import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintWriter;  
import java.net.BindException;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
import javax.swing.JFrame;  
import javax.swing.JMenu;  
import javax.swing.JMenuBar;  
import javax.swing.JMenuItem;  
import javax.swing.JOptionPane;  
import javax.swing.JScrollPane;  
import javax.swing.JTextArea;  
  
  
public class MessageServer extends JFrame {  
  
    private JTextArea output;  
    private final int port = 10000;  
    public MessageServer(String title) {  
        super(title);  
        init();  
    }  
    private void init(){  
        output = new JTextArea(10, 30);  
        output.setEditable(false);  
  
        output.setForeground(Color.blue);  
        JScrollPane jsp = new JScrollPane(output);  
        JMenuBar menuBar = new JMenuBar();  
          
        JMenu fileMenu = new JMenu("File");  
        fileMenu.setMnemonic('F');  
        JMenuItem exitItem = new JMenuItem("Exit...");  
        exitItem.setMnemonic('X');  
        exitItem.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent event) {  
                  
                System.exit(0);  
            }  
        });  
        fileMenu.add(exitItem);  
          
        JMenu editMenu = new JMenu("Edit");  
        editMenu.setMnemonic('E');  
        JMenuItem clearItem = new JMenuItem("Clear");  
        clearItem.setMnemonic('C');  
        clearItem.addActionListener(new ActionListener(){  
            public void actionPerformed(ActionEvent event){  
                output.setText("");  
            }  
        });  
        editMenu.add(clearItem);  
          
        JMenu helpMenu = new JMenu("Help");  
        helpMenu.setMnemonic('H');  
        JMenuItem aboutItem = new JMenuItem("About...");  
        aboutItem.setMnemonic('A');  
        aboutItem.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent event) {  
                JOptionPane.showMessageDialog(MessageServer.this,  
                        "Socket学习", "About", JOptionPane.PLAIN\_MESSAGE);  
            }  
        });  
        helpMenu.add(aboutItem);  
  
        menuBar.add(fileMenu);  
        menuBar.add(editMenu);  
        menuBar.add(helpMenu);  
        setJMenuBar(menuBar);  
  
        add(jsp, BorderLayout.CENTER);  
        setSize(400, 300);  
        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
        this.setResizable(false);  
        setVisible(true);  
    }  
      
    private void service(){  
        try {  
            ServerSocket server = new ServerSocket(port);  
              
              
            while (true) {  
                final Socket socket = server.accept();  
                System.out.println("Accepted from " + socket.getInetAddress()  
                        + " with port:"+ socket.getPort());  
                  
                output.append(" Accepted from "  
                        + socket.getInetAddress() +" ,port:"+ socket.getPort()+" \r\n");  
                new Thread(new Runnable(){  
                    public void run(){  
                        try {  
                            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
                              
                            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);  
                              
                              
                            while(true){  
                                String line = in.readLine();  
                                System.out.println("Incept message:" + line);  
                                output.append("Incept message:" + line+" \r\n");  
                                output.setCaretPosition(output.getText().length());  
                                out.println("your input is " + line);  
                                out.flush();  
                                if (line.equals("end")||line==null){  
                                    out.println("Good Bye!");  
                                    break;  
                                }  
                            }  
                            in.close();  
                            out.close();  
                            socket.close();  
                            System.out.println("close the connection");  
                        } catch (IOException e) {  
                            e.printStackTrace();  
                        }  
                    }  
                }).start();  
            }  
  
        }catch(BindException e){  
            System.out.println("端口已被占用!");  
            JOptionPane.showMessageDialog(null,"端口已被占用，请检查服务器是否启动或者查看是否其它程序战用了此端口,换一个端口重试!","端口被占用",JOptionPane.ERROR\_MESSAGE);  
            System.exit(0);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
          
          
    }  
      
    public static void main(String[] args) {  
        MessageServer serverFrame = new MessageServer("Server");  
        serverFrame.init();  
          
        serverFrame.service();  
    }  
  
}  
  
客户端  
  
  
import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintWriter;  
import java.net.ConnectException;  
import java.net.Socket;  
import java.net.SocketException;  
  
import javax.swing.JButton;  
import javax.swing.JComponent;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.JScrollPane;  
import javax.swing.JTextArea;  
import javax.swing.JTextField;  
import javax.swing.KeyStroke;  
  
  
public class MessageClient extends JFrame {  
    private Socket socket;  
    private BufferedReader in;  
    private PrintWriter out;  
    protected String host = "192.168.1.3";  
    protected int port = 10000;  
    public JTextArea output;  
    protected JTextField input;  
    public JTextField content;  
      
    public MessageClient(String title){  
        super(title);  
        init();  
    }  
      
    public void init(){  
          
        try{  
              
            socket = new Socket(host,port);  
              
            output = new JTextArea(15, 32);  
            output.setEditable(false);  
  
            output.setForeground(Color.black);  
            JScrollPane jsp = new JScrollPane(output);  
            add(jsp,BorderLayout.NORTH);  
            JPanel inputPanel = new JPanel();  
            content = new JTextField(10);  
              
            JButton sendButton = new JButton("Send");  
              
              
              
            /\*\*响应回车事件\*/  
            content.registerKeyboardAction(new Listener(),KeyStroke.getKeyStroke(KeyEvent.VK\_ENTER, 0, false),JComponent.WHEN\_FOCUSED);  
            /\*\*响应点击Send按扭事件\*/  
            sendButton.addMouseListener(new Listener());  
            /\*\*响应点击Exit按扭事件\*/  
            JButton exitButton = new JButton("Exit");  
            exitButton.addMouseListener(new ExitListener());  
              
            inputPanel.add(content);  
            inputPanel.add(sendButton);  
            inputPanel.add(exitButton);  
            add(inputPanel,BorderLayout.SOUTH);  
            addWindowListener(new WindowAdapter(){  
                public void windowClosing(WindowEvent event){  
                    new ExitListener().executeClick();  
                    System.out.println("已关闭");  
                }  
  
            });  
            setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);  
            setSize(400,340);  
            output.append("已建立连接|使用端口:" + socket.getPort()+ "... \r\n");  
        }catch(ConnectException connEx){  
              
            JOptionPane.showMessageDialog(this,  
                    "系统错误，没有找到服务器", "提示", JOptionPane.PLAIN\_MESSAGE);  
            System.exit(0);  
        }catch(Exception e){  
              
            e.printStackTrace();  
        }  
    }  
      
    public static void main(String[] args){  
        new MessageClient("客户端").setVisible(true);  
    }  
  
      
    class Listener extends MouseAdapter implements ActionListener{  
  
        public void actionPerformed(ActionEvent e) {  
            executeClick();  
              
        }  
        public void mouseClicked(MouseEvent event){  
            executeClick();  
        }  
          
          
        private void executeClick(){  
            Thread temp = new Thread(new Runnable(){  
  
                public void run() {  
                    try {  
                        if(socket==null || socket.isClosed()){  
                            int reConnect = JOptionPane.showConfirmDialog(null,"连接已断开，是否重新建立连接？","确认",JOptionPane.YES\_NO\_OPTION,JOptionPane.QUESTION\_MESSAGE );  
                            if(reConnect == 0){  
                                socket = new Socket("192.168.1.3",10000);  
                            }  
                            else  
                                return;  
                        }  
                        out = new PrintWriter(socket.getOutputStream());  
                      
                        String contentText = content.getText();  
                        content.setText("");  
                        content.requestFocus();  
                        if(contentText==null)  
                            return;  
                        System.out.println("发送数据" + contentText);  
                        output.append("发送数据" + contentText+" \r\n");  
                        out.println(contentText);  
                        out.flush();  
                        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
                        String getResult=in.readLine();  
                        while(getResult!=null){  
                            System.out.println("接收到数据 " + getResult);  
                            output.append("接收到数据" + getResult+" \r\n");  
                                break;  
                        }  
                        output.setCaretPosition(output.getText().length());  
                        if(contentText.equals("end")){  
                              
                            in.close();  
                            out.close();  
                            socket.close();  
                        }  
                        return;  
                    } catch(ConnectException connEx){  
                        System.out.println("未连接到服务器，请检查网络!");  
                        JOptionPane.showMessageDialog(null,"未连接到服务器，请检查网络!","未连接",JOptionPane.ERROR\_MESSAGE);  
                    }catch (SocketException e){  
                        System.out.println("没有服务器");  
                        output.append("没有服务器 ");  
                        JOptionPane.showMessageDialog(null,"连接已断开，请检查服务器并重试!","连接已断开",JOptionPane.ERROR\_MESSAGE);  
                    }catch (IOException e) {  
                        e.printStackTrace();  
                    }  
                }  
                  
            });  
            temp.start();  
        }  
          
    }  
      
    class ExitListener extends MouseAdapter{  
        public void mouseClicked(MouseEvent event){  
            executeClick();  
        }  
        private void executeClick(){  
            Thread temp = new Thread(new Runnable() {  
  
                public void run() {  
                    try {  
                        if (socket != null && !socket.isClosed()) {  
                            out = new PrintWriter(socket.getOutputStream(),  
                                    true);  
                            out.println("end");  
                            socket.close();  
                        }  
                    } catch (IOException e) {  
                        e.printStackTrace();  
                    }  
                    System.exit(0);  
  
                }  
  
            });  
            temp.start();  
        }  
    }  
}