Leila Abdollahi Vayghan **Ranjan Batra**

Assignment 3 Document

This assignment is focussed towards transforming the assignment 2 into a project working on web-services. For implementing that, we used the web-service annotations.

The implementation of the end point class is as follows:

```
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface DSMSServerIF {

	@WebMethod void createDoctorRecord(String firstName, String lastName, String address, String phone,
			String specialization, String location, String managerID);
	@WebMethod void createNurseRecord(String firstName, String lastName, String designation, String status, String date,
			String location, String managerID);
	@WebMethod void editRecord(String recordID, String fieldName, String fieldValue);
	@WebMethod void transferRecord(String managerID, String recordID, String remoteClinicServerName);
	@WebMethod String getRecord();
	@WebMethod void udprun() throws InterruptedException;

}
```

We implemented a publisher class to create WSDL at the desired ports.

```
Endpoint.publish("http://localhost:8070/ws/DSMS", new DSMSServer());
Endpoint.publish("http://localhost:8080/ws/DSMS", new DSMSServer());
Endpoint.publish("http://localhost:8090/ws/DSMS", new DSMSServer());
```

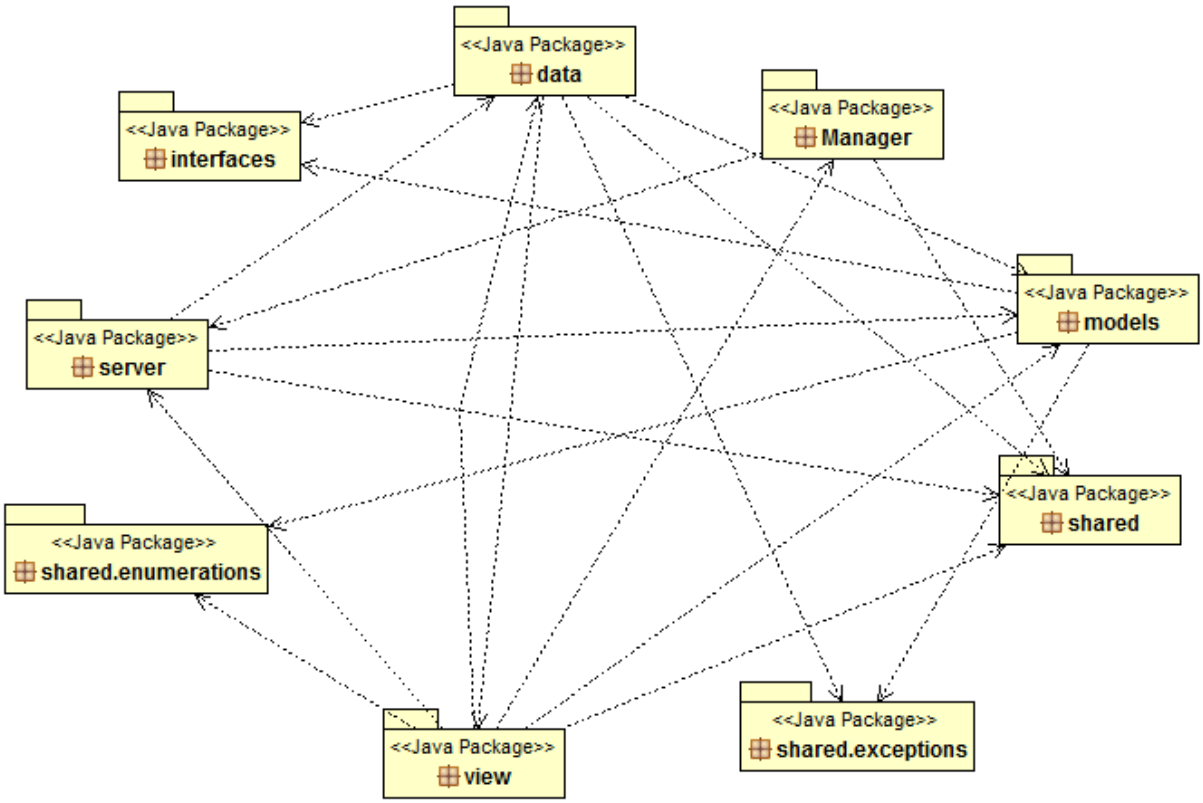
-Communication Standards

The client communicates with server using web service. However, the communication between servers are in UDP. In other words, total staff count and transfer record are done through UDP.

Leila Abdollahi Vayghan **Ranjan Batra**

Assignment 3 Document

Package Distributions:



<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> server

This package contains all the classes required for implementing web services including an interface, server implementation, and the publisher for wsdl.

<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.

Leila Abdollahi Vayghan **Ranjan Batra**

Assignment 3 Document

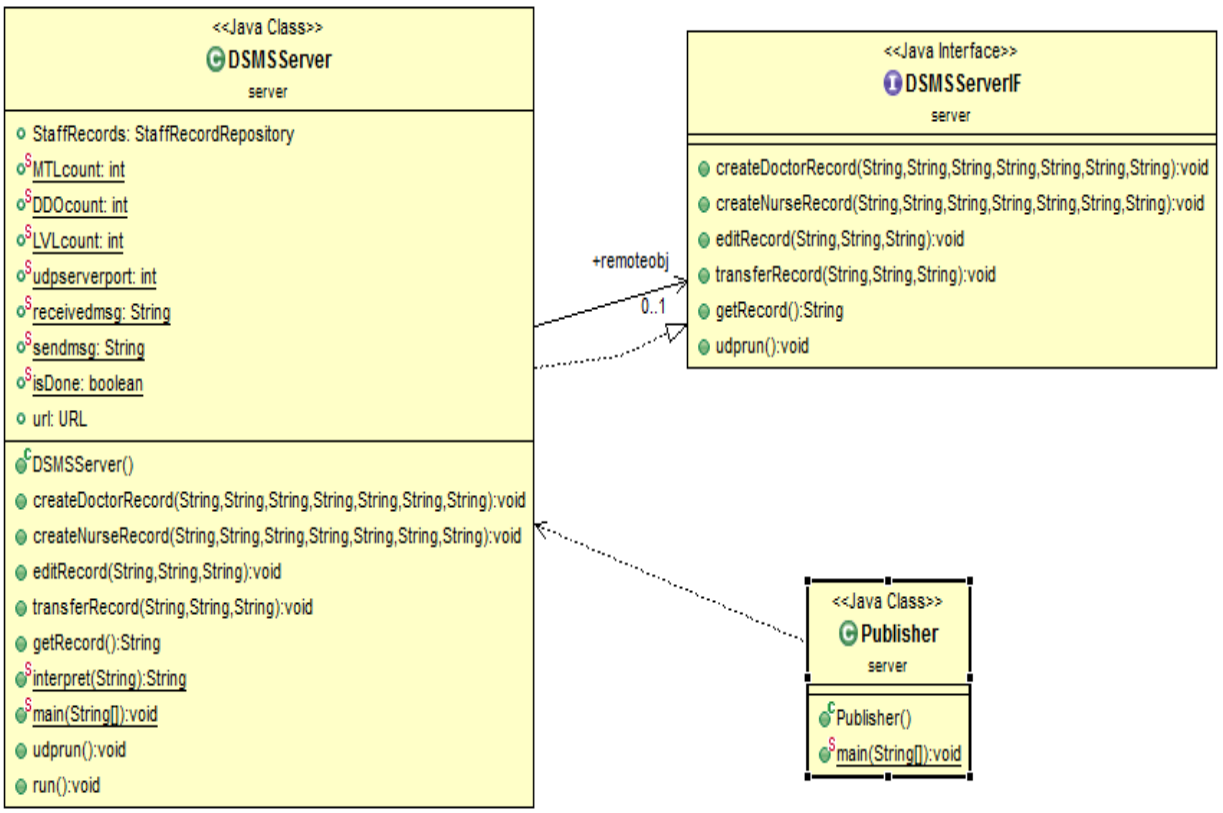
<java package> Manager
This package deals with the functionalities client or manager in this case can have.

<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:

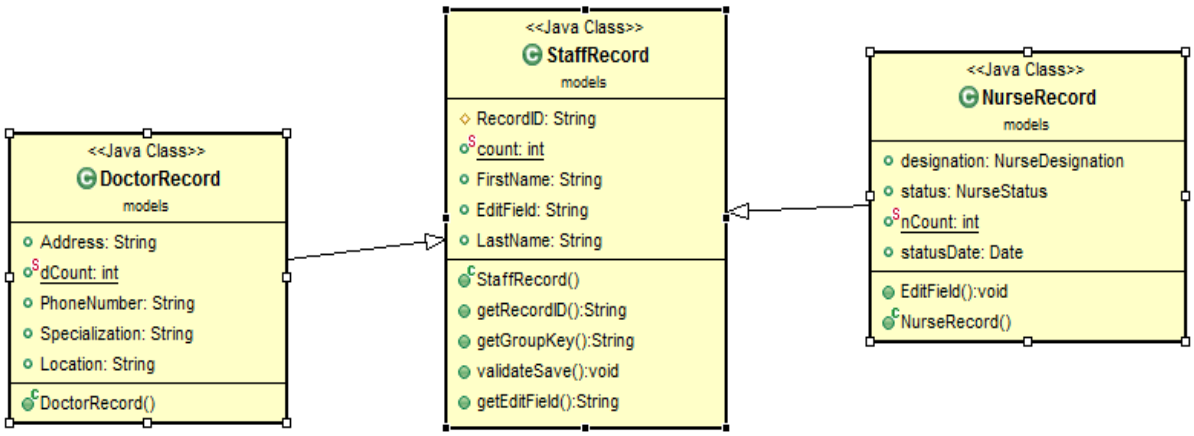
- Web Service Implementation:



All the implementation is done in **DSMSServer.java** and web-service is applied to particular port in **Publisher.java** and **DSMSServerIF.java** is the interface contains all the methods required to

accessed by web-services by implementing DSMSServerIF by the main implementation class DSMSServer.

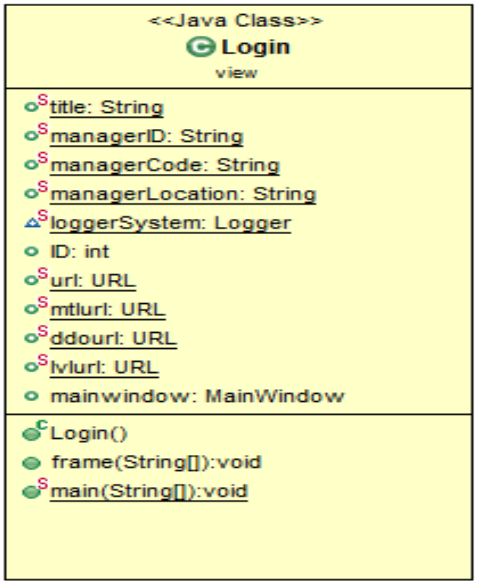
- ## Doctor Records and Nurse Records:



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”.

Assignment 3 Document

- ## Client Side Execution:

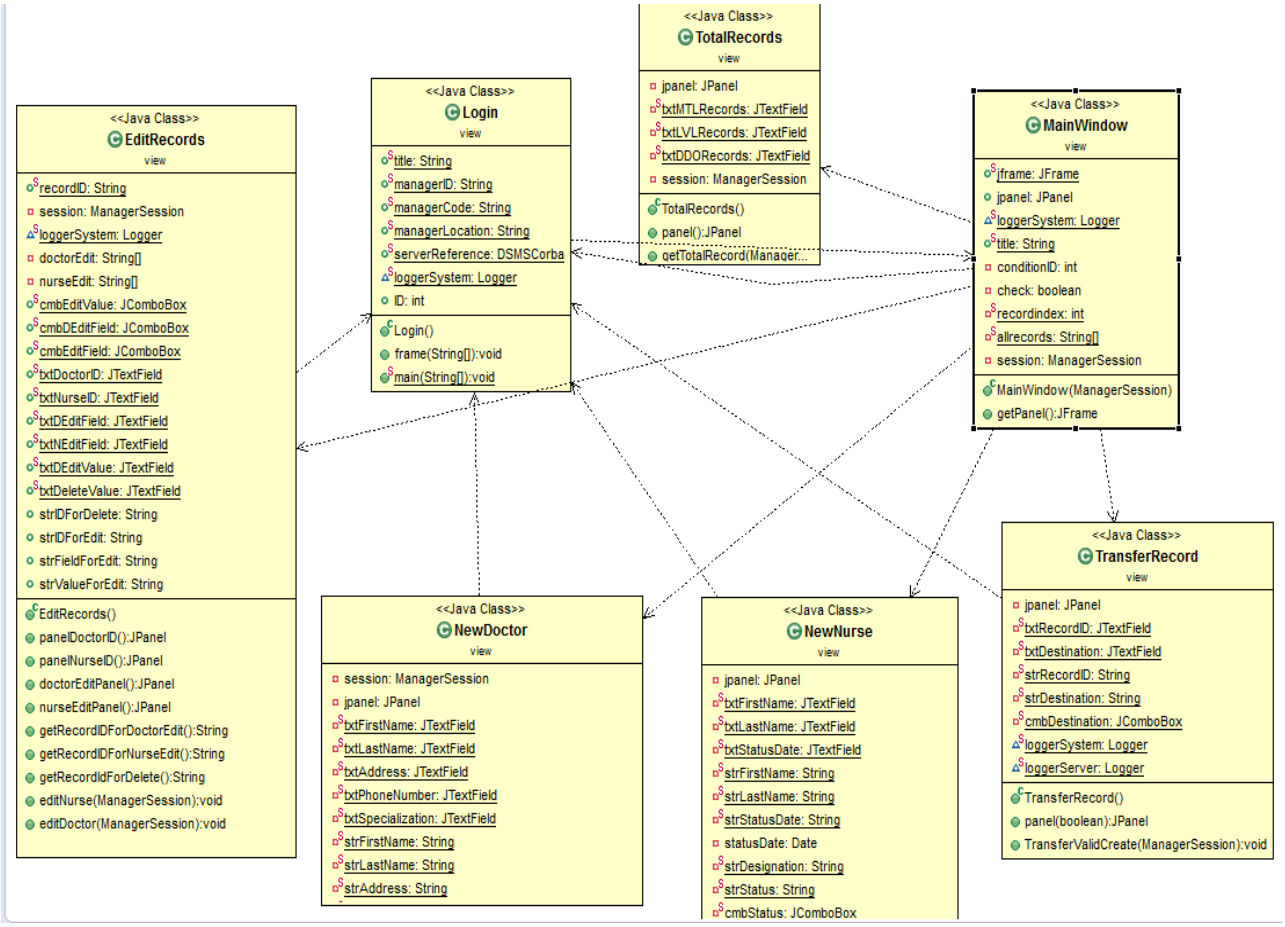


Login.java contains binding implementation of client object to server skeletons through **web-services** on the basis of clinic location selected. Web-service implemented through using the components required for web-service including url, port, qname, service.

Leila Abdollahi Vayghan **Ranjan Batra**

Assignment 3 Document

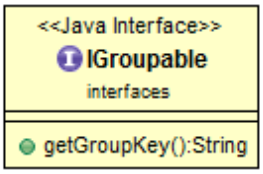
- ## User Interface:



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 along with(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.

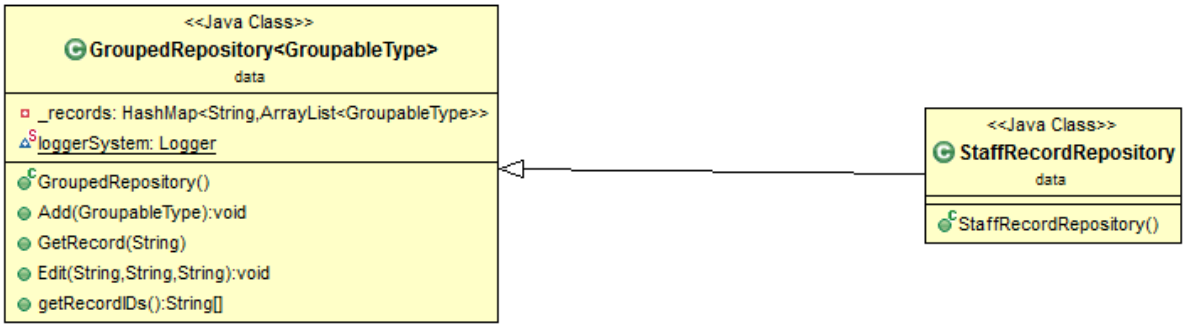
Assignment 3 Document

- ## 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.

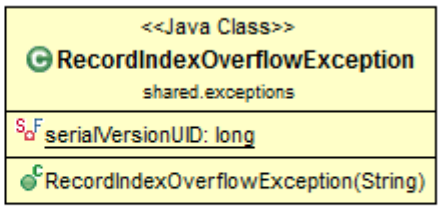
- Data



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.

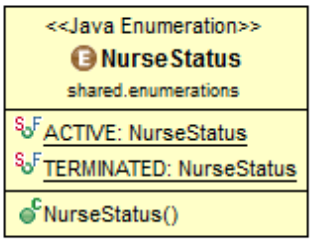
Assignment 3 Document

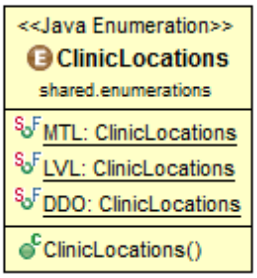
- **Exceptions:**

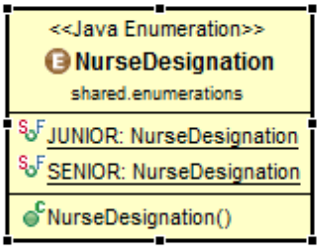


This section of application deals with the exceptions that might be arise from any other section of the application during their execution. Some of these exceptions are listed above like NoRecordTypeException exception that may be arose due to type other than doctor and nurse or RecordIndexOverflowException 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>>();
```

Assignment 3 Document

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

  Communication through web-services and making the compatibility of UDP implementation with web-services became a little concern for us. Otherwise, the rest was based on annotations we required we need to implement in right way.

- ## 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

  To run the project, we need to just run two classes: **publisher.java** and then **login.java** and make sure you enter four digit number after the code in login.