

SISTEMAS LEGADOS

Práctica 2

Programa de Tareas

Martín Gascón Laste (764429)

Daniel Huici Meseguer (758635)

Eduardo Ruiz (764539)

Resumen

En esta segunda práctica se ha creado una GUI para el programa tareas.c que se ejecuta sobre un mainframe IBM ESA/390 con el sistema operativo MUSIC/SP. Para realizar esta tarea ha sido necesario usar el emulador s3270 que está desarrollado para realizar aplicaciones screen-scraping. Y a partir de ese emulador se ha encapsulado tareas.c. En este documento se muestra la estructura de la aplicación, los problemas que se han presentado, así como la organización entre los miembros que conformamos el grupo.

Aplicación de tareas

Se ha usado el patrón arquitectural MVC (Model-View-Controller) para la arquitectura de nuestra aplicación. De esta forma se encapsula toda la interacción con el programa tareas.c en la clase Handler3270. Para poder mostrar las tareas y que el usuario puede insertar tareas desde la interfaz se ha implementado el patrón Observer.

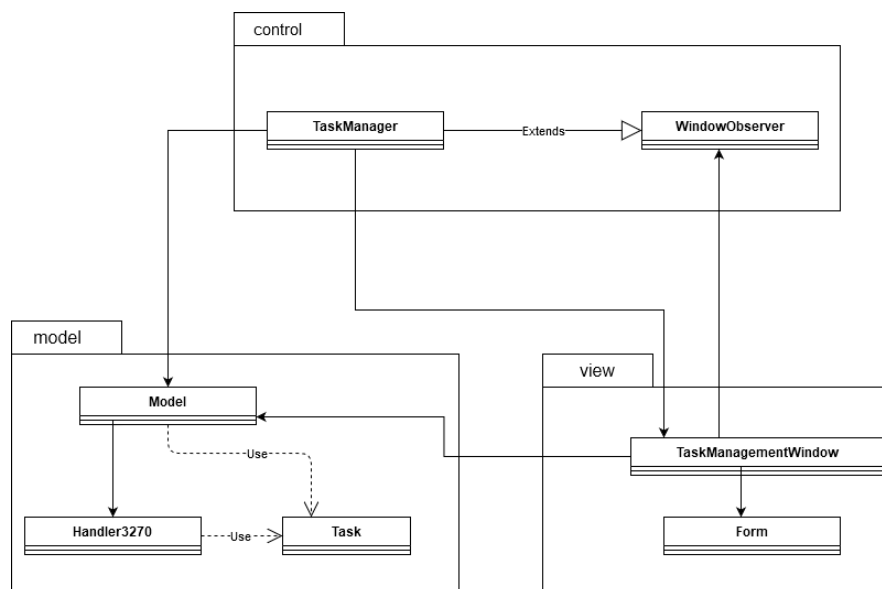


Imagen 1

Interprete del emulador 3270

Para poder ejecutar las ordenes de nuestra aplicación en el programa tareas.c ha sido necesario crear una clase que sea la intermediaria con el emulador.

Clase Handler3270

Esta clase es la encargada de realizar el papel de intermediario entre el emulador [s3270](#) y nuestra aplicación, permitiendo al usuario ver y añadir tareas usando el programa tareas.c. Para poder realizar esta función se ha usado el programa [x3270if](#) el cual proporciona una interfaz entre los scripts y el emulador [s3270](#).

Para poder ejecutar los comandos en el emulador, la primera instrucción del constructor es iniciar el emulador [s3270](#) y se ha creado el método *executeCommand* (*String command*). En este método se inicia y se ejecuta el programa [x3270if](#) pasando por parámetro el comando que se quiere ejecutar. Se ha añadido un sleep, debido a que se han detectados posibles fallos al intentar ejecutar otro comando antes de que termine el anterior.

```
private void executeCommand(String command) throws IOException, InterruptedException {
    Runtime.getRuntime().exec(new String[]{X3270IF_PATH, "-t", EMULATOR_PORT, command}).waitFor();
    Thread.sleep( millis: 750);
}
```

Imagen 2

Para poder iniciar tareas.c, añadir una tarea y obtener tareas es necesario realizar una cadena de comandos en cada una. Se han realizado pruebas de forma manual usando el emulador [x3270](#) para detectar que comandos eran necesarios ejecutar. Una vez se han tenido las cadenas de comandos se ha creado una lista en cada uno de los métodos que implementan dichas funcionalidades, y se llama a un método que corre la lista y ejecuta uno a uno los comandos. Esto se puede ver en la imagen 3, la cual muestra el método usado para iniciar tareas.c.

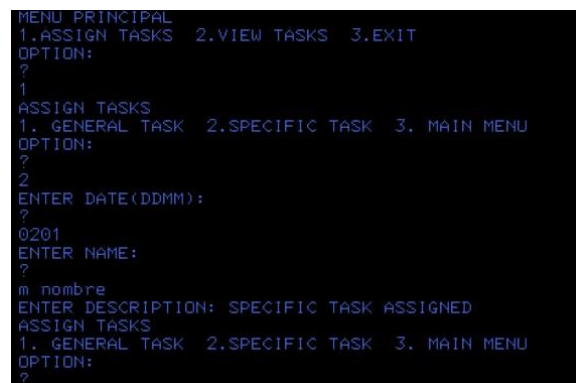
```
private void startProgram() throws IOException, InterruptedException {
    System.out.println("Starting mainframe connection...");
    List<String> startCommands = List.of("connect(" + IP_HOST + ":" + PORT + ")", "Enter",
        "String(\"" + USER + "\")", "Enter", "String(\"" + PASSWORD + "\")", "Enter", "Enter",
        "String(\"tareas.c\")", "Enter", "wait(InputField)");
    executeCommands(startCommands);

    System.out.println("Ready. Awaiting user inputs..");
}
```

Imagen 3

Problemas solucionados

Al realizar pruebas con el emulador [x3270](#) se detectaron varios problemas que se iban a tener que solucionar al en la clase Handler3270. Uno de estos problemas era que el programa tareas.c no soluciona correctamente al introducir en el nombre o descripción al añadir una nueva tarea. El programa tareas.c detecta que las palabras escritas después del espacio es la siguiente entrada que se va a introducir, se puede observar en la imagen 4. Para solucionar este problema se ha creado un método que cambia los espacios de entrada por | y al devolver se llama otro método que realiza la función contraria.



```
MENU PRINCIPAL
1.ASSIGN TASKS  2.VIEW TASKS  3.EXIT
OPTION:
?
1
ASSIGN TASKS
1. GENERAL TASK  2.SPECIFIC TASK  3. MAIN MENU
OPTION:
?
2
ENTER DATE(DDMM):
?
0201
ENTER NAME:
?
m nombre
ENTER DESCRIPTION: SPECIFIC TASK ASSIGNED
ASSIGN TASKS
1. GENERAL TASK  2.SPECIFIC TASK  3. MAIN MENU
OPTION:
?
```

Imagen 4

Otro problema que se detectó en las pruebas y se ha solucionado es que después añadir una nueva tarea u obtener tareas hay que volver al menú principal para poder ejecutar correctamente la siguiente acción. Para ello se ha creado un método que introduce la opción 3 en el emulador y simula la pulsación de Enter hasta que el programa se encuentre en el menú principal. Otro problema que se ha solucionado es que al agregar una tarea si se ponía un nombre o descripción demasiado largos. El problema es que al mostrar la tarea se solapa con la siguiente tarea guardada para ello se ha puesto un limite de caracteres a estos dos campos.

Problemas con estado del programa

En este apartado se va a explicar problemas que tienen que ver con el estado del programa y que no se detectaron al realizar las pruebas, pero se solucionaron en la implementación. El primer problema que se soluciono fue que al iniciar el programa tareas.c el tiempo que se tarda en iniciar varía de ejecución a ejecución. Para solucionar este problema después de iniciar el programa se ejecuta el comando *wait (InputField)*, este comando hace que el emulador espere hasta que se detecte que se puede escribir.

Los problemas que más ha costado solucionar han sido que daba cuando la pantalla del emulador se llenaba. Para solucionar los problemas que se producían al introducir una entrada se ha creado un método que simula la pulsación de Enter hasta que tareas.c está en modo lectura. A parte de al introducir dato este evento también producía errores siempre que se quería obtener una información específica de la

pantalla. Uno de estos casos es cuando al comprobar si el programa se encontraba en el menú principal, si la pantalla estaba completa y no aparecía “?” (símbolo por el que se hace la partición) saltaba una excepción. Para solucionar este error el programa ahora comprueba si tareas.c esta en modo lectura, y en caso de no estarlo se realiza la partición por otra cadena de caracteres, esto se puede ver en la imagen 5.

```
private Boolean isNotMainMenu() throws IOException, InterruptedException {
    String screen = getScreen();
    String[] linesScreen;
    int position = 0;
    if(emulatorIsNotReading()) {
        linesScreen = screen.split( regex: "-----T-----");
        linesScreen = linesScreen[linesScreen.length - 2].split( regex: "\\?");
        position = linesScreen.length - 1;
    }else {
        linesScreen = screen.split( regex: "\\?");
        position = linesScreen.length - 2;
    }
    if(linesScreen[position].contains("MENU PRINCIPAL")) {
        return false;
    }
    return true;
}
```

Imagen 5

Los mayores problemas se han encontrado al obtener la lista de tareas guardadas, esto se debe al evento explicado anteriormente. Dependiendo de donde se completaba la pantalla la forma de obtener las tareas era diferente, ya que tareas.c puede haber mostrado o no alguna tarea en pantalla y puede que queden tareas por mostrar. Para solucionar este error antes de obtener las tareas se comprueba si la pantalla esta completa y si se ha mostrado alguna tarea en la pantalla actual. En caso de que se cumplan ambos casos se obtienen las tareas de la pantalla actual, se llama a un método que pone la pantalla nueva y se obtienen las tareas mostradas en la nueva pantalla. Si no se cumplen esas condiciones se obtienen las tareas de la pantalla actual. Se puede ver esta implementación en la imagen 6.

```
if (emulatorIsNotReading() && screenTasks()) {
    String[] linesScreen = getScreen().split( regex: "VIEW TASK");
    String[] lastData = linesScreen[linesScreen.length - 1].split( regex: "\\n");
    for (String data : lastData) {
        if (data.startsWith("TASK ")) {
            tasks.add(parseTask(data));
        }
    }
}

checkMainFrameStatus();
lastData = getScreen().split( regex: "\\?");
lastData = lastData[lastData.length - 1].split( regex: "\\n");
for (String data : lastData) {
    if (data.startsWith("TASK ")) {
        tasks.add(parseTask(data));
    }
}
} else {
    String[] linesScreen = getScreen().split( regex: "TOTAL TASK");
    String[] lastData = linesScreen[linesScreen.length - 2].split( regex: "\\?");
    lastData = lastData[lastData.length - 1].split( regex: "\\n");
    for (String data : lastData) {
        if (data.startsWith("TASK ")) {
            tasks.add(parseTask(data));
        }
    }
}
```

Imagen 6

Distribución del trabajo.

Tarea	Miembro del equipo	Tiempo invertido/persona
Ejecutar , probar y conectar con emulador	Eduardo, Daniel y Martín	4,5h
Crear wrapper	Eduardo y Daniel	5h
Solución problemas wrapper	Eduardo	6,5h
Solución problemas wrapper	Daniel	4h
Solución problemas wrapper	Martín	2h
Creación de GUI y estructura de aplicación	Martín	4h
Realización de la memoria.	Eduardo, Martín y Daniel	1,5h