Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

PRÁCTICA 2 - MUSIC/SP

GRUPO 3 - Sistemas Legados

David Zandundo Fuster (780500)
Álvaro Andrada Gimeno (795326)
Rubén Albacete Seren (736650)

Introducción

Para esta práctica se pide implementar una interfaz que se comunique con una sencilla aplicación legada, cuya función es la de mostrar una lista de tareas. Se pueden añadir nuevas tareas, divididas en Generales y Específicas, y mostrarlas después.
Esta aplicación se encuentra sobre un mainframe IBM ESA/390 con el sistema operativo MUSIC/SP. Para comunicarnos con el mainframe se utiliza la técnica de 'scraping' a través de un terminal IBM 3270.
Para crear el programa con interfaz gráfica se ha decidido utilizar Java con la biblioteca Java FX que permite crear pantallas fácilmente mediante un formato tipo XML.

Configuración

A continuación se detallan los elementos utilizados para desarrollar la aplicación:

- Un proyecto de JavaFX en el entorno de programación IntelliJ, que proporciona un esqueleto inicial para crear las pantallas necesarias.
- Toda la aplicación está hecha en el lenguaje Java, para la lógica de la aplicación y la comunicación por scraping. Para la creación de las pantallas de la aplicación se han escrito en formato 'fxml'.
- El terminal displayless utilizado ha sido 'ws3270.exe', su ejecutable para Windows.
- Para entender cómo funcionaba la aplicación legada y saber los comandos que había que introducir en cada momento se ha utilizado el programa con la interfaz gráfica llamado wx3270

Acceso al emulador de terminal x3270

El programa x3270 es el que se encarga de gestionar toda la comunicación con el mainframe. Una vez iniciado, acepta comandos (por la entrada estándar o a través de un puerto TCP) y devuelve la salida de los comandos. Se distribuye en dos variantes: la versión gráfica (x3270) y la versión para usarse en scrapers (s3270). Algunos de los comandos que acepta x2370 y que han sido utilizados son:

- Connect(<ip>:<port>) para abrir una conexión con un host
- Disconnect para desconectar
- String("<str>") para escribir texto
- Enter para enviar el texto escrito
- Tab para enviar una pulsación del tabulador
- Snap Ascii para imprimir por pantalla una captura previamente guardada

Desarrollo

Toda la comunicación con la terminal ws3270 se instancia en la misma clase denominada ‘WS3270.java’. Para mantener una misma sesión durante toda la ejecución, dentro de la clase se mantiene una variable static de la propia clase. De este modo, en vez de llamar a su constructor cada vez, lo que hacemos es ejecutar una función que verifica si aún no se ha establecido la comunicación (null) o, en su defecto, si ya existe una sesión activa.

```
private static WS3270 session = null;
```

```
public static WS3270 getSession() {
    if (session == null) {
        session = new WS3270();
    }
    return session;
}
```

Finalmente, en el constructor lo que hacemos es crear la comunicación y capturar la entrada y la salida estándar para su uso posterior.

```
this.emulator = Runtime.getRuntime().exec(ws3270exe);
lectura = this.emulator.getInputStream();
teclado = new PrintWriter(new OutputStreamWriter(this.emulator.getOutputStream()));
```

En este fichero (‘WS3270.java’) también se encuentran todas las funciones que interactúan directamente con la terminal del ws3270. Cabe destacar las funciones ‘escribirLinea()’ que escribe en la salida capturada anteriormente y ‘leerPantalla()’ que almacena en un buffer todos los caracteres de entrada capturada anteriormente, estas están desarrolladas de la siguiente manera:

```
public void escribirLinea(String cadena) throws InterruptedException {
    do {
        cadena += "\n";
        this.teclado.write(cadena);
        this.teclado.flush();
    } while (leerPantalla().toString().contains(OK));
    Thread.sleep(delay);
}
```

```
public StringBuilder leerPantalla() {
    StringBuilder cadena = new StringBuilder();
    try {
        while (lectura.available() == 0); //Espera a que se llene el buffer
        while (lectura.available() > 0) {
            cadena.append((char) lectura.read());
        }
    } catch (IOException ex) {
        cadena = null;
        System.out.println("IOException");
    } finally {
        System.out.println(cadena);
        return cadena;
    }
}
```

Toda la lógica de la aplicación se encuentra en el fichero que hemos denominado 'Actions.java'. Aquí tenemos métodos que encapsulan, por un lado, la conexión con el sistema MUSIC/SP y, por el otro, la interacción con la aplicación de tareas.
Por ejemplo, esto es lo que se hace para conectarse:

```
public void conectar() throws InterruptedException {
    session.conectar(ip, puerto);
    session.enter();
}
```

Al tener que hacer screen-scraping para comunicarnos, hay que mandar al terminal el texto correspondiente en cada momento. En este caso, primero se ejecuta 'conectar()' con una ip (155.210.152.51) y un puerto (3270), previamente proporcionados, que lo que hará dentro de WS3270 es llamar a la función 'escribirLinea' escribiendo "connect ip:puerto". Después de eso, pulsaremos enter para confirmar, y del mismo modo, esto se hace escribiendo "ENTER" en la terminal.
Una vez que el terminal está conectado necesitamos introducir el grupo (usuario) al que pertenecemos y la contraseña (pass), para ello se le presenta una pantalla al usuario (GUI)
Una vez escritos por el usuario, se llama a la siguiente función (login) que escribe en la terminal los comandos necesarios ('escribirCadena' es una función idéntica a 'escribirlinea' pero antepone la cadena 'string ' para que la terminal lo asimile como un string y no como un comando propio de WS3270)

```
public int login(String usuario, String pass) throws InterruptedException {
    //Escribir nombre de usuario
    escribirCadena(usuario);
    enter();
    // Si el usuario no es valido pulsa F3 para cancelar
    if (buscarCadena("Userid is not authorized")) {
        System.out.println("Userid is not authorized");
        teclaFuncion(3);
        enter();
        return -1; }
    //Escribir pass
    escribirCadena(pass);
    enter();
    // Si pass no es valida pulsa F3 para cancelar
    if (buscarCadena("Password incorrect!")) {
        teclaFuncion(3);
        enter();
        return -2; }
    // Si el ID esta en uso se fuerza el acceso
    if (buscarCadena("Userid is in use.")) {
        escribirCadena("OK");
        enter(); }
    // Si esta correcto devuelve true
    if (buscarCadena("Press ENTER to continue...")) {
        enter();
        comenzarPrograma();
        return 0; }
    // Por defecto devuelve false
    return -10;
}
```

Cuando se comprueba que el grupo y la contraseña son correctos se llama a la función ‘comenzarPrograma’ que lo único que hace es escribir en la terminal “tareas.c” y pulsar ENTER. Después de esto, se le presenta al usuario una pantalla con todas las posibles acciones que puede realizar (añadir tarea general/específica y listar tareas general/específica).

Como ejemplo de ejecución durante ‘tareas.c’, se muestra a continuación la función que se ejecuta cuando el usuario quiere añadir una nueva tarea de tipo general:

```
session.escribirCadena("1");
enterTareas();
session.escribirCadena("1");
enterTareas();
session.escribirCadena(dia+mes);
enterTareas();
session.escribirCadena(descripcion);
enterTareas();
session.escribirCadena("3");
enterTareas();
```

Este es un buen ejemplo de lo que ha sido escribir palabras específicas en la terminal. Para añadir una nueva tarea mientras estás en el menú principal de ‘tareas.c’, debes escribir ‘1’ y pulsar ‘enter’, lo cual te lleva a elegir si quieres añadir una tarea general o específica. Si queremos que sea general pulsamos de nuevo ‘1’ y ‘enter’, momento en el cual la aplicación pregunta por el día y mes para la tarea y, una vez introducidos, por una descripción de la misma. Después de añadirse correctamente la tarea, lo que hacemos es pulsar ‘3’, porque sino no vuelve al menú principal de ‘tareas.c’.
Para añadir una tarea específica es idéntico pero la aplicación también te pide una campo de nombre.

Para las funcionalidades de listar se muestra a continuación la función de listar las tareas generales:

```
public ArrayList<TareaGeneral> getTareasGenerales() {
    try {
        session.escribirCadena("2");
        session.enter();
        session.escribirCadena("1");
        session.enter();
        session.ascii();
        String s = session.leerPantalla().toString();
        if(s.contains("More...")) {
            session.enter();
        }
        session.escribirCadena("3");
        session.enter();
        ArrayList<TareaGeneral> tareas = new ArrayList<~>();
        int start = s.indexOf("data: TASK ");
        int end = s.indexOf( str: "\r\n",start);
        int next = s.indexOf( str: "data: TASK ",end);
        while(start != -1) {
            System.out.println("While");
            if((s.indexOf( str: " GENERAL",start) - start) < 17) {
                System.out.println("if");
                end = s.indexOf( str: "\r\n",start);
                String sAux;
                if(end != -1) {sAux = s.substring(start,end);}
```

Este es un buen ejemplo de lo que ha sido hacer el scraping de la terminal.
Para listar las tareas generales mientras estás en el menú principal de ‘tareas.c’, debes escribir ‘2’ y pulsar ‘enter’, lo cual te lleva a elegir si quieres listar tareas generales o específicas. Si queremos las tareas generales pulsamos ‘1’ y ‘enter’, momento en el cual la aplicación lista por pantalla de la terminal todas las tareas generales siguiendo un patrón, la función lee todos los caracteres de la pantalla, detecta el patrón y por consiguiente las tareas. Una vez que se capturan todas las tareas son enviadas a una función que las presenta por pantalla al usuario (GUI). Después de listar correctamente las tareas, lo que hacemos es pulsar ‘3’, porque sino no vuelve al menú principal de ‘tareas.c’.
Para listar las tareas específicas es idéntico pero la aplicación enseña también el contenido del campo nombre.

Principales dificultades encontradas

- El principal problema inicial que tuvimos fue conseguir tener en ejecución la terminal en java. No entendiamos como funcionaba y estuvimos probando bastante tiempo con varias de las opciones que se encuentran en wx3270. Después de varios intentos, al final conseguimos hacerlo con ws3270.

- Una vez comenzada la interacción, observamos que a veces conseguimos conectarnos mientras que otras no. Haciendo uso de acciones como 'ASCII' para visualizar lo que ocurría en pantalla, pudimos ver que en ocasiones las acciones introducidas se sobreescriben unas a otras, ya que para hacer cualquier cosa hay que mandar varias órdenes a la terminal y, además, pulsar 'ENTER' para confirmarlas todas. Esto hacía que el mainframe no tuviese tiempo de reaccionar a todo lo que recibía. Nuestra solución a esto fue crear un delay artificial cada vez que escribimos algo en la pantalla, por eso todas las funciones tienen una excepción 'InterruptedException':

```
public void escribirLinea(String cadena) throws InterruptedException {
    do {
        cadena += "\n";
        this.teclado.write(cadena);
        this.teclado.flush();
    } while (leerPantalla().toString().contains(OK));
    Thread.sleep(delay);
}
```

- Otro problema fue controlar la pantalla durante la ejecución de 'tareas.c'. Cada vez que escribes algo, la pantalla se va llenando hasta que llega hasta abajo y te muestra una cadena 'More…'. Cuando esto pasa, debes pulsar 'ENTER' para que se borre entera, muestre el texto que falta y puedas continuar normalmente con la ejecución. Esto puede ocurrir en cualquier momento, mientras te muestra un listado o estás introduciendo comandos, lo cual llevaba a introducir parámetros cuando no tocaba con los consiguientes errores en la ejecución. Para solucionar esto, se creó un 'ENTER' especial que solo se ejecuta mientras estamos en 'tareas.c'. La idea es que después de cada valor introducido, se compruebe si ha aparecido en la pantalla una cadena 'More…'. Si es así, se pulsa un 'ENTER' extra y continúa con la ejecución. En caso contrario, simplemente continúa con la ejecución.

```
private void enterTareas() throws InterruptedException {
    session.enter();
    session.hayMasTexto();
}
```

```
public void hayMasTexto() throws InterruptedException {
    ascii();
    String cadenaAux = leerPantalla().toString();
    if(cadenaAux.contains(MORE)) {
        enter();
    }
}
```

Demostración del funcionamiento

En el siguiente enlace se muestra un video donde se prueban todas las funcionalidades de la aplicación.

https://drive.google.com/file/d/13lI7mHyKYajzRyeKHBYbaX3zZUPNv3hh/view?usp=sharing

Ejecución de la práctica

Abrir proyecto en carpeta “practica_2” con Eclipse, y ejecutar desde ahí (cambiar directorio de emulador ws3270.exe en WS3270.java (en /src/main/java/com/example/practica_2/comunication), se ha incluido el ejecutable en el mismo directorio que la carpeta del proyecto).