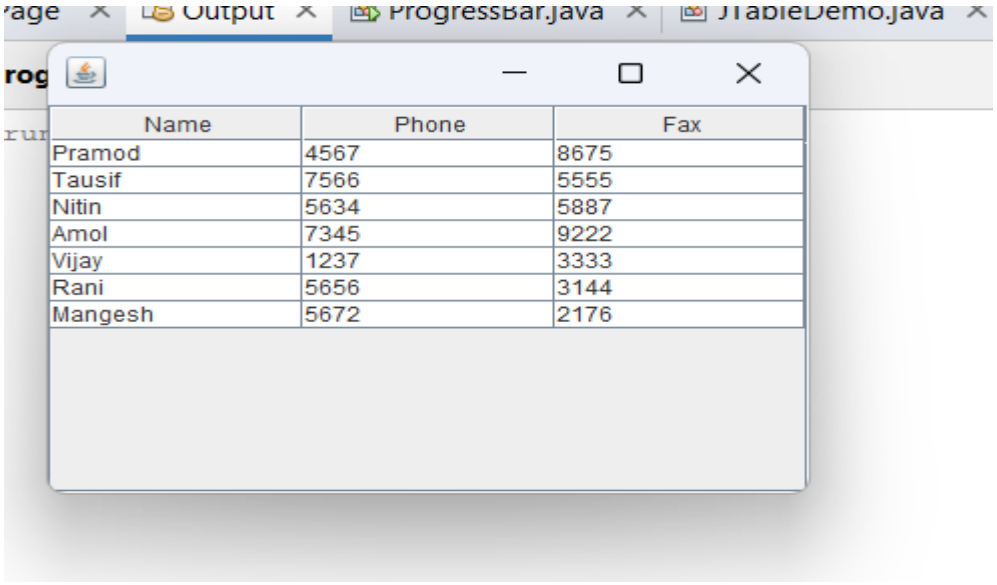
## Practical No : 8

Write a Program to create a JTable .

```
import java.awt.*;
import javax.swing.*;
public class JTableDemo extends JFrame {
JTableDemo()
{
      final String[] colHeads = { "Name", "Phone", "Fax" };
      final Object[][] data = {
      { "Pramod", "4567", "8675" },
      { "Tausif", "7566", "5555" },
      { "Nitin", "5634", "5887" },
      { "Amol", "7345", "9222" },
      { "Vijay", "1237", "3333" },
      { "Rani", "5656", "3144" },
      { "Mangesh", "5672", "2176" }
      };


      JTable table = new JTable(data, colHeads);
      int v = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
      int h = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
      JScrollPane jsp = new JScrollPane(table, v, h);
      add(jsp);
}
   public static void main(String a[]) {
   JTableDemo jt=new JTableDemo();
   jt.setSize(300,300);
   jt.setVisible(true);
   }}
```

Output:

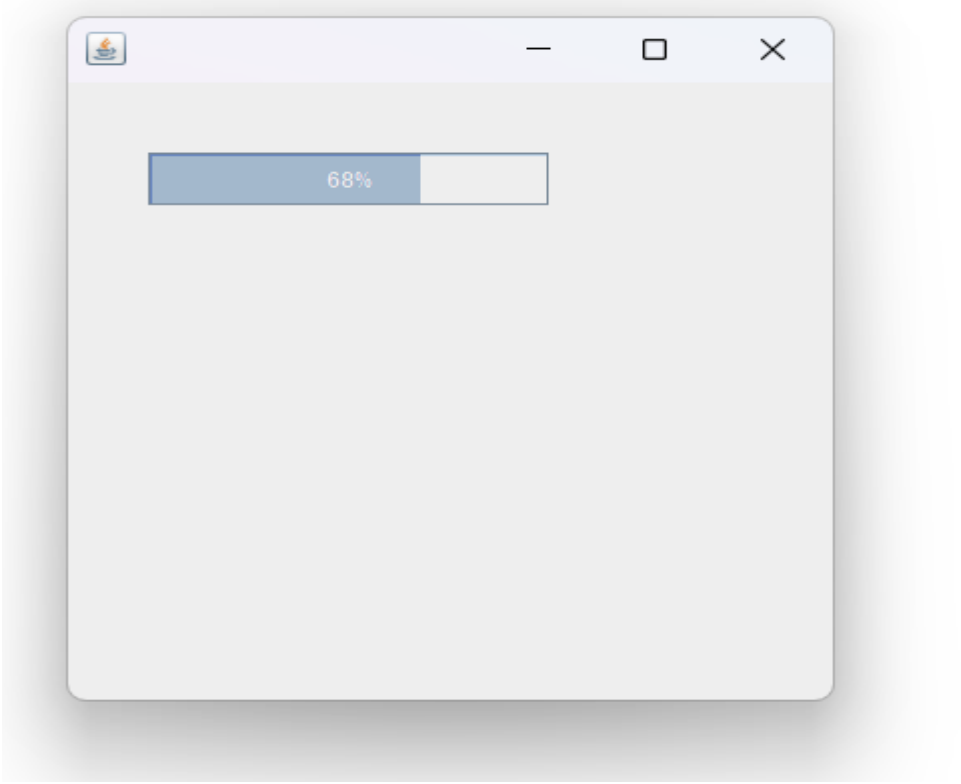


**Practical No :9**

Write a program to launch a JProgressBar..

```
import javax.swing.*;

public class MyProgress extends JFrame
{
   JProgressBar jb;
   int i=0,num=0;
   MyProgress()
   {
      jb=new JProgressBar(0,2000);
      jb.setBounds(40,40,200,30);
      jb.setValue(0);
      jb.setStringPainted(true);
      add(jb);
      setSize(400,400);
      setLayout(null);
   }


      public void iterate()
      {
         while(i<=2000)
         {
            jb.setValue(i);
            i=i+20;
          try
         {
            Thread.sleep(150);
         }
            catch(Exception e){}
         }
      }
      public static void main(String[] args)
      {
         MyProgress m=new MyProgress();
         m.setVisible(true);
         m.iterate();
      }
```

}

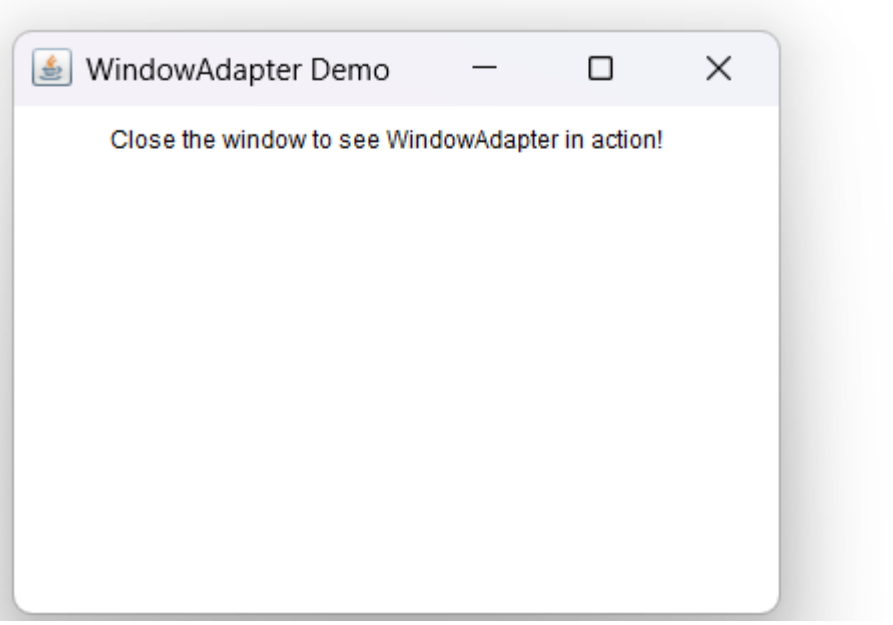

68%

**Practical No: 13**

**Write a program to demonstrate the use of windowAdapter class .**

```
import java.awt.*;
import java.awt.event.*;

public class WindowAdapterDemo extends Frame {

   public WindowAdapterDemo() {
      // Set the frame title
      setTitle("WindowAdapter Demo");
      setSize(400, 300);
      setLayout(new FlowLayout());

      // Add a label to the frame
      Label label = new Label("Close the window to see WindowAdapter in
action!");
      add(label);

      // Add a WindowAdapter to handle window events
      addWindowListener(new WindowAdapter() {
         // Override the windowClosing method
         @Override
         public void windowClosing(WindowEvent e) {
            System.out.println("Window is closing...");
            dispose(); // Close the frame
         }

         // Optionally, override other methods (e.g., windowOpened)
         @Override
         public void windowOpened(WindowEvent e) {
            System.out.println("Window opened!");
         }
      });
   }

   public static void main(String[] args) {
      // Create an instance of the frame
      WindowAdapterDemo frame = new WindowAdapterDemo();
```

```
        // Make the frame visible
        frame.setVisible(true);
    }
}
```




Practical No:16

Write a program to implement chat server using servers socket and socket class.

---

Server Program

```
import java.net.*;
import java.io.*;

public class MyServer1_Demo {
    public static void main(String[] args) {
        ServerSocket serverSocket = null;

        try {
            // Create a server socket on port 3333
            serverSocket = new ServerSocket(3333);
            System.out.println("Server is running and waiting for a client...");

            // Accept client connection
            Socket socket = serverSocket.accept();
            System.out.println("Client connected!");

            // Create input and output streams
            DataInputStream din = new DataInputStream(socket.getInputStream());
```

```
            DataOutputStream dout = new DataOutputStream(socket.getOutputStream());
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

            String clientMessage = "", serverResponse = "";

            // Communication loop
            while (!clientMessage.equals("stop")) {
                clientMessage = din.readUTF(); // Read message from client
                System.out.println("Client says: " + clientMessage);

                // Read response from server console
                serverResponse = br.readLine();
                dout.writeUTF(serverResponse); // Send response to client
                dout.flush();
            }

            // Close resources
            din.close();
            dout.close();
            socket.close();
            System.out.println("Server stopped.");
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        } finally {
            try {
                if (serverSocket != null) {
                    serverSocket.close(); // Release the port
                }
            } catch (IOException e) {
                System.err.println("Failed to close server socket: " + e.getMessage());
            }
        }
    }
}
```

Client program

```
import java.net.*;
import java.io.*;

public class MyClient1_Demo {
    public static void main(String[] args) {
        Socket socket = null;

        try {
            // Connect to the server on localhost and port 3333
            socket = new Socket("localhost",3333);
            System.out.println("Connected to the server!");

            // Create input and output streams
            DataInputStream din = new DataInputStream(socket.getInputStream());
            DataOutputStream dout = new DataOutputStream(socket.getOutputStream());
```

```
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        String clientMessage = "", serverResponse = "";

        // Communication loop
        while (!clientMessage.equals("stop")) {
            // Read message from client console
            clientMessage = br.readLine();
            dout.writeUTF(clientMessage); // Send message to server
            dout.flush();

            // Read response from server
            serverResponse = din.readUTF();
            System.out.println("Server says: " + serverResponse);
        }

        // Close resources
        dout.close();
        din.close();
        System.out.println("Client disconnected.");
    } catch (IOException e) {
        System.err.println("Error: " + e.getMessage());
    } finally {
        try {
            if (socket != null) {
                socket.close(); // Close the client socket
            }
        } catch (IOException e) {
            System.err.println("Failed to close client socket: " + e.getMessage());
        }
    }
  }
}
```

```
run:
Connected to the server!

hello
Server says: hello
Server says: i like
ok bro
Server says: stop
stop
Server says: stop
Client disconnected.
BUILD SUCCESSFUL (total time: 3 minutes 13 seconds)
```


```
run:
Connected to the server!

hello
Server says: hello
Server says: i like
ok bro
Server says: stop
stop
Server says: stop
Client disconnected.
BUILD SUCCESSFUL (total time: 3 minutes 13 seconds)
```

Practical No:17
Write a program to demonstrate use of datagram Socket and Datagram Packet .

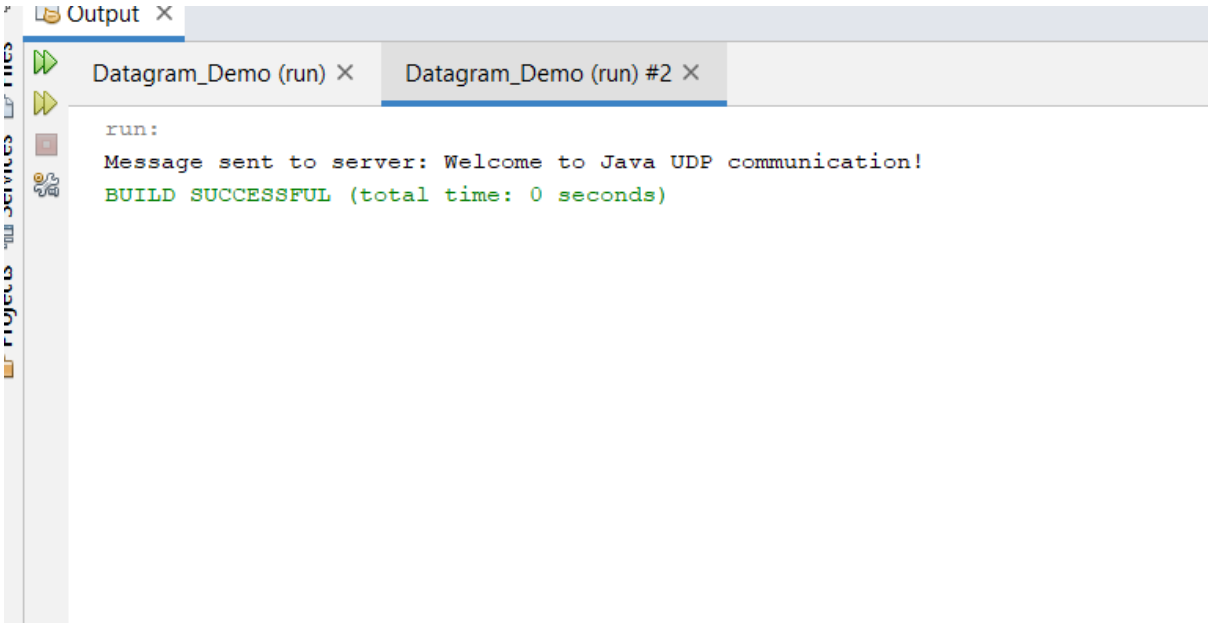
Datagram_Clent

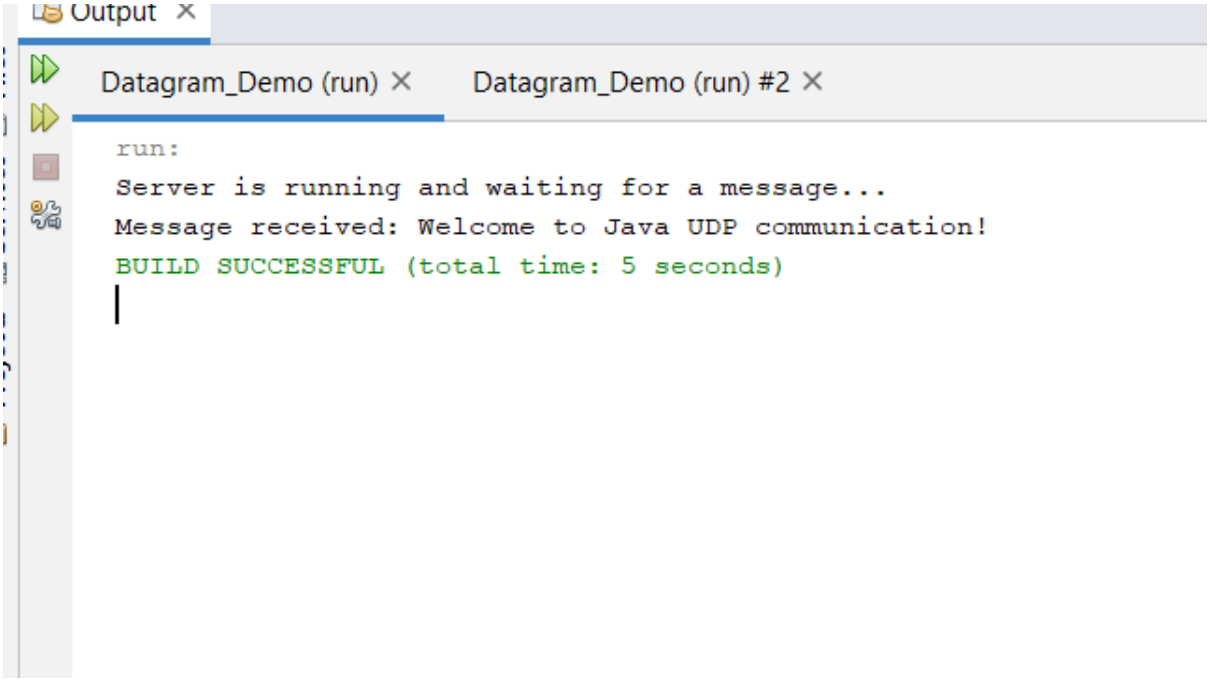
```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class DClient {
   public static void main(String[] args) {
      try {
         // Create a DatagramSocket to send data
         DatagramSocket ds = new DatagramSocket();

         // Message to send
         String message = "Welcome to Java UDP communication!";
         byte[] messageBytes = message.getBytes();

         // Get the server's IP address (localhost in this case)
         InetAddress serverAddress = InetAddress.getByName("127.0.0.1");

         // Create a DatagramPacket to send the message
         DatagramPacket dp = new DatagramPacket(messageBytes, messageBytes.length,
serverAddress, 3000);

         // Send the packet
         ds.send(dp);
         System.out.println("Message sent to server: " + message);

         // Close the socket
         ds.close();
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
}
```

Datagram_Server

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class DServer {
   public static void main(String[] args) {
      try {
         // Create a DatagramSocket to listen on port 3000
         DatagramSocket ds = new DatagramSocket(3000);
         System.out.println("Server is running and waiting for a message...");

         // Buffer to store incoming data
```

```
            byte[] buf = new byte[1024];

            // Create a DatagramPacket to receive data
            DatagramPacket dp = new DatagramPacket(buf, buf.length);

            // Receive the packet (blocking call)
            ds.receive(dp);

            // Convert the received data to a string
            String receivedMessage = new String(dp.getData(), 0, dp.getLength());
            System.out.println("Message received: " + receivedMessage);

            // Close the socket
            ds.close();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```




Output
Datagram_Demo (run)
Datagram_Demo (run) #2
run:
Server is running and waiting for a message...
Message received: Welcome to Java UDP communication!
BUILD SUCCESSFUL (total time: 5 seconds)