SOURCE CODE FOR RDP PROTOCOL:

In this project we have server and client programs. We will see each and every program of both server and client.

Server Code:

In server we have 5 programs. We store these programs in a package and if we run main program then all server programs will run and we will get our required output.

5 programs are

- Start.java
- Set_password.java
- InitConnection.java
- Send_screen.java
- Receive_events.java

We will see each and every programs in order.

Start.java

```
public class Start
{
        public static void main(String arg[])
        {
                SetPassword frame1=new SetPassword();
                frame1.setSize(300,80);
                frame1.setLocation(500,300);
                frame1.setVisible(true);
        }
}
```

The above program is main program to the server and by running this program we will start server

Set_password.java

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class SetPassword extends JFrame implements ActionListener
{
        static String port="4903";
        JButton submit;
        JPanel panel;
        JTextField text1,text2;
        String value1,value2;
        JLabel label,label1,label2;

        SetPassword()
        {
                label1=new JLabel();
                label1.setText("set password");
                text1=new JTextField(15);
                label=new JLabel();
                label.setText("");
                this.setLayout(new BorderLayout());
                submit=new JButton("submit");
                panel=new JPanel(new GridLayout(2,1));
                panel.add(label1);
                panel.add(text1);
                panel.add(label);
                panel.add(submit);
                add(panel,BorderLayout.CENTER);
                submit.addActionListener(this);
                setTitle("setting password for client");
        }
        public void actionPerformed(ActionEvent e)
        {
                value1=text1.getText();
                dispose();
                new InitConnection(Integer.parseInt(port),value1);

        }

        public String getValue1()
        {
                return value1;
        }

        public static void main(String arg[])
        {
                SetPassword frame1=new SetPassword();
                frame1.setSize(300,80);
                frame1.setLocation(500,300);
                frame1.setVisible(true);
        }
```

The above program is used to set password to the server desktop.

InitConnection.java

```
import java.awt.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class InitConnection
{
	ServerSocket socket=null;
	DataInputStream password=null;
	DataOutputStream verify=null;
	String width="";
	String height="";

	InitConnection(int port,String value1)
	{
		Robot robot=null;
		Rectangle rectangle=null;
		try
		{
			System.out.println("waiting for client to get connected.....");
			socket=new ServerSocket(port);
			GraphicsEnvironment gEnv=GraphicsEnvironment.getLocalGraphicsEnvironment();
			GraphicsDevice gDev=gEnv.getDefaultScreenDevice();

			while(true)
			{
				Socket sc=socket.accept();
				password=new DataInputStream(sc.getInputStream());
				verify=new DataOutputStream(sc.getOutputStream());

				String psword=password.readUTF();
				if(psword.equals(value1))
				{
					verify.writeUTF("valid");
					verify.writeUTF(width);
					verify.writeUTF(height);
					new SendScreen(sc,robot,rectangle);
					new ReceiveEvents(sc,robot);
				}
				else
				{
					verify.writeUTF("invalid");
				}
			}


		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}

	private void drawGUI()
	{}


}
```

The above program is use to form connection between client and server. When client program run after entering IP address and password, if both password and IP address are correct then it form connection between client and server.

Send_screen.java

```
import java.awt.*;
import java.io.*;
import java.net.Socket;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;

public class SendScreen extends Thread
{
	Socket socket=null;
	Robot robot=null;
	Rectangle rectangle=null;
	boolean continueLoop=true;
	OutputStream oos=null;

	public SendScreen(Socket socket,Robot robot,Rectangle rect)
	{
	this.socket=socket;
	this.robot=robot;
	rectangle=rect;
	start();
	}

	public void run()
	{
		try
		{
			oos=socket.getOutputStream();
		 catch(IOException e)
		 {
			e.printStackTrace();
		 }

		 while(continueLoop)
		 {
			BufferedImage image=robot.createScreenCapture(rectangle);
			try
			{
				ImageIO.write(image,"jpeg",oos);
			}
			catch(IOException e)
			{
				e.printStackTrace();
			}
			try
			{
				Thread.sleep(10);
			}
			catch(InterruptedException e)
			{
				e.printStackTrace();
			}
		 }

	}
}
```

When client program want to access the server screen then, the above program is used to share server screen to client desktop.

Receive_events.java

```
import java.awt.Robot;
import java.io.*;
import java.net.Socket;
import java.util.*;

public class ReceiveEvents extends Thread
{
	Socket socket=null;
	Robot robot=null;
	boolean continueLoop=true;

	public ReceiveEvents(Socket socket,Robot robot)
	{
		this.socket=socket;
		this.robot=robot;
		start();

	}


	p
	{			while(continueLoop)
				{
					int command=scanner.nextInt();
					switch(command)
					{
						case-1:
						robot.mousePress(scanner.nextInt());
						break;
						case-2:
						robot.mouseRelease(scanner.nextInt());
						break;
						case-3:
						robot.keyPress(scanner.nextInt());
						break;
						case-4:
						robot.keyRelease(scanner.nextInt());
						break;
						case-5:
						robot.mouseMove(scanner.nextInt(),scanner.nextInt());
						break;
					}
				}
		}

		catch(IOException e)
		{
			e.printStackTrace();
		}
	}
```

The above program is used when we do any events like mouse clicking, opening a file, typing in client desktop then server will receive events from client can achieve by this program.

CLIENT CODE:

In client we have 6 programs. We store these programs in a package and if we run main program then all client programs will run and we will get our required output.

6 programs are

- Start.java
- Authentication.java
- CreateFrame.java
- ReceivingScreen.java
- SendEvents.java
- Commands.java

We will see each and every programs in order.

Start.java

```
import java.net.*;
import javax.swing.*;
public class Start
{
 static String port="4903";
 public static void main(String arg[])
 {

        String ip=JOptionPane.showInputDialog("please enter server ip address");
        new Start().initialize(ip,Integer.parseInt(port));
 }
 public void initialize(String ip,int port)
 {
        try
        {
                Socket sc=new Socket(ip,port);
                System.out.println("connecting to the server....");
                Authentication frame1=new Authentication(sc);
                frame1.setSize(300,80);
                frame1.setLocation(500,300);
                frame1.setVisible(true);

        }
        catch(Exception e)
        {
                e.printStackTrace();
    }


 }
}
```

The above program is the main program to start client program. Don't confuse with Start.java name there are 2 Start.java programs one is in server and another one is client.

Authentication.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
class Authentication extends JFrame implements ActionListener
{
        private Socket cSocket=null;
        DataOutputStream passchk=null;
        DataInputStream verification=null;
        String verify="";
        JButton submit;
        JPanel panel;
        JLabel label,label1;
        String width="",height="";
        JTextField text1;
        Authentication(Socket cSocket)
    {
                label1=new JLabel();
                label1.setText("password");
                text1=new JTextField(15);
                this.cSocket=cSocket;
                label=new JLabel();
                label.setText("");
                this.setLayout(new BorderLayout());
                submit=new JButton("submit");
                panel=new JPanel(new GridLayout(2,1));
                 passchk.writeUTF(value1);
                 verify=verification.readUTF();
        }
        catch(IOException e)
        {
                e.printStackTrace();
        }
        if(verify.equals("valid"))
        {
                try
                {
                        width=verification.readUTF();
                        height=verification.readUTF();
                }
                catch(IOException e)
                {
                        e.printStackTrace();
                }
                CreateFrame abc=new CreateFrame(cSocket,width,height);
                dispose();
        }
        else
        {
                System.out.println("please enter valid password");
                JOptionPane.showMessageDialog(this,"password is incorrect","Error",JOptionPane.ERROR_MESSAGE);
                dispose();
        }
    }
}
```

CreateFrame.java

```
import javax.swing.*;
import java.net.Socket;
import java.io.InputStream;
import java.beans.PropertyVetoException;
import java.awt.BorderLayout;
import java.util.zip.*;
import java.io.IOException;

class CreateFrame extends Thread
{
	String width="",height="";
	private JFrame frame=new JFrame();
	private JDesktopPane desktop=new JDesktopPane();
	private Socket cSocket=null;
	private JInternalFrame interFrame=new JInternalFrame("Server Screen",true,true,true);
	private JPanel cPanel=new JPanel();
	public CreateFrame(Socket cSocket,String width,String height)
	{
		this.width=width;
		this.height=height;
		this.cSocket=cSocket;
		start();
	}
	public void drawGUI()
	{
		frame.add(desktop,BorderLayout.CENTER);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setExtendedState(frame.getExtendedState()|JFrame.MAXIMIZED_BOTH);
		try
		{
			interFrame.setMaximum(true);
		}
		catch(PropertyVetoException ex)
		{
			ex.printStackTrace();

		}
		cPanel.setFocusable(true);
		interFrame.setVisible(true);

	}


	public void run()
	{
		InputStream in=null;
		drawGUI();
		try
		{
			in=cSocket.getInputStream();
		}
		catch(IOException e)
		{
			e.printStackTrace();
		}
		new ReceivingScreen(in,cPanel);
		new SendEvents(cSocket,cPanel,width,height);


	}


}
```

ReceivingScreen.java

```
import java.io.ObjectInputStream;
import java.io.InputStream;
import javax.swing.JPanel;
import java.awt.Image;
import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.io.IOException;
import java.io.ByteArrayInputStream;


class ReceivingScreen extends Thread
{
	private ObjectInputStream cObjectInputStream=null;
	private JPanel cPanel=null;
	private boolean continueLoop=true;
	InputStream oin=null;
	Image image1=null;

	public ReceivingScreen(InputStream in,JPanel p)
	{
		oin=in;
		cPanel=p;
		start();
	}

	public void run()
	{
		try
		{
		  while(true)
		  {
			byte[] bytes=new byte[1024*1024];
			int count=0;
			do
			{
				count+=oin.read(bytes,count,bytes.length-count);
			}while(!(count>4&&bytes[count-2]==(byte)-1&&bytes[count-1]==(byte)-39));
			image1=ImageIO.read(new ByteArrayInputStream(bytes));
			image1=image1.getScaledInstance(cPanel.getWidth(),cPanel.getHeight(),Image.SCALE_FAST);

			Graphics graphics=cPanel.getGraphics();
			graphics.drawImage(image1,0,0,cPanel.getWidth(),cPanel.getHeight(),cPanel);
		  }
		}
		catch(IOException e)
		{
			e.printStackTrace();
	}
	}


}
```

This program is use to receive screen from server to client desktop.

SendEvents.java

```
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.io.PrintWriter;
import java.io.IOException;
import java.net.Socket;
import javax.swing.JPanel;


class SendEvents implements KeyListener,MouseMotionListener,MouseListener
{
		private Socket cSocket=null;
		private JPanel cPanel=null;
		private PrintWriter writer=null;
		String width="",height="";
		double w;
		double h;

		SendEvents(Socket s,JPanel p,String width,String height)
		{
			cSocket=s;
			cPanel=p;
			this.width=width;
			this.height=height;
			w=Double.valueOf(width.trim()).doubleValue();
			h=Double.valueOf(width.trim()).doubleValue();

	    cPanel.addKeyListener(this);
			cPanel.addMouseMotionListener(this);
			cPanel.addMouseListener(this);

			try
			{
			writer=new PrintWriter(cSocket.getOutputStream());
			}
			catch(IOException e)
			{
			e.printStackTrace();

		public void mouseDragged(MouseEvent e)
		{
		}

		public void mouseMoved(MouseEvent e)
		{
			double xScale=(double)w/cPanel.getWidth();
			double yScale=(double)h/cPanel.getHeight();
			writer.println(Commands.MOVE_MOUSE.getAbbrev());
			writer.println((int)(e.getX()*xScale));
			writer.println((int)(e.getX()*yScale));
			writer.flush();
		}


		public void mouseClicked(MouseEvent e)
		{}


		public void mousePressed(MouseEvent e)
		{
			writer.println(Commands.PRESS_MOUSE.getAbbrev());
			int button=e.getButton();
			int xButton=16;
			if(button==3)
			{
				xButton=4;
			}
			writer.println(xButton);
			writer.flush();
		}
```

```
		public void mouseReleased(MouseEvent e)
		{
			writer.println(Commands.RELEASE_MOUSE.getAbbrev());
			int button=e.getButton();
			int xButton=16;
			if(button==3)
			{
				xButton=4;
			}
			writer.println(xButton);
			writer.flush();
		}

		public void mouseEntered(MouseEvent e)
		{}

		public void mouseExited(MouseEvent e)
		{}

		public void keyTyped(KeyEvent e)
		{}

		public void keyPressed(KeyEvent e)
		{
			writer.println(Commands.PRESS_KEY.getAbbrev());
			writer.println(e.getKeyCode());
			writer.flush();
		}

		public void keyReleased(KeyEvent e)
		{
			writer.println(Commands.RELEASE_KEY.getAbbrev());
			writer.println(e.getKeyCode());
			writer.flush();
		}

	}
```

The above program is used to transfer events from client desktop to server desktop.

Commands.java

```
public enum Commands
{
	PRESS_MOUSE(-1),
	RELEASE_MOUSE(-2),
	PRESS_KEY(-3),
	RELEASE_KEY(-4),
	MOVE_MOUSE(-5);

	private int abbrev;
	Commands(int abbrev)
	{
		this.abbrev=abbrev;
	}
	public int getAbbrev()
	{
		return abbrev;
	}
}
```

Through these 11 programs we can achieve remote desktop controller of one desktop to another.