

## Andy Nguyen (27333870) Assignment 2 Design Documentation

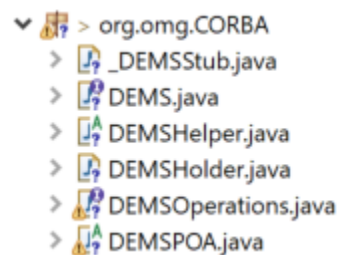
### DEMS.IDL (Distributed Employee Management System Interface Definition Language)

The IDL: allows the development of language and location-independent interfaces to distributed objects.

```
module CORBA {  
    interface DEMS {  
        string createMRecord(in string managerID, in string firstName, in string lastName, in string employeeID, in string emailID, in string projectInfo, in string location);  
        string createERRecord(in string managerID, in string firstName, in string lastName, in string employeeID, in string emailID, in string projectID );  
        string getRecordCounts(in string managerID);  
        boolean editRecord(in string managerID, in string recordID, in string fieldName, in string newValue);  
        boolean transferRecord(in string managerID, in string recordID, in string remoteCenterServerName);  
        string printData();  
        string printRecord(in string managerID, in string recordID);  
    };  
};
```

The functions from the previous RMI assignment remains, with the addition of transferRecord function and the parameter managerID.

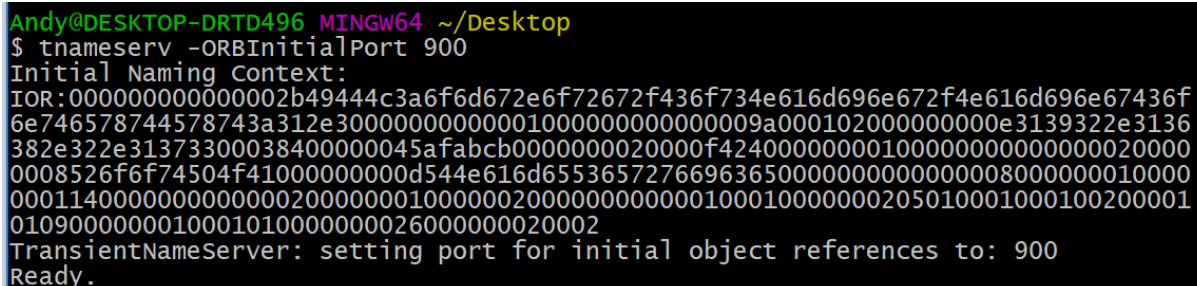
Doing the command “idlj –fall DEMS.idl” in the project will wield these auto created files.



A screenshot of an IDE showing a file explorer for the package `org.omg.CORBA`. The files listed are `_DEMSStub.java`, `DEMS.java`, `DEMSHelper.java`, `DEMSHolder.java`, `DEMSOperations.java`, and `DEMSPOA.java`.

- 1) Client Stub that accesses to IDL Operations
- 2) The signature Interface, which supplies static methods to manipulate type.
- 3) A Helper class which cast object references to the proper type.
- 4) A Holder class that hold public instance of type DEMS.
- 5) Operations Interface that contain all the methods declared in the interface.
- 6) Portable Object Adapter, that assist ORB to deliver client requests. The Adapter interacts with the Server Skeleton to perform Data Marshalling to invoke proper methods.

To Start Server “tnameserv -ORBInitialPort 900 “In Command Prompt.



A screenshot of a terminal window with a black background and green text. It shows the command `$ tnameserv -ORBInitialPort 900` being executed. The output includes the initial naming context (IOR), a long hexadecimal string, and a message from the TransientNameServer setting the port to 900 and becoming ready.

## CenterServerImpl.java

```
public class CenterServerImpl extends DEMSPOA
```

The Server extends the Distributed Employee Management System Portable Object Adapter, because it will be registered to the Portable Object Adapter. The functions will then be recognized and messages dispatched by Object Adapter.

```
50     private ORB orb;  
51  
52     public void setORB(ORB orb_val) {  
53         this.orb = orb_val;  
54     }  
55
```

—Set the ORB when servant initiated.

Setting up the Server:

```
//Servants to be registered to ORB.  
CenterServerImpl serverCA = new CenterServerImpl(CAserverID);  
CenterServerImpl serverUS = new CenterServerImpl(USserverID);  
CenterServerImpl serverUK = new CenterServerImpl(UKserverID);  
  
try {  
  
    ORB orb = ORB.init(args,null);  
    POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));  
    rootPOA.the_POAManager().activate();  
  
    serverCA.setORB(orb);  
    serverUS.setORB(orb);  
    serverUK.setORB(orb);  
  
    org.omg.CORBA.Object refCA = rootPOA.servant_to_reference(serverCA);  
    org.omg.CORBA.Object refUS = rootPOA.servant_to_reference(serverUS);  
    org.omg.CORBA.Object refUK = rootPOA.servant_to_reference(serverUK);  
    DEMS demsServer1 = DEMSHelper.narrow(refCA);  
    DEMS demsServer2 = DEMSHelper.narrow(refUS);  
    DEMS demsServer3 = DEMSHelper.narrow(refUK);  
  
    // Get the root Naming Context  
    org.omg.CORBA.Object objRef = orb.resolve_initial_references(Config.CORBA.NAME_SERVICE);  
    NamingContextExt namingContextRef = NamingContextExtHelper.narrow(objRef);  
  
    // Bind the object reference to the Naming Context  
    NameComponent path1[] = namingContextRef.to_name(CAserverID.name());  
    NameComponent path2[] = namingContextRef.to_name(USserverID.name());  
    NameComponent path3[] = namingContextRef.to_name(UKserverID.name());  
  
    namingContextRef.rebind(path1, demsServer1);  
    namingContextRef.rebind(path2, demsServer2);  
    namingContextRef.rebind(path3, demsServer3);  
}
```

- We connect the ORB core to get the references to the RootPOA.

- We set the servants to the orb. (Each server has its own ORB)
- Get the reference to the RootPOA.
- Initialize the servant and register it to the ORB.
- Get the root naming context.
- Then create object references to bind it with the naming context.
- Run the Threads and finally orb.run() to start the server.

## ManagerClient.java

```
private static void prepareORB(String args[]) throws Exception {
    // Initiate client ORB
    orb = ORB.init(args, null);

    // Get object reference to the Naming Service
    namingContextObj = orb.resolve_initial_references(Config.CORBA.NAME_SERVICE);

    // Narrow the NamingContext object reference to the proper type to be usable (like any CORBA object)
    namingContextRef = NamingContextExtHelper.narrow(namingContextObj);
}

public static void main(String args[]) throws Exception {
    String ClientName;
    String ProjectName;
    prepareORB(args);
    boolean validateRecordID=false;
}
```

I added a prepare Object Request Broker function so that when main function is invoked, the ORB will be ready to take in and use the data passed.

Similar to Assignment 1, but new functionalities added, like transfer record.

```
case 5:
    System.out.print("Enter Record ID to be Tranfered: ");
    recordID = scanner.nextLine().toUpperCase();
    System.out.print("Enter Server Name: ");
    String serverName = scanner.nextLine().toUpperCase();
    DEMS TransferTheRecord = DEMSHelper.narrow(namingContextRef.resolve_str(serverID.name()));
    boolean isSuccess = TransferTheRecord.transferRecord(managerID, recordID, serverName);
    if (isSuccess) {
        LOGGER.info(recordID + " transferred to " + serverName);
        System.out.println(recordID + " transferred to " + serverName);
    }
    else {
        LOGGER.info(recordID + " failed to transfer to " + serverName);
        System.out.println(recordID + " failed to transfer to " + serverName);
    }
    break;
}

//CenterServer CANADAServer = connectToRmiServer((Config.Server_ID.CA));
DEMS CANADAServer = DEMSHelper.narrow(namingContextRef.resolve_str(Config.Server_ID.CA.name()));
```

The key difference from assignment 1 and 2 is that in A2 , we don't create a server object to connect to Java RMI server. But instead create a CORBA object with the

Helper Class to **narrow** it to the appropriate type. This replaces the **lookup()** function of `javax.naming.Context` Interface. Corba uses Internet Inter-ORB Protocol, which makes it possible for distributed programs written in different programming languages to communicate. Java RMI-IIOP provides a mechanism to narrow the Object you have received from your lookup, to the appropriate type.

So the only change in the Client file that I made was replace connect to RMI with narrow with the naming context.

## Multi-Threading

In the transfer record function, we need synchronous to keep the system's data integrity. Lock the `mapRecord` ArrayList so that multiple threads can modify the ArrayList safely.

```
public boolean transferRecord(String managerID, String recordID, String remoteCenterServerName) {
    if (remoteCenterServerName.compareTo(this.serverID.name()) != 0) {
        System.out.println("Hey Man");

        String result = "";

        synchronized (mapRecords) {
            for (ArrayList<Record> recordsList : this.mapRecords.values()) {
                Record recordFound = null;
                Iterator<Record> iterator = recordsList.iterator();
                // ...
            }
        }
    }
}
```

This prevents unpredictable behaviors of the ArrayList, ensures the recount count is always true and logs updated and reflects server's activities correctly.