PROYECTO FINAL DE ENTORNOS DE DESARROLLO

Descripción del proyecto.

Vamos a realizar un proyecto que incluya todas las actividades estudiadas durante el curso. Para esto, cada grupo deberá plantear un software que cumpla los requisitos que se detallarán a continuación. No se pretende que sea una gran aplicación de gestión, puede ser una aplicación sencilla (una calculadora, un juego, gestión de datos sobre unos datos con una tabla, etc.).

Organización.

- Se realizará por grupos. Se identificarán en el foro correspondiente.

Actividades a desarrollar.

- ***Descripción del problema a resolver. Deberá de haber dos tipos de usuarios o actores. Cada uno de ellos tendrá que realizar como mínimo dos acciones en el sistema. Esto no implica tener un usuario o contraseña, sino que pueda existir un usuario normal y uno experto, dos niveles de juego, etc.***

Se va a plantear el diseño de una aplicación llamada "Repeated File Finder" (RFF), enfocada en la búsqueda de archivos y la comprobación de su presencia en diferentes directorios. El programa debe cumplir el objetivo de encontrar los archivos a través de una búsqueda que inserte valores en un campo de texto mediante el teclado; y, además, proporcionará una opción para localizar archivos que se encuentre duplicados en dos directorios diferentes, con el propósito de que el usuario pueda localizarlos fácilmente, ayudando tanto a limpiar espacio en su dispositivo de almacenamiento como a verificar la presencia de los archivos deseados en una copia de seguridad.

- ***Análisis de requisitos. Se detallarán:***
    - ***Requisitos funcionales (se implementará por lo menos uno)***
    - ***Requisitos no funcionales (no se implementarán)***

<u>Requisitos funcionales:</u>

1. *Búsqueda de archivos.* La aplicación permitirá al usuario realizar búsquedas de archivos por nombre, mediante la introducción del mismo a través del teclado en un campo de texto.

2. *Visualización de los resultados*. El programa mostrará los archivos cuyo nombre contenga la información introducida por el usuario en el campo de texto.
3. *Búsqueda en directorios concretos*. La aplicación debe reconocer directorios concretos para realizar las comparaciones correctamente.
4. *Búsqueda de archivos duplicados*. El programa proporcionará una opción para buscar archivos que se encuentran duplicados en dos directorios distintos.
5. *Comprobación de la presencia de los archivos.* La aplicación comprobará si existen archivos en los directorios especificados.

Requisitos no funcionales:

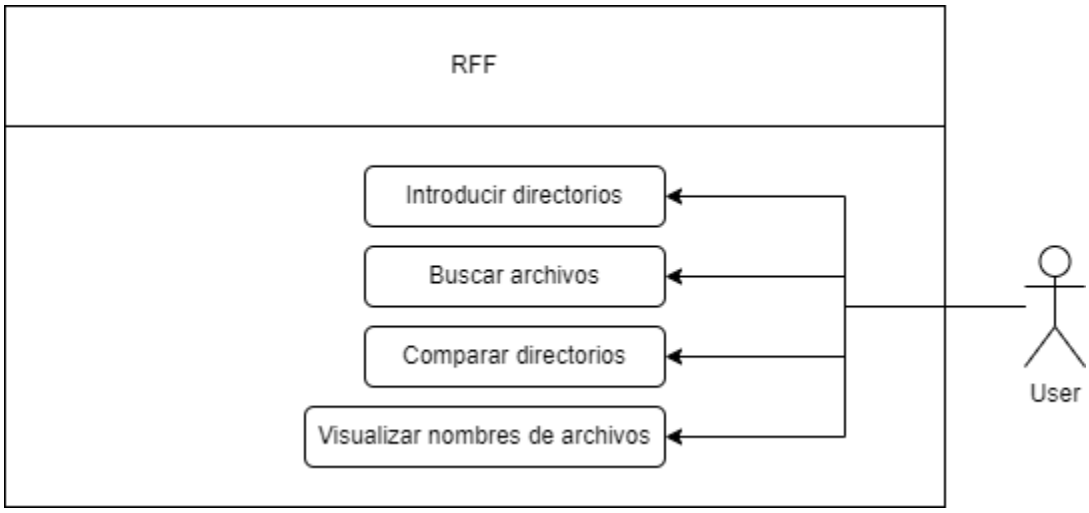
1. *Rendimiento*. La aplicación debe realizar rápidamente las búsquedas y las comparaciones entre directorios.
2. *Interfaz de usuario*. La interfaz del programa debe ser intuitiva y sencilla, facilitando la búsqueda para que pueda ser realizada por cualquier usuario.
3. *Mantenimiento*. El software debe aportar una documentación detallada que permita una manutención y actualización periódicas, facilitando las labores de mantenimiento y resolución de problemas al administrador del sistema.
4. *Portabilidad.* La aplicación debe ser compatible con varios sistemas operativos.

- ***Diseño de casos de uso.***

  En este apartado, se detallan los casos de uso de la aplicación:

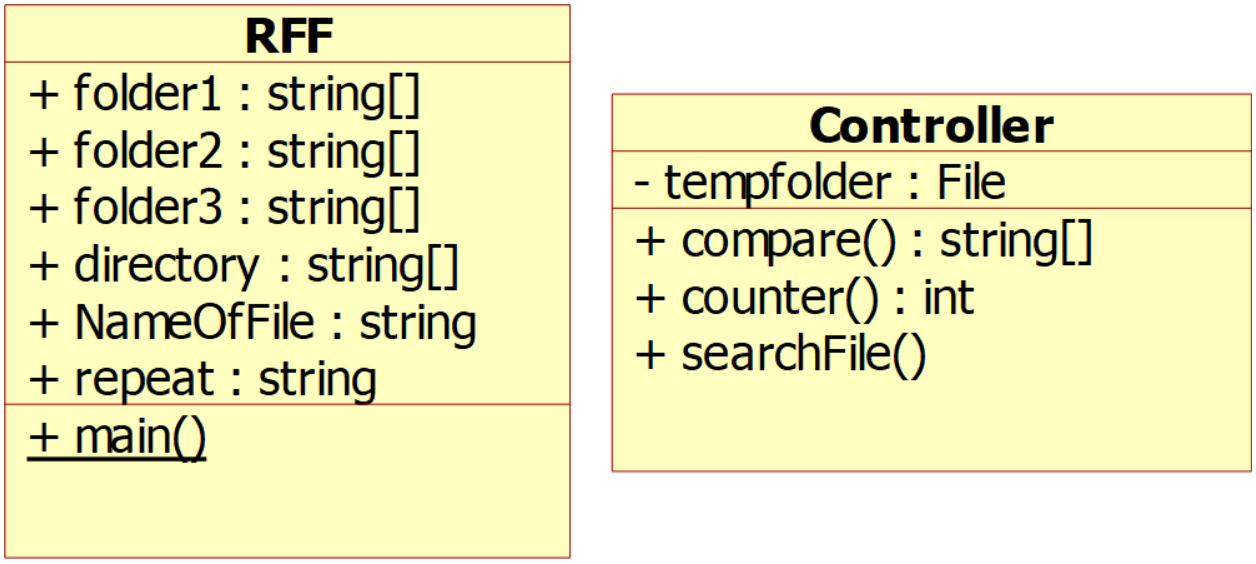
  - Búsqueda de archivos por nombre. Cuando el usuario desea buscar archivos concretos en un directorio, debe abrir el programa RFF y especificar el directorio en el que quiere buscar y, además, tendrá que escribir el nombre del archivo que desea encontrar. El programa efectuará una búsqueda en directorios y subdirectorios, mostrando finalmente la ruta absoluta del archivo que contenga el nombre especificado por el usuario.

  - Búsqueda de archivos repetidos. Cuando el usuario desea buscar posibles archivos duplicados en dos directorios distintos, debe abrir la aplicación RFF y especificar los dos directorios en los que desea realizar la búsqueda. El programa se encargará de ejecutar la búsqueda y mostrar los archivos con el mismo nombre que estén presentes en ambos directorios.



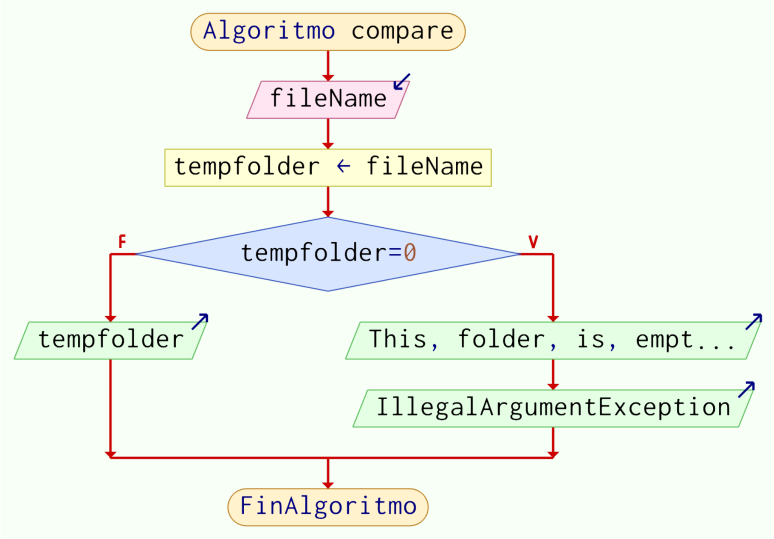
- ***Diagramas de clases. Se realizará la implementación con el código para probar su funcionamiento.***

El diagrama de clases obtenido para este programa sería el siguiente:



- ***Diseño del algoritmo de alguno de los requisitos funcionales (un cálculo, un proceso con datos, etc.).***

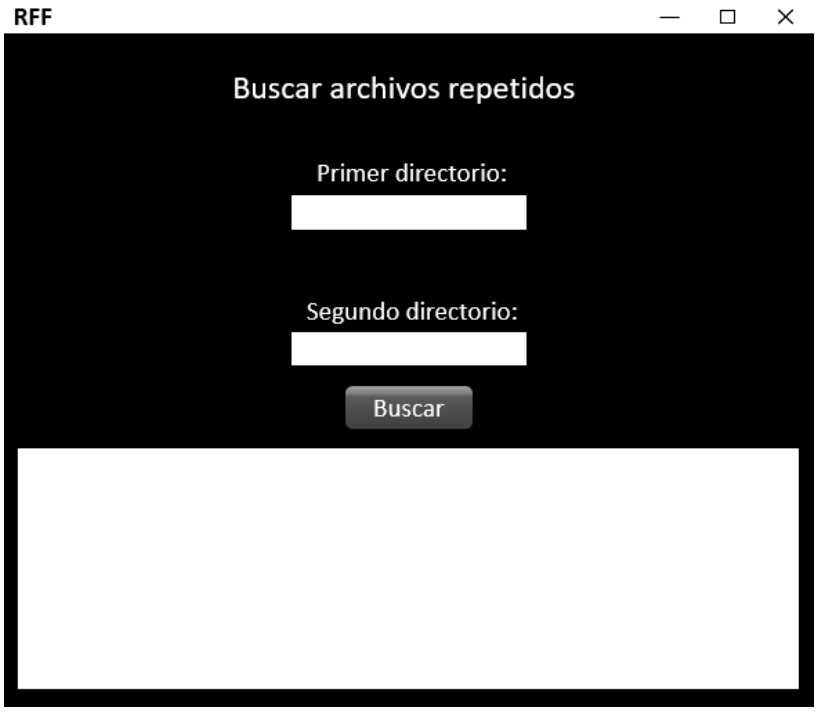
Se implementa el requisito funcional “Búsqueda de archivos duplicados” con la ayuda del programa PSeInt, obteniendo como resultado una representación del método compare() en pseudocódigo.




```
Algoritmo compare
    Leer fileName
    tempfolder←fileName
    Si tempfolder = 0 Entonces
        Escribir This folder is empty, it does not exist, or could not find repeated files, please try another one
        Escribir IllegalArgumentException
    SiNo
        Escribir tempfolder
    Fin Si
FinAlgoritmo
```

- ***Diseño de la interfaz.***


RFF
Buscador de archivos
Directorio:
Nombre de archivo:
Buscar

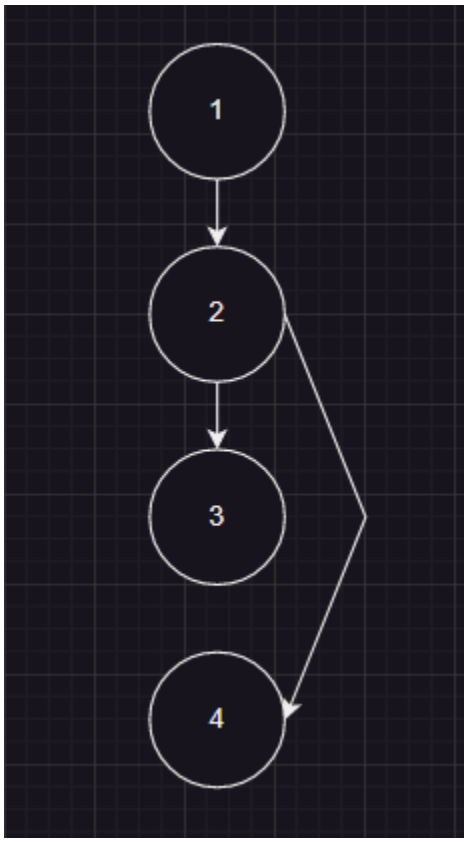

RFF
Buscar archivos repetidos
Primer directorio:
Segundo directorio:
Buscar

- ***Diseño de pruebas unitarias.***

Caja Blanca:

Se elaborarán las pruebas de caja blanca sobre el método “compare” de la aplicación RFF.

```
public String[] compare(String fileName){
    tempfolder = new File(fileName);
    if (tempfolder.length() == 0) {
        throw new IllegalArgumentException("This folder is empty, it doesn't exist, " +
                "or couldn't find repeated files, please try another one: ");
    }
    return tempfolder.list();
}
```



Caja Negra:

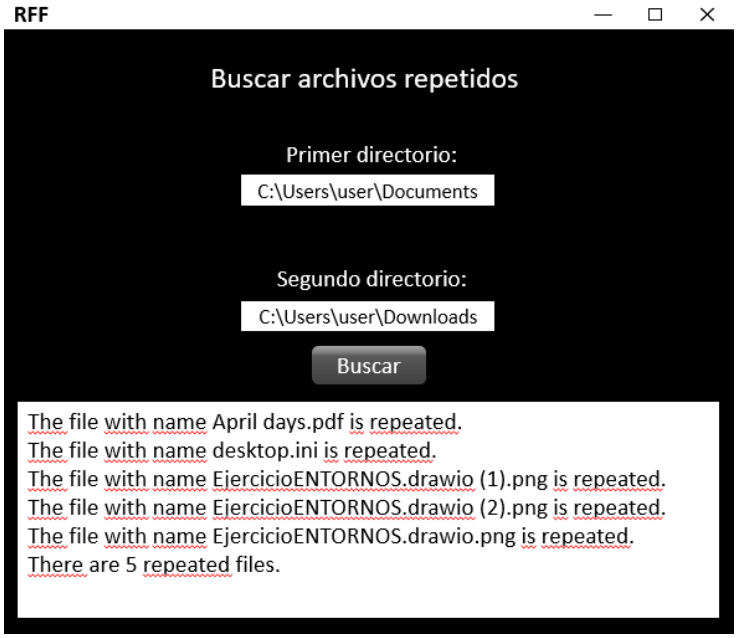
1. Casos Válidos ->
   fileName(nombre del archivo): hay que escribir el nombre de un archivo válido.

2. Casos no Válidos->
   Si no se escribe nada o el archivo está mal escrito, se devolverá un error.

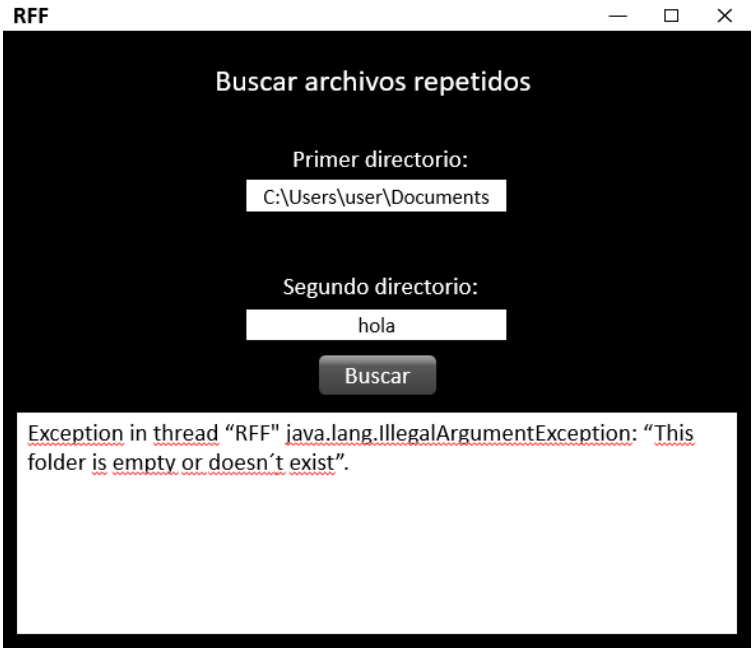
3. Casos de prueba:

- En el primer caso se muestra la introducción correcta de un directorio y un campo vacío para el segundo, obteniéndose el siguiente error.

- En el segundo caso se muestra un uso correcto del programa, en el que se facilitan dos rutas correctas y se muestran los archivos repetidos.



- En el tercer caso se observa el error que surge tras proporcionar un directorio correcto y uno incorrecto, ubicando una palabra que no indica ninguna ruta. Esto se debe a que la aplicación necesita dos rutas absolutas para realizar correctamente la búsqueda.



- ***Implementación de alguno de los requisitos (no tiene que incluir interfaz gráfica).***

Este es el requisito funcional de: Búsqueda de archivos.
Comprueba si dicho archivo existe o no en un directorio del sistema.
Si lo encuentra, se mostrará que encontró la ruta absoluta de dicho archivo.

```
public boolean searchFile(String directoryPath, String NameOfFile) {

    File directory = new File(directoryPath);

    if (directory.isDirectory()) {
        File[] files = directory.listFiles();

        if (files != null) {
            for (File file : files) {
                if (NameOfFile.equalsIgnoreCase(file.getName())) {
                    System.out.println("Found file: " + file.getAbsolutePath());
                    return true;
                }
            }
        }
    }
    System.out.println("File not found");
    return false;
}
```

- ***Documentación utilizando JAVADOC.***

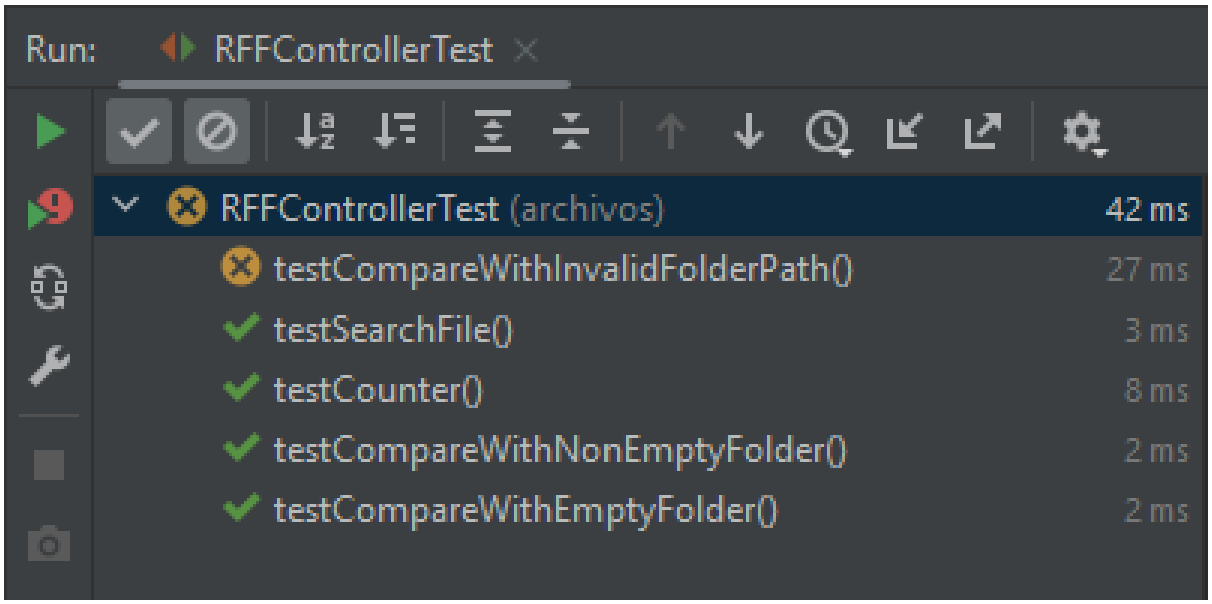
La documentación se encuentra ubicada dentro del propio programa.

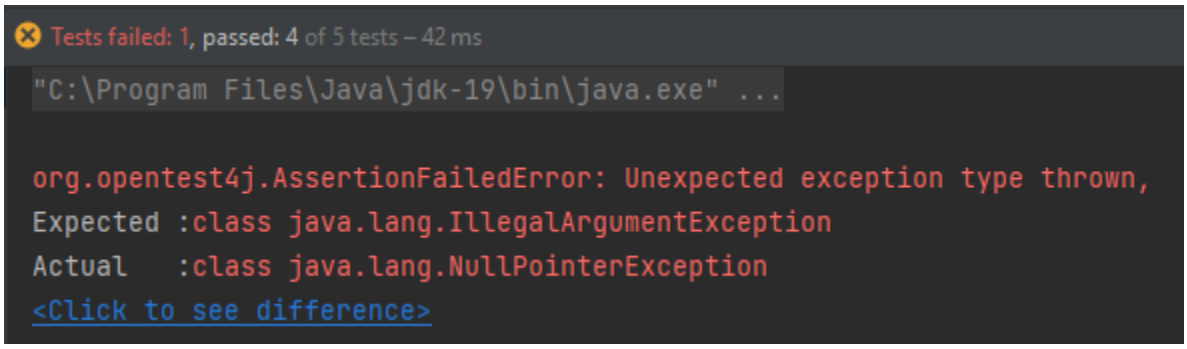
- ***Realización de las pruebas y sus resultados con JUNIT. Errores detectados y correcciones implementadas.***

La realización de las pruebas JUNIT se incluye en el archivo llamado “RFFControllerTest.java”, ubicado dentro de la carpeta del programa.

Asimismo, se proporciona una captura en la que se muestran las pruebas realizadas.

En este caso, la prueba que testea la posibilidad de introducir una ruta inválida en el método compare() da un fallo y no transcurre como cabría esperar. Desgraciadamente, no ha sido posible averiguar el motivo de este error. Sin embargo, el programa funciona correctamente si se introducen los datos adecuadamente, pues devuelve la información esperada.





Se creará un repositorio donde se incluya todo el proyecto con el código generado, las clases, main, pruebas JUNIT, documentación JAVADOC y un documento PDF con la inclusión de los distintos apartados del proyecto, pasos realizados, problemas detectados, soluciones.

Se valorará la implementación completa del proyecto, funcionalidad y la realización de todos los pasos del ciclo de vida de una aplicación software.