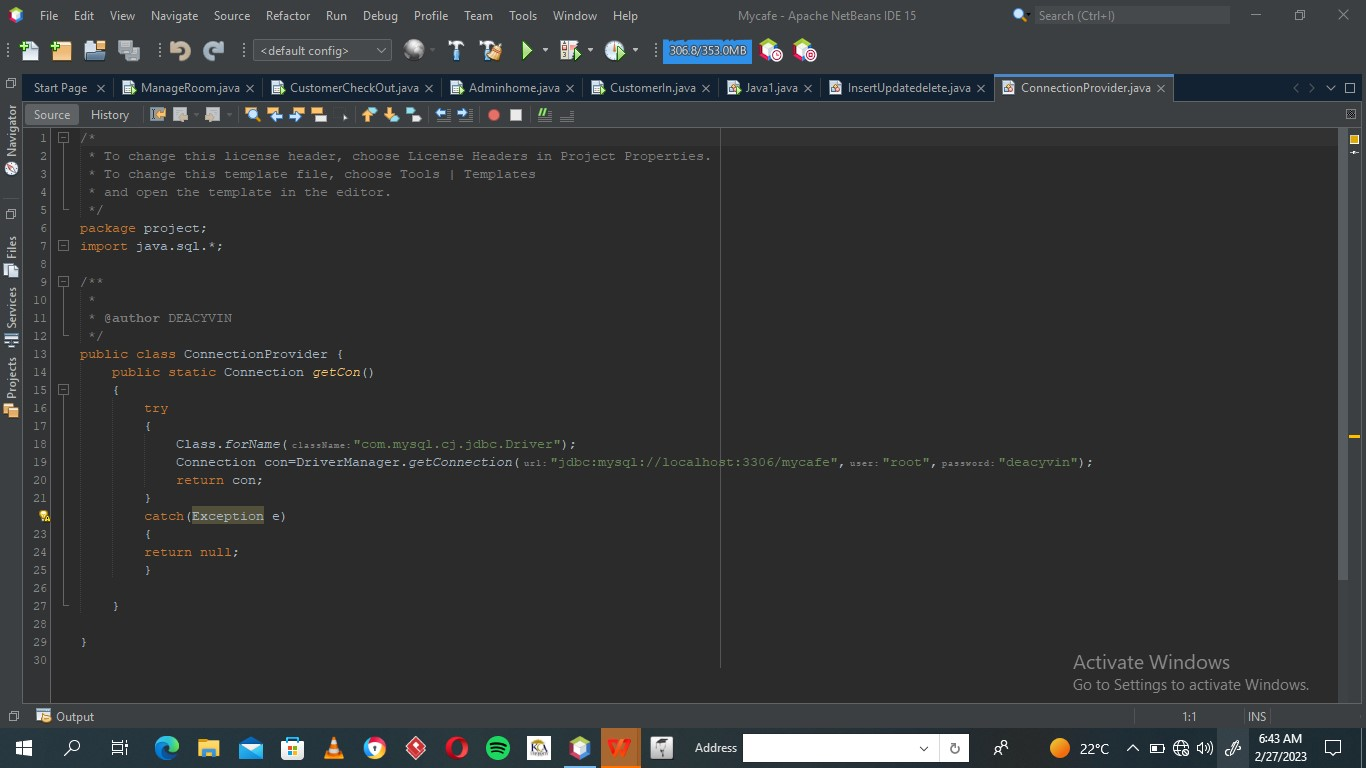
VIN JAVA



* RMI

**import** java.rmi.\*;

**public** **interface** Search **extends** Remote

{

    // Declaring the method prototype

**public** String query(String search) **throws** RemoteException;

}

|  |
| --- |
| // Java program to implement the Search interface  **import** java.rmi.\*;  **import** java.rmi.server.\*;  **public** **class** SearchQuery **extends** UnicastRemoteObject  **implements** Search  {      // Default constructor to throw RemoteException      // from its parent constructor      SearchQuery() **throws** RemoteException      {  **super**();      }        // Implementation of the query interface  **public** String query(String search)  **throws** RemoteException      {          String result;  **if** (search.equals("Reflection in Java"))              result = "Found";  **else**              result = "Not Found";    **return** result;      }  } |

// Java program for server application

**import** java.rmi.\*;

**import** java.rmi.registry.\*;

**public** **class** SearchServer

{

**public** **static** **void** main(String args[])

    {

**try**

        {

            // Create an object of the interface

            // implementation class

            Search obj = **new** SearchQuery();

            // rmiregistry within the server JVM with

            // port number 1900

            LocateRegistry.createRegistry(1900);

            // Binds the remote object by the name

            // geeksforgeeks

            Naming.rebind("<rmi://localhost:1900>"+

                          "/geeksforgeeks",obj);

        }

**catch**(Exception ae)

        {

            System.out.println(ae);

        }

    }

}

// Java program for client application

**import** java.rmi.\*;

**public** **class** ClientRequest

{

**public** **static** **void** main(String args[])

    {

        String answer,value="Reflection in Java";

**try**

        {

            // lookup method to find reference of remote object

            Search access =

                (Search)Naming.lookup("<rmi://localhost:1900>"+

                                      "/geeksforgeeks");

            answer = access.query(value);

            System.out.println("Article on " + value +

                            " " + answer+" at GeeksforGeeks");

        }

**catch**(Exception ae)

        {

            System.out.println(ae);

        }

    }

}

Network programming

**Establishing communications through sockets (server programming)**

// A Java program for a Server

import java.net.\*;

import java.io.\*;

public class Server

{

//initialize socket and input stream

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

// constructor with port

public Server(int port)

{

// starts server and waits for a connection

try

{

server = new ServerSocket(port);

System.out.println("Server started");

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client accepted");

// takes input from the client socket

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

String line = "";

// reads message from client until "Over" is sent

while (!line.equals("Over"))

{

try

{

line = in.readUTF();

System.out.println(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

System.out.println("Closing connection");

// close connection

socket.close();

in.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Server server = new Server(5000);

}

}