Enunciado Trabajo Práctico

Laberinto

Redes de Datos

Ing. Diego Azcurra – Ing. Damián Santos – Lic. Gustavo Fernandez

UNLA

2017

|  |  |  |
| --- | --- | --- |
| DNI | Nombre | Evaluación Individual |
| 39461683 | Andrés Filoso |  |
| 39587249 | Lucio Trucco |  |
| 39371903 | Joel Stella |  |
| 37750014 | Sergio Alejandro Peralta |  |
|  |  |  |
| Evaluación Trabajo |  | |

## Objetivo

Desarrollar una aplicación que permita jugar en un laberinto bajo el esquema de cliente y servidor.

## Descripción

## Laberinto

El laberinto es un área que cuenta con una única entrada y una única salida. Desde esa entrada surgen varios caminos de los cuales al menos uno permite llegar a la salida. Para salir, primero es necesario encontrar una llave. A su vez, a lo largo del laberinto se podrá encontrar oro. El oro sirve para “pagar” a los guardias que vigilan los caminos. Si nos cruzamos con un guardia y no tenemos oro, nos matará y tendremos que empezar de nuevo.

El laberinto es una matriz cuadrada de una determinada dimensión X. En cada posición, se podrá encontrar un carácter que especificará:

* E: Entrada.
* P: Pared.
* C: Camino.
* K: Llave.
* S: Salida.
* O: Oro.
* G: Guardia.

## Objetivo

El objetivo del juego es encontrar la salida recolectando la mayor cantidad de oro posible.

## Arquitectura

El juego se desarrollará bajo una arquitectura cliente servidor. La aplicación cliente será el jugador y se conectará al servidor para solicitar todos los elementos del juego.

## Desarrollo del juego

Para poder jugar, el cliente deberá conectarse al servidor mediante un usuario y contraseña.

El servidor posiciona al jugador en la casilla indicada con la letra E. En todos los casos, el servidor va revelar solo partes del laberinto: matrices cuadradas armadas desde un parámetro Y (desde la ubicación del jugador, se arma un cuadrado avanzando Y casillas en todos los sentidos). Desde esa posición, el jugador podrá moverse siguiendo los casilleros marcados con C. Al posicionarse en la nueva ubicación, solicitará al servidor que revele las casillas correspondientes.

Por ejemplo, tomemos una parte de un determinado laberinto:

000000000111111

123456789012345

01|PPPPPPPPPPPPPPP

02|ECCCCPPPCCCCCCP

03|PPPPCPPPCPPPPPP

04|PPPPCCCCCPPPPPP

05|PPPPCPPPCCCCCCP

06|PPPPCPCCCPPCPPP

07|PPPCCPPPCCCCPPP

El juego comienza en la posición (2,1). Suponiendo Y= 2, el servidor en este caso va a revelar únicamente:

000

123

01|PPP

02|ECC

03|PPP

04|PPP

Desde ahí, en este caso sólo podrá moverse en sentido horizontal a la posición (2,2). Para esto, el cliente enviará un comando de movimiento a la derecha y el servidor responderá con la nueva matriz:

0000

1234

01|PPPP

02|ECCC

03|PPPP

04|PPPP

La secuencia continúa en búsqueda de oro, de la llave y de la salida del laberinto.

## Requerimientos

Se pide desarrollar un sistema cliente/servidor que permita:

1. Servidor:
   * El servidor deberá permitir loggear jugadores.
   * Deberá administrar el juego. Recibirá comandos desde los jugadores y enviará respuestas.
2. Cliente:
   * Deberá poder conectarse a un servidor ingresando IP y puerto del mismo.
   * Luego de recibir la posición inicial del jugador, deberá interpretar los comandos ingresados por teclado, enviarlos al servidor y actuar en consecuencia.
3. Protocolo:
   * Todos los comandos que se intercambian entre cliente y servidor deberán respetar un protocolo definido por cada grupo. Como mínimo, se deberán contemplar los siguientes campos:
     + Encabezado:
       - Tipo de comando (Conexión, Movimiento, Tomar (para el oro o las llaves), Pagar, Abrir Puerta, etc).
       - Tamaño del mensaje: salvo que se decida trabajar con tamaños fijos.
     + Datos:
       - Datos de los distintos mensajes en el formato que decida el grupo.
   * El protocolo deberá ser documentado detalladamente. Tomar como referencia algún estándar de la industria (HTTP, FTP, telnet…). Por ejemplo:
     + - <https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto>
       - <https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP>
4. Seguridad:

El sistema deberá implementar al menos dos servicios de seguridad informática. El grupo seleccionará qué servicios y con qué implementaciones.

1. Consola cliente:

Se deberá desarrollar otra aplicación cliente que permita interactuar con el servidor mediante mensajes de pregunta y respuesta en formato texto (de la misma forma que lo hace el juego pero en forma manual).

El protocolo definido tendrá una cantidad de operaciones a las que se podrá acceder mediante un esquema de request/response. Esta aplicación deberá permitir que el usuario ingrese por consola un comando para que el cliente lo envíe al servidor y muestre la respuesta obtenida.

Por ejemplo, si el protocolo define que el mensaje de login es:

LOG|us:Damian|pass:Damian

Y la respuesta es:

LOG|200|ok

El usuario podrá ingresar esa cadena y deberá recibir la respuesta esperada. Si algún dato está mal, deberá recibir un código de error.

## Presentación

23/10/17: **Entrega del TP**.

06/11/17: **Recuperatorio del TP**.

**Condiciones de entrega**.

El trabajo entregado deberá contener:

* Una copia impresa del enunciado del trabajo práctico (TODO este documento, incluyendo los anexos).
* Una copia impresa de la estrategia de resolución del trabajo práctico. Es un texto descriptivo de cómo se estructuró la aplicación, como se separaron las capas, relaciones entre las entidades, es decir, todo aquello que consideren significativo para explicar la resolución del trabajo.
* El detalle del protocolo utilizado.
* El detalle de los servicios de seguridad informática aplicados y la forma en que fueron implementados.
* Una copia impresa de todos los archivos del proyecto. Poner como encabezado de cada hoja el nombre del archivo. Pueden imprimirse dos páginas por hoja, en formato horizontal.
* Un CD conteniendo en formato digital todos los puntos anteriores y el proyecto completo.
* La presentación deberá ser en un folio o carpeta, en forma prolija y debidamente identificada. Los CDs deberán contener el número de grupo y el nombre y los padrones o documentos de identidad de cada uno de los integrantes y deberán estar correctamente adjuntos al resto del trabajo práctico de forma tal que no puedan perderse. Además, deberá incluirse **todo** el proyecto desarrollado (**la carpeta completa** generada por el IDE, con los archivos del proyecto y el código fuente) incluyendo los archivos de pruebas.
* Todas las hojas deben estar numeradas.

El incumplimiento de cualquiera de las normas de entrega implicará la desaprobación del trabajo práctico.

Metodología de evaluación:

La Evaluación de los trabajos prácticos contará con una etapa grupal y una individual.

* Grupal: Se realizará un conjunto de pruebas sobre el trabajo presentado por los alumnos en presencia de los mismos. Se deberá aprobar la totalidad de las pruebas. En caso de que una prueba falle, los alumnos podrán intentar corregir el código mientras dure la evaluación.
* Individual: Se realizará una evaluación individual oral o escrita para cada alumno. Los temas a evaluar podrán ser, por ejemplo: preguntas teóricas sobre el contenido de la materia, preguntas sobre el trabajo práctico, codificación de alguna primitiva o modificación del trabajo práctico, etc.

La nota final del trabajo se calculará en función de las notas obtenidas en forma grupal e individual. La nota grupal será el promedio entre la primer presentación y el recuperatorio (en caso de necesitarlo). Por este motivo, SOLO deberán presentarse aquellos grupos que hayan concluido TODO el trabajo práctico ya que no se harán evaluaciones parciales.

## Revisiones

Se agregó el punto 5 dentro de los requerimientos.

Anexo – Correcciones

|  |  |  |
| --- | --- | --- |
| Prueba | Resultado | Comentario |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Anexo – Correcciones (copia para los alumnos)

|  |  |  |
| --- | --- | --- |
| Prueba | Resultado | Comentario |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Introducción

Este Laberinto fue desarrollado en Java, mediante el programa (IDE) Eclipse Oxigen. Se comenzó por realizar diferentes aplicaciones sencillas del tipo Cliente/Servidor (Chat, calculadora) a nivel de capacitación para poder lograr la resolución del juego.

Estrategia

Nuestra estrategia se desarrolla de la siguiente manera: Al abrir el servidor, éste predeterminadamente queda a la espera de usuarios que deseen unirse para jugar escuchando el puerto “9020”. Un usuario puede unirse al servidor mediante el formulario de inicio de sesión en donde envía sus credenciales, el puerto y el host desde el cual se conecta. Una vez que el servidor autorice las credenciales, devuelve las coordenadas de entrada. Cuando el cliente recibe las coordenadas de entrada, cierra el formulario de inicio de sesión y se posiciona sobre la coordenada recibida.

Al posicionarse en la entrada o cada vez que el jugador haga un movimiento, el cliente pide al servidor todos los valores de casilleros que se encuentran alrededor y los muestra en el laberinto. Todas las posiciones conocidas son guardadas por el cliente para no volver a pedirlas al servidor.

Por defecto, el Servidor tiene solo un mapa cargado. A nivel paliativo, se encuentran diferentes modelos de laberintos en la ruta src/extras/ Mapa de laberinto.txt los cuales pueden servir para editar el predefinido en Servidor.java. Una oportunidad de mejora para este laberinto sería que el servidor elija un mapa aleatoriamente al autorizar al cliente (Esto no está implementado ya que no era parte de los requisitos del tp)

El cliente es el encargado de verificar en qué posición se encuentra y determinar si es necesario tomar oro, pagar a un guardia o tomar llave. Así mismo, se encuentran predefinidas ciertas reglas para el que jugador no pueda pasar por paredes.

El servidor tiene únicamente dos tareas. Por un lado, cuando se inicia queda a la espera de autorizar credenciales y luego durante el resto del juego devuelve el valor de las coordenadas que le solicitan.

Se encuentra implementando una consola que permite iniciar un servidor sobre un puerto determinado, loguear a un usuario a un servidor determinado, mostrar el formulario de inicio de sesión, etc... Todos estos comandos se encuentran predefinidos y pueden ser consultados con el comando “help” desde la consola

Protocolo

Para realizar este juego utilizamos predeterminada el puerto 9020, el cual permitirá aceptar una nueva conexión por parte de un usuario y poder unirse al servidor. Como se explicó en la estrategia, el programa cuenta con una consola capaz de iniciar un servidor en cualquier puerto y loguear clientes de forma personalizada utilizando comandos predefinidos. Todas las trasferencias de datos desde y hacia el servidor utilizan los puertos predefinidos por el usuario al iniciar el juego.

Seguridad

En este punto utilizamos una función de encriptación mediante funciones de conversión de los datos que se van a enviar mediante los sockets. Una vez que los mensajes viajan hacia el servidor, se desencriptan para poder entenderlos, se obtienen los datos que el cliente necesita, se encriptan y se devuelven. Finalmente, cuando el cliente recibe los datos, los desencripta mediante otro método que le permite comprender los datos que acaba de recibir de forma útil. En cuanto al acceso al juego, el servidor valida credenciales para aceptar una conexión y enviar las primeras Coordenadas del laberinto, las cuales son imprescindibles para posicionarse en el mapa.

package modelo;

import java.awt.AWTException;

import java.awt.Color;

import java.awt.Font;

import java.awt.Robot;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.DataInputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class Cliente {

public static void main(String[] args) {

VentanaLogin ventanaLogin = new VentanaLogin();

ventanaLogin.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

class VentanaLogin extends JFrame implements ActionListener {

JButton btnLogin;

JTextField ip;

JTextField puerto;

JTextField user;

JTextField password;

JTextArea respuesta;

public VentanaLogin() {

JLabel titulo = new JLabel("Login");

titulo.setBounds(10, 10, 50, 20);

add(titulo);

JLabel ipLabel = new JLabel("IP:");

ipLabel.setBounds(10, 50, 50, 20);

add(ipLabel);

ip = new JTextField("localhost");

ip.setBounds(70, 50, 100, 20);

add(ip);

JLabel puertoLabel = new JLabel("Puerto: ");

puertoLabel.setBounds(10, 80, 50, 20);

add(puertoLabel);

puerto = new JTextField("9020");

puerto.setBounds(70, 80, 100, 20);

add(puerto);

JLabel userLabel = new JLabel("Usuario: ");

userLabel.setBounds(10, 110, 50, 20);

add(userLabel);

user = new JTextField("admin");

user.setBounds(70, 110, 100, 20);

add(user);

JLabel passLabel = new JLabel("Clave: ");

passLabel.setBounds(10, 140, 50, 20);

add(passLabel);

password = new JTextField("admin");

password.setBounds(70, 140, 100, 20);

add(password);

btnLogin = new JButton();

btnLogin.setText("Ingresar");

btnLogin.setBounds(10, 180, 150, 20); // (x, y, ancho, alto)

btnLogin.addActionListener(this); // para poder programar en el boton

add(btnLogin);

JLabel respuestaLabel = new JLabel("Respuesta del servidor: ");

respuestaLabel.setBounds(10, 290, 150, 20);

add(respuestaLabel);

respuesta = new JTextArea(" Esperando...");

respuesta.setBounds(10, 320, 150, 20);

add(respuesta);

setLayout(null); // para que los controles no esten uno encima del otro

setTitle("Cliente Login");

setSize(400, 400); // tamaño del layout

setResizable(false);

setLocationRelativeTo(null);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnLogin) {

try {

boolean esperando = true;

Coordenada entradaXY = new Coordenada();

Login login = new Login(ip.getText(), puerto.getText(), user.getText(), password.getText());

Socket cliente = new Socket(ip.getText(), Integer.parseInt(puerto.getText()));

ObjectOutputStream out = new ObjectOutputStream(cliente.getOutputStream());

out.writeObject(login);

while (esperando == true) {

ObjectInputStream entrada = new ObjectInputStream(cliente.getInputStream());

entradaXY = (Coordenada) entrada.readObject();

if (entradaXY != null) {

esperando = false;

}

}

setVisible(false);

VentanaCliente ventanaCliente = new VentanaCliente(entradaXY, ip.getText(), puerto.getText());

ventanaCliente.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

} catch (Exception ex) {

System.out.println("Error en cliente " + ex.getMessage());

}

}

}

}

class VentanaCliente extends JFrame {

int oro = 0;

boolean llave = false;

public static int key = 346;

int cSize = 30;

int posx = 70;

int posy = 50;

int xEntrada = 0;

int yEntrada = 0;

public static Coordenada posicionActual;

public static KeyListener listener;

JTextArea[][] laberinto = new JTextArea[20][20];

Coordenada[][] mapa = new Coordenada[20][20];

ArrayList<Coordenada> oroYaTomado = new ArrayList<Coordenada>();

JTextArea casillero;

JTextArea txtMensajes;

JLabel oroLabel;

Configuracion config;

public VentanaCliente(Coordenada entrada, String ip, String puerto) {

posicionActual = entrada;

xEntrada = entrada.getX();

yEntrada = entrada.getY();

listener = new MyKeyListener();

addKeyListener(listener);

setFocusable(true);

config = new Configuracion(ip, puerto);

JLabel entradaLabel = new JLabel("Coordenadas de entrada: " + entrada.getX() + "; " + entrada.getY());

entradaLabel.setBounds(10, 10, 200, 20);

add(entradaLabel);

oroLabel = new JLabel("Oro: " + oro);

oroLabel.setBounds(300, 10, 200, 20);

oroLabel.setFont(new Font("Arial", Font.PLAIN, 25));

add(oroLabel);

txtMensajes = new JTextArea();

txtMensajes.setBounds(450, 20, 300, 1000);

// add(txtMensajes);

txtMensajes.setEditable(false);

consola("Inicio de juego.");

dibujarLaberinto(20, false);

initMapa();

setLayout(null);

setTitle("Laberinto");

setSize(800, 800);

setLocationRelativeTo(null);

setVisible(true);

try {

mostrarVecinos();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public void initMapa() {

System.out.println("Inicializando mapa");

for (int i = 0; i < 20; i++) {

for (int j = 0; j < 20; j++) {

mapa[i][j] = new Coordenada(i, j, false);

}

}

mapa[xEntrada][yEntrada].setKnown(false);

}

public JTextArea posicionar(JTextArea casillero, Coordenada c) {

casillero.setBounds(laberinto[c.getX()][c.getY()].getX(), laberinto[c.getX()][c.getY()].getY(), 35, 35);

return casillero;

}

public void consola(String mensaje) {

Calendar calendario = new GregorianCalendar();

txtMensajes.append(calendario.get(Calendar.HOUR) + ":" + calendario.get(Calendar.MINUTE) + ":"

+ calendario.get(Calendar.SECOND));

txtMensajes.append(": " + mensaje + "\n");

}

public void mostrarVecinos() throws NumberFormatException, UnknownHostException, IOException {

ArrayList<Coordenada> coordenadas = new ArrayList<Coordenada>();

if (mapa[posicionActual.getX()][posicionActual.getY()].isKnown() == false) {

Coordenada izquierda = new Coordenada(posicionActual.getX() - 1, posicionActual.getY());

Coordenada derecha1 = new Coordenada(posicionActual.getX() + 1, posicionActual.getY());

Coordenada arriba = new Coordenada(posicionActual.getX(), posicionActual.getY() - 1);

Coordenada arriba1 = new Coordenada(arriba.getX() + 1, arriba.getY());

Coordenada abajo = new Coordenada(posicionActual.getX(), posicionActual.getY() + 1);

Coordenada abajo1 = new Coordenada(abajo.getX() + 1, abajo.getY());

coordenadas.add(posicionActual);

coordenadas.add(izquierda);

coordenadas.add(derecha1);

coordenadas.add(arriba);

coordenadas.add(arriba1);

coordenadas.add(abajo);

coordenadas.add(abajo1);

} else {

Coordenada izquierda = new Coordenada(posicionActual.getX() - 1, posicionActual.getY());

Coordenada derecha1 = new Coordenada(posicionActual.getX() + 1, posicionActual.getY());

Coordenada arriba = new Coordenada(posicionActual.getX(), posicionActual.getY() - 1);

Coordenada abajo = new Coordenada(posicionActual.getX(), posicionActual.getY() + 1);

coordenadas.add(izquierda);

coordenadas.add(derecha1);

coordenadas.add(arriba);

coordenadas.add(abajo);

}

for (Coordenada c : coordenadas) {

if (c.getX() >= 0 && c.getX() <= 19 && c.getY() >= 0 && c.getY() <= 19) {

mapa[c.getX()][c.getY()].setLetra(devolverLetra(config.getIp(), config.getPuerto(), c));

if (mapa[c.getX()][c.getY()].isKnown() == false) {

add(posicionar(format(new JTextArea(), mapa[c.getX()][c.getY()].getLetra(), c), c), c);

}

SwingUtilities.updateComponentTreeUI(this);

mapa[c.getX()][c.getY()].setKnown(true);

}

add(posicionar(format(new JTextArea(), " ", posicionActual), posicionActual));

}

}

public static String devolverLetra(String ip, String puerto, Coordenada c)

throws NumberFormatException, UnknownHostException, IOException {

int letraEncript = 0;

String letra = "";

Socket cliente = new Socket(ip, Integer.parseInt(puerto));

ObjectOutputStream pedido = new ObjectOutputStream(cliente.getOutputStream());

c = encriptarCoordenada(c);

pedido.writeObject(c);

DataInputStream recibir = new DataInputStream(cliente.getInputStream());

letraEncript = recibir.readInt();

letra = desencriptarLetra(letraEncript);

cliente.close();

return letra;

}

public static Coordenada encriptarCoordenada(Coordenada c) {

Coordenada encript = new Coordenada();

encript.setX(c.getX() \* key);

encript.setY(c.getY() \* key);

return encript;

}

public static String desencriptarLetra(int encript) {

String decript = "";

int aux = (encript / key);

switch (aux) {

case 1: decript = "P"; break;

case 2: decript = "C"; break;

case 3: decript = "O"; break;

case 4: decript = "G"; break;

case 5: decript = "E"; break;

case 6: decript = "S"; break;

case 7: decript = "K"; break;

}

return decript;

}

public JTextArea format(JTextArea j, String nombre, Coordenada c) {

j.setText(nombre);

j.setFont(new Font("Arial", Font.PLAIN, 25));

j.setEditable(false);

if (j.getText().equalsIgnoreCase("C"))

j.setBackground(Color.GREEN);

if (j.getText().equalsIgnoreCase("P"))

j.setBackground(Color.DARK\_GRAY);

if (j.getText().equalsIgnoreCase("O"))

j.setBackground(Color.YELLOW);

if (j.getText().equalsIgnoreCase("G"))

j.setBackground(Color.MAGENTA);

if (j.getText().equalsIgnoreCase("K"))

j.setBackground(Color.RED);

if (j.getText().equalsIgnoreCase(" "))

j.setBackground(Color.WHITE);

if (posicionActual.equals(c))

j.setBackground(Color.ORANGE);

return j;

}

public void dibujarLaberinto(int size, boolean reiniciar) {

if (!reiniciar) {

int cSize = 30;

JTextArea txtmensajes;

int a = 70, b = 50;

int i = 0, j = 0;

while (i < size) {

txtmensajes = new JTextArea();

txtmensajes.setFont(new Font("Arial", Font.PLAIN, 25));

txtmensajes.setBounds(a, b, cSize, cSize);

txtmensajes.setEditable(false);

laberinto[i][j] = txtmensajes;

while (j < size - 1) {

b += cSize + 5;

j++;

txtmensajes = new JTextArea();

txtmensajes.setFont(new Font("Arial", Font.PLAIN, 25));

txtmensajes.setBounds(a, b, cSize, cSize);

txtmensajes.setEditable(false);

laberinto[i][j] = txtmensajes;

}

b = 50;

j = 0;

a += cSize + 5;

i++;

}

}

}

public boolean checkOro() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

boolean tomado = false;

try {

for (Coordenada c : oroYaTomado) {

if (c.equals(posicionActual)) {

consola("Aca no hay mas oro. Ya lo agarraste!");

tomado = true;

}

}

if (tomado == false) {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("O")) {

oro++;

respuesta = true;

laberinto[posicionActual.getX()][posicionActual.getY()].setText("X");

oroYaTomado.add(new Coordenada(posicionActual.getX(), posicionActual.getY()));

}

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN ORO: " + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean checkGuardia() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

try {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("G")) {

if (oro == 0) {

JOptionPane.showMessageDialog(null, "No tiene oro para pagar al guardia. Perdirse!");

posicionActual.setX(xEntrada);

posicionActual.setY(yEntrada);

respuesta = true;

salir();

} else {

oro--;

JOptionPane.showMessageDialog(null, "Guardia! Perdiste 1 oro. Ahora tienes: " + oro + " de oro");

}

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN GUARDIA" + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean checkSalida() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

try {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("S")) {

if (llave == true) {

JOptionPane.showMessageDialog(null, "Ganaste! Llegaste a la salida");

posicionActual.setX(xEntrada);

posicionActual.setY(yEntrada);

respuesta = true;

salir();

} else {

JOptionPane.showMessageDialog(null, "No podes salir, todavia no encontraste la llave!");

respuesta = false;

}

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN GUARDIA" + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean checkLlave() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

try {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("K")) {

llave = true;

JOptionPane.showMessageDialog(null, "Encontraste la llave. Ahora encontra la salida!");

respuesta = true;

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN ORO: " + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean limites(String direccion) throws NumberFormatException, UnknownHostException, IOException {

boolean resultado = true;

if (direccion.equals("derecha")) { // Derecha

if (posicionActual.getX() == 19) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX() + 1, posicionActual.getY())).equalsIgnoreCase("P")) {

resultado = false;

}

}

if (direccion.equals("arriba")) { // Arriba

if (posicionActual.getY() == 0) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY() - 1)).equalsIgnoreCase("P")) {

resultado = false;

}

}

if (direccion.equals("abajo")) { // Abajo

if (posicionActual.getY() == 19) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY() + 1)).equalsIgnoreCase("P")) {

resultado = false;

}

}

if (direccion.equals("izquierda")) { // Izquierda

if (posicionActual.getX() == 0) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX() - 1, posicionActual.getY())).equalsIgnoreCase("P")) {

resultado = false;

}

}

return resultado;

}

public void salir() throws NumberFormatException, UnknownHostException, IOException {

devolverLetra(config.getIp(), config.getPuerto(), new Coordenada(-1, -1));

this.setVisible(false);

System.exit(0);

return;

}

public void mover(String direccion) {

try {

if (limites(direccion)) {

if (direccion.equals("izquierda"))

posicionActual.setX(posicionActual.getX() - 1);

if (direccion.equals("arriba"))

posicionActual.setY(posicionActual.getY() - 1);

if (direccion.equals("derecha"))

posicionActual.setX(posicionActual.getX() + 1);

if (direccion.equals("abajo"))

posicionActual.setY(posicionActual.getY() + 1);

if (checkOro()) {

format(new JTextArea(), " ", posicionActual);

}

checkGuardia();

checkSalida();

if (llave == false)

checkLlave();

mostrarVecinos();

}

} catch (NumberFormatException | IOException e1) {

e1.printStackTrace();

}

}

class MyKeyListener implements KeyListener {

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

// System.out.println("keyPressed=" + e.getKeyCode());

oroLabel.setText("Oro: " + Integer.toString(oro));

String direccion = "";

switch (e.getKeyCode()) {

case 37: // Izquierda

direccion = "izquierda";

break;

case 38: // Arriba

direccion = "arriba";

break;

case 39: // Derecho

direccion = "derecha";

break;

case 40: // Abajo

direccion = "abajo";

break;

}

mover(direccion);

}

@Override

public void keyReleased(KeyEvent e) {

// TODO Auto-generated method stub

}

}

public void keyReleased(KeyEvent e) {

}

}

class Login implements Serializable {

private String ip;

private String puerto;

private String user;

private String pass;

public Login(String ip, String puerto, String user, String pass) {

this.ip = ip;

this.puerto = puerto;

this.user = user;

this.pass = pass;

}

public Login() {

this.user = "unlogged";

}

public String getIp() {

return ip;

}

public String getPuerto() {

return puerto;

}

public void setUser(String user) {

this.user = user;

}

public String getUser() {

return user;

}

public String getPass() {

return pass;

}

}

class Configuracion {

private String ip;

private String puerto;

public Configuracion(String ip, String puerto) {

this.ip = ip;

this.puerto = puerto;

}

public String getIp() {

return ip;

}

public void setIp(String ip) {

this.ip = ip;

}

public String getPuerto() {

return puerto;

}

public void setPuerto(String puerto) {

this.puerto = puerto;

}

package modelo;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

public class Servidor extends JFrame implements Runnable {

JTextArea txtMensajes;

String authUser = "admin";

String authPass = "admin";

int puerto;

int matrizSize = 20;

JTextArea[][] laberinto = new JTextArea[matrizSize][matrizSize];

public Servidor(int p) {

puerto = p;

txtMensajes = new JTextArea();

txtMensajes.setBounds(450, 20, 300, 1000);

txtMensajes.setEditable(false);

// add(txtMensajes);

setLayout(null);

setTitle("Servidor");

setSize(800, 800);

setLocationRelativeTo(null);

setResizable(false);

setVisible(true);

Thread hilo = new Thread(this);

hilo.start(); // llama al metodo run que creamos

}

public static void main(String[] args) {

new Servidor(9020);

}

@Override

public void run() {

try {

ServerSocket servidorJuego = new ServerSocket(puerto);

Socket cliente; // socket para guardar las llamadas

System.out.println("Servidor iniciado en puerto: " + puerto);

consola("Servidor iniciado en puerto: " + puerto);

boolean juego = false;

while (juego == false) {

cliente = servidorJuego.accept(); // esperando una llamada

ObjectInputStream autorizacion = new ObjectInputStream(cliente.getInputStream());

Login login = (Login) autorizacion.readObject();

System.out.println("Usuario a validar: " + login.getUser());

System.out.println("Password a validar: " + login.getPass());

if (autorizar(login.getUser(), login.getPass())) {

consola("Bienvenido: " + login.getUser());

consola("Cerrando login y creando mapa de juego...");

consola("Tamaño del laberinto: " + matrizSize);

laberinto = dibujarLaberinto(matrizSize);

Coordenada entradaXY = coordenadasEntrada(laberinto);

ObjectOutputStream devolucionCoordenada = new ObjectOutputStream(cliente.getOutputStream());

devolucionCoordenada.writeObject(entradaXY);

consola("Enviando coordenadas de entrada");

juego = true; // arranca el juego

} else {

Coordenada denegado = new Coordenada (100,100);

ObjectOutputStream devolucionCoordenada = new ObjectOutputStream(cliente.getOutputStream());

devolucionCoordenada.writeObject(denegado);

consola("Acesso denegado.. Servidor cerrado");

//servidorJuego.close();

}

}

while (juego == true) {

try {

Coordenada aux;

// socket para guardar las llamadas

consola("Esperando peticion de casillero...");

// System.out.println("Esperando peticion de casillero...");

cliente = servidorJuego.accept(); // esperando una llamada

// System.out.println("Peticion recibida");

consola("Petición recibida");

ObjectInputStream solicitudMovimiento = new ObjectInputStream(cliente.getInputStream());

aux = (Coordenada) solicitudMovimiento.readObject();

aux = desencriptarCoordenada(aux);

if (aux.getX() == -1 && aux.getY() == -1) {

consola("Cerrando Servidor de juego por coordenadas -1;-1");

juego = false;

servidorJuego.close();

setVisible(false);

break;

}

String valor = devolverValor(aux, laberinto);

consola("Se devuelve valor: " + valor);

// System.out.println("Se devuelve letra: " + valor);

DataOutputStream devolver = new DataOutputStream(cliente.getOutputStream());

int encript = encriptarValorLetra(valor);

// System.out.println("El valor de la letra es: " + encript);

devolver.writeInt(encript);

// System.out.println("Se devuelve letra: " + valor);

} catch (Exception e) {

// JOptionPane.showMessageDialog(null, e.getMessage());

System.out.println(e.getMessage());

// e.printStackTrace();

}

} // while juego == true

} catch (IOException e) {

// JOptionPane.showMessageDialog(null, e.getMessage());

System.out.println(e.getMessage());

// e.printStackTrace();

} catch (ClassNotFoundException e) {

JOptionPane.showMessageDialog(null, e.getMessage());

// e.printStackTrace();

}

}

public void consola(String mensaje) {

Calendar calendario = new GregorianCalendar();

txtMensajes.append(calendario.get(Calendar.HOUR) + ":" + calendario.get(Calendar.MINUTE) + ":"+ calendario.get(Calendar.SECOND));

txtMensajes.append(": " + mensaje + "\n");

}

public boolean autorizar(String user, String pass) {

boolean resultado = false;

if (user.equals(authUser) && pass.equals(authPass)) {

resultado = true;

} else {

System.out.println("Credenciales incorrectas");

}

return resultado;

}

public String devolverValor(Coordenada c, JTextArea[][] array) {

String valor = "";

valor = array[c.getX()][c.getY()].getText();

return valor;

}

public Coordenada desencriptarCoordenada(Coordenada c) {

Coordenada decript = new Coordenada();

decript.setX(c.getX() / VentanaCliente.key);

decript.setY(c.getY() / VentanaCliente.key);

return decript;

}

public int encriptarValorLetra(String letra) {

int encript = 0;

switch (letra) {

case "P":

encript = (1 \* VentanaCliente.key);

break;

case "C":

encript = (2 \* VentanaCliente.key);

break;

case "O":

encript = (3 \* VentanaCliente.key);

break;

case "G":

encript = (4 \* VentanaCliente.key);

break;

case "E":

encript = (5 \* VentanaCliente.key);

break;

case "S":

encript = (6 \* VentanaCliente.key);

break;

case "K":

encript = (7 \* VentanaCliente.key);

break;

}

return encript;

}

public Coordenada coordenadasEntrada(JTextArea[][] array) {

int x = 0;

int y = 0;

for (int k = 0; k < matrizSize; k++) {

for (int l = 0; l < matrizSize; l++) {

if (array[k][l].getText().equalsIgnoreCase("E")) {

x = k;

y = l;

}

}

}

return new Coordenada(x, y);

}

public JTextArea[][] dibujarLaberinto(int size) {

String fila1[] = { "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "S", "P" };

String fila2[] = { "E", "C", "C", "C", "C", "P", "P", "P", "O", "P", "C", "G", "C", "C", "P", "O", "C", "C", "C", "P" };

String fila3[] = { "P", "P", "P", "P", "C", "P", "P", "P", "C", "P", "C", "P", "P", "C", "P", "P", "P", "C", "O", "P" };

String fila4[] = { "P", "P", "P", "P", "C", "P", "P", "P", "C", "P", "C", "P", "P", "G", "C", "O", "C", "G", "C", "P" };

String fila5[] = { "P", "O", "C", "C", "C", "C", "C", "C", "C", "P", "C", "P", "P", "C", "P", "P", "P", "C", "P", "P" };

String fila6[] = { "P", "P", "P", "P", "P", "P", "C", "P", "G", "P", "C", "P", "P", "C", "P", "P", "P", "C", "P", "P" };

String fila7[] = { "P", "C", "C", "C", "P", "P", "G", "P", "C", "P", "O", "C", "C", "G", "C", "O", "C", "G", "C", "P" };

String fila8[] = { "P", "C", "P", "C", "P", "P", "C", "P", "C", "P", "C", "P", "P", "P", "P", "P", "P", "P", "C", "P" };

String fila9[] = { "P", "O", "P", "C", "C", "C", "C", "P", "O", "P", "C", "P", "P", "P", "P", "P", "P", "P", "C", "P" };

String fila10[] = { "P", "C", "P", "P", "P", "P", "P", "P", "P", "P", "C", "P", "C", "C", "C", "O", "C", "C", "C", "P" };

String fila11[] = { "P", "C", "P", "P", "P", "P", "P", "P", "P", "P", "C", "P", "C", "P", "P", "p", "P", "P", "C", "P" };

String fila12[] = { "P", "C", "C", "C", "C", "O", "C", "C", "C", "C", "G", "P", "O", "C", "C", "G", "C", "O", "O", "P" };

String fila13[] = { "P", "P", "P", "P", "P", "P", "P", "P", "C", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P" };

String fila14[] = { "P", "O", "O", "P", "P", "P", "P", "P", "C", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P" };

String fila15[] = { "P", "O", "O", "C", "C", "G", "C", "C", "C", "C", "C", "C", "O", "C", "C", "C", "C", "C", "P", "P" };

String fila16[] = { "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "C", "P", "P" };

String fila17[] = { "P", "O", "K", "O", "G", "C", "C", "C", "C", "C", "C", "C", "C", "O", "G", "P", "P", "C", "C", "P" };

String fila18[] = { "P", "P", "P", "P", "P", "P", "P", "P", "G", "P", "P", "P", "P", "P", "P", "P", "P", "P", "C", "P" };

String fila19[] = { "P", "O", "O", "G", "C", "C", "C", "C", "C", "C", "C", "C", "C", "O", "C", "C", "C", "C", "O", "P" };

String fila20[] = { "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P" };

String[][] filas = { fila1, fila2, fila3, fila4, fila5, fila6, fila7, fila8, fila9, fila10, fila11, fila12,

fila13, fila14, fila15, fila16, fila17, fila18, fila19, fila20 };

int cSize = 30;

JLabel sizeLabel;

sizeLabel = new JLabel("Tamaño del laberinto: " + size);

sizeLabel.setBounds(10, 10, 200, 20);

add(sizeLabel);

JTextArea txtmensajes;

JTextArea[][] array = new JTextArea[size][size];

int a = 70, b = 50;

int i = 0, j = 0;

while (i < size) {

txtmensajes = new JTextArea();

txtmensajes.setFont(new Font("Arial", Font.PLAIN, 25));

txtmensajes.setBounds(a, b, cSize, cSize);

txtmensajes.setEditable(false);

array[i][j] = txtmensajes;

txtmensajes.setText(filas[j][i]);

add(array[i][j]);

while (j < size - 1) {

b += cSize + 5;

j++;

txtmensajes = new JTextArea();

txtmensajes.setFont(new Font("Arial", Font.PLAIN, 25));

txtmensajes.setBounds(a, b, cSize, cSize);

txtmensajes.setEditable(false);

array[i][j] = txtmensajes;

txtmensajes.setText(filas[j][i]);

add(array[i][j]);

}

b = 50;

j = 0;

a += cSize + 5;

i++;

}

for (int k = 0; k < size; k++) {

for (int l = 0; l < size; l++) {

array[k][l].setBackground(Color.LIGHT\_GRAY);

}

}

return array;

}

}

class Coordenada implements Serializable {

private int x;

private int y;

private boolean known;

private String letra;

public Coordenada(int x, int y) {

this.x = x;

this.y = y;

this.known = false;

this.letra = "";

}

public Coordenada(int x, int y, boolean known) {

this.x = x;

this.y = y;

this.known = known;

this.letra = "";

}

public Coordenada() {

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public boolean isKnown() {

return known;

}

public void setKnown(boolean known) {

this.known = known;

}

public String getLetra() {

return letra;

}

public void setLetra(String letra) {

this.letra = letra;

}

@Override

public String toString() {

return x + " ;" + y + ", known=" + known;

}

public boolean equals(Coordenada c) {

boolean respuesta = false;

if (this.getX() == c.getX() && this.getY() == c.getY()) {

respuesta = true;

} else {

respuesta = false;

}

return respuesta;

}

package modelo;

import java.awt.AWTException;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Point;

import java.awt.Robot;

import java.awt.event.ActionEvent;

import java.awt.event.KeyEvent;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.sound.midi.Synthesizer;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

public class ConsolaComandos {

public static void main(String[] args) {

VentanaConsola ventanaConsola = new VentanaConsola();

ventanaConsola.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

class VentanaConsola extends JFrame {

boolean juego = false;

JTextArea bash;

JTextField bashInput = new JTextField();

JButton btnEnviar;

JScrollPane jp;

String init = "Consola Cliente para Laberinto v.1\nIngrese \"help\" para mas información\n\n";

Login loginBash = new Login();

public VentanaConsola() {

bash = new JTextArea(init);

bash.setFont(new Font("Arial", Font.PLAIN, 15));

bash.setEditable(false);

jp = new JScrollPane(bash);

add(jp, BorderLayout.CENTER);

bashInput.addActionListener(action);

bashInput.setFont(new Font("Arial", Font.BOLD, 15));

bashInput.setPreferredSize(new Dimension(0, 30));

add(bashInput, BorderLayout.SOUTH);

setTitle("Consola de Comandos");

setSize(800, 500);

setLocationByPlatform(true);

setVisible(true);

}

Action action = new AbstractAction() {

String msg = "";

@Override

public void actionPerformed(ActionEvent e) {

Calendar calendario = new GregorianCalendar();

String time = (calendario.get(Calendar.HOUR) + ":" + calendario.get(Calendar.MINUTE) + ":"

+ calendario.get(Calendar.SECOND));

msg = e.getActionCommand();

if (loginBash.getUser().equalsIgnoreCase("unlogged")) {

bash.append(time + " | " + msg + "\n");

bashInput.setText(null);

procesar(msg);

} else {

bash.append(time + " | " + loginBash.getUser() + " | " + msg + "\n");

bashInput.setText(null);

procesar(msg);

}

jp.getViewport().setViewPosition((new Point(0, bash.getHeight()))); // Scroll siempre abajo

};

};

public void procesar(String msg) {

try {

String parametros[] = msg.split(" ");

String comando = parametros[0];

String noLogin = "Error: Antes de ejecutar un comando de movimiento tiene que conectarse al servidor\n"

+ "Utilice el comando login user pass ip port para autenticar\n\n";

for (int i = 0; i < parametros.length; i++) {

System.out.println(parametros[i]);

}

switch (comando) {

case "help":

VentanaFAQS ventanaFAQS = new VentanaFAQS();

break;

case "init":

if (parametros[1].equalsIgnoreCase("server")) {

int puerto = Integer.parseInt(parametros[2]);

new Servidor(puerto);

} else if (parametros[1].equalsIgnoreCase("login") && parametros[2].equals("gui")) {

System.out.println("login gui");

VentanaLogin ventanaLogin = new VentanaLogin();

}

break;

case "login":

if (!msg.equalsIgnoreCase("login")) {

bash.append("Autorizando usuario: " + parametros[1] + "\n");

Login login = new Login(parametros[3], parametros[4], parametros[1], parametros[2]);

loguear(login);

} else {

bash.append("Error: Para utilizar el comando login tiene que ingresar credenciales, ip y puerto\n\n");

}

break;

case "exit":

if (!loginBash.getUser().equalsIgnoreCase("Unlogged")) {

cerrar();

} else {

bash.append("Error: Antes de ejecutar el comando \"exit\" tiene que conectarse al servidor.\n\n");

}

break;

case "status":

if (juego == true) {

bash.append("Posicion Actual: " + VentanaCliente.posicionActual.getX() + " ;"

+ VentanaCliente.posicionActual.getY() + " \n\n");

}

break;

case "clear":

bash.setText(init);

break;

case "izquierda":

case "arriba":

case "derecha":

case "abajo":

if (juego == false) {

bash.append(noLogin);

} else {

// No esta desarrollado

}

break;

default:

bash.append("Error: \"" + msg+ "\" no se reconoce como un comando válido. Ingrese \"help\" para mas información\n\n");

throw new IllegalArgumentException("Error: \"" + msg

+ "\" no se reconoce como un comando válido. Ingrese \"help\" " + "para mas información\n\n");

}

} catch (Exception e) {

// loginBash.setUser("Unlogged");

System.out.println(e.getMessage());

}

}

public boolean loguear(Login login) {

boolean resultado = true;

Coordenada entradaXY = new Coordenada();

Socket cliente;

try {

cliente = new Socket(login.getIp(), Integer.parseInt(login.getPuerto()));

ObjectOutputStream out = new ObjectOutputStream(cliente.getOutputStream());

out.writeObject(login);

ObjectInputStream entrada = new ObjectInputStream(cliente.getInputStream());

entradaXY = (Coordenada) entrada.readObject();

if(entradaXY.getX() == 100 && entradaXY.getY() == 100) {

juego = false;

bash.append("Servidor | " + login.getPuerto() + " | Credenciales incorrectas\n");

}else {

bash.append("Servidor | " + login.getPuerto() + " | Autorizado\n");

loginBash = login;

juego = true;

VentanaCliente ventanaCliente = new VentanaCliente(entradaXY, login.getIp(), login.getPuerto());

ventanaCliente.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

} catch (NumberFormatException | IOException | ClassNotFoundException e) {

bash.append("Error en cliente: " + e.getMessage() + "\n\n");

System.out.println(e.getMessage());

}

return resultado;

}

public void cerrar() throws NumberFormatException, UnknownHostException, IOException {

bash.append("Cerrando Servidor por instruccion de usuario\n\n");

VentanaCliente.devolverLetra(loginBash.getIp(), loginBash.getPuerto(), new Coordenada(-1, -1));

}

}

class VentanaFAQS extends JFrame {

JLabel titulo, help, clear, initServer, loginGui, login, exit, izquierda, arriba, derecha, abajo, status;

int y = 50;

int x = 10;

int w = 300;

int h = 20;

public VentanaFAQS() {

titulo = new JLabel("Guia de comandos");

titulo.setBounds(x, 10, w, h);

titulo.setFont(new Font("Arial", Font.PLAIN, 20));

add(titulo);

help = new JLabel("help");

help.setBounds(x, y, w, h);

help.setFont(new Font("Arial", Font.BOLD, 15));

add(help);

y += 30;

clear = new JLabel("clear");

clear.setBounds(x, y, w, h);

clear.setFont(new Font("Arial", Font.BOLD, 15));

add(clear);

y += 30;

initServer = new JLabel("init server port");

initServer.setBounds(x, y, w, h);

initServer.setFont(new Font("Arial", Font.BOLD, 15));

add(initServer);

y += 30;

loginGui = new JLabel("init login gui");

loginGui.setBounds(x, y, w, h);

loginGui.setFont(new Font("Arial", Font.BOLD, 15));

add(loginGui);

y += 30;

login = new JLabel("login user pass ip port");

login.setBounds(x, y, w, h);

login.setFont(new Font("Arial", Font.BOLD, 15));

add(login);

y += 30;

status = new JLabel("status");

status.setBounds(x, y, w, h);

status.setFont(new Font("Arial", Font.BOLD, 15));

add(status);

y += 30;

exit = new JLabel("exit");

exit.setBounds(x, y, w, h);

exit.setFont(new Font("Arial", Font.BOLD, 15));

add(exit);

y += 30;

setLayout(null); // para que los controles no esten uno encima del otro

setTitle("Comandos");

setSize(400, 500); // tamaño del layout

setLocationRelativeTo(null);

setResizable(false);

setVisible(true);

}

}