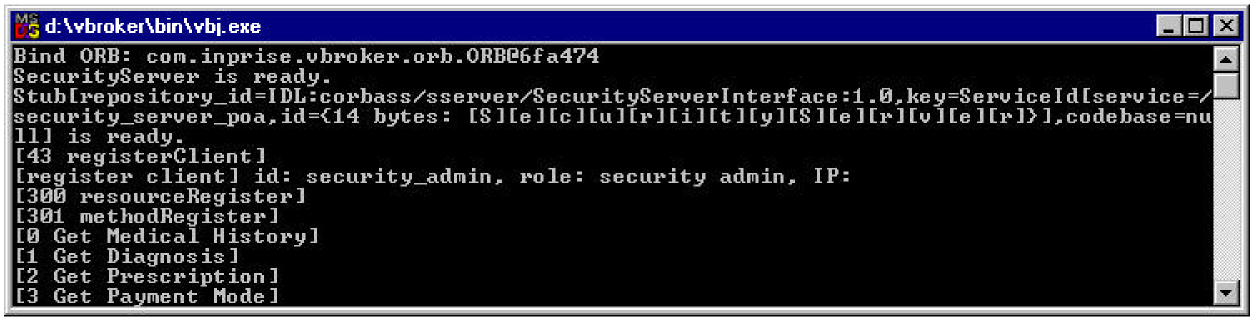
- start vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService corbass.pdbclient.PDBClient

In Unix:

- start vbj -DORBagentAddr=137.99.10.209 -DSVCnameroot=NameService corbass.pdbclient.PDBClient &

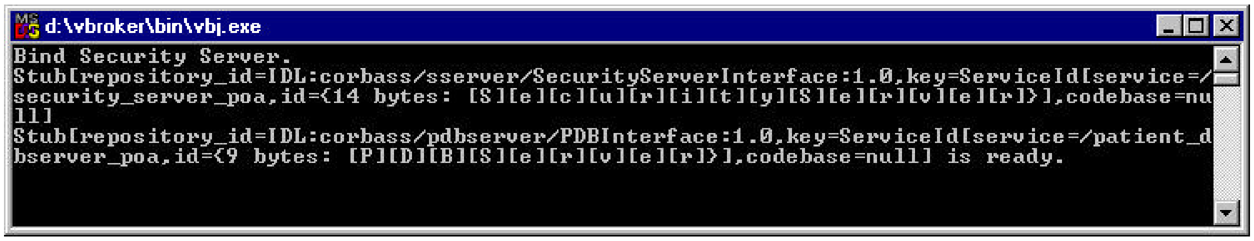
Appendix B: System Demo with Screen Shots

1. Security Server



```
Bind ORB: com.inprise.vbroker.orb.ORB@6fa474
SecurityServer is ready.
Stub[repository_id=IDL:corbass/sserver/SecurityServerInterface:1.0,key=ServiceId[service=/
security_server_poa,id={14 bytes: [S][e][c][u][r][i][t][y][S][e][r][v][e][r]}],codebase=nu
ll] is ready.
[43 registerClient]
[register client] id: security_admin, role: security admin, IP:
[300 resourceRegister]
[301 methodRegister]
[0 Get Medical History]
[1 Get Diagnosis]
[2 Get Prescription]
[3 Get Payment Mode]
```

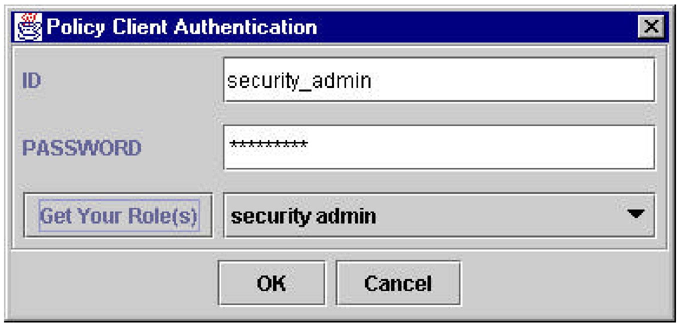
2. Patient DB Server



```
Bind Security Server.
Stub[repository_id=IDL:corbass/sserver/SecurityServerInterface:1.0,key=ServiceId[service=/
security_server_poa,id={14 bytes: [S][e][c][u][r][i][t][y][S][e][r][v][e][r]}],codebase=nu
ll]
Stub[repository_id=IDL:corbass/pdbserver/PDBInterface:1.0,key=ServiceId[service=/patient_d
bserver_poa,id={9 bytes: [P][D][B][S][e][r][v][e][r]}],codebase=null] is ready.
```

3. Policy Client

3.1 Authentication Dialog

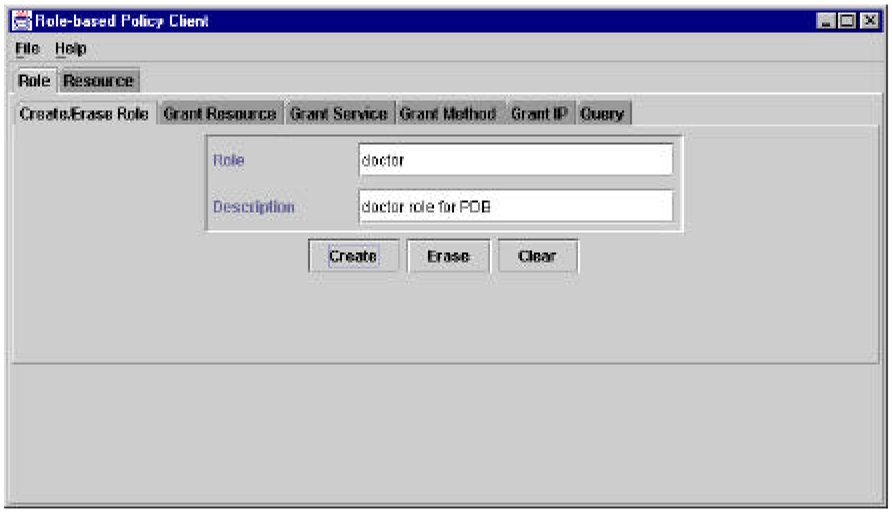


ID: security_admin

PASSWORD: **********

security admin

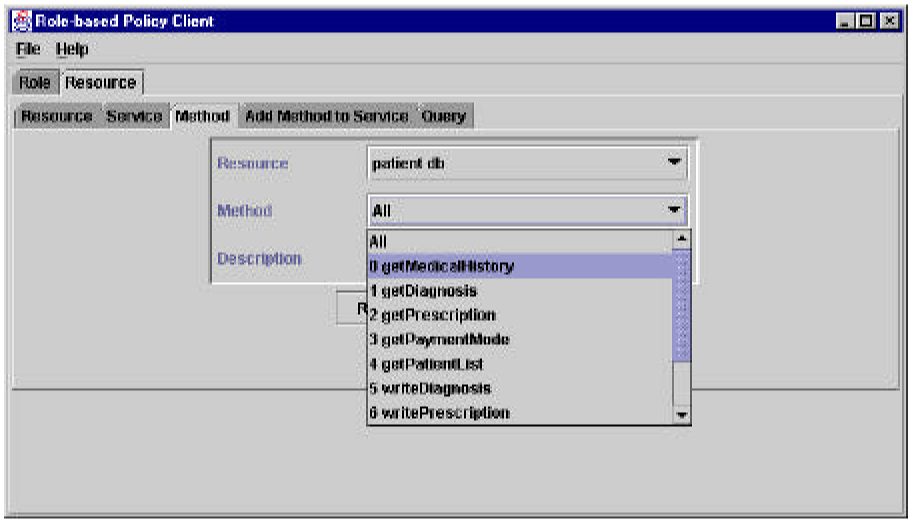
3.2 Role Panel


Role-based Policy Client
File Help
Role Resource
Create/Erase Role Grant Resource Grant Service Grant Method Grant IP Query
Create Erase Clear

Role: doctor

Description: doctor role for PDB

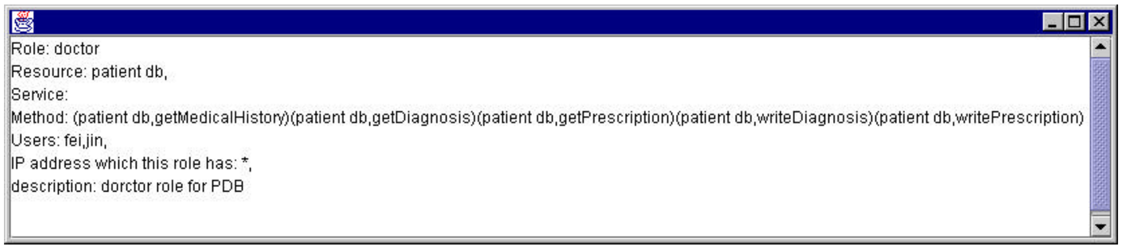
3.3 Resource Panel



Resource: patient db

Method: All

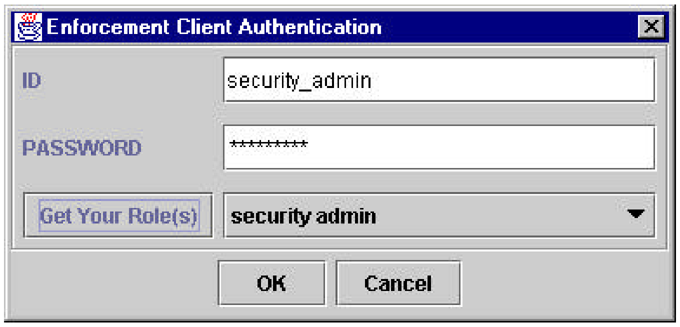
3.4 Role-Query window



```
Role: doctor
Resource: patient db,
Service:
Method: (patient db,getMedicalHistory)(patient db,getDiagnosis)(patient db,getPrescription)(patient db,writeDiagnosis)(patient db,writePrescription)
Users: fei,jin,
IP address which this role has: *,
description: dorctor role for PDB
```

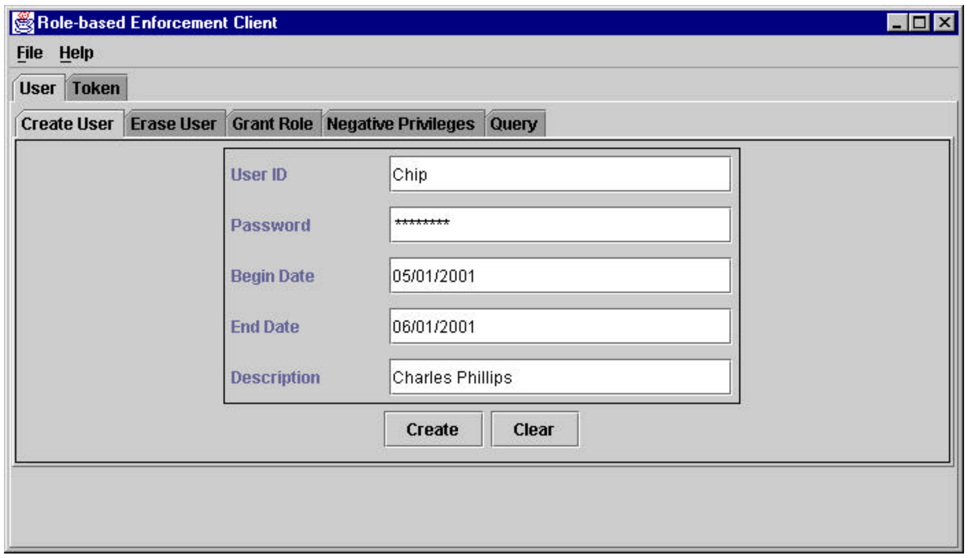
4. Enforcement Client

4.1 Authentication Dialog



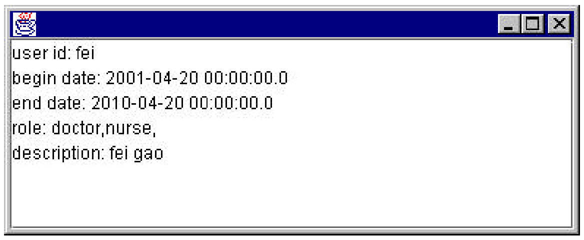
ID: security_admin
PASSWORD: **********
security admin

4.2 User Panel



User ID	Chip
Password	*********
Begin Date	05/01/2001
End Date	06/01/2001
Description	Charles Phillips

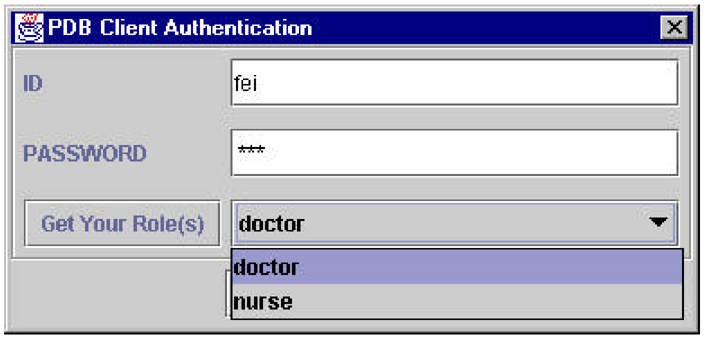
4.3 User-Query window



```
user id: fei
begin date: 2001-04-20 00:00:00.0
end date: 2010-04-20 00:00:00.0
role: doctor,nurse,
description: fei gao
```

5. Patient DB Client

5.1 Authentication window

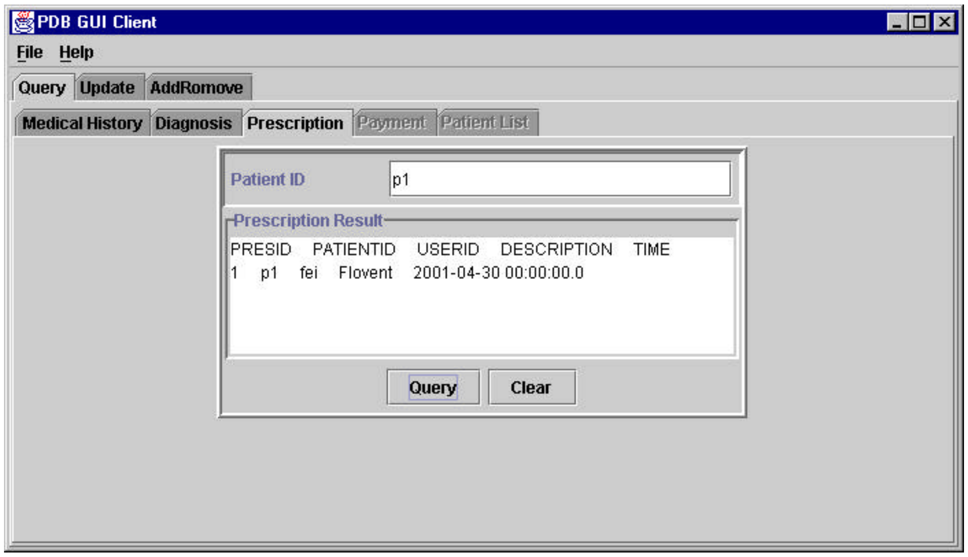


ID: fei

PASSWORD: ***

Role: doctor

5.2 Query Panel

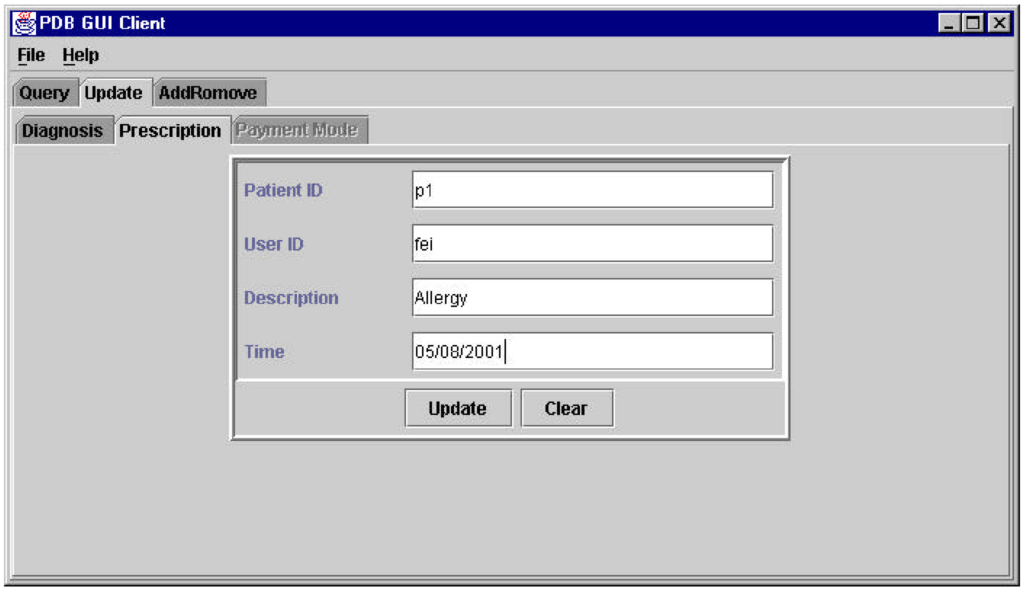


Patient ID: p1

Prescription Result

```
PRESID   PATIENTID   USERID   DESCRIPTION   TIME
1    p1    fei    Flovent    2001-04-30 00:00:00.0
```

5.3 Update Panel



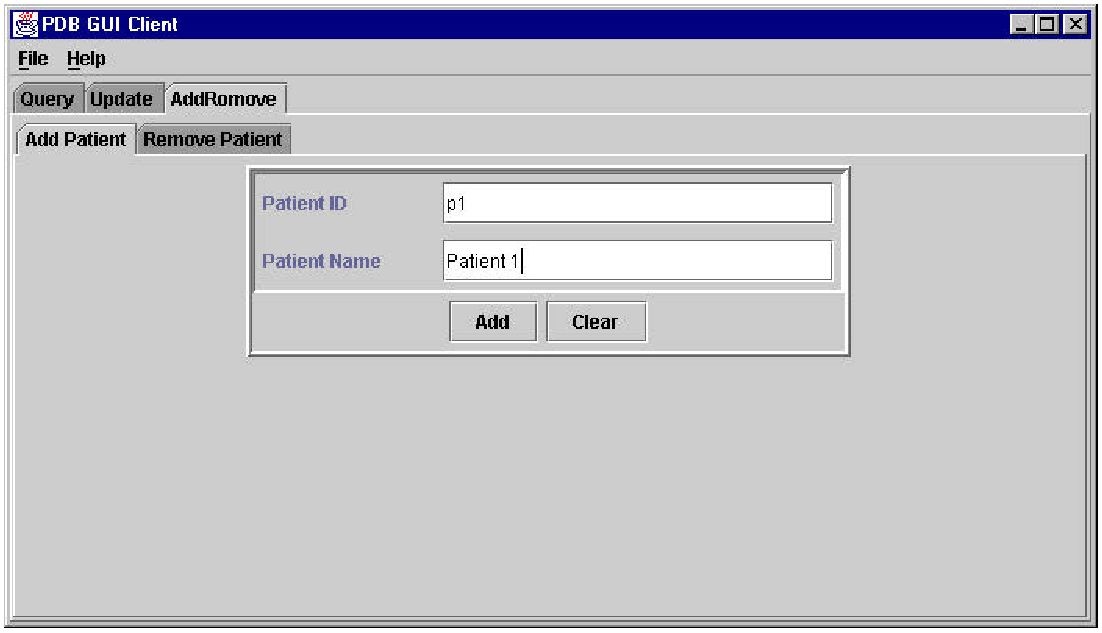
Patient ID: p1

User ID: fei

Description: Allergy

Time: 05/08/2001

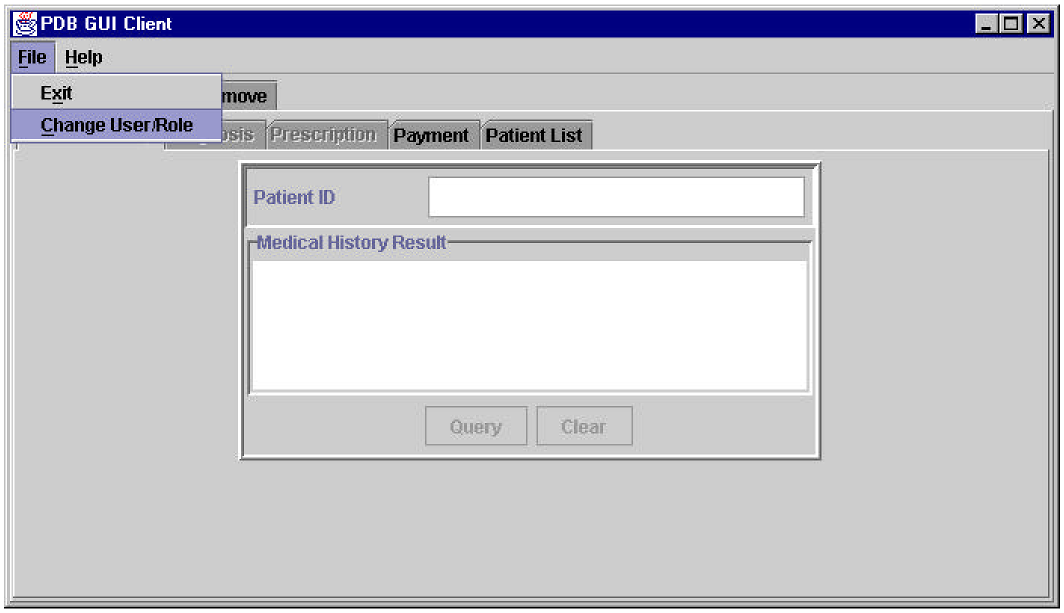
5.4 Add/Remove Panel



Patient ID: p1

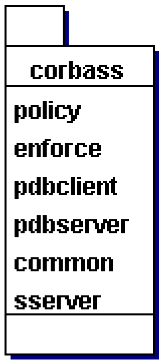
Patient Name: Patient 1

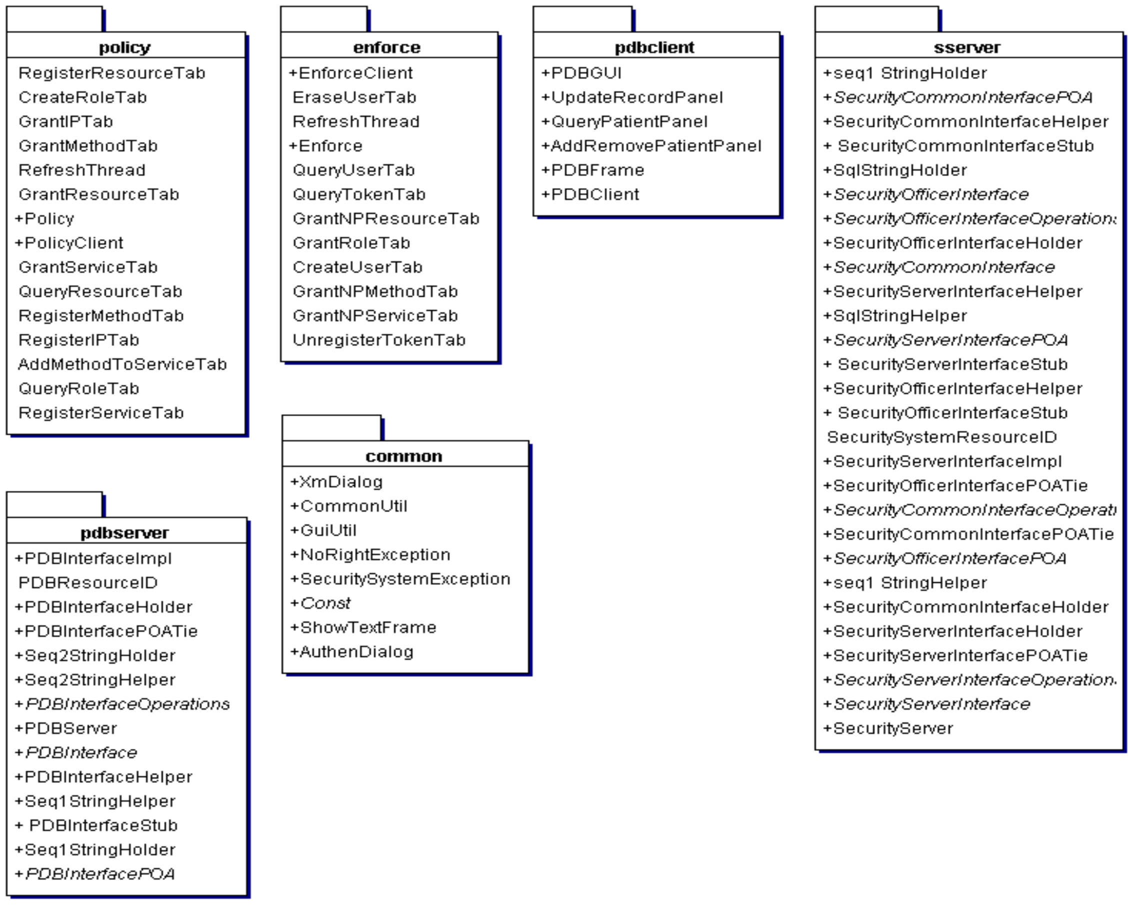
5.5 Change User/Role



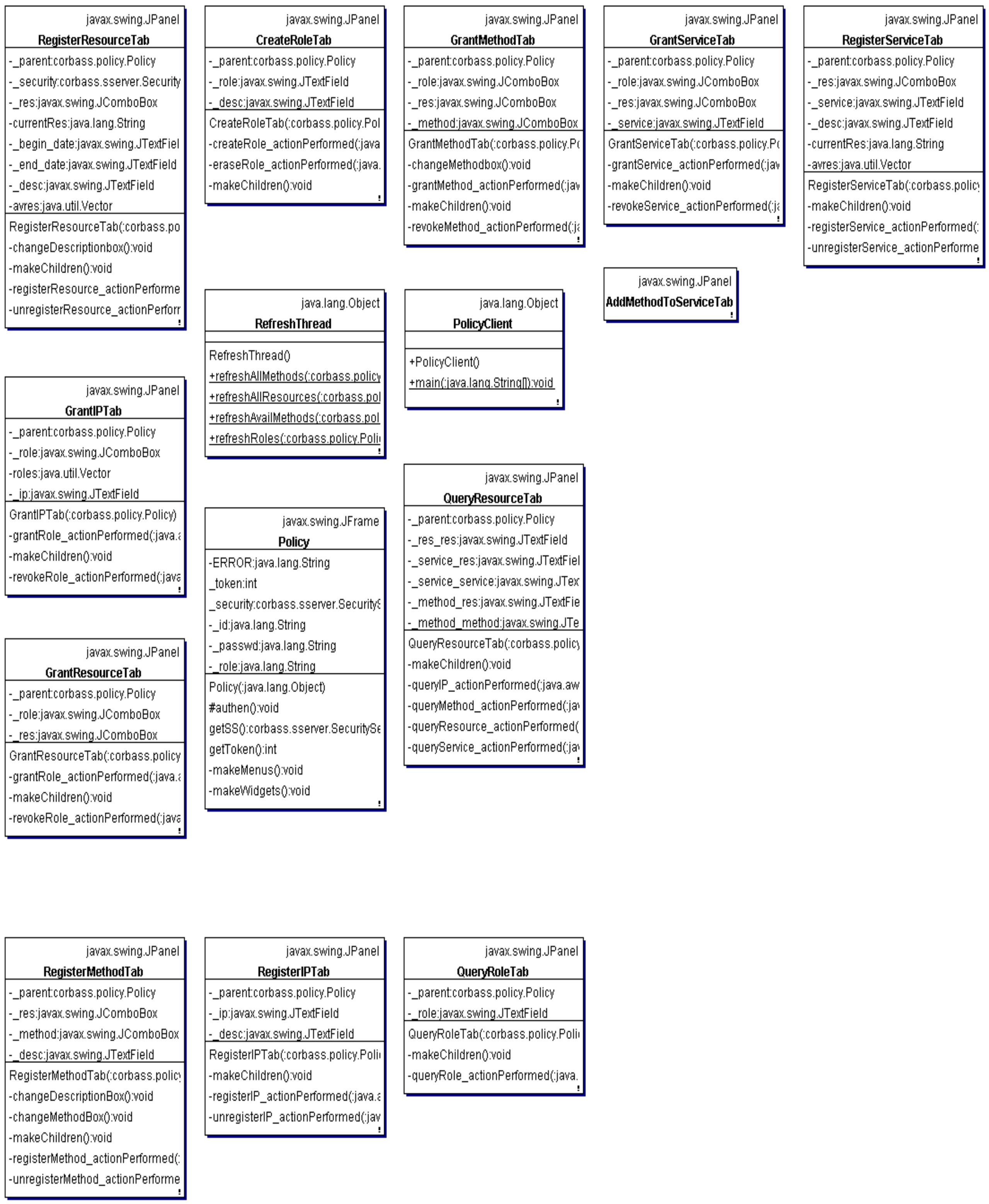
Appendix C: UML Diagrams

1. Packages

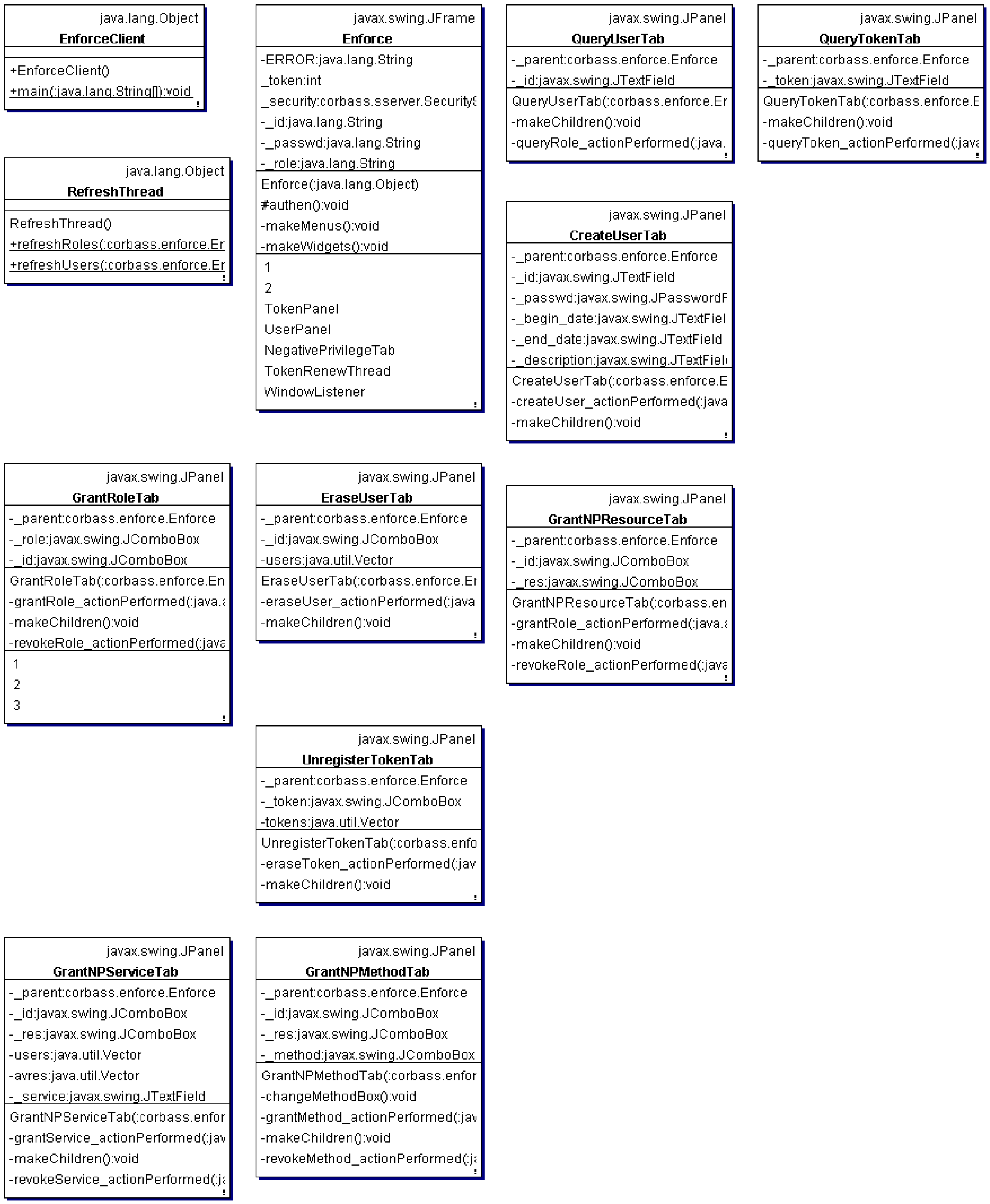




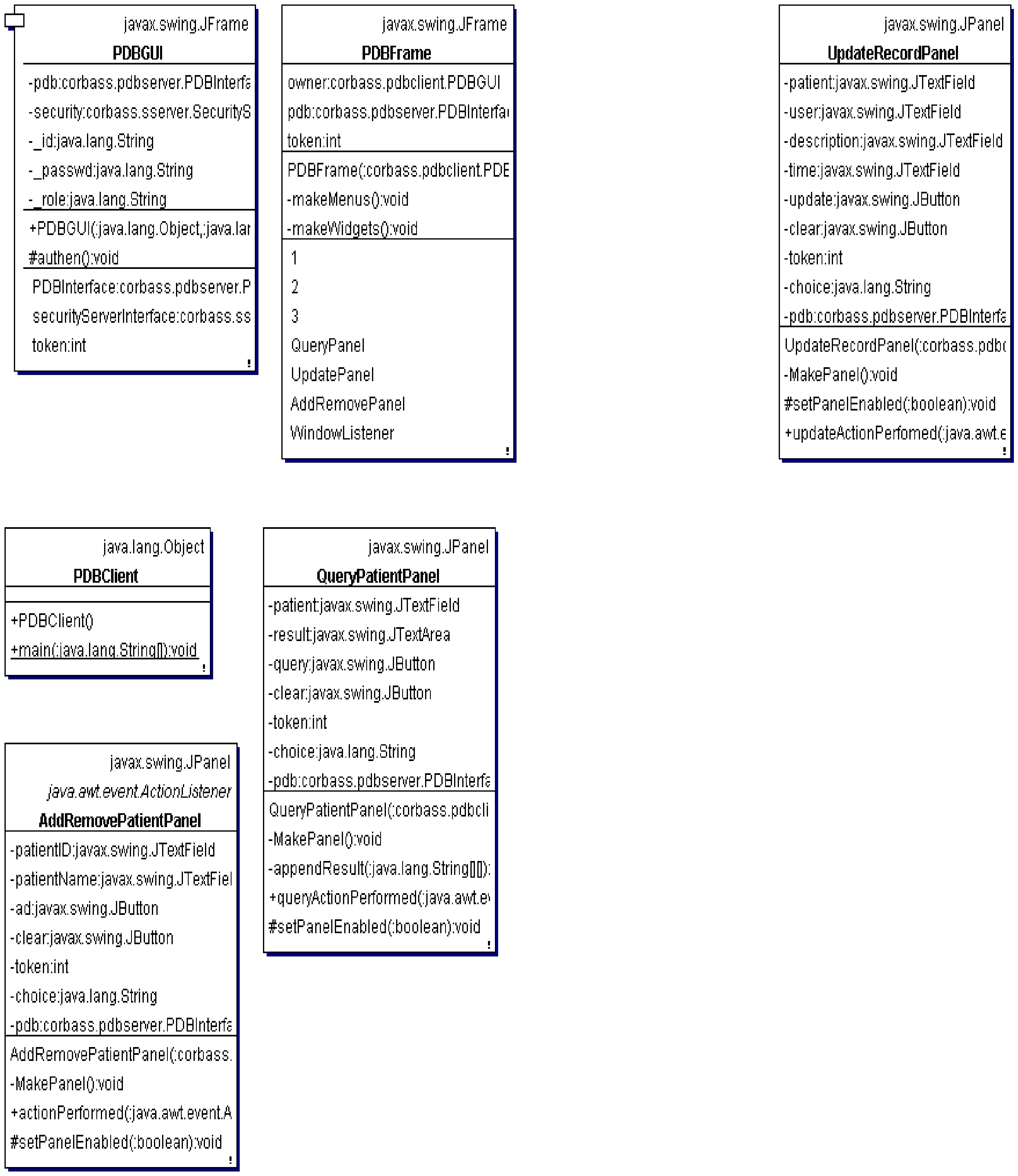
2. Package: corbass.policy


javax.swing.JPanel
RegisterResourceTab
-_parent:corbass.policy.Policy
-_security:corbass.sserver.Security
-_res:javax.swing.JComboBox
-currentRes:java.lang.String
-_begin_date:javax.swing.JTextFiel
-_end_date:javax.swing.JTextField
-_desc:javax.swing.JTextField
-avres:java.util.Vector
RegisterResourceTab(:corbass.po
-changeDescriptionbox():void
-makeChildren():void
-registerResource_actionPerforme
-unregisterResource_actionPerforr
javax.swing.JPanel
CreateRoleTab
-_parent:corbass.policy.Policy
-_role:javax.swing.JTextField
-_desc:javax.swing.JTextField
CreateRoleTab(:corbass.policy.Pol
-createRole_actionPerformed(:java
-eraseRole_actionPerformed(:java.
-makeChildren():void
javax.swing.JPanel
GrantMethodTab
-_parent:corbass.policy.Policy
-_role:javax.swing.JComboBox
-_res:javax.swing.JComboBox
-_method:javax.swing.JComboBox
GrantMethodTab(:corbass.policy.P
-changeMethodbox():void
-grantMethod_actionPerformed(:jav
-makeChildren():void
-revokeMethod_actionPerformed(:j
javax.swing.JPanel
GrantServiceTab
-_parent:corbass.policy.Policy
-_role:javax.swing.JComboBox
-_res:javax.swing.JComboBox
-_service:javax.swing.JTextField
GrantServiceTab(:corbass.policy.P
-grantService_actionPerformed(:jav
-makeChildren():void
-revokeService_actionPerformed(:j
javax.swing.JPanel
RegisterServiceTab
-_parent:corbass.policy.Policy
-_res:javax.swing.JComboBox
-_service:javax.swing.JTextField
-_desc:javax.swing.JTextField
-currentRes:java.lang.String
-avres:java.util.Vector
RegisterServiceTab(:corbass.policy
-makeChildren():void
-registerService_actionPerformed(:
-unregisterService_actionPerforme
javax.swing.JPanel
AddMethodToServiceTab
java.lang.Object
RefreshThread
RefreshThread()
+refreshAllMethods(:corbass.policy
+refreshAllResources(:corbass.pol
+refreshAvailMethods(:corbass.pol
+refreshRoles(:corbass.policy.Poli
java.lang.Object
PolicyClient
+PolicyClient()
+main(:java.lang.String[]):void
javax.swing.JPanel
GrantIPTab
-_parent:corbass.policy.Policy
-_role:javax.swing.JComboBox
-roles:java.util.Vector
-_ip:javax.swing.JTextField
GrantIPTab(:corbass.policy.Policy)
-grantRole_actionPerformed(:java.
-makeChildren():void
-revokeRole_actionPerformed(:java
javax.swing.JFrame
Policy
-ERROR:java.lang.String
_token:int
_security:corbass.sserver.Security
-_id:java.lang.String
-_passwd:java.lang.String
-_role:java.lang.String
Policy(:java.lang.Object)
#authen():void
getSS():corbass.sserver.SecuritySe
getToken():int
-makeMenus():void
-makeWidgets():void
javax.swing.JPanel
QueryResourceTab
-_parent:corbass.policy.Policy
-_res_res:javax.swing.JTextField
-_service_res:javax.swing.JTextFiel
-_service_service:javax.swing.JTex
-_method_res:javax.swing.JTextFie
-_method_method:javax.swing.JTe
QueryResourceTab(:corbass.policy
-makeChildren():void
-queryIP_actionPerformed(:java.aw
-queryMethod_actionPerformed(:ja
-queryResource_actionPerformed(
-queryService_actionPerformed(:ja
javax.swing.JPanel
GrantResourceTab
-_parent:corbass.policy.Policy
-_role:javax.swing.JComboBox
-_res:javax.swing.JComboBox
GrantResourceTab(:corbass.policy
-grantRole_actionPerformed(:java.
-makeChildren():void
-revokeRole_actionPerformed(:java
javax.swing.JPanel
RegisterMethodTab
-_parent:corbass.policy.Policy
-_res:javax.swing.JComboBox
-_method:javax.swing.JComboBox
-_desc:javax.swing.JTextField
RegisterMethodTab(:corbass.policy
-changeDescriptionBox():void
-changeMethodBox():void
-makeChildren():void
-registerMethod_actionPerformed(:
-unregisterMethod_actionPerforme
javax.swing.JPanel
RegisterIPTab
-_parent:corbass.policy.Policy
-_ip:javax.swing.JTextField
-_desc:javax.swing.JTextField
RegisterIPTab(:corbass.policy.Poli
-makeChildren():void
-registerIP_actionPerformed(:java.
-unregisterIP_actionPerformed(:jav
javax.swing.JPanel
QueryRoleTab
-_parent:corbass.policy.Policy
-_role:javax.swing.JTextField
QueryRoleTab(:corbass.policy.Poli
-makeChildren():void
-queryRole_actionPerformed(:java.

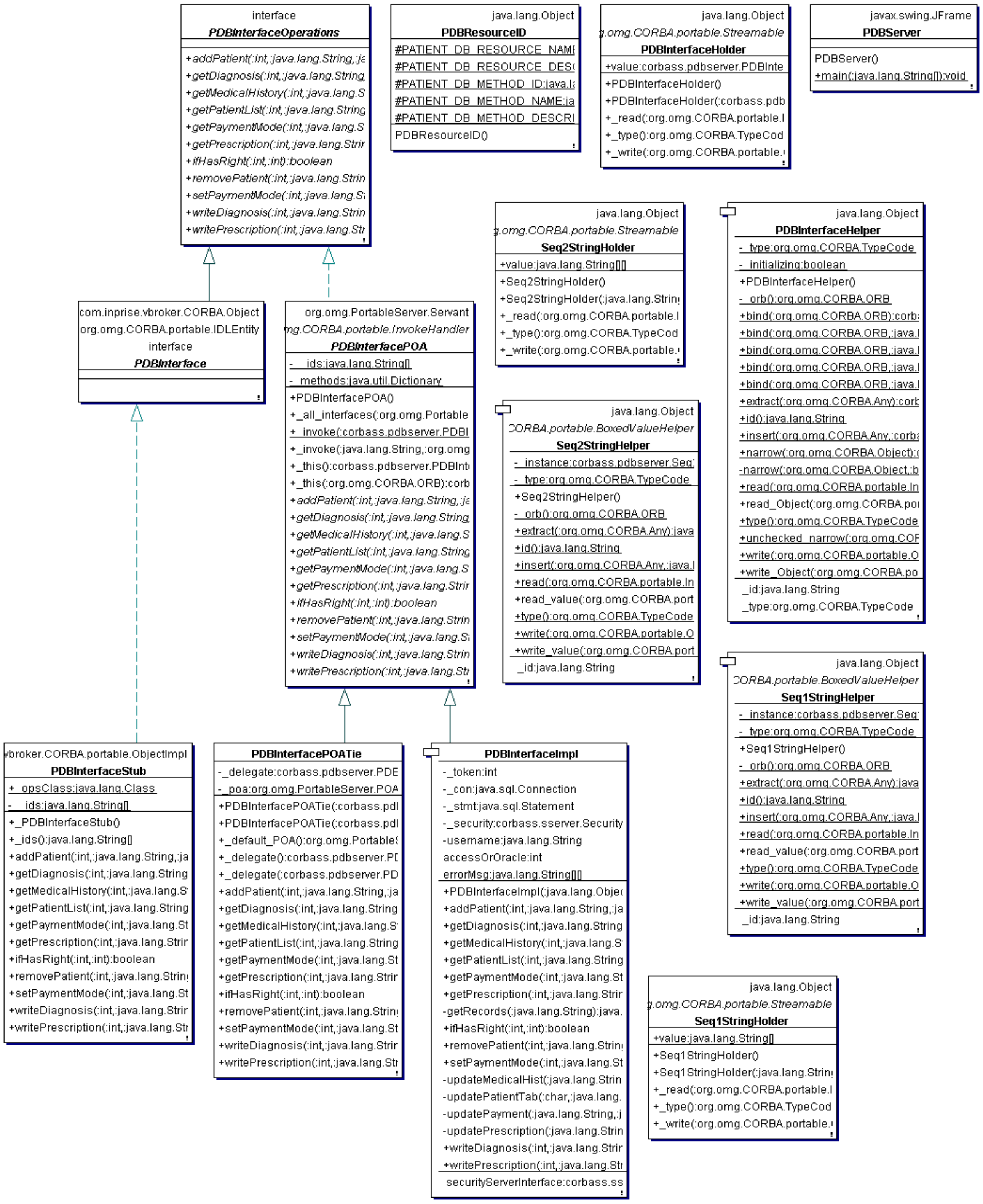
3. Package: corbass.enforce


java.lang.Object
EnforceClient
+EnforceClient()
+main(:java.lang.String[]):void
java.lang.Object
RefreshThread
RefreshThread()
+refreshRoles(:corbass.enforce.Er
+refreshUsers(:corbass.enforce.Er
javax.swing.JFrame
Enforce
-ERROR:java.lang.String
_token:int
_security:corbass.sserver.Security
-_id:java.lang.String
-_passwd:java.lang.String
-_role:java.lang.String
Enforce(:java.lang.Object)
#authen():void
-makeMenus():void
-makeWidgets():void
1
2
TokenPanel
UserPanel
NegativePrivilegeTab
TokenRenewThread
WindowListener
javax.swing.JPanel
QueryUserTab
-_parent:corbass.enforce.Enforce
-_id:javax.swing.JTextField
QueryUserTab(:corbass.enforce.Er
-makeChildren():void
-queryRole_actionPerformed(:java.
javax.swing.JPanel
QueryTokenTab
-_parent:corbass.enforce.Enforce
-_token:javax.swing.JTextField
QueryTokenTab(:corbass.enforce.E
-makeChildren():void
-queryToken_actionPerformed(:java
javax.swing.JPanel
CreateUserTab
-_parent:corbass.enforce.Enforce
-_id:javax.swing.JTextField
-_passwd:javax.swing.JPasswordF
-_begin_date:javax.swing.JTextFiel
-_end_date:javax.swing.JTextField
-_description:javax.swing.JTextFiel
CreateUserTab(:corbass.enforce.E
-createUser_actionPerformed(:java
-makeChildren():void
javax.swing.JPanel
GrantRoleTab
-_parent:corbass.enforce.Enforce
-_role:javax.swing.JComboBox
-_id:javax.swing.JComboBox
GrantRoleTab(:corbass.enforce.En
-grantRole_actionPerformed(:java.
-makeChildren():void
-revokeRole_actionPerformed(:java
1
2
3
javax.swing.JPanel
EraseUserTab
-_parent:corbass.enforce.Enforce
-_id:javax.swing.JComboBox
-users:java.util.Vector
EraseUserTab(:corbass.enforce.E
-eraseUser_actionPerformed(:java
-makeChildren():void
javax.swing.JPanel
GrantNPResourceTab
-_parent:corbass.enforce.Enforce
-_id:javax.swing.JComboBox
-_res:javax.swing.JComboBox
GrantNPResourceTab(:corbass.en
-grantRole_actionPerformed(:java.
-makeChildren():void
-revokeRole_actionPerformed(:java
javax.swing.JPanel
UnregisterTokenTab
-_parent:corbass.enforce.Enforce
-_token:javax.swing.JComboBox
-tokens:java.util.Vector
UnregisterTokenTab(:corbass.enfo
-eraseToken_actionPerformed(:jav
-makeChildren():void
javax.swing.JPanel
GrantNPServiceTab
-_parent:corbass.enforce.Enforce
-_id:javax.swing.JComboBox
-_res:javax.swing.JComboBox
-users:java.util.Vector
-avres:java.util.Vector
-_service:javax.swing.JTextField
GrantNPServiceTab(:corbass.enfor
-grantService_actionPerformed(:jav
-makeChildren():void
-revokeService_actionPerformed(:j
javax.swing.JPanel
GrantNPMethodTab
-_parent:corbass.enforce.Enforce
-_id:javax.swing.JComboBox
-_res:javax.swing.JComboBox
-_method:javax.swing.JComboBox
GrantNPMethodTab(:corbass.enfor
-changeMethodBox():void
-grantMethod_actionPerformed(:jav
-makeChildren():void
-revokeMethod_actionPerformed(:j

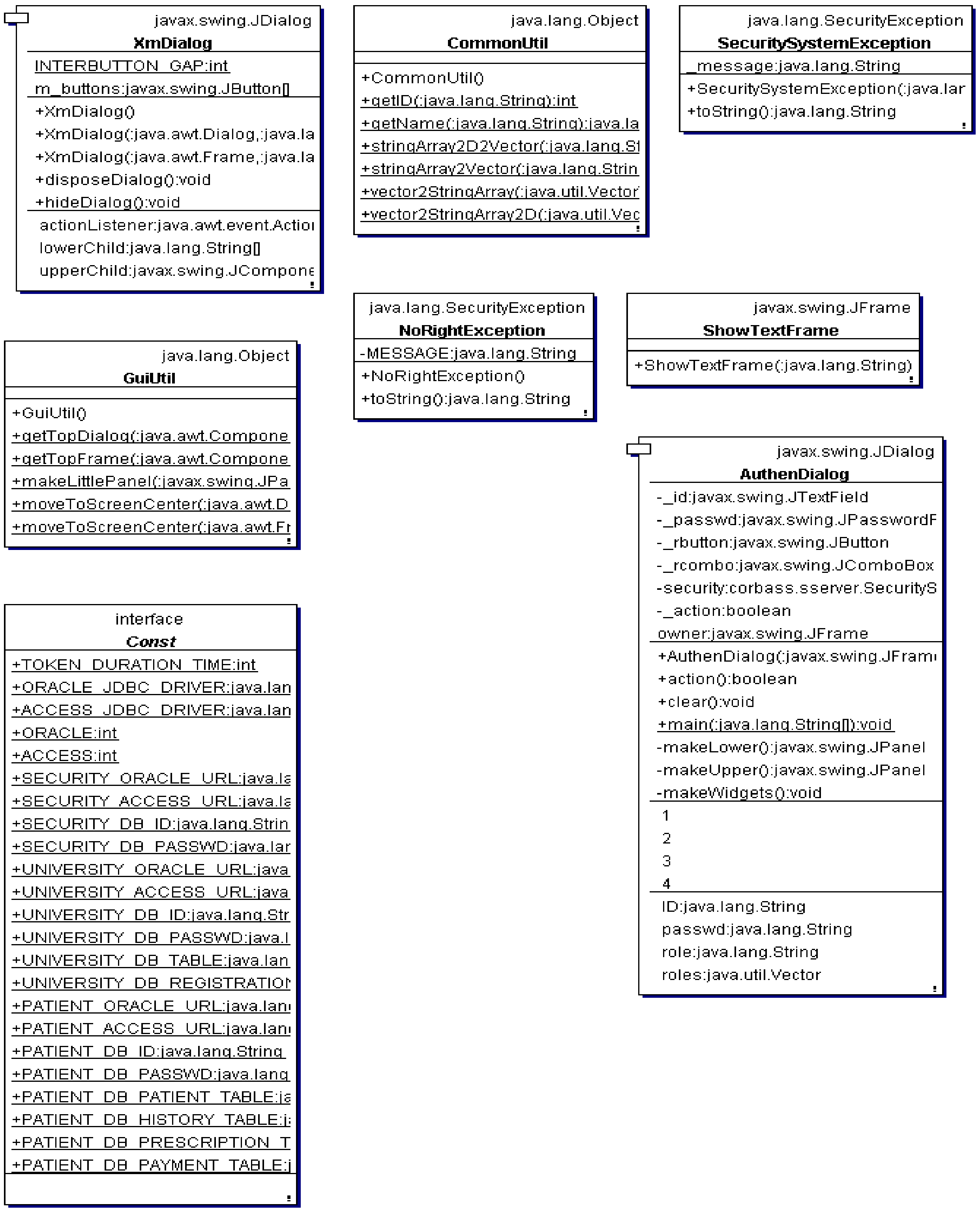
4. Package: corbass.pdbclient


javax.swing.JFrame
PDBGUI
-pdb:corbass.pdbserver.PDBInterfa
-security:corbass.sserver.SecurityS
-_id:java.lang.String
-_passwd:java.lang.String
-_role:java.lang.String
+PDBGUI(:java.lang.Object,:java.lar
#authen():void
PDBInterface:corbass.pdbserver.P
securityServerInterface:corbass.ss
token:int
javax.swing.JFrame
PDBFrame
owner:corbass.pdbclient.PDBGUI
pdb:corbass.pdbserver.PDBInterfa
token:int
PDBFrame(:corbass.pdbclient.PDE
-makeMenus():void
-makeWidgets():void
1
2
3
QueryPanel
UpdatePanel
AddRemovePanel
WindowListener
javax.swing.JPanel
UpdateRecordPanel
-patient:javax.swing.JTextField
-user:javax.swing.JTextField
-description:javax.swing.JTextField
-time:javax.swing.JTextField
-update:javax.swing.JButton
-clear:javax.swing.JButton
-token:int
-choice:java.lang.String
-pdb:corbass.pdbserver.PDBInterfa
UpdateRecordPanel(:corbass.pdb
-MakePanel():void
#setPanelEnabled(:boolean):void
+updateActionPerfomed(:java.awt.e
java.lang.Object
PDBClient
+PDBClient()
+main(:java.lang.String[]):void
javax.swing.JPanel
QueryPatientPanel
-patient:javax.swing.JTextField
-result:javax.swing.JTextArea
-query:javax.swing.JButton
-clear:javax.swing.JButton
-token:int
-choice:java.lang.String
-pdb:corbass.pdbserver.PDBInterfa
QueryPatientPanel(:corbass.pdbcli
-MakePanel():void
-appendResult(:java.lang.String[][]):
+queryActionPerformed(:java.awt.ev
#setPanelEnabled(:boolean):void
javax.swing.JPanel
java.awt.event.ActionListener
AddRemovePatientPanel
-patientID:javax.swing.JTextField
-patientName:javax.swing.JTextFiel
-ad:javax.swing.JButton
-clear:javax.swing.JButton
-token:int
-choice:java.lang.String
-pdb:corbass.pdbserver.PDBInterfa
AddRemovePatientPanel(:corbass.
-MakePanel():void
+actionPerformed(:java.awt.event.A
#setPanelEnabled(:boolean):void

5. Package: corbass.pdbserver


interface
PDBInterfaceOperations
+addPatient(:int,:java.lang.String,:ja
+getDiagnosis(:int,:java.lang.String
+getMedicalHistory(:int,:java.lang.S
+getPatientList(:int,:java.lang.String
+getPaymentMode(:int,:java.lang.S
+getPrescription(:int,:java.lang.Strir
+ifHasRight(:int,:int):boolean
+removePatient(:int,:java.lang.Strin
+setPaymentMode(:int,:java.lang.S
+writeDiagnosis(:int,:java.lang.Strin
+writePrescription(:int,:java.lang.Str
java.lang.Object
PDBResourceID
#PATIENT_DB_RESOURCE_NAME
#PATIENT_DB_RESOURCE_DES
#PATIENT_DB_METHOD_ID:java.l
#PATIENT_DB_METHOD_NAME:ja
#PATIENT_DB_METHOD_DESCRI
PDBResourceID()
java.lang.Object
g.omg.CORBA.portable.Streamable
PDBInterfaceHolder
+value:corbass.pdbserver.PDBInte
+PDBInterfaceHolder()
+PDBInterfaceHolder(:corbass.pdb
+_read(:org.omg.CORBA.portable.I
+_type():org.omg.CORBA.TypeCod
+_write(:org.omg.CORBA.portable.
javax.swing.JFrame
PDBServer
PDBServer()
+main(:java.lang.String[]):void
java.lang.Object
g.omg.CORBA.portable.Streamable
Seq2StringHolder
+value:java.lang.String[][]
+Seq2StringHolder()
+Seq2StringHolder(:java.lang.Strin
+_read(:org.omg.CORBA.portable.I
+_type():org.omg.CORBA.TypeCod
+_write(:org.omg.CORBA.portable.
java.lang.Object
PDBInterfaceHelper
-_type:org.omg.CORBA.TypeCode
-_initializing:boolean
+PDBInterfaceHelper()
-_orb():org.omg.CORBA.ORB
+bind(:org.omg.CORBA.ORB):corb
+bind(:org.omg.CORBA.ORB,:java.l
+bind(:org.omg.CORBA.ORB,:java.l
+bind(:org.omg.CORBA.ORB,:java.l
+bind(:org.omg.CORBA.ORB,:java.l
+extract(:org.omg.CORBA.Any):corb
+id():java.lang.String
+insert(:org.omg.CORBA.Any,:corba
+narrow(:org.omg.CORBA.Object):
-narrow(:org.omg.CORBA.Object,:b
+read(:org.omg.CORBA.portable.In
+read_Object(:org.omg.CORBA.por
+type():org.omg.CORBA.TypeCode
+unchecked_narrow(:org.omg.COF
+write(:org.omg.CORBA.portable.O
+write_Object(:org.omg.CORBA.po
_id:java.lang.String
_type:org.omg.CORBA.TypeCode
com.inprise.vbroker.CORBA.Object
org.omg.CORBA.portable.IDLEntity
interface
PDBInterface
org.omg.PortableServer.Servant
mg.CORBA.portable.InvokeHandler
PDBInterfacePOA
-__ids:java.lang.String[]
-_methods:java.util.Dictionary
+PDBInterfacePOA()
+_all_interfaces(:org.omg.Portable
+_invoke(:corbass.pdbserver.PDBI
+_invoke(:java.lang.String,:org.omg
+_this():corbass.pdbserver.PDBInt
+_this(:org.omg.CORBA.ORB):corb
+addPatient(:int,:java.lang.String,:ja
+getDiagnosis(:int,:java.lang.String
+getMedicalHistory(:int,:java.lang.S
+getPatientList(:int,:java.lang.String
+getPaymentMode(:int,:java.lang.S
+getPrescription(:int,:java.lang.Strir
+ifHasRight(:int,:int):boolean
+removePatient(:int,:java.lang.Strin
+setPaymentMode(:int,:java.lang.S
+writeDiagnosis(:int,:java.lang.Strin
+writePrescription(:int,:java.lang.Str
java.lang.Object
CORBA.portable.BoxedValueHelper
Seq2StringHelper
-_instance:corbass.pdbserver.Seq
-_type:org.omg.CORBA.TypeCode
+Seq2StringHelper()
-_orb():org.omg.CORBA.ORB
+extract(:org.omg.CORBA.Any):java
+id():java.lang.String
+insert(:org.omg.CORBA.Any,:java.l
+read(:org.omg.CORBA.portable.In
+read_value(:org.omg.CORBA.port
+type():org.omg.CORBA.TypeCode
+write(:org.omg.CORBA.portable.O
+write_value(:org.omg.CORBA.port
_id:java.lang.String
java.lang.Object
CORBA.portable.BoxedValueHelper
Seq1StringHelper
-_instance:corbass.pdbserver.Seq
-_type:org.omg.CORBA.TypeCode
+Seq1StringHelper()
-_orb():org.omg.CORBA.ORB
+extract(:org.omg.CORBA.Any):java
+id():java.lang.String
+insert(:org.omg.CORBA.Any,:java.l
+read(:org.omg.CORBA.portable.In
+read_value(:org.omg.CORBA.port
+type():org.omg.CORBA.TypeCode
+write(:org.omg.CORBA.portable.O
+write_value(:org.omg.CORBA.port
_id:java.lang.String
vbroker.CORBA.portable.ObjectImpl
PDBInterfaceStub
+_opsClass:java.lang.Class
-__ids:java.lang.String[]
+_PDBInterfaceStub()
+_ids():java.lang.String[]
+addPatient(:int,:java.lang.String,:ja
+getDiagnosis(:int,:java.lang.String
+getMedicalHistory(:int,:java.lang.S
+getPatientList(:int,:java.lang.String
+getPaymentMode(:int,:java.lang.St
+getPrescription(:int,:java.lang.Strir
+ifHasRight(:int,:int):boolean
+removePatient(:int,:java.lang.Strin
+setPaymentMode(:int,:java.lang.St
+writeDiagnosis(:int,:java.lang.Strir
+writePrescription(:int,:java.lang.Str
PDBInterfacePOATie
-_delegate:corbass.pdbserver.PDE
-_poa:org.omg.PortableServer.POA
+PDBInterfacePOATie(:corbass.pdl
+PDBInterfacePOATie(:corbass.pdl
+_default_POA():org.omg.Portable
+_delegate():corbass.pdbserver.PD
+_delegate(:corbass.pdbserver.PD
+addPatient(:int,:java.lang.String,:ja
+getDiagnosis(:int,:java.lang.String
+getMedicalHistory(:int,:java.lang.S
+getPatientList(:int,:java.lang.String
+getPaymentMode(:int,:java.lang.St
+getPrescription(:int,:java.lang.Strir
+ifHasRight(:int,:int):boolean
+removePatient(:int,:java.lang.Strin
+setPaymentMode(:int,:java.lang.St
+writeDiagnosis(:int,:java.lang.Strir
+writePrescription(:int,:java.lang.Str
PDBInterfaceImpl
-_token:int
-_con:java.sql.Connection
-_stmt:java.sql.Statement
-_security:corbass.sserver.Security
-username:java.lang.String
accessOrOracle:int
errorMsg:java.lang.String[][]
+PDBInterfaceImpl(:java.lang.Objec
+addPatient(:int,:java.lang.String,:ja
+getDiagnosis(:int,:java.lang.String
+getMedicalHistory(:int,:java.lang.S
+getPatientList(:int,:java.lang.String
+getPaymentMode(:int,:java.lang.St
+getPrescription(:int,:java.lang.Strir
-getRecords(:java.lang.String):java.
+ifHasRight(:int,:int):boolean
+removePatient(:int,:java.lang.Strin
+setPaymentMode(:int,:java.lang.St
-updateMedicalHist(:java.lang.Strin
-updatePatientTab(:char,:java.lang.
-updatePayment(:java.lang.String,:j
-updatePrescription(:java.lang.Strin
+writeDiagnosis(:int,:java.lang.Strir
+writePrescription(:int,:java.lang.Str
securityServerInterface:corbass.ss
java.lang.Object
g.omg.CORBA.portable.Streamable
Seq1StringHolder
+value:java.lang.String[]
+Seq1StringHolder()
+Seq1StringHolder(:java.lang.Strin
+_read(:org.omg.CORBA.portable.I
+_type():org.omg.CORBA.TypeCod
+_write(:org.omg.CORBA.portable.

6. Package:corbass.common


javax.swing.JDialog
XmDialog
INTERBUTTON_GAP:int
m_buttons:javax.swing.JButton[]
+XmDialog()
+XmDialog(:java.awt.Dialog,:java.la
+XmDialog(:java.awt.Frame,:java.la
+disposeDialog():void
+hideDialog():void
actionListener:java.awt.event.Actio
lowerChild:java.lang.String[]
upperChild:javax.swing.JCompone
java.lang.Object
CommonUtil
+CommonUtil()
+getID(:java.lang.String):int
+getName(:java.lang.String):java.la
+stringArray2D2Vector(:java.lang.St
+stringArray2Vector(:java.lang.Strin
+vector2StringArray(:java.util.Vector
+vector2StringArray2D(:java.util.Vec
java.lang.SecurityException
SecuritySystemException
_message:java.lang.String
+SecuritySystemException(:java.lar
+toString():java.lang.String
java.lang.SecurityException
NoRightException
-MESSAGE:java.lang.String
+NoRightException()
+toString():java.lang.String
javax.swing.JFrame
ShowTextFrame
+ShowTextFrame(:java.lang.String)
java.lang.Object
GuiUtil
+GuiUtil()
+getTopDialog(:java.awt.Compone
+getTopFrame(:java.awt.Compone
+makeLittlePanel(:javax.swing.JPa
+moveToScreenCenter(:java.awt.D
+moveToScreenCenter(:java.awt.Fr
javax.swing.JDialog
AuthenDialog
-_id:javax.swing.JTextField
-_passwd:javax.swing.JPasswordF
-_rbutton:javax.swing.JButton
-_rcombo:javax.swing.JComboBox
-security:corbass.sserver.SecurityS
-_action:boolean
owner:javax.swing.JFrame
+AuthenDialog(:javax.swing.JFram
+action():boolean
+clear():void
+main(:java.lang.String[]):void
-makeLower():javax.swing.JPanel
-makeUpper():javax.swing.JPanel
-makeWidgets():void
1
2
3
4
ID:java.lang.String
passwd:java.lang.String
role:java.lang.String
roles:java.util.Vector
interface
Const
+TOKEN_DURATION_TIME:int
+ORACLE_JDBC_DRIVER:java.lan
+ACCESS_JDBC_DRIVER:java.lan
+ORACLE:int
+ACCESS:int
+SECURITY_ORACLE_URL:java.la
+SECURITY_ACCESS_URL:java.la
+SECURITY_DB_ID:java.lang.Strin
+SECURITY_DB_PASSWD:java.lar
+UNIVERSITY_ORACLE_URL:java
+UNIVERSITY_ACCESS_URL:java
+UNIVERSITY_DB_ID:java.lang.Str
+UNIVERSITY_DB_PASSWD:java.l
+UNIVERSITY_DB_TABLE:java.lan
+UNIVERSITY_DB_REGISTRATIOI
+PATIENT_ORACLE_URL:java.lan
+PATIENT_ACCESS_URL:java.lan
+PATIENT_DB_ID:java.lang.String
+PATIENT_DB_PASSWD:java.lang
+PATIENT_DB_PATIENT_TABLE:ja
+PATIENT_DB_HISTORY_TABLE:j
+PATIENT_DB_PRESCRIPTION_T
+PATIENT_DB_PAYMENT_TABLE:j

7. Package:corbass.sserver