Tema 1. Acceso a ficheros

Boletín 2

*Todos los ejercicios de este boletín deben estar dentro de un **package** llamado **tema01** (Siguiendo las reglas que hemos visto en clase, ej: com.germangascon.tema01).*

Para la realización de los siguientes ejercicios se recomienda crear una carpeta específica donde copiaremos algunos directorios y archivos de tu disco duro (los que tu quieras) a modo de ejemplo. Todas las rutas deben hacer referencia, única y exclusivamente, a archivos ubicados dentro de dicha carpeta. Un ejemplo de ruta absoluta para la carpeta de los ejemplos podría ser: */home/usuario/tema01/ficheros* Para evitar problemas, no utilices espacios en blanco en la ruta absoluta de la carpeta.

1. Haz un método que reciba como parámetro un fichero con 20 DNI aleatorios (sin letra) y:
   a) compruebe que todos los DNI tienen una longitud de 8 dígitos, sino es así, deberá rellenar con 0's por la izquierda.
   b) calcule la letra correspondiente a cada DNI y la añada por la derecha.
   c) Guarde el resultado en otro archivo cuyo nombre sea el resultado de concatenar el nombre del archivo original más "_conLetras" en la parte del nombre.

2. Haz un programa que permita añadir y eliminar alumnos de un archivo. Para ello, crea las sigiuentes clases:
   a) Alumno, con los siguientes atributos: nia, nombre, apellido1, apellido2 y fechaNacimiento. Como métodos tendremos un constructor que recibe todos los atributos y los correspondientes getters.
   b) GestionAlumnos que tenga los siguientes métodos:
      - Un constructor que recibirá como parámetro la ruta del archivo donde se almacenan los alumnos. En caso de no existir dicho archivo, se creará al invocar al constructor.
      - insertarAlumnos(Alumno[] alumnos), para insertar los todos elementos de un array en el archivo. Será un array con los alumnos de clase. Se debe insertar cada Alumno en una línea diferente.
      - eliminarAlumno(Alumno alumno), que recibe como parámetro el Alumno que se quiere eliminar del archivo, y lo elimine.
      - Sobreescribir el método toString() para que muestre los alumnos que actualmente están en el

archivo.

Crea un objeto GestionAlumnos en el método main y realiza varias pruebas añadiendo, modificando y mostrando los alumnos.

3. Crea un programa que permita validar el acceso de un usuario. Al iniciar mostrará el siguiente menú:

****

1. Validar acceso

****

La opción **Validar acceso**:

a) Pide la contraseña

b) Lee el archivo properties y comprueba que el acceso es válido. Ten en cuenta que la contraseña no se guarda directamente, en su lugar se guarda el hash en **SHA-1** para evitar que si alguien tiene acceso al archivo, averigue la contraseña (pista: busca la clase **MessageDigest** en Javadoc o ejemplos por Internet).

NOTA: Como la primera vez que accedas no existirá aún el archivo properties la contraseña por defecto será "S3cret@".

Al validar el acceso se mostrará el siguiente menú:

****

1. Modificar contraseña

2. Salir

****

La opción **Modificar contraseña**, en primer lugar solicitará al usuario la contraseña anterior y la validará, esto se hace por si el usuario se ausenta de su puesto de trabajo unos minutos mientras está en este menú cualquiera podría cambiarle la contraseña. Después solicitará la nueva contraseña, que como mínimo deberá tener 8 caracteres con al menos un letra en mayúsculas, una en minúsculas, un número y un caracter no alfanumérico, y guardará su hash en el archivo properties.

La opción **Salir** finalizará el programa.

4. Hacer el ejercicio anterior pero para múltiples usuarios utilizando un archivo JSON.

5. Diseña una clase para gestionar el estado de una partida del 3 en raya. No es necesario que diseñes todas las clases del juego, sólo la clase para "congelar" el estado de una partida. Es decir, la posición de las fichas, a qué jugador le tocaría jugar, y la puntuación acumulada (número de manos que ha ganado cada jugador).

Implementar también una clase llamada GameStorage que gestione el guardado en disco del estado de la

partida y su posterior recuperación. Realiza esta implementación utilizando la técnica de la serialización propia de Java con ObjectOutputStream y ObjectInputStream.

6. Crea un programa que calcule números primos. Cada número primo encontrado será añadido a un archivo. Al iniciar el programa comprobará si existe el archivo, en caso afirmativo, continuará por el último número primo calculado en la sesión anterior, en caso negativo, comenzará por el principio.

7. Ampliación

Haz que el programa anterior sea multihilo. Para ello cada hilo debe realizar la búsqueda de números en rangos diferentes. Por ejemplo, el hilo0 puede buscar en el rango [0 - 999_999], el hilo1 en el rango [1_000_000 - 1_999_999] y así sucesivamente, y posteriormente incrementar los rangos según la cantidad de hilos.

El método Runtime.*getRuntime*().availableProcessors(); permite obtener cuantos núcleos tiene la CPU.