Distributed Class Management System (DCMS) using Web Services

APIs used

1. **Web Services**: for remote methods invocation from clients to servers
2. **Java Socket Programming**: for communication between server instances (MTL, LVL, DDO)
3. **Java Multi-Threading**: for improving system's performance
    a. Clients invoking multiple methods concurrently
    b. Servers handling multiple invocations concurrently
    c. Separated threads to handle special operations: running UDP server
4. **Log4j**: for logging on both clients and servers side

Data Structures

HashMap used to help accessing a part of the Map without locking the whole Map to improve concurrency. Whenever a record need to be created, edited or transferred, the ArrayList storing that record will be locked. This approach prevents unexpected behaviors when multiple threads trying to access the same shared data, while allowing multiple threads to perform unrelated operations concurrently, hence improve performance of the system.

Test cases

- Test all functions one-by-one to make sure the transition from CORBA to Web Services has no problem.
- Create multiple records having the same last name's initial character concurrently. The expected result would be all the records will be created successfully without any duplicated recordID.
- Running multiple threads trying to edit and transfer two records concurrently. The expected result would be any attempt to edit a record before it being transferred will be success, and any attempt to edit a record after it being transferred will be failed. Any editted information would come along with the record to new server.

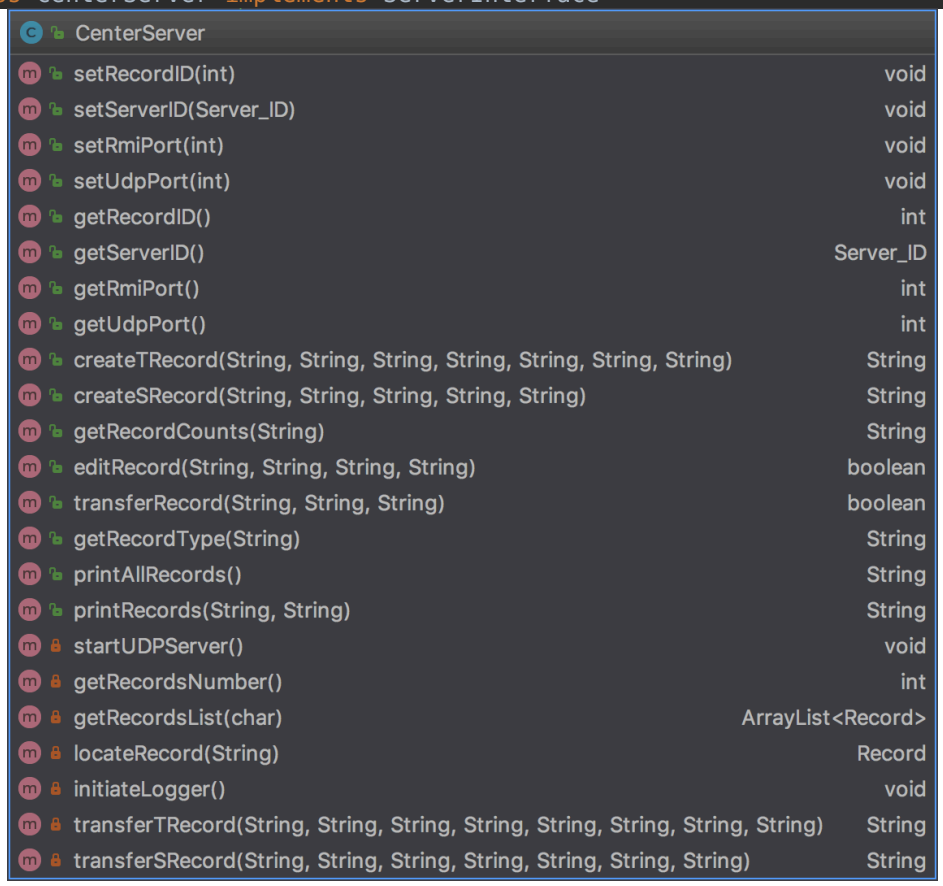
Steps to implement Web services

Define ServerInterface.java

```
@WebService
public interface ServerInterface extends Remote {
    @WebMethod
    String createTRecord(String managerID, String firstName, String lastName, String
address, String phone, String specialization, String location) throws RemoteException,
ServerNotActiveException;
    @WebMethod
    String createSRecord(String managerID, String firstName, String lastName, String
coursesRegistered, String status) throws RemoteException, ServerNotActiveException;
    @WebMethod
    String getRecordCounts(String managerID) throws RemoteException;
    @WebMethod
    boolean editRecord(String managerID, String recordID, String fieldName, String
newValue) throws RemoteException;
    @WebMethod
    boolean transferRecord(String managerID, String recordID, String
remoteCenterServerName) throws RemoteException;
    @WebMethod
    String printRecords(String managerID, String recordID) throws RemoteException;
    @WebMethod
    String printAllRecords() throws RemoteException;
    @WebMethod
    String getRecordType(String recordID) throws RemoteException;
}
```

Implement the interface in CenterServer.java

```
public class CenterServer implements ServerInterface
```



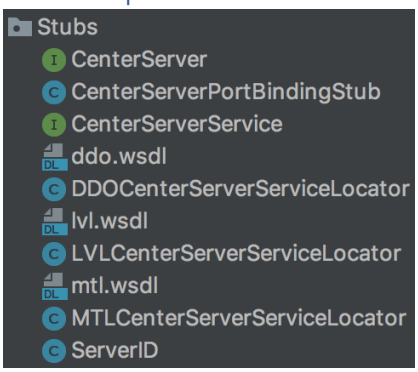
Publish 3 server instances' WSDL in ServersPublisher.java

```
public class ServersPublisher {
    public static void main(String args[]) {
        try {
            CenterServer serverMtl = new CenterServer(Config.Server_ID.MTL);
            Endpoint endpointMtl = Endpoint.publish("http://localhost:8888/mtl",
serverMtl);
            System.out.println("MTL Server published : " + endpointMtl.isPublished());

            CenterServer serverLvl = new CenterServer(Config.Server_ID.LVL);
            Endpoint endpointLvl = Endpoint.publish("http://localhost:8888/lvl",
serverLvl);
            System.out.println("LVL Server published : " + endpointLvl.isPublished());

            CenterServer serverDdo = new CenterServer(Config.Server_ID.DDO);
            Endpoint endpointDdo = Endpoint.publish("http://localhost:8888/ddo",
serverDdo);
            System.out.println("DDO Server published : " + endpointDdo.isPublished());
        } catch (Exception e) {
            System.out.println("ERROR: " + e);
            e.printStackTrace(System.out);
        }
    }
}
```

Generate the Stubs for the client from published WSDL's URLs



Connect clients to servers using the generated stubs

```
private static CenterServer connectToServer(Config.Server_ID serverID) throws
ServiceException {
    switch (serverID) {
        case MTL: {
            Stubs.MTLCenterServerServiceLocator locator = new
Stubs.MTLCenterServerServiceLocator();
            return locator.getCenterServerPort();
        }
        case LVL: {
            Stubs.LVLCenterServerServiceLocator locator = new
Stubs.LVLCenterServerServiceLocator();
            return locator.getCenterServerPort();
        }
        case DDO: {
            Stubs.DDOCenterServerServiceLocator locator = new
Stubs.DDOCenterServerServiceLocator();
            return locator.getCenterServerPort();
        }
    }
    return null;
}
```

Server UML diagram