Leila Abdollahi Vayghan **Ranjan Batra**

Design Documentation - Assignment 2

This project is the implementation of the assignment one methods and also the transfer record method. In this assignment, all the communications between client and server are in CORBA, and the communication between servers are by using sockets and UDP protocol. The manager ID field is also added to all functions.

The first step was to write the following IDL:

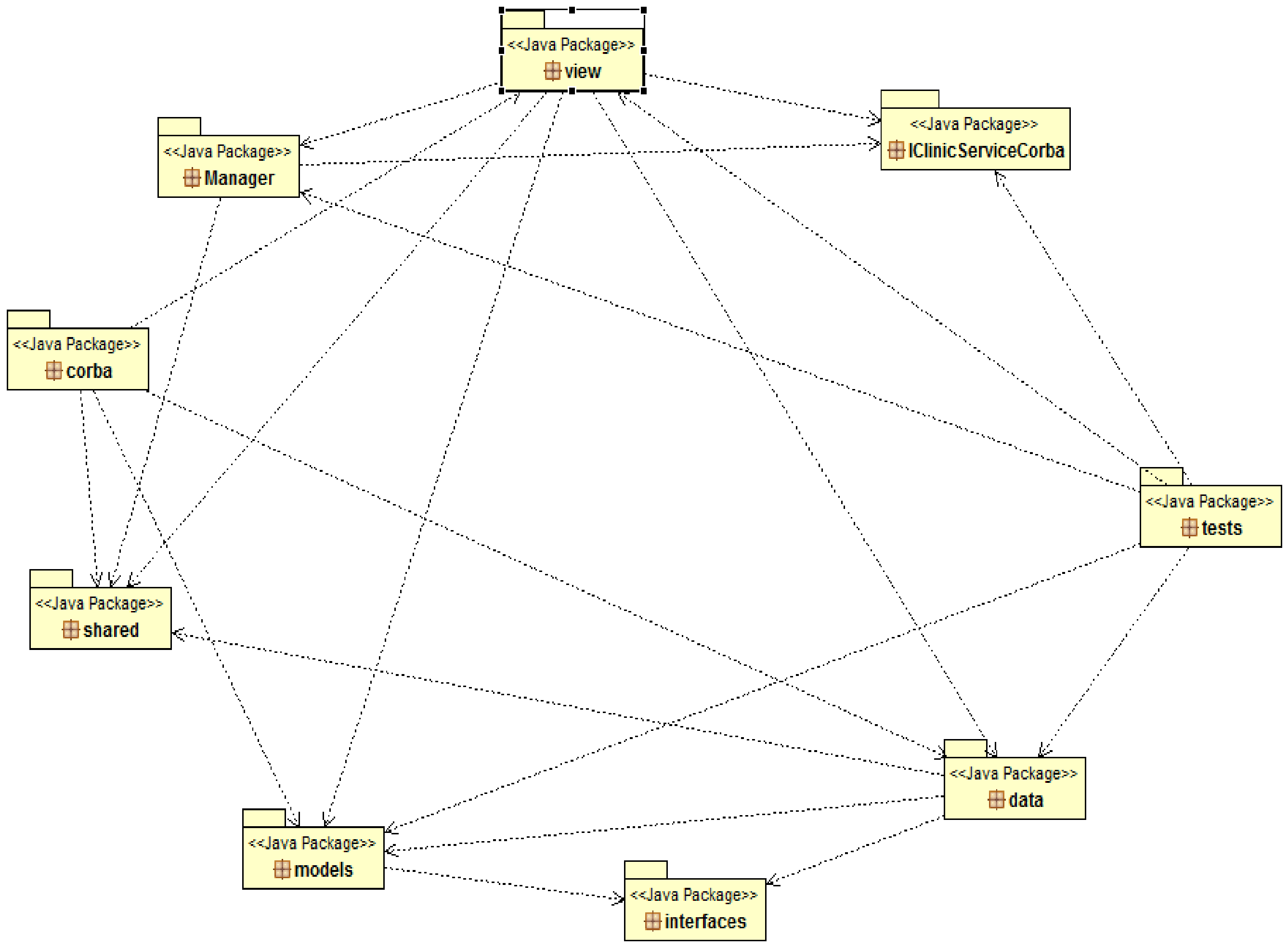
```
module IClinicServiceCorba {

	interface DSMSCorba
	{
	oneway void createDoctorRecord(in string firstName , in string
lastName, in string address, in string phone, in string specialization,in
string location, in string managerID);

	oneway void createNurseRecord(in string firstName , in string lastName,
in string designation, in string status, in string date, in string location,
in string managerID);

	oneway void editRecord(in string recordID, in  string fieldName, in
string fieldValue);

	oneway void transferRecord(in string managerID, in string recordID, in
string remoteClinicServerName);

	oneway void deleteRecord(in string recordID);

	string getRecord();
	};
};
```

And after compiling the IDL file, we started with the implementation.

Package Distributions:

Leila Abdollahi Vayghan **Ranjan Batra**

Design Documentation - Assignment 2



<java package> data

This package contains all the methods manipulating the data like creating doctor or nurse and editing a record depending upon the record type, overall acting as a repository of data.

<java package> corba

This package contains all the functionalities required by the corba server for the execution of distributed system, binding the objects to corba client and executing the functions all described in the assignment.

<java package> IClinicServiceCorba

This package contains all the classes after compiling the idl file including stub, helper,holder and operations and POA file used to implement CORBA system.

<java package> shared

This package contains the shared components distributed in the programming of the application like exception class specifying particular exception depending upon the condition or enumerations for nurse records consisting nurse designation and nurse status.

<java package> interfaces
This package contains interfaces with methods definition required for complete execution of distributed system.

<java package> tests
This package consists of all the tests required for the confirmation of functionalities of the code written for development of the application.

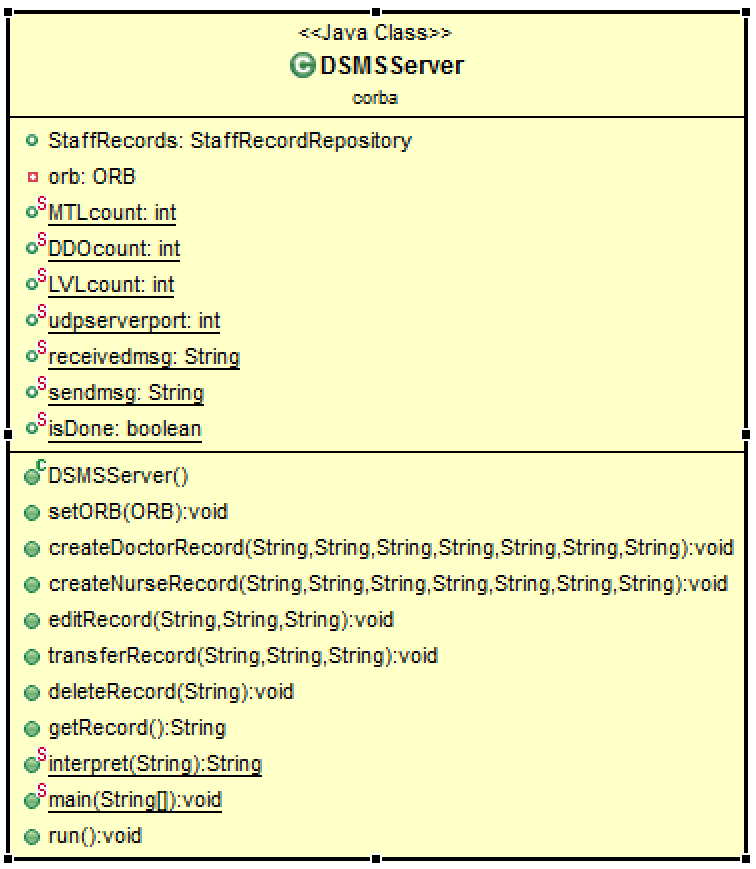
<java package> Manager
This package deals with the functionalities client or manager in this case can have.

<java package> view
This package consists of all the classes that makes up the client side user interface of this application allowing the graphical execution of the operations for completion of this system.

<java package> models
This package contains all the model side information including the doctor record information or attributes or nurse record information or attributes.

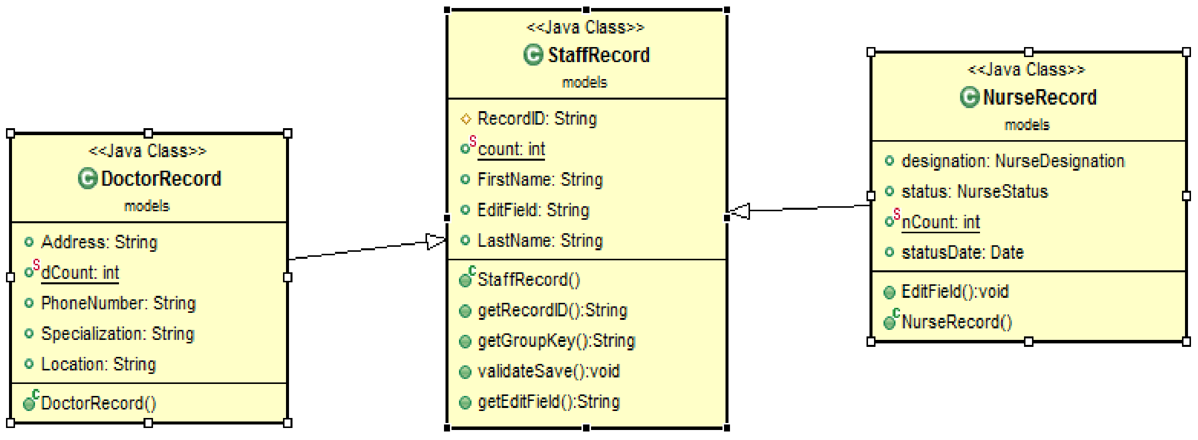
Implementation:

- CORBA Server Implementation:


<<Java Class>>
DSMSServer
corba
StaffRecords: StaffRecordRepository
orb: ORB
MTLcount: int
DDOcount: int
LVLcount: int
udpserverport: int
receivedmsg: String
sendmsg: String
isDone: boolean
DSMSServer()
setORB(ORB):void
createDoctorRecord(String,String,String,String,String,String,String):void
createNurseRecord(String,String,String,String,String,String,String):void
editRecord(String,String,String):void
transferRecord(String,String,String):void
deleteRecord(String):void
getRecord():String
interpret(String):String
main(String[]):void
run():void

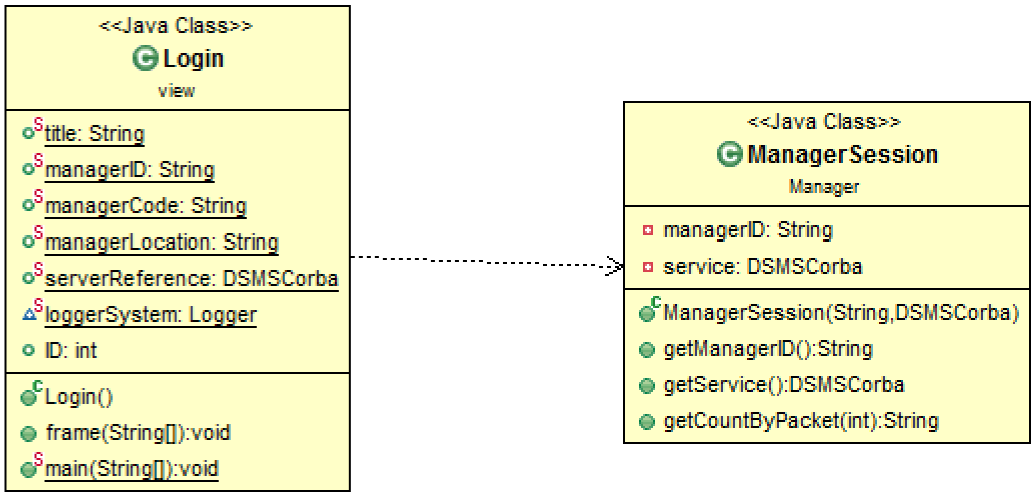
Server skeletons implementation is carried out in this class under the package corba with operations of creating , editing and transferring doctor, nurse records. StaffRecords being the class inherited from GroupedRepository containing all the code-point implementation of operations.

- **Doctor Records and Nurse Records:**



Doctor records and Nurse records were executed along with the execution of inheritance. As both doctors and nurses made up staff, Staffrecord.java is the base class consisting the common attributes of both the sort of staff like FirstName, LastName and contains methods that were to be implemented on both sorts of records like getting the recordID,getting the group key on which hashmaps are to be grouped. For Doctor records , StaffRecords is inherited to class DoctorRecords.java with other attributes required for doctors and the constructor taking care of the unique recordID with “DR” prefix for doctor records. Similarly, Nurse records are made up by inheriting StaffRecord.java and other attributes mentioned in NurseRecord.java with constructor NurseRecord taking care of unique recordID with prefix “NR”.

- Client Side Execution:

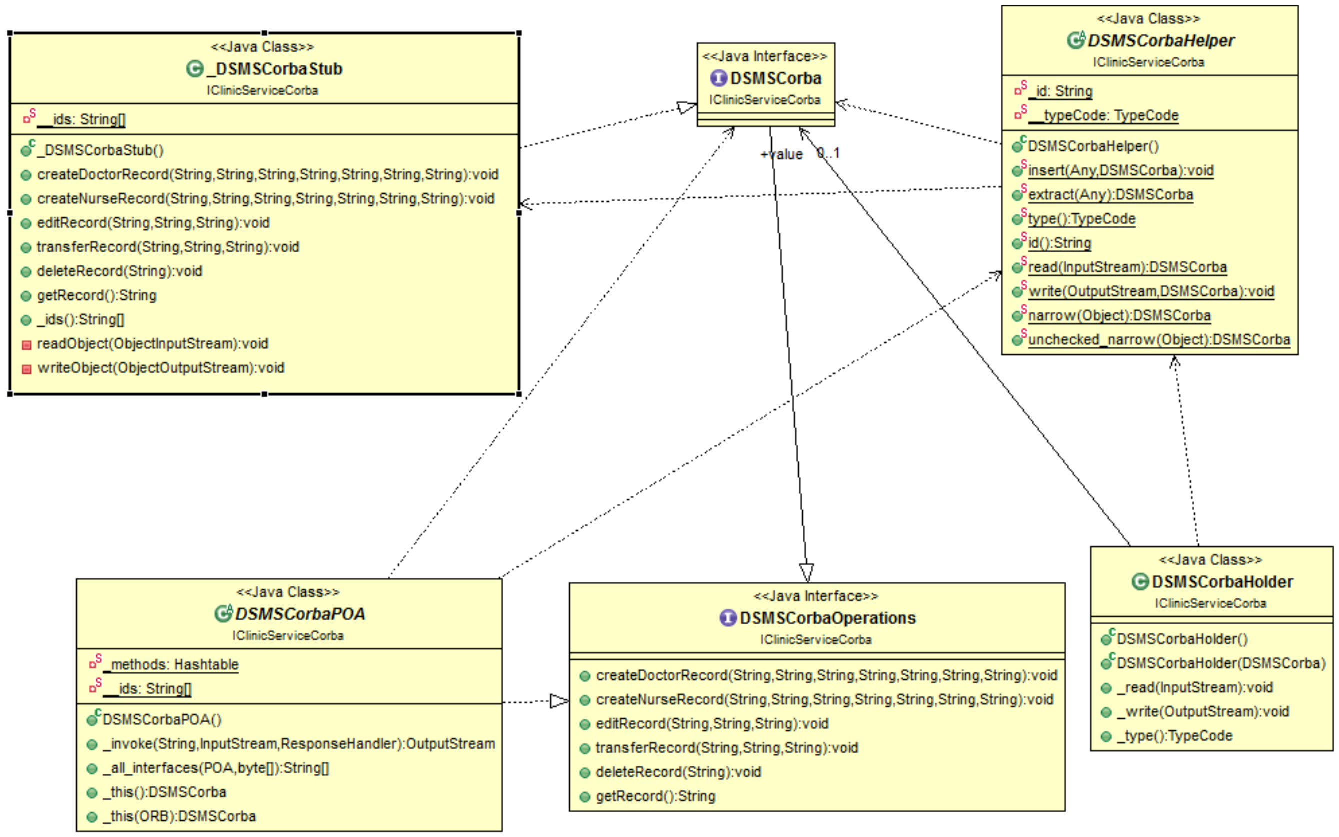


Login.java contains binding implementation of corba client object to server skeletons on the basis of clinic location selected. ManagerSession contains the functionalities that will be used by the client for further implementation of operations.

- Corba implementation:

Leila Abdollahi Vayghan **Ranjan Batra**

Design Documentation - Assignment 2

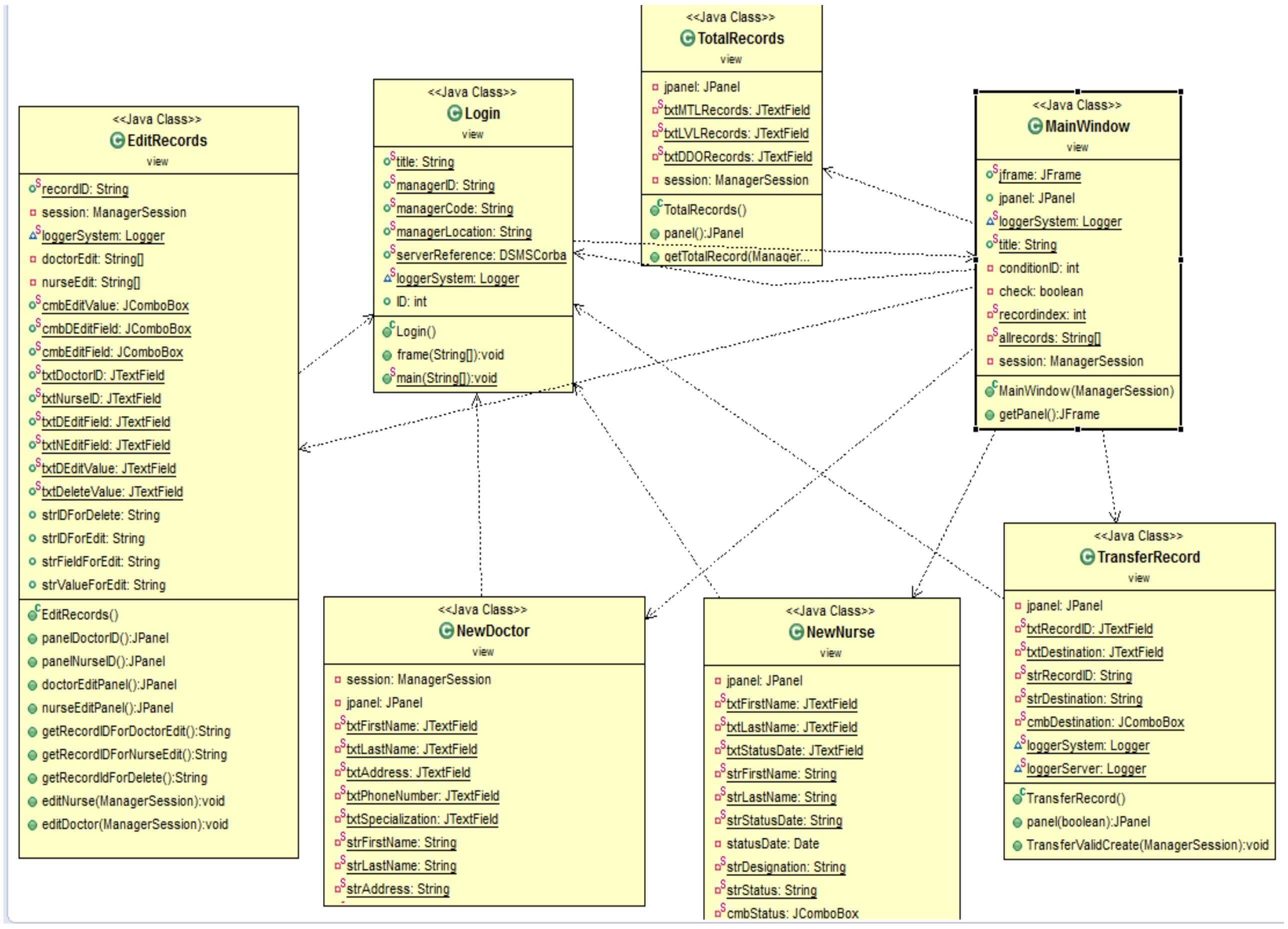

<<Java Class>>
_DSMSCorbaStub
IClinicServiceCorba
__ids: String[]
_DSMSCorbaStub()
createDoctorRecord(String,String,String,String,String,String,String):void
createNurseRecord(String,String,String,String,String,String,String):void
editRecord(String,String,String):void
transferRecord(String,String,String):void
deleteRecord(String):void
getRecord():String
_ids():String[]
readObject(ObjectInputStream):void
writeObject(ObjectOutputStream):void
<<Java Interface>>
DSMSCorba
IClinicServiceCorba
+value 0..1
<<Java Class>>
DSMSCorbaHelper
IClinicServiceCorba
_id: String
__typeCode: TypeCode
DSMSCorbaHelper()
insert(Any,DSMSCorba):void
extract(Any):DSMSCorba
type():TypeCode
id():String
read(InputStream):DSMSCorba
write(OutputStream,DSMSCorba):void
narrow(Object):DSMSCorba
unchecked_narrow(Object):DSMSCorba
<<Java Class>>
DSMSCorbaPOA
IClinicServiceCorba
_methods: Hashtable
__ids: String[]
DSMSCorbaPOA()
_invoke(String,InputStream,ResponseHandler):OutputStream
_all_interfaces(POA,byte[]):String[]
_this():DSMSCorba
_this(ORB):DSMSCorba
<<Java Interface>>
DSMSCorbaOperations
IClinicServiceCorba
createDoctorRecord(String,String,String,String,String,String,String):void
createNurseRecord(String,String,String,String,String,String,String):void
editRecord(String,String,String):void
transferRecord(String,String,String):void
deleteRecord(String):void
getRecord():String
<<Java Class>>
DSMSCorbaHolder
IClinicServiceCorba
DSMSCorbaHolder()
DSMSCorbaHolder(DSMSCorba)
_read(InputStream):void
_write(OutputStream):void
_type():TypeCode

This package contains all the implementation for corba language independent framework aiding in easying the task required to carry out. For eg: creating doctor record, nurse record and editing them through client stubs along with helper and holder classes and POA and operations file.

- **User Interface:**

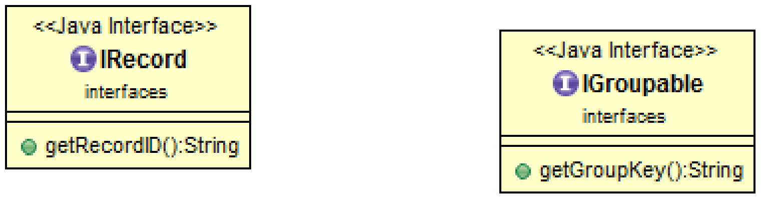
Leila Abdollahi Vayghan **Ranjan Batra**

Design Documentation - Assignment 2



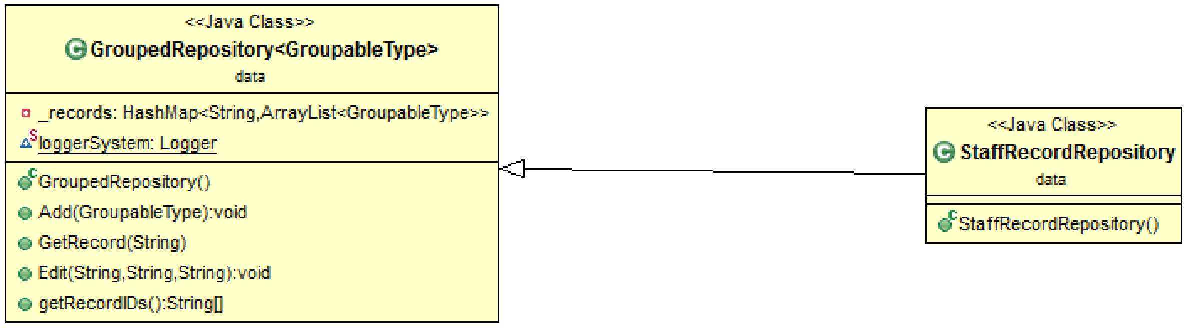
For user interface, the flow goes from Login.java , a class acting as corba client permitting the manager to input his id with clinic code alongwith(4 digit numeric identifier),the clinic locations to access the operations allowed from this application to MainWindow.java class containing menu items for creating doctor, creating nurse, editing nurse, editing doctor, getting total number of records of doctors and nurse,transferring the records from one clinic to another .NewDoctor.java containing JFrame specifications of attributes required for creating a doctor. Similarly NewNurse contains the same functionality for creating the nurse. EditRecords contains the functionality for editing both doctors and nurses. TransferRecord containing functionality for transferring from one clinic to another.

- Interfaces



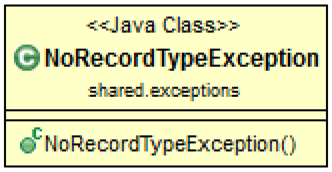
IGroupable is another interface that will be applied to staffrecords and contains the method for grouping the key in hashmaps. IRecord is another interface that will be implemented by the repository so that staff members either doctor records or nurse records can extends it and implement operations for getting the recordID for further execution of operations.

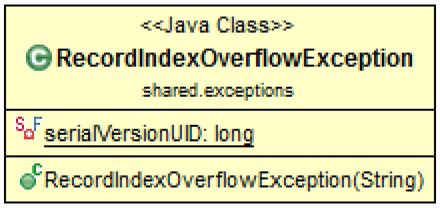
- Data



This package contains two java classes, GroupedRepository.java is the generic class extending IGroupable and IRecord as its wildcards. The main purpose of this class is to allow the execution of operations on both kinds of operations including doctor record and nurse record irrespective of type acting as generic operations execution class. StaffRecordRepository.java class extends GroupedRepository.java through which we can implement the operations of GroupedRepository by considering the objects of this class.

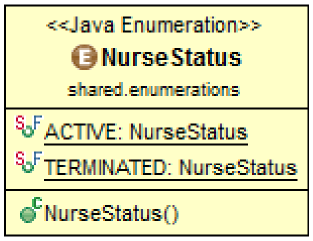
- Exceptions:

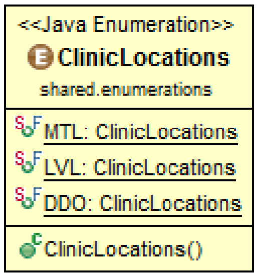


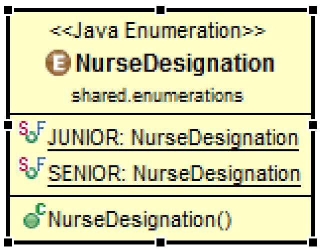

<<Java Class>>
RecordIndexOverflowException
shared.exceptions
serialVersionUID: long
RecordIndexOverflowException(String)

This section of application deals with the exceptions that might be arise from any other section of the application during their execution. Some of these exceptions are listed above like NoRecordTypeException exception that may be arose due to type other than doctor and nurse or RecordIndexOverflowException that will be arise when the records entered will take count more than 5-digit unique number.

- Enumerations:







This section of application deals with all the information that may be represented in enumerations having fixed values in their parameters. Some of the enumerations used in this application are listed above NurseStatus, ClinicLocations and NurseDesignation.

- **Data Structures & Techniques Used**:

- HashMaps

```
private HashMap<String, ArrayList<GroupableType>> _records = new
HashMap<String, ArrayList<GroupableType>>();
```

Hashmaps are used to group the records, Groupable here means any record extending from IGroupable and IRecord with String being first letter of last name. These records are stored in arraylists in form of instances of groupable. Basically, HashMap data structure is used to map the arraylist of instances with special string key.

- ArrayLists:

```
ArrayList<GroupableType> existingGroup = _records.get(key);
existingGroup.add(record);
```

The arraylist, data structure is used to store the instances of Groupabletype, in this application GroupableType may be Doctor Record or Nurse Record.

- Generics:

  Generics in this application is implemented to classes and functions to execute what is desired in this application.

```
public class GroupedRepository<GroupableType extends IGroupable &
IRecord>
```

Groupedrepository is the generic class that will take any type GroupableType as its type that will be extended by IGroupable and IRecord.

```
public void Add(GroupableType record)
{
ArrayList<GroupableType> newGroup = new ArrayList<GroupableType>();
newGroup.add(record);
}
```

Methods to create nurse or doctor is a function taking generic type as its parameter and can be used to store the record again in arraylist of generic type.

```
public void Edit(String recordID, String FieldName, String FieldValue)
throws Exception
{
GroupableType record = GetRecord(recordID);
Field[] recordFields = record.getClass().getFields();
if(field.getName().equals(FieldName))
{
if(field.getType().toString().equals(FieldValue))
{
field.set(record, FieldValue);
```

Editing method for doctor record or nurse record is also generic beginning with attaining the record and afterwards using java reflection to get the field of the object and then validating that field to the field to be edited and after validations assigning the new value to that field.

- ## Communication and difficulties

  As mentioned above, the communication between client and server is carried out using CORBA standard. However, the communication between servers is by using sockets and UDP protocol. Using threads, three different UDP servers are run when the DSMSServer class is run. It is in this class that servers communicate with each other and transfer their records. The difficulties faced in this assignment was mainly in this part, as the CORBA implementation and understanding the concepts of this standard would seem difficult at first.

- **Atomicity in transfer record**
  The transfer record is implemented in a way that a record is deleted only when it is transferred to the remote server and vice versa. When the remote server creates the record, it sends a UDP message that it has done that and the other side deletes the record after it receives this UDP message. The same procedure is done for the remote server to keep the record and make sure that there is no duplicates.

- **How to run this project**
  as it was clearly mentioned in class, we first start an ORB with the initial port of 1050 (it is an optional port which we chose). Then we run DSMSServer.java and Login.java.