ProgMain.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class ProgMain {

	static Chat_Kernel kr = null;

	public static void main(String[] args) {
		// TODO 自動生成されたメソッド・スタブ
		String	s_myPort  = null;
		int	myPort    = 0;
		boolean	init	  = true;
		BufferedReader std_in   = null;

		/**--------------------初期処理-----------------------------*/
		std_in = new BufferedReader(new InputStreamReader(System.in));
		while(init){
			try{
				System.out.println("適当なポート番号を入力してください:");
				s_myPort = std_in.readLine();
				if(s_myPort.equals("")){
					System.out.println("入力が正しくありません。");
					s_myPort = null;
				}else{
					myPort = Integer.parseInt(s_myPort);
					init = false;
				}
			}catch(Exception e){
				e.printStackTrace();
			}
		}
		kr = new Chat_Kernel(myPort);
		/**----------------------------------------------------------*/
	}

	public static Chat_Kernel getKernel(){
		return kr;
	}

}
```

Chat_Kernel.java

```
public class Chat_Kernel {
	MainFrm	frame	= null;
	SettingFrame s_frame	= null;
	ServSock	srv	= null;
	HostSock	host	= null;
	String	hostIPaddr = null;
	int	hostPort	= 0;

	public Chat_Kernel(int port){
		frame	= new MainFrm();
		s_frame = new SettingFrame();
		frame.setVisible(true);		//メイン画面の起動
		srv = new ServSock(port,60);	//サーバ機能の起動
	}
	public boolean Connection(String addr, int port){
		try{
			host = new HostSock(addr, port);
		}catch(Exception e){
			s_frame.showErrorDialog("接続できません");
			return false;
		}
		return true;
	}

	public void msgSend(String msg){
		if(host.equals(null)){
			frame.showErrorDialog("接続が確立されていません");
		}else{
			host.msgSend(msg);
		}
	}
	public void showMsg(String msg){
		frame.pane.showMsg(msg+"\n");
	}
	public void setButtonEnabled(boolean bl){
		frame.pane.setButtonEnabled(bl);
	}
}
```

MainFrm.java

```
/**
 * MainFrm.java
 * 自分のポート番号を入力した後に表示される画面の設計を行う。
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class MainFrm {

	JFrame frame;
	MyPane pane;

	public MainFrm(){
		frame = new JFrame("Test Chat Program.");
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setSize(400,400);
		frame.setContentPane(pane = new MyPane());
		initMenu();
	}
	public void setVisible(boolean bl){
		frame.setVisible(bl);
	}
	public void showErrorDialog(String err){
		JOptionPane.showMessageDialog(frame, err, "alert", JOptionPane.INFORMATION_MESSAGE);
	}
	public class MyPane extends JPanel{
		/**-----GUI部品-------------------*/
		final JButton   button;
		final JTextArea send_txt;
		final JTextArea log_txt;
		/**-------------------------------*/
		public MyPane(){
			setLayout(new BorderLayout());

			/**--- GUI部品の設定-----------------------------------------------*/
			button   = new JButton("送信");
			button.setPreferredSize(new Dimension(20,20));
			button.setEnabled(false); //接続設定がなされていない時は送信できないように
する
			send_txt = new JTextArea(1,50);
```

```
                        log_txt  = new JTextArea();
                        /**-------------------------------------------------------------*/
                        button.addActionListener(new ActionListener(){
                                public void actionPerformed(ActionEvent e){
                                        SendButton_Click(send_txt.getText());
                                }
                        });

                        log_txt.setBorder(new LineBorder(Color.BLACK));
                        log_txt.setLineWrap(true);
                        JScrollPane vpane = new JScrollPane(log_txt);

                        add(send_txt, BorderLayout.NORTH);
                        add(vpane, BorderLayout.CENTER);
                        add(button, BorderLayout.SOUTH);
                }
                public void setButtonEnabled(boolean bl){
                        button.setEnabled(bl);
                }
                public void showMsg(String msg){
                        log_txt.append(msg);
                }
        }
        /**
         * initMenu()
         * メニューバーの設定。
         * [接続]
         *    |
         *    └[接続設定]…接続設定画面(SettingFrame.javaで設計)を呼び出す
         */
        private void initMenu(){
                /**--------メニュー部品-----------------------*/
                JMenuBar  bar     = new JMenuBar();
                JMenu     m_conn  = new JMenu("接続");
                JMenuItem mi_conn = new JMenuItem("接続設定");
                /**-----------------------------------------*/
                mi_conn.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent e){
                                //接続設定画面を呼び出す
                                ProgMain.getKernel().s_frame.setVisible(true);
                        }
                });
                frame.setJMenuBar(bar);
                bar.add(m_conn);
```

```
                    m_conn.add(mi_conn);
          }
          private void SendButton_Click(String msg){
                    if(msg.equals("")){
                              JOptionPane.showMessageDialog(frame,"送信するメッセージを入力してください
","alert",

JOptionPane.INFORMATION_MESSAGE);
                              return;
                    }else{
                              ProgMain.getKernel().msgSend(msg);
                    }
          }
}
```

SettingFrame.java

```
/**
 * SettingFrame.java
 * メイン画面で「接続」→「接続設定」をクリック
 * した時に呼び出される画面のGUIを設計。
 *
 * ホストのIPアドレス、及びポート番号を入力し、
 * 接続ボタンを押すとそのホストへ接続を行う。こ
 * の後にホストとの対話が可能になる。
 */
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import javax.swing.border.*;

public class SettingFrame {
          JFrame  s_frame;
          SubPane s_pane;
          public SettingFrame(){
                    s_frame = new JFrame("Settings");
                    s_frame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
                    s_frame.setSize(200,300);
                    s_frame.setContentPane(s_pane = new SubPane());
          }
          public void setVisible(boolean bl){
                    s_frame.setVisible(bl);
          }
```

```
	public void showErrorDialog(String err){
		JOptionPane.showMessageDialog(s_frame, err, "alert", JOptionPane.INFORMATION_MESSAGE);
	}
	private class SubPane extends JPanel{
		/**---GUI部品----------------*/
		final JTextArea IPAddr;
		final JTextArea connectPort;

		final JLabel    l_addr;
		final JLabel    l_connectport;

		final JButton   btnConnect;

		JPanel AddrZone    = null;
		JPanel conPortZone = null;
		JPanel ButtonZone  = null;
		/**-------------------------*/
		public SubPane(){
			AddrZone    = new JPanel();
			AddrZone.setLayout(new GridLayout(2,1));
			conPortZone = new JPanel();
			conPortZone.setLayout(new GridLayout(2,1));
			ButtonZone  = new JPanel();
			ButtonZone.setLayout(new GridLayout(1,1));
			this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

			/**-------GUIの設定----------------------------------*/
			l_addr     = new JLabel("相手のIPアドレス");
			l_connectport  = new JLabel("相手のポート番号");
			btnConnect = new JButton("接続");
			IPAddr     = new JTextArea();
			connectPort    = new JTextArea();
			btnConnect.setPreferredSize(new Dimension(240,25));

			IPAddr.setBorder(new LineBorder(Color.BLACK));
			connectPort.setBorder(new LineBorder(Color.BLACK));

			AddrZone.add(l_addr);
			AddrZone.add(IPAddr);
			conPortZone.add(l_connectport);
			conPortZone.add(connectPort);

			btnConnect.addActionListener(new ActionListener(){
				public void actionPerformed(ActionEvent e){
```

```
                                        String s_conport = connectPort.getText();
                                        String s_addr    = IPAddr.getText();
                                        if(s_addr.equals("") || s_conport.equals("")){
                                                JOptionPane.showMessageDialog(s_frame, "無
効な入力です", "alert",

                                JOptionPane.INFORMATION_MESSAGE);
                                        }else{
                                                ConnectButton_Click(e, s_addr,
Integer.parseInt(s_conport));
                                        }
                                }
                        });
                        ButtonZone.add(btnConnect);

                        add(AddrZone);
                        add(conPortZone);
                        add(ButtonZone);
                        /**-------------------------------------------------*/
                }
                private void ConnectButton_Click(ActionEvent e, String addr, int con_port){
                        /**
                         * Chat_KernelのConnectionを呼び出す。
                         * 入力されたIPアドレス、ポート番号を持つホストに接続する。
                         */
                        Chat_Kernel kr;
                        kr = ProgMain.getKernel();
                        if(kr.Connection(addr, con_port)){
                                s_frame.setVisible(false);
                                ProgMain.getKernel().setButtonEnabled(true); //メイン画面の[送
信]ボタンを有効にする.
                        }
                }
        }
}
```

HostSock.java

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

public class HostSock {
        Socket         host   = null;
```

```
	PrintWriter	writer = null;

	public HostSock(String addr, int port){
		try{
			host = new Socket(addr,port);
			writer = new PrintWriter(new OutputStreamWriter(host.getOutputStream()));
		}catch(Exception e){
			e.printStackTrace();
		}
	}
	public void msgSend(String msg){
		try{
			writer.println(msg);
			writer.flush();
			ProgMain.getKernel().showMsg(msg);
		}catch(Exception e){
		}
	}
}
```

ServSock.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

public class ServSock extends Thread implements Runnable {
	ServerSocket	serv	= null;
	Socket		client	= null;
	BufferedReader in_sock	= null;
	public ServSock(int port, int con){
		try{
			serv = new ServerSocket(port, con);
			this.start();
		}catch(Exception e){
			e.printStackTrace();
		}
	}
	public void run(){
		String msg;
		BufferedReader in_sock;
```

```
                    try{
                              client = serv.accept();
                              in_sock = new BufferedReader(new
InputStreamReader(client.getInputStream()));
                              System.out.println("Host accepted.");
                              while(true){
                                        msg = in_sock.readLine();
                                        ProgMain.getKernel().showMsg(">"+msg);
                              }
                    }catch(IOException ioe){

                    }
          }
}
```