Practical -10 Socket programming and Action Listener Event

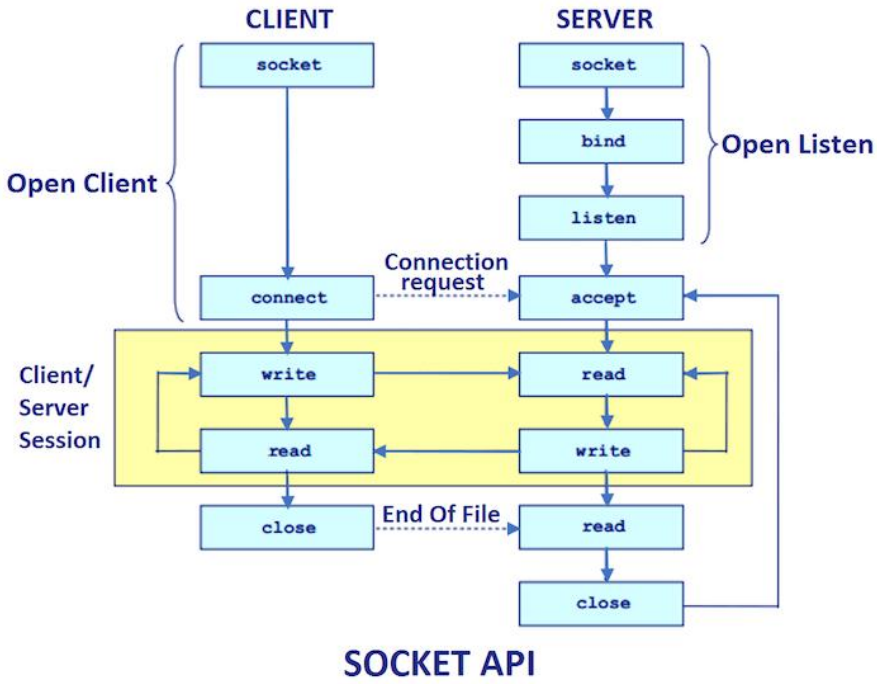
Java Socket programming is used for communication between the applications running on different JRE.
Java Socket programming can be connection-oriented or connection-less.
Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

TCP/IP is used in connection-oriented Programming
UDP is used in Connection-less programming.

TCP is suited for applications that require high reliability, and transmission time is relatively less critical
UDP is suitable for applications that need fast, efficient transmission, such as games.

● Socket Class

A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.
There are Two kinds of TCP sockets in Java. One is for servers, and the other is for clients.
1. **The ServerSocket class** is designed to be a "listener," which waits for clients to connect before doing anything.
2. The Socket class is designed to connect to server sockets and to initiate protocol exchanges. The Socket class is for clients.he creation of a Socket object implicitly establishes a connection between the client and server.



For socket programming at client side IP Address of Server, and Port number is required.

Socket class

The Socket class can be used to create a socket and which is used to communicatie with server.

Create Client Sockets

Socket(String hostname,int port) throws UnknownHostException,IOException	Create a socket connected to named host and port
Socket(InetAddress ipaddress,int port) throws UnknownHostException,IOException	Create a socket connected to given ipaddress and port

Methods for Socket

InetAddress getInetAddress()	Returns the InetAddress associated with Socket objet. It returns null if the socket is not connected
Int getPort()	Returns the remote port to which the Socket is connected. Return 0 if the socket is not connected
Int getLocalPort()	Returns the local port to which the invoking socket object is bound. Return -1 if it is not bound
public InputStream getInputStream()	returns the InputStream attached with this socket.
public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
void close()	closes this socket

ServerSocket class

ServerSocket class can be used to create a socket to serverside.

Create Server Sockets

Syntax ServerSocket ServerSocket(int portno);

public Socket accept()	returns the socket and establish a connection between server and client.
void close()	closes the server socket.
public InputStream getInputStream()	returns the InputStream attached with this

	socket.
public OutputStream getOutputStream()	returns the OutputStream attached with this socket.

- ## Steps for implementing Server

1. Open the Server Socket

```
DataOutputStream os;
DataInputStream is;
ServerSocket server = new ServerSocket(PORT);
```

2. Wait for the Client Request

```
Socket client = server. accept();
```

3. Create I/O streams for communicating to the client

```
is = new DataInputStream(client.getInputStream());
os = new DataOutputStream(client.getOutputStream());
```

4.Perform communication with client

```
String line = is.readLine();
os.writeBytes("Hello\n");
```

5. Close sockets

```
client.close();
```

- ## Implementing Client

1. Create a Socket Object:

```
client = new Socket(server, port_id );
```

2 Create I/O streams for communicating with the server..

```
is = new DataInputStream(client.getInputStream() );
os = new DataOutputStream( client.getOutputStream() );
```

3. Perform I/O or communication with the server:

```
String line = is.readLine();
os.writeBytes("Hello\n");
```

4.Close the socket when done:

```
client.close();
```

```
import java.io.*;
import java.net.*;
import java.util.*;
class Server2 {

        public static void main(String args[])
                throws Exception
        {
```

```
                // Create server Socket
                ServerSocket ss = new ServerSocket(5544);

                // connect it to client socket
                Socket s = ss.accept();
                System.out.println("Connection established");

                // to send data to the client BufferedWriter
                PrintStream ps
                        = new PrintStream(s.getOutputStream());

    //PrintWriter ps  = new PrintWriter(s.getOutputStream(), true);
        // to read data coming from the client
                BufferedReader br
                        = new BufferedReader(
                                new InputStreamReader(
                                        s.getInputStream()));

                // to read data from the keyboard
            Scanner input = new Scanner(System.in);



                // server executes continuously
                while (true) {

                        String str, str1;

                        // repeat as long as the client
                        // does not send a null string

                        // read from client
                        while ((str = br.readLine()) != null) {
                                System.out.println(str);

                  str1 = input.nextLine();

//
                                // send to client
                                ps.println(str1);

                }
       //  str1=input.nextLine();
         System.out.println("End" );
                                // close connection
                //        dos.close();
                                br.close();
                                //kb.close();
                                ss.close();
                                s.close();
```

```
                        // terminate application
                        System.exit(0);

                } // end of while
        }
}
```

Client program

```
import java.io.*;
import java.net.*;
import java.util.*;
class Client2 {

        public static void main(String args[])
                throws Exception
        {
        // Create client socket
                Socket s = new Socket("127.0.0.1", 5544);
                // to send data to the server
                DataOutputStream dos
                        = new DataOutputStream(
                                s.getOutputStream());
                // to read data coming from the server
                BufferedReader br
                        = new BufferedReader(
                                new InputStreamReader(
                                        s.getInputStream()));

      DataInputStream dis  = new DataInputStream(s.getInputStream());

        // to read data from the keyboard

      Scanner input = new Scanner(System.in);
        String str, str1;
                // repeat as long as exit
                // is not typed at client
            str=input.nextLine();
                while (!(str.equals("end"))) {
                        // send to the server
                        dos.writeChars(str + "\n");
                        //wait from server response
                        str1 = br.readLine();
                        System.out.println(str1);
                        str = input.nextLine();
```

```
                }
                // close connection.
                dos.close();
                br.close();
                //kb.close();
                s.close();
        }
}
```

- Creating Frame

```
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

import javax.swing.*;

public class SwingDemo implements WindowListener {
 JFrame window ;
 JButton jb;
 JLabel jl;
 public SwingDemo() {

   window = new JFrame("Window State Listener");
  JPanel jp = new JPanel();
   jp.setBounds(40,80,200,200);
   jb = new JButton("click me");

   jl = new JLabel("Swing demo");
   window.setBounds(30, 30, 300, 300);
   window.addWindowListener(this);
   window.setVisible(true);
//    window.add(jl);
   //window.add(jb);
   jp.add(jl);
   jp.add(jb);
   window.add(jp);
 }

 public static void main(String[] args) {
   new SwingDemo();
 }

 public void windowClosing(WindowEvent e) {
   System.out.println("Closing");
```

```
    window.dispose();

  }

  public void windowOpened(WindowEvent e) {
    System.out.println("Opened");
  }

  public void windowClosed(WindowEvent e) {
    System.out.println("Closed");
    System.exit(0);
  }

  public void windowIconified(WindowEvent e) {
    System.out.println("Iconified");
  }

  public void windowDeiconified(WindowEvent e) {
    System.out.println("Deiconified");
  }

  public void windowActivated(WindowEvent e) {
    System.out.println("Activated");
  }

  public void windowDeactivated(WindowEvent e) {
    System.out.println("Deactivated");
  }
}
```

ActionListner

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class SumSwingApp implements ActionListener{
    private JTextField textField1, textField2;
    private JButton button1, button2;
    private JLabel jb1,jb2 ,jb3;
    JFrame jf;
    public SumSwingApp() {
       jf = new JFrame("test");
```

```
        jf.setTitle("Sum Swing App");
         jf.setSize(300, 200);
        jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);

         jf.setLayout(new FlowLayout());

         textField1 = new JTextField(10);
         textField2 = new JTextField(10);
         jb1 = new JLabel("Enter Number1");
             jb2 = new JLabel("Enter Number2");
             button1 = new JButton("Calculate Sum");
        button2 = new JButton("Click OK");
        button1.addActionListener( this);
        button2.addActionListener( this);
        jf.add(jb1);
        jf.add(textField1);
        jf.add(jb2);
        jf.add(textField2);
        jf.add(button1);
        jf.add(button2);
        jb3 = new JLabel("Message here");
        jf.add(jb3);
        jf.setVisible(true);
    }

    public static void main(String[] args) {
        new SumSwingApp();
    }

            public void actionPerformed(ActionEvent e) {
              if(e.getSource() == button1){
              int num1 = Integer.parseInt(textField1.getText());
               int num2 = Integer.parseInt(textField2.getText());
               int sum = num1 + num2;
                jb3.setText("Sum is :" + sum);
            //   JOptionPane.showMessageDialog(null, "The sum is: " + sum);
            }
              if(e.getSource() == button2){
               jb3.setText("Cilck ok");
            }
           }
}
```

1.	Create an chat application which communicate between client and server using socket programming
2.	Create a Frame which contains two textfield and three button and

	one lable. Called add,sub,mul. On click of the button it should take values from textfield and display message in Jlabel respective operations