Finolex Academy of Management & Technology, Ratnagiri

Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

PRACTICAL NO. 4

Remote Object Communication

LOB4: Understand the mechanism of database access with remote objects.
LO4: Implement programs based on database access using remote objects.

- **Java Remote Object model**

In the Java distributed object model, a *remote object* is one whose methods can be invoked from another Java Virtual Machine, potentially on a different host. An object of this type is described by one or more *remote interfaces*, which are Java interfaces that declare the methods of the remote object.

Remote method invocation (RMI) is the action of invoking a method of a remote interface on a remote object. Most importantly, a method invocation on a remote object has the same syntax as a method invocation on a local object.

The Java distributed object model is similar to the Java object model in the following ways:

- A reference to a remote object can be passed as an argument or returned as a result in any method invocation (local or remote).
- A remote object can be cast to any of the set of remote interfaces supported by the implementation using the built-in Java syntax for casting.
- The built-in Java instanceof operator can be used to test the remote interfaces supported by a remote object.

The Java distributed object model differs from the Java object model in these ways:

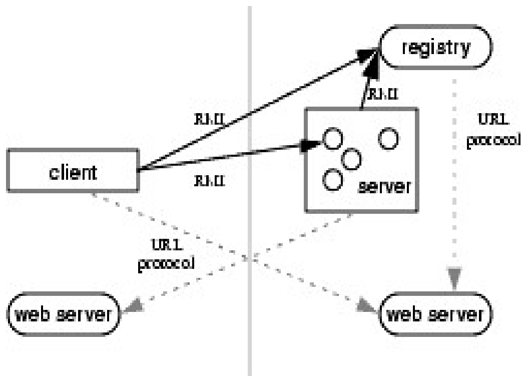
- Clients of remote objects interact with remote interfaces, never with the implementation classes of those interfaces.
- Non-remote arguments to, and results from, a remote method invocation are passed by copy rather than by reference. This is because references to objects are only useful within a single virtual machine.
- A remote object is passed by reference, not by copying the actual remote implementation.
- The semantics of some of the methods defined by class Object are specialized for remote objects.
- Since the failure modes of invoking remote objects are inherently more complicated than the failure modes of invoking local objects, clients must deal with additional exceptions that can occur during a remote method invocation.

RMI applications are often comprised of two separate programs: a server and a client. A typical server application creates a number of remote objects, makes references to those remote objects accessible, and waits for clients to invoke methods on those remote objects. A typical client application gets a remote reference to one or more remote objects in the server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a distributed object application.

Distributed object applications need to:

- ***Locate remote objects:*** Applications can use one of two mechanisms to obtain references to remote objects. An application can register its remote objects with RMI's simple naming facility, the rmiregistry, or the application can pass and return remote object references as part of its normal operation.
- ***Communicate with remote objects:*** Details of communication between remote objects are handled by RMI; to the programmer, remote communication looks like a standard method invocation.
- ***Load class bytecodes for objects that are passed as parameters or return values:*** Because RMI allows a caller to pass objects to remote objects, RMI provides the necessary mechanisms for loading an object's code as well as transmitting its data.

The illustration below depicts an RMI distributed application that uses the registry to obtain references to a remote object. The server calls the registry to associate a name with a remote object. The client looks up the remote object by its name in the server's registry and then invokes a method on it. The illustration also shows that the RMI system uses an existing web server to load bytecodes of classes written in the Java programming language, from server to client and from client to server, for objects when needed. RMI can load class bytecodes using any URL protocol (e.g., HTTP, FTP etc.) that is supported by the Java platform.



RMI mechanism for remote object access

Exercise:

a) **Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept.**

DBServiceSr.java:

```
    package library;


import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class DBServiceSr {

    public DBServiceSr() {
            super();
            // TODO Auto-generated constructor stub
    }
    public static void main(String[] args) {
            // TODO Auto-generated method stub
            try {
                    LibraryDBInf skeleton = new DBOperationService();
                    LocateRegistry.createRegistry(1900);
                    Naming.rebind("rmi://localhost:1900/ROCforLibraryDB", skeleton);

            }catch (Exception e) {
                    // TODO: handle exception
                    e.printStackTrace();
            }

    }

}
```

LibraryDBInf.java:

```
package library;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface LibraryDBInf extends Remote {
    public String getData(String strQry) throws RemoteException;
    public String insertData(String strQry) throws RemoteException;

}
```

DBOperationService.java:

```
package library;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import com.mysql.cj.jdbc.result.ResultSetMetaData;

public class DBOperationService extends UnicastRemoteObject implements LibraryDBInf  {

    Connection con;
    Statement stmt;
    ResultSet rs;
    ResultSetMetaData rsmd;
    String colStr, resultStr;
    public DBOperationService() throws RemoteException {
            super();
```

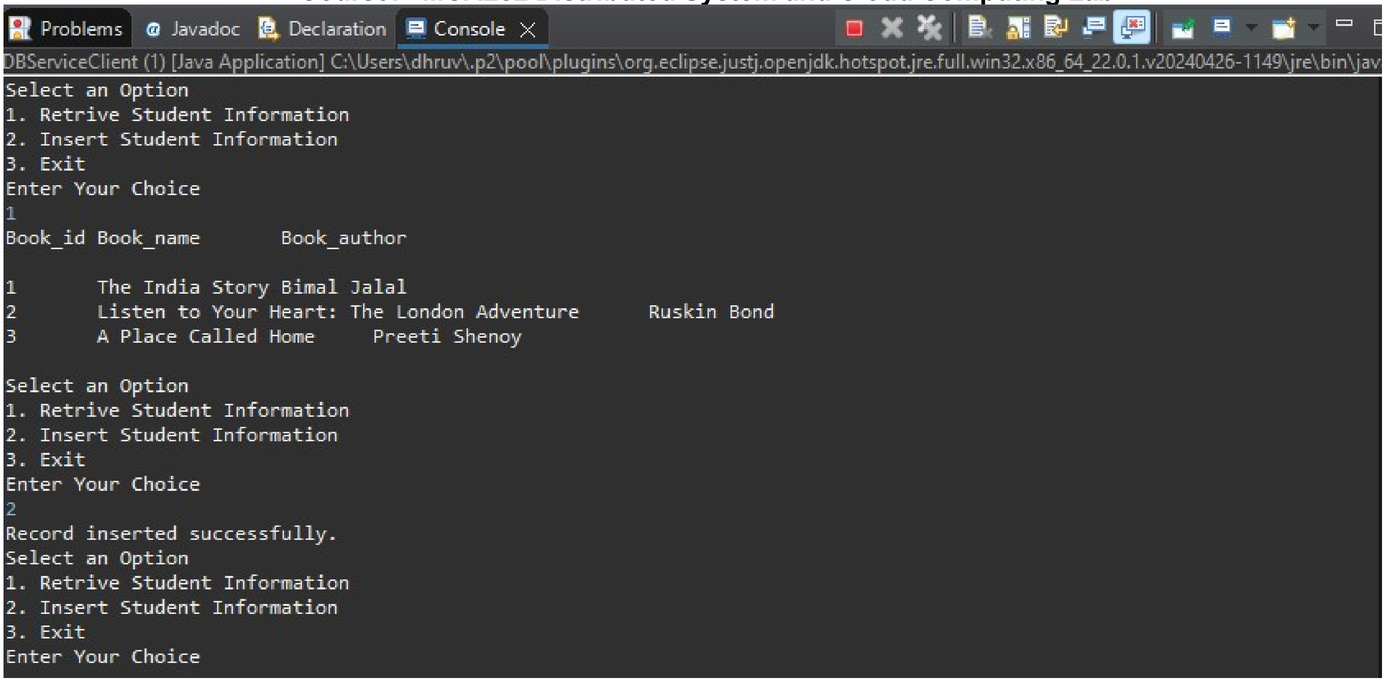
```
		con = null;
		stmt = null;
		rs = null;
		rsmd = null;
		colStr = "";
		resultStr = "";
		// TODO Auto-generated constructor stub
	}

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;

	public void setDBCon() {
		try {
			String URL = "jdbc:mysql://localhost:3307/library";
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection(URL,"root","");
		}
		catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}
	}

	@Override
	public String getData(String strQry) throws RemoteException {
		// TODO Auto-generated method stub
		try {
			setDBCon();
			System.out.println("Server Registered.");
			stmt = con.createStatement();
			rs = stmt.executeQuery(strQry);
			rsmd = (ResultSetMetaData) rs.getMetaData();
			for(int i=1; i<=rsmd.getColumnCount();i++) {
				colStr = colStr + rsmd.getColumnName(i) + "\t";
			}
			while(rs.next()){
				for(int i=1; i<=rsmd.getColumnCount();i++) {
					resultStr = resultStr + rs.getString(i) + "\t";
				}
				resultStr = resultStr + "\n";
			}
		}
		catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}
		return colStr + "\n\n" + resultStr;
	}

	@Override
	public String insertData(String strQry) throws RemoteException {
		// TODO Auto-generated method stub
		try {
			setDBCon();
			System.out.println("Server Registered.");
			stmt=con.createStatement();
			int recordInserted=stmt.executeUpdate(strQry);
			if(recordInserted != 0)
			{
				resultStr="Record inserted successfully.";
			}
			else
			{
				resultStr="Record not Inserted successfully.";
			}

		}
		catch (Exception e) {
			e.printStackTrace();
			// TODO: handle exception
		}
		return resultStr;
	}

}
```

DBServiceClient.java:

```
package library;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.rmi.Naming;

public class DBServiceClient {

	public DBServiceClient() {
	super();
	// TODO Auto-generated constructor stub
	}
	public static void main(String[] args) {
		String sql = "", ch = "";
		// TODO Auto-generated method stub
		try {
		LibraryDBInf stub = (LibraryDBInf) Naming.lookup("rmi://localhost:1900/ROCforLibraryDB");
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
                while (true) {
                        System.out.println("Select an Option");
                        System.out.println("1. Retrive Student Information ");
                        System.out.println("2. Insert Student Information ");
                        System.out.println("3. Exit");
                        System.out.println("Enter Your Choice");

                        ch = br.readLine();
                        if (ch.equals("1")) {
                                sql = "SELECT * FROM Book";
                                sql = stub.getData(sql);
                        } else if (ch.equals("2")) {
                                sql = "INSERT INTO Book VALUES(4,'Nation Calling','IAS officer Sonal Goel')";
                                sql = stub.insertData(sql);
                        } else if (ch.equals("3")) {
                                System.exit(0);
                        } else {
                                sql = "Please select Valid Option";
                        }
                        System.out.println(sql);

                }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }

    }

}
```

Output:



```
DBServiceClient (1) [Java Application] C:\Users\dhruv\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\jav
Select an Option
1. Retrive Student Information
2. Insert Student Information
3. Exit
Enter Your Choice
1
Book_id Book_name       Book_author

1       The India Story Bimal Jalal
2       Listen to Your Heart: The London Adventure      Ruskin Bond
3       A Place Called Home     Preeti Shenoy

Select an Option
1. Retrive Student Information
2. Insert Student Information
3. Exit
Enter Your Choice
2
Record inserted successfully.
Select an Option
1. Retrive Student Information
2. Insert Student Information
3. Exit
Enter Your Choice
```

b) Using MySQL create Electric_Bill database. Create table Bill (consumer_name, bill_due_date, bill_amount) and retrieve the bill information from the Electric_Bill database using Remote Object Communication concept.

DBServerService.java:

```
package electricity;

    import java.rmi.Naming;
    import java.rmi.registry.LocateRegistry;


    public class DBServiceSr {
        public DBServiceSr() {
                super();
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                try {
                        ElectricityDBInf skeleton = new DBOperationService();
                        LocateRegistry.createRegistry(1900);
                        Naming.rebind("rmi://localhost:1900/ROCforElectricityDB", skeleton);

                }catch (Exception e) {
                        // TODO: handle exception
                        e.printStackTrace();
                }


    }
        }
```

ElectricityDBInf.java:

```
package electricity;
```

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ElectricityDBInf extends Remote {
      public String getData(String strQry) throws RemoteException;
      public String insertData(String strQry) throws RemoteException;
}
```

DBOperationService.java

```
            package electricity;

            import java.rmi.RemoteException;
            import java.rmi.server.UnicastRemoteObject;
            import java.sql.Connection;
            import java.sql.DriverManager;
            import java.sql.ResultSet;
            import java.sql.Statement;

            import com.mysql.cj.jdbc.result.ResultSetMetaData;

            public class DBOperationService extends UnicastRemoteObject implements ElectricityDBInf {
                Connection con;
                Statement stmt;
                ResultSet rs;
                ResultSetMetaData rsmd;
                String colStr, resultStr;

                public DBOperationService() throws RemoteException {
                            super();
                            con = null;
                            stmt = null;
                            rs = null;
                            rsmd = null;
                            colStr = "";
                            resultStr = "";
                            // TODO Auto-generated constructor stub
                }

                /**
                 *
                 */
                private static final long serialVersionUID = 1L;

                public void setDBCon() {
                            try {
                                        String URL = "jdbc:mysql://localhost:3307/bill";
                                        Class.forName("com.mysql.cj.jdbc.Driver");
                                        con = DriverManager.getConnection(URL, "root", "");
                            } catch (Exception e) {
                                        // TODO: handle exception
                                        e.printStackTrace();
                            }
                }

                @Override
                public String getData(String strQry) throws RemoteException {
                            // TODO Auto-generated method stub
                            try {
                                        setDBCon();
```

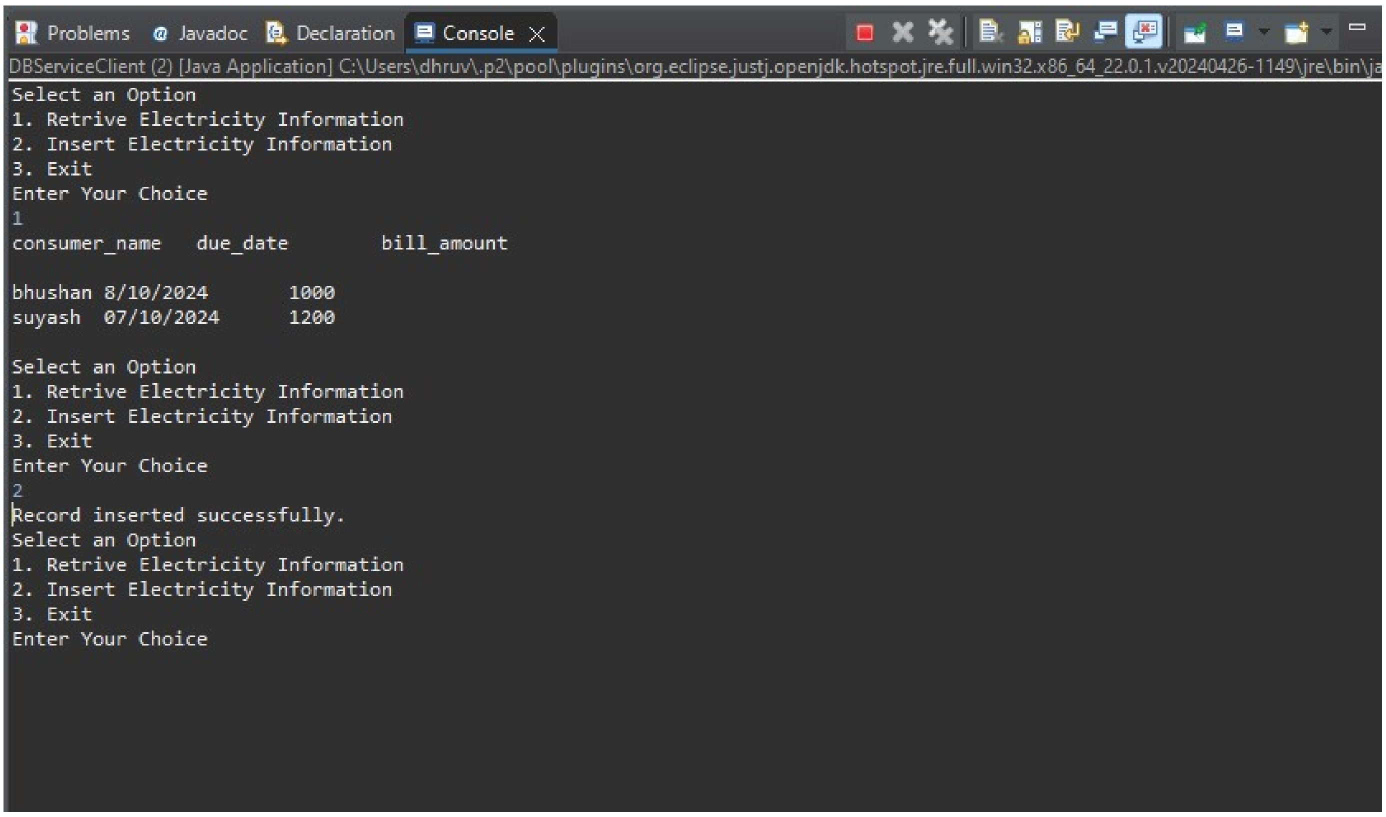
```
                        System.out.println("Server Registered.");
                        stmt = con.createStatement();
                        rs = stmt.executeQuery(strQry);
                        rsmd = (ResultSetMetaData) rs.getMetaData();
                        for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                                colStr = colStr + rsmd.getColumnName(i) + "\t";
                        }
                        while (rs.next()) {
                                for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                                        resultStr = resultStr + rs.getString(i) + "\t";
                                }
                                resultStr = resultStr + "\n";
                        }
                } catch (Exception e) {
                        // TODO: handle exception
                        e.printStackTrace();
                }
                return colStr + "\n\n" + resultStr;
        }

        @Override
        public String insertData(String strQry) throws RemoteException {
                // TODO Auto-generated method stub
                try {
                        setDBCon();
                        System.out.println("Server Registered.");
                        stmt = con.createStatement();
                        int recordInserted = stmt.executeUpdate(strQry);
                        if (recordInserted != 0) {
                                resultStr = "Record inserted successfully.";
                        } else {
                                resultStr = "Record not Inserted successfully.";
                        }

                } catch (Exception e) {
                        e.printStackTrace();
                        // TODO: handle exception
                }
                return resultStr;
        }

      }
```

DBServiceClient.java:

```
        package electricity;

      import java.io.BufferedReader;
      import java.io.InputStreamReader;
      import java.rmi.Naming;

      public class DBServiceClient {

        public DBServiceClient() {
                super();
                // TODO Auto-generated constructor stub
        }
        public static void main(String[] args) {
                String sql = "", ch = "";
                // TODO Auto-generated method stub
                try {
                ElectricityDBInf                                stub                                =
   (ElectricityDBInf)Naming.lookup("rmi://localhost:1900/ROCforElectricityDB");
```

```
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                while (true) {
                        System.out.println("Select an Option");
                        System.out.println("1. Retrive Electricity Information ");
                        System.out.println("2. Insert Electricity Information ");
                        System.out.println("3. Exit");
                        System.out.println("Enter Your Choice");

                        ch = br.readLine();
                        if (ch.equals("1")) {
                                sql = "SELECT * FROM bill";
                                sql = stub.getData(sql);
                        } else if (ch.equals("2")) {
                                sql = "INSERT INTO bill VALUES('aditya','06/10/2024',1000)";
                                sql = stub.insertData(sql);
                        } else if (ch.equals("3")) {
                                System.exit(0);
                        } else {
                                sql = "Please select Valid Option";
                        }
                        System.out.println(sql);

                }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }

        }

}
```

Output:



Problems Javadoc Declaration Console

```
DBServiceClient (2) [Java Application] C:\Users\dhruv\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\ja
Select an Option
1. Retrive Electricity Information
2. Insert Electricity Information
3. Exit
Enter Your Choice
1
consumer_name   due_date        bill_amount

bhushan 8/10/2024       1000
suyash  07/10/2024      1200

Select an Option
1. Retrive Electricity Information
2. Insert Electricity Information
3. Exit
Enter Your Choice
2
Record inserted successfully.
Select an Option
1. Retrive Electricity Information
2. Insert Electricity Information
3. Exit
Enter Your Choice
```