TAREA 6

EJERCICIO 1:

En este ejercicio, debemos crear una aplicación en un solo archivo llamado AuthApp.java, en esta aplicación, debemos de crear un login que nos pida un nombre de usuario con 8 letras minúsculas, y tras esto solicitamos al usuario el nombre del fichero que desea mostrar, el cual debe de tener 8 caracteres como máximo de nombre, seguido de un punto y 3 caracteres más para la extensión del archivo. Tras realizar estas dos peticiones, se visualizará en pantalla el contenido del fichero, el cual deberá estar localizado en la carpeta raíz del proyecto.

Para realizar esto, vamos a hacer uso de un Scanner para recibir los datos introducidos por el usuario, además también usaremos FileHandler para que se cree un archivo .log con los logs de la ejecución anterior del programa y por último usaremos Logger, para registrar así todos los logs de nuestra aplicación.

Crearemos 4 funciones, la primera será la función main, donde tendremos la lógica de nuestro programa. La segunda es validateUsername, la cuál nos servirá para validar que nuestro nombre de usuario tiene 8 letras minúsculas exactamente. La tercera es validateFileName, donde validaremos que el fichero tenga como máximo 3 caracteres, un punto y 3 una extensión de 3 caracteres. Y por último, tenemos la función contenidoFichero, aquí recibimos el nombre del archivo, y devuelve el contenido del archivo.

EJERCICIO 2:

Para realizar la firma digital de la aplicación, debemos generar un KeyStore, para ello haremos uso del siguiente comando:

keytool -genkeypair -alias keyAuthApp -keyalg RSA -keystore authappkeystore.jks -storepass admin123 -validity 365

Tras usar ese comando, nos realizará una serie de preguntas, a las cuales tendremos que responder para generar nuestra KeyStore.

Cuando ya tenemos generada nuestra KeyStore nos aparecerá un archivo llamado authappkeystore.jks, el cual contiene nuestras keys.

Ahora, vamos a exportar un certificado público, para ello haremos uso del siguiente comando:

keytool -export -alias keyAuthApp -keystore authappkeystore.jks -file authappcert.cer -storepass admin123

Por último, debemos firmar el archivo JAR, así que, primero para poder generar el archivo .jar, haremos lo siguiente:

- Ve a File → Project Structure → Artifacts.
- En la pestaña Artifacts, haz clic en + y selecciona:
    - JAR → From modules with dependencies.
    - Selecciona el módulo principal (AuthApp).
    - Marca la opción Extract to the target JAR.
- Guarda la configuración y cierra la ventana.
- Ve a Build → Build Artifacts → Build.

Y para poder firmar nuestro .jar, usaremos el siguiente comando para ello:

jarsigner -keystore authappkeystore.jks -storepass admin123 "C:\Users\fluna\OneDrive\ESTUDIOS\DAM\2 DAM\ASIGNATURAS\Programación de Servicios y Procesos\TAREAS\TAREA 6\luna_raya_francisco_Tarea_UT6PSP\out\artifacts\luna_raya_francisco_Tarea_UT6PSP_jar\luna_raya_francisco_Tarea_UT6PSP.jar" keyAuthApp

Para concluir esta actividad, queremos configurar las políticas de acceso para que la aplicación solo pueda leer los datos del directorio “c:/datos”. Para ello, vamos a crear un archivo llamado AuthApp.policy en la misma carpeta donde tenemos nuestro .jar y escribiremos lo siguiente en él:

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

Tras haber creado este archivo, para ejecutar la aplicación con la política que hemos configurado, usaremos el siguiente comando:

```
java -Djava.security.manager -Djava.security.policy=AuthApp.policy -jar
luna_raya_francisco_Tarea_UT6PSP.jar
```

Y de esa manera, ya estaríamos ejecutando nuestro .jar con las políticas de acceso.