```
#****************************************
#  Chat.idl
#****************************************
# Makefile - Java version
#
# use: make
# or:  make test
#

NAME = Chat

NAMESERVERPORT = 8050

JAVA  = /usr/bin/java
JAVAC = /usr/bin/javac
IDLJ  = /usr/bin/idlj

GENERATED = $(NAME)App

all:
        -@ echo "  "
        -@ echo "  make target                - build the project"
        -@ echo "  make orbd|client|server  - run the individual components"
        -@ echo "  make clean                 - clean temporary files"
        -@ echo "  make clobber               - wipe everything that is generated"
        -@ echo "  "


clean:
        -@touch ./abc~ core
        -@rm -f *~ core

clobber: clean
        -@touch ./abc.class
        -@rm -rf *.class $(GENERATED) orb.db

idl::
        -$(IDLJ) -fall $(NAME).idl

c::
        $(JAVAC) $(NAME)Client.java  $(NAME)App/*.java

s::
        $(JAVAC) $(NAME)Server.java $(NAME)App/*.java

target::
        make clobber
        make idl
        make c
        make s

orbd::
        -@echo "Starting orbd"
        -@rm -rf ./orb.db
        orbd -ORBInitialPort $(NAMESERVERPORT) -ORBInitialHost localhost

server::
        $(JAVA) $(NAME)Server -ORBInitialPort $(NAMESERVERPORT) -ORBInitialHost
            localhost

client::
        $(JAVA) $(NAME)Client -ORBInitialPort $(NAMESERVERPORT) -ORBInitialHost
            localhost
```

```
/***************************************
Chat.idl
***************************************/
module ChatApp {
  interface ChatCallback {
    void callback(in string message);
  };
  interface Chat {
    string join(in ChatCallback objref, in string nickname);
    void post(in ChatCallback objref, in string nickname, in string msg);
    void list(in ChatCallback objref, in string nickname);
    void leave(in ChatCallback objref, in string nickname);

  };
  interface GameCallback {
    void boardupdate(in string gbstate);
    void startgame(in string nickname, in char colour);
    void closegame();
  };
  interface Game {
    boolean join(in ChatCallback chatref, in GameCallback gbref, in string nickname, in
        char colour);
    boolean makemove(in ChatCallback chatref, in string nickname, in string move);
    void passturn();
    void list(in ChatCallback chatref);
    void leave(in ChatCallback chatref, in GameCallback gbref, in string nickname);
    void reset(in boolean manReset);

  };
};
```

```
/**************************************
ChatClient.java
**************************************/
import ChatApp.*;           // The package containing our stubs
import org.omg.CosNaming.*; // HelloClient will use the naming service.
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;     // All CORBA applications need these classes.
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.*;


class ChatCallbackImpl extends ChatCallbackPOA {
    private ORB orb;

    public void setORB(ORB orb_val) {
        orb = orb_val;
    }

    public void callback(String notification){
        System.out.println(notification);
    }
}

public class ChatClient {
    static Chat chatImpl;
    static Game gameImpl;

    public static void main(String args[]){
        try {
            // create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // create servant (impl) and register it with the ORB
            ChatCallbackImpl chatCallbackImpl = new ChatCallbackImpl();
            chatCallbackImpl.setORB(orb);

            /* extra bit */
            GameCallbackImpl gameCallbackImpl = new GameCallbackImpl();
            gameCallbackImpl.setORB(orb);
            /* /extra bit */

            // get reference to RootPOA and activate the POAManager
            POA rootpoa =
                POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootpoa.the_POAManager().activate();

            // get the root naming context
            org.omg.CORBA.Object objRef =
                orb.resolve_initial_references("NameService");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            // resolve the object reference in naming
            String name = "Chat";
            chatImpl = ChatHelper.narrow(ncRef.resolve_str(name));
            String name2 = "Game";
            gameImpl = GameHelper.narrow(ncRef.resolve_str(name2));

            // obtain callback reference for registration w/ server
            org.omg.CORBA.Object ref =
                rootpoa.servant_to_reference(chatCallbackImpl);
            ChatCallback cref = ChatCallbackHelper.narrow(ref);

            /* extra bit */
            ref = rootpoa.servant_to_reference(gameCallbackImpl);
            GameCallback gref = GameCallbackHelper.narrow(ref);

            // Application code goes below
            String nickname = "";
            String[] input;
```

```
Scanner in = new Scanner(System.in);
boolean Active = false;
boolean Playing = false;

// Welcome Message
System.out.println("\u001b[32;1m\n"
                    + "Welcome to ShitShat!\n"
                    + "Commands available: (shorts available with \\, i.e. \\j to join)\n\u001b[0m"
                    + "(j)oin <nick>            \u001b[35mJoin chat \u001b[0m\n"
                    + "pos(t) <msg>             \u001b[35mPost to chat \u001b[0m\n"
                    + "(l)ist                   \u001b[35mList connected users \u001b[0m\n"
                    + "pl(a)y <color>           \u001b[35mPlay game \u001b[0m\n"
                    + "passtur(n)               \u001b[35mPass turn in-game\u001b[0m\n"
                    + "(g)list                  \u001b[35mList connected players \u001b[0m\n"
                    + "reset                    \u001b[35mReset the game board\u001b[0m\n"
                    + "lea(v)e                  \u001b[35mLeave chat \u001b[0m\n"
                    + "(p)ut <coordinate XY>    \u001b[35mMake a move \u001b[0m\n"
                    + "(q)uit                   \u001b[35mQuit ShitShat \u001b[0m\n");

while(true){

    input = in.nextLine().split(" ");

    // Join
    if (input[0].equals("join") || input[0].equals("\\j")){
        if (input.length < 2){
            System.out.println("\u001b[31;1m No name given at command line!\u001b[0m");
        }
        else if (!Active){
            nickname = chatImpl.join(cref, input[1]);
            Active = true;
            if (nickname.equals("active")){
                Active = false;
            }
        }
        else{
            System.out.println("\u001b[31;1mDon't join twice, " + nickname + "\u001b[0m");
        }
    }

    // Post
    if (input[0].equals("post") || input[0].equals("\\t")){
        if (Active){
            String msg = "";
            for(int i= 1; i < input.length; i++) {
                msg = msg + " " + input[i];
            }
            chatImpl.post(cref, nickname, msg);
        }
        else{
            System.out.println("\u001b[31;1mGo active first!\u001b[0m");
        }
    }

    // List
    if (input[0].equals("list") || input[0].equals("\\l")){
        chatImpl.list(cref, nickname );
```

```
}

// List (game)
if (input[0].equals("glist") || input[0].equals("\\g")){
    gameImpl.list(cref);
}

// Leave (game)
if (input[0].equals("leave") || input[0].equals("\\v")){
    if (Active){
        chatImpl.leave(cref, nickname);
        Active = false;

        if (Playing){
            gameImpl.leave(cref, gref, nickname);
            Playing = false;
        }
    }
    else{
        System.out.println("\u001b[31;1mJoin before leaving!\u001b[0m");
    }
}

// Play (game)
if(input[0].equals("play") || input[0].equals("\\a")){
    if (Active){
        if (input.length > 1){
            char color = input[1].charAt(0);
            gameImpl.join(cref, gref, nickname, color);
            Playing = true;
        }
    }
    else{
        System.out.println("\u001b[31;1mJoin first omg\u001b[0m");
    }
}

// Put (game)
//Note-to-self: int (char(a)) == 97
if (input[0].equals("put") || input[0].equals("\\p")){
    if (Playing){
        if (input.length > 1){
            String pos = input[1];
            if (pos.matches("([a-h]|[A-H])+([1-8])")){
                gameImpl.makemove(cref, nickname, pos);
            }
            else{
                System.out.println("\u001b[31;1mSyntax: \"put [a-h
                    ][1-8]\", i.e. \"put a3\"\u001b[0m");
            }
        }
        else{
            System.out.println("Enter a 2-digit coordinate to put");
        }
    }
    else{
        System.out.println("\u001b[31;1mJoin a game first\u001b[0m");
    }
}

//Pass Turn
if (input[0].equals("passturn") || input[0].equals("\\n")) {
    System.out.println("\u001b[36mPassed turn!\u001b[0m");
    gameImpl.passturn();
}

//Reset
if (input[0].equals("reset")) {
    boolean manReset = true;
    gameImpl.reset(manReset);
```

```
                }

                // Quit
                if (input[0].equals("quit") || input[0].equals("\\q")){
                    if (Active){
                        chatImpl.leave(cref, nickname);
                        Active = false;
                        System.out.println("\u001b[31;1mStill active, leaving ...\u001b
                            [0m");
                    }

                    System.out.println("\u001b[36mBye " + nickname + "!\u001b[0m");
                    System.exit(0);
                }
            }
        }
        catch(Exception e){
            System.out.println("\u001b[31;1mERROR : " + e + "\u001b[0m");
            e.printStackTrace(System.out);
        }
    }
}
```

```
/**************************************
ChatServer.java
**************************************/
import ChatApp.*;           // The package containing our stubs.
import org.omg.CosNaming.*; // HelloServer will use the naming service.
import org.omg.CosNaming.NamingContextPackage.*; // ..for exceptions.
import org.omg.CORBA.*;     // All CORBA applications need these classes.
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.*;

class ChatImpl extends ChatPOA {
  private ORB orb;
  Map<String, ChatCallback> clients = new HashMap<String, ChatCallback>();

  public void setORB(ORB orb_val) {
    orb = orb_val;
  }

    // ### Join ###
    public String join(ChatCallback objref, String nickname){
        if(clients.containsKey(nickname)){
            objref.callback("\u001b[31;1m" + nickname + "␣is␣already␣an␣active␣chatter\
                u001b[0m");
            return "active";
        }
        for (Map.Entry<String,ChatCallback> eent : clients.entrySet()) {
            try {
                eent.getValue().callback("\u001b[33m" + nickname + "␣has␣joined!\u001b[0
                    m"); // goes out to everyone
            }
            catch(Exception e){
                /* Remove zombie peers */
                System.out.println("\u001b[31;1mLost␣connection␣to␣peer␣" + eent.getKey
                    () + "!␣\u001b[0m");
                clients.remove(eent.getKey());
            }
        }

        objref.callback("\u001b[36mWelcome␣" + nickname + "!\u001b[0m");
        clients.put(nickname, objref);
        return nickname;

    }

    // ### post ###
    public void post(ChatCallback objref, String nickname, String msg){
        for (Map.Entry<String,ChatCallback> eent : clients.entrySet()) {
            try {
                eent.getValue().callback("\u001b[34;1m" + nickname + ":\u001b[0m" + msg)
                    ;
            }
            catch(Exception e){
                /* Remove zombie peers */
                System.out.println("\u001b[31;1mLost␣connection␣to␣peer␣" + eent.getKey
                    () + "!␣\u001b[0m");
                clients.remove(eent.getKey());
            }
        }
    }

    // ### list ###
    public void list(ChatCallback objref, String nickname){
        objref.callback("\u001b[36mList␣of␣registered␣users:␣\u001b[0m");

        for(String joinedNicks : clients.keySet()){
            objref.callback(joinedNicks);
        }
    }
```

```
    // ### leave ###
    public void leave(ChatCallback objref, String nickname){
        clients.remove(nickname); // remove post in hash
        for (Map.Entry<String,ChatCallback> eent : clients.entrySet()) {
            try {
                eent.getValue().callback("\u001b[33m" + nickname + " has left.\u001b[0m"
                    ); // broadcast message
            }
            catch(Exception e){
                /* Remove zombie peers */
                System.out.println("\u001b[31;1mLost connection to peer " + eent.getKey
                    () + "!\u001b[0m");
                clients.remove(eent.getKey());
            }
        }

        objref.callback("Cheers " + nickname);
    }
}

public class ChatServer {
    public static void main(String args[]) {
        try {
            // create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // create servant (impl) and register it with the ORB
            ChatImpl chatImpl = new ChatImpl();
            chatImpl.setORB(orb);

            /* extra bit */
            GameImpl gameImpl = new GameImpl(chatImpl);
            gameImpl.setORB(orb);
            /* /extra bit */

            // get reference to rootpoa & activate the POAManager
            POA rootpoa =
                POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootpoa.the_POAManager().activate();

            // get the root naming context
            org.omg.CORBA.Object objRef =
                orb.resolve_initial_references("NameService");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            // obtain object reference from the servant (impl)
            org.omg.CORBA.Object ref =
                rootpoa.servant_to_reference(chatImpl);
            Chat cref = ChatHelper.narrow(ref);

            /* extra bit */
            org.omg.CORBA.Object ref2 =
                rootpoa.servant_to_reference(gameImpl); //Enough?
            Game gref = GameHelper.narrow(ref2);
            /* /extra bit */

            // bind the object reference in naming
            String name = "Chat";
            String name2 = "Game";
            NameComponent path[] = ncRef.to_name(name);
            ncRef.rebind(path, cref);

            /* extra bit */
            NameComponent path2[] = ncRef.to_name(name2);
            ncRef.rebind(path2, gref);
            /* /extra bit */

            System.out.println("\u001b[32;1m\nChatServer ready and waiting ...\u001b[0m"
                );
```

```
            // wait for invocations from clients
            orb.run();
        }

        catch(Exception e) {
            System.err.println("\u001b[31;1mERROR : " + e + "\u001b[0m");
            e.printStackTrace(System.out);
        }

        System.out.println("ChatServer Exiting ...");
    }

}
```

```
/**************************************
GameCallbackImpl.java
**************************************/
import ChatApp.*;
import org.omg.CosNaming.*; // HelloServer will use the naming service.
import org.omg.CosNaming.NamingContextPackage.*; // ..for exceptions.
import org.omg.CORBA.*;       // All CORBA applications need these classes.
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.*;

class GameCallbackImpl extends GameCallbackPOA
{
    private ORB orb;

    private static final int maxX = 8; // W
    private static final int maxY = 8; // H

    private GuiTextArea gbGUI;
    private char[][] mirrored_gameBoard = new char[maxX][maxY];

    //*** Methods ***
    public void setORB(ORB orb)
    {
        this.orb = orb;
    }

    public void setGUI(GuiTextArea gbgui)
    {
        this.gbGUI = gbgui;
    }

    public void startgame(String nickname, char colour)
    {
        gbGUI = new GuiTextArea("Othello - " + nickname + " on team " + colour);
    }

    public void closegame()
    {
        gbGUI = null;
    }

    public void boardupdate(String gbstate)
    {
        for (int i = 0 ; i < maxX ; ++i) {
            for (int j = 0 ; j < maxY ; ++j) {
                mirrored_gameBoard[i][j] = gbstate.charAt(j + i*8);
            }
        }
        renderboard();
    }

    private void renderboard()
    {
        gbGUI.clear();
        gbGUI.println("  | a b c d e f g h |");
        gbGUI.println("--|-----------------|--");

        for (int j = 0 ; j < maxY ; ++j) {
            gbGUI.print((j+1) + " | ");
            for (int i = 0 ; i < maxX ; ++i) {
                gbGUI.print(mirrored_gameBoard[i][j] + " ");
            }
            gbGUI.println("|");
        }

        gbGUI.println("--|-----------------|--");
        gbGUI.println("  |                 |  ");
    }
}
```

```
/**************************************
GameImpl.java
**************************************/
import ChatApp.*;
import org.omg.CosNaming.*; // HelloServer will use the naming service.
import org.omg.CosNaming.NamingContextPackage.*; // ..for exceptions.
import org.omg.CORBA.*;       // All CORBA applications need these classes.
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.*;


class GameImpl extends GamePOA
{
    private ORB orb;
    private ChatImpl chatImpl;

    private static final int maxX = 8; // W
    private static final int maxY = 8; // H

    private Map<String, GameCallback> clients = new HashMap<String, GameCallback>(); //
        Linking player to client GB
    private Map<String, String> players = new HashMap<String, String>(); //Linking
        player to a colour
    private char[][] gameBoard = new char[maxX][maxY];
    private boolean[][] legalMoves = new boolean[maxX][maxY];
    private char activeColour;
    private char opposingColour;

    //*** Initialization, Construction ***
    public GameImpl(ChatImpl chatImpl)
    {
        this.chatImpl = chatImpl;

        boolean manReset = false;
        reset(manReset);
    }


    //*** Methods ***
    public void setORB(ORB orb)
    {
        this.orb = orb;
    }

    public boolean join(ChatCallback chatref, GameCallback gbref, String nickname, char
        colour)
    {
        if (clients.containsKey(nickname)) {
            //Already playing
            chatref.callback("\u001b[31;1m" + nickname + "␣is␣playing␣already!\u001b[0m"
                );
            return false;
        }
        else {
            String colourString = String.valueOf(colour);

            System.out.println(colourString);

            clients.put(nickname, gbref);
            players.put(nickname, colourString);
            //Announce
            for (ChatCallback client : chatImpl.clients.values()) {
                if (client != chatref)
                    client.callback(nickname + "␣is␣now␣playing␣Othello␣on␣team␣" +
                        colourString + "!");
                else
                    client.callback("Joined␣Othello.");
            }
            //Send gameboard data to player
```

```
            gbref.startgame(nickname, colour);
            gbref.boardupdate( gbStringify() );
            return true;
        }
    }

    public boolean makemove(ChatCallback chatref, String nickname, String move)
    {
        System.out.println("makemove1");

        System.out.println(players.get(nickname));

        if (players.get(nickname).charAt(0) != activeColour) {
            chatref.callback("It's not your turn.");
            return false;
        }

        System.out.println("makemove2");

        int x = move.charAt(0) - 97; //int(char('a')) == 97
        int y = move.charAt(1) - 49;  //1 to 8 --> 0 to 7

        if (!inbounds(x, y)) {
            chatref.callback("Out of bounds.");
            return false;
        }

        System.out.println("makemove3");

        if (legalMoves[x][y] == false) {
            chatref.callback("You can't make that move.");
            return false;
        }

        //Perform move
        gameBoard[x][y] = activeColour;
        flip_affected(x, y);

        //Turn change
        turn_change_ao(opposingColour, activeColour);

        //Calculate legal moves for next turn.
        calc_legalMoves();

        //Update client gameboards
        for (GameCallback gbref : clients.values()) {
            gbref.boardupdate( gbStringify() );
        }

        return true;
    }

    public void passturn()
    {
        //Turn change
        turn_change_ao(opposingColour, activeColour);

        //Calculate legal moves for next turn.
        calc_legalMoves();
    }

    public void list(ChatCallback chatref)
    {
        Set<String> playersSet = players.keySet();
        Iterator<String> it = playersSet.iterator();
        Vector<String> teamx = new Vector();
        Vector<String> teamo = new Vector();
        String temp;

        while (it.hasNext()) {
```

```
            temp = it.next();
            if (players.get(temp).charAt(0) == 'x')
                teamx.add(temp);
            else
                teamo.add(temp);
        }

        chatref.callback(players.size() + " players playing Othello.");
        chatref.callback("Team x:");
        for (String player : teamx)
            chatref.callback(player);

        chatref.callback("Team o:");
        for (String player : teamo)
            chatref.callback(player);
    }

    public void leave(ChatCallback chatref, GameCallback gbref, String nickname)
    {
        gbref.closegame();
        clients.remove(nickname);
        players.remove(nickname);
        for (ChatCallback client : chatImpl.clients.values()) {
            if (client != chatref)
                client.callback(nickname + " stopped playing Othello.");
            else
                client.callback("You have stopped playing.");
        }
    }

    public void reset(boolean manReset)
    {
        if (manReset == true) {
            for (ChatCallback client : chatImpl.clients.values()) {
                    client.callback("\u001b[31;1m The gameboard was manually reset.\
                        u001b[0m");
            }
        }
        //Reset pieces.
        for (char[] column : gameBoard){
            Arrays.fill(column, '.');
        }
        gameBoard[3][3] = 'x';
        gameBoard[3][4] = 'o';
        gameBoard[4][3] = 'o';
        gameBoard[4][4] = 'x';

        //x is first to act. 'x' and 'o' becomes default colours.
        turn_change_ao('x', 'o');

        //Calculate legal moves for next turn.
        calc_legalMoves();

        //Update client gameboards
        for (GameCallback gbref : clients.values()) {
            gbref.boardupdate( gbStringify() );
        }
    }

    private String gbStringify()
    {
        String retString = new String();

        for (int i = 0 ; i < maxX ; ++i) {
            for (int j = 0 ; j < maxY ; ++j) {
                retString = retString + gameBoard[i][j];
            }
        }
        return retString;
    }
```

```
private boolean inbounds(int x, int y)
{
    return (x >= 0 &&
            y >= 0 &&
            x < maxX &&
            y < maxY);
}

private void turn_change_ao(char newactive, char newopposing)
{
    activeColour = newactive;
    opposingColour = newopposing;
}

private void calc_legalMoves()
{
    for (int x = 0 ; x < maxX ; ++x) {
        for (int y = 0; y < maxY ; ++y) {
            //(x,y) is illegal until we prove it's not.
            legalMoves[x][y] = false;

            //Check that (x,y) is empty
            if (gameBoard[x][y] != '.')
                continue;

            //Check all immediate neighbouring squares (x+i,y+j).
            outerloop:
            for (int i = -1 ; i < 2 ; ++i) {
                for (int j = -1 ; j < 2 ; ++j) {
                    if (inbounds(x+i, y+j) && !(i == 0 && j == 0)) {
                        if (gameBoard[x+i][y+j] == opposingColour) {
                            //Opposing piece found in neighbouring square, direction
                                (i,j).
                            for (int n = 1 ; inbounds(x+n*i, y+n*j) ; ++n) {
                                //Check if consecutive line of opposing piece can be
                                     made in direction (i,j) to a friendly piece.
                                if (gameBoard[x+n*i][y+n*j] == opposingColour)
                                    continue;
                                if (gameBoard[x+n*i][y+n*j] == '.')
                                    break;
                                if (gameBoard[x+n*i][y+n*j] == activeColour) {
                                    //Sequence of o x ... x o or vice versa found. (
                                        x,y) is a legal move.
                                    legalMoves[x][y] = true;
                                    break outerloop;
                                }
                            }

                        }
                    }
                }
            }
        }
    }
}

private void flip_affected(int x, int y)
{
    //Check all immediate neighbouring squares (x+i,y+j).
    for (int i = -1 ; i < 2 ; ++i) {
        for (int j = -1 ; j < 2 ; ++j) {
            if (inbounds(x+i, y+j)) {
                if (gameBoard[x+i][y+j] == opposingColour) {
                    //Opposing piece found in neighbouring square, direction (i,j).
                    for (int n = 1 ; inbounds(x+n*i, y+n*j) ; ++n) {
                        //Check if consecutive line of opposing piece can be made in
                             direction (i,j) to a friendly piece.
                        if (gameBoard[x+n*i][y+n*j] == opposingColour)
                            continue;
```

```
                        if (gameBoard[x+n*i][y+n*j] == '.')
                            break;
                        if (gameBoard[x+n*i][y+n*j] == activeColour) {
                            //Sequence of o x ... x o or vice versa found. Flip
                                pieces.
                            for (int m = 1 ; m < n ; ++m) {
                                gameBoard[x+m*i][y+m*j] = activeColour;
                            }
                            break;
                        }
                    }

                }
            }
        }
    }

}
```

```
/*****************************************
GuiTextArea.java
*****************************************/
import javax.swing.*;
import java.awt.Font;

public class GuiTextArea {

    JTextArea myArea;

    //---------------------
    GuiTextArea(String title) {

        //Create and set up the window
        JFrame frame = new JFrame(title);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        myArea = new JTextArea(20, 40);
        myArea.setEditable(false);
        myArea.setFont(new Font("monospaced", Font.PLAIN, 12));
        JScrollPane scrollPane =
            new JScrollPane(myArea,
                            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

        frame.getContentPane().add(scrollPane);

        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }

    //---------------------
    public void print(String s)   {
        myArea.append(s);
        myArea.setCaretPosition(myArea.getDocument().getLength());
    }
    public void println(String s) { print(s+"\n"); }
    public void println()         { print("\n"); }

    public void clear() {
        myArea.setText(null);
    }

}
```