```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.*;
import java.util.Scanner;


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;

  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);

      // 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));

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

      // Application code goes below
```

```
58          System.out.println("Welcome to the chat server!");
59          System.out.println("Please input command...");
60
61          boolean quit = false;
62          Scanner scanner = new Scanner(System.in);
63
64          while(!quit)
65          {
66            //Get user input
67            String input = scanner.nextLine();
68
69            //First word is command, rest is argument
70            String command = input.split(" ")[0];
71            String arguments = input.substring(input.indexOf(" ")+1, input.length());
72
73            //Look which command was used
74            if (command.equals("join"))
75            {
76              chatImpl.join(cref, arguments);
77            }
78            else if(command.equals("leave"))
79            {
80              chatImpl.leave(cref);
81            }
82            else if(command.equals("list"))
83            {
84              chatImpl.list(cref);
85            }
86            else if(command.equals("post"))
87            {
88              chatImpl.post(cref, arguments);
89            }
90            else if(command.equals("help"))
91            {
92              String errorMsg = "Available commands:\n";
93              errorMsg+="join <username>\nleave\nlist\npost <message>\nquit\ngomoku
                <color>\nadd <x> <y>";
94              System.out.println(errorMsg);
95            }
96            else if(command.equals("quit"))
97            {
98              //Leave user, then quit
99              chatImpl.leave(cref);
100             quit = true;
101           }
102           else if(command.equals("gomoku"))
103           {
104             chatImpl.gomoku(cref, arguments);
105           }
106           else if(command.equals("add") || command.equals("a"))
107           {
108             try
109             {
110               //Get first and second argument of the arguments
111               int x = Integer.parseInt(arguments.split(" ")[0]);
112               int y = Integer.parseInt(arguments.split(" ")[1]);
113               //Subtract 1 to get correct coordinates
```

```
114             //(correct in the means of how a coordinate system is represented)
115             chatImpl.add(cref, x-1, y-1);
116           }
117           catch(Exception e)
118           {
119             System.out.println(arguments + " are not valid coordinates");
120           }
121         }
122         else
123         {
124           System.out.println("\n'" + command + "' is not a valid command\nUse
              command 'help' for a list of available commands");
125         }
126       }
127     }
128     catch(Exception e) {
129         System.out.println("ERROR : " + e);
130         e.printStackTrace(System.out);
131     }
132   }
133 }
134
```