Tarea para PSP06.

Detalles de la tarea de esta unidad.

Enunciado.

La tarea de la unidad está dividida en dos actividades.

Actividad 6.1. Crea una aplicación que realice los siguientes pasos:

- Solicita el nombre del usuario que va a utilizar la aplicación. El login tiene una longitud de 8 caracteres y está compuesto únicamente por letras minúsculas.
- Solicita al usuario el nombre de un fichero que quiere mostrar. El nombre del fichero es como máximo de 8 caracteres y tiene una extensión de 3 caracteres.
- Visualiza en pantalla el contenido del fichero.

Es importante tener en cuenta que se tiene que realizar una validación de los datos de entrada y llevar un registro de la actividad del programa.

Actividad 6.2. Utilizando la aplicación desarrollada en la actividad anterior, configura las políticas de acceso para:

- Firmar digitalmente la aplicación.
- Que sólo pueda leer los datos del directorio `c:/datos`.

Criterios de puntuación. Total 10 puntos.

- **Actividad 6.1** (5 puntos).
- **Actividad 6.2** (5 puntos).

Recursos necesarios para realizar la Tarea.

Para realización de la actividad tan sólo es necesario tener instalado el entorno de desarrollo de Java.

Consejos y recomendaciones.

Leer detenidamente el contenido de la unidad.

Indicaciones de entrega.

Una vez realizada la tarea elaborarás un único documento donde figuren las respuestas correspondientes. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_SIGxx_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna **Begoña Sánchez Mañas para la sexta unidad del MP de PSP**, debería nombrar esta tarea como...

sanchez_manas_begona_PSP06_Tarea

Memoria

Ejercicio01

Mi aplicación es una clase principal que tiene un logger que se configura con el método configurarLogger(); y luego tiene 3 partes en el método main:

solicitarNombreUsuario(): Meto en un do-while el bucle de pedir usuario hasta que introduzca uno que cumpla el patrón diseñado: [a-z]{8} En cualquier caso, siempre creo un log de nivel INFO con lo que el usuario está metiendo.

solicitarNombreArchivo(): Meto en un do-while el bucle de pedir archivo hasta que introduzca uno que cumpla el patrón diseñado: [a-zA-Z{1,8}\\.[a-zA-Z]{3} En cualquier caso, siempre creo un log de nivel INFO con lo que el usuario está metiendo.

visualizarArchivo(): Si el usuario y el archivo pasan el control de pattern, trato de leer el fichero con un try y capturo las posibles excepciones que se puedan dar, una de tipo IOException y otra de tipo Exception, cada una con un mensaje personalizado y en todos los casos guardo un registro de la actividad en el log y su correspondiente nivel que esta vez serán WARNING para IOException y SEVERE para Exception. Finalmente, entro en un finally que informará de que la aplicación se termina.

configurarLogger(): indica en que archivo se guarda el log.

capturarMomento(): pequeño método para escribir en el log la fecha en formato aaaammdd hh:mm:ss

Ejercicio02

Voy a la ubicación de la clase en el explorador de archivos, escribo cmd en la barra de direcciones y se abre el símbolo de sistema con la ruta a la clase preparada:

Generar Archivo.JAR:

En vez de usar los comandos **javac NombreClase.java** para compilar el archivo .class y **jar cvf Nombre.jar NombreClase.class** para desplegar el archivo .jar, en mi caso, lo que hago es que hago botón derecho sobre el proyecto > Propiedades > Run > Main Class y elijo la clase que será main (se añadirá al manifest) y luego hago clean and build y se despliega el archivo .jar en la carpeta \dist\ del proyecto.

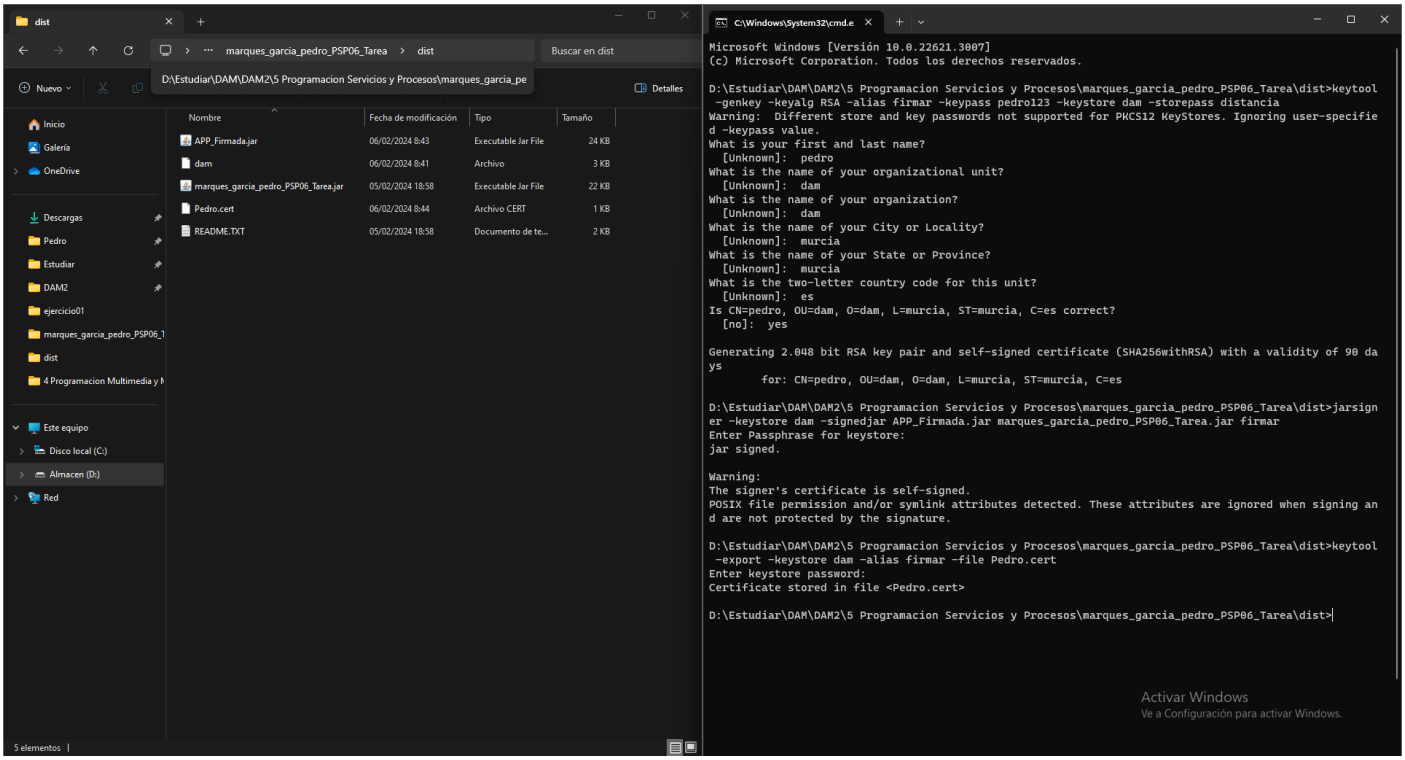
Firmarlo:

Voy a la carpeta \dist\ del proyecto, escribo CMD en la barra de direcciones y en la consola de comandos
Ejecuto: **keytool -genkey -keyalg RSA -alias firmar -keypass pedro123 -keystore dam -storepass distancia** para generar las claves públicas y privada.
Is CN=pedro, OU=dam, O=dam, L=murcia, ST=murcia, C=es correct? > yes
Ejecuto el comando **jarsigner -keystore dam -signedjar APP_Firmada.jar marques_garcia_pedro_PSP06_Tarea.jar firmar** para crear el archivo APP_Firmada.jar (firmado) a partir del jar que desplegué (habrá que introducir la contraseña de storepass: distancia).
Ejecuto: **keytool -export -keystore dam -alias firmar -file Pedro.cert** para exportar el certificado.

Con esto ya está firmado digitalmente el archivo APP_Firmada.jar y el certificado el Pedro.cert

Si hubiese que importar el certificado:
keytool -importcert -alias Pedro -file Pedro.cert -keystore dam y luego pulsar yes.

Dejo los archivos que he generado en la carpeta \dist\ del proyecto para poder ser accesibles.



Limitar los permisos:

Intento abrir el policytool, pero el comando ya no está disponible. He consultado por internet y resulta que:

> A partir de JDK 9, la herramienta `policytool` se eliminó como parte del conjunto estándar de herramientas de JDK. Puedes encontrar más información sobre esto en la documentación oficial de Oracle.

En java 17 el archivo de seguridad de java es: C:\Program Files\Java\jdk-17\lib\security\ default.policy
Con lo cual hay que abrir este archivo, e introducir las siguientes líneas:

```
grant codeBase "file:/D:/Estudiar/DAM/DAM2/5 Programacion Servicios y Procesos/marques_garcia_pedro_PSP06_Tarea/dist/APP_Firmada.jar"
    permission java.io.FilePermission "C:/datos/*", "read";
};
```

Con esto, cuando ejecuto APP_Firmada.jar solo tiene permisos de lectura sobre la carpeta seleccionada.

Después de algunas pruebas, no funciona correctamente. La solución ha sido:

1) Usar el siguiente código en el default.policy:

```
grant
    permission java.io.FilePermission "C:/datos/*", "read";
};
```

2) En NetBeans > Proyecto (boton derecho) > run > VM Options:

```
-Djava.security.manager
```