Ejercicios.

1. Diseña un programa Java que cree un archivo en el que se guarde una cadena cualquiera carácter a carácter.
2. Diseña un programa Java que cree un archivo en el que se guarden las cadenas existentes en un arrayList de tipo String.
3. Diseña un programa Java que solicite palabras (sin espacios) al usuario (método next() de Scanner) y las guarde en un archivo. El proceso de petición finaliza cuando el usuario teclee "fin", que no debe escribirse en el archivo.
4. Diseña un programa Java que tenga una clase con un método estático que permita:
    a. Leer líneas de un archivo de texto cualquiera y guardarlas en un arrayList.
    b. Ordenar el arrayList
    c. Volver a escribir las palabras en el archivo.

5. Diseña un aplicación Java que permita añadir texto a un archivo existente. Puedes ayudarte de algún archivo creado en ejercicios anteriores. La idea es añadir al archivo las cadenas contenidas en un arrayList.

    Para poder añadir texto a un archivo existente, es necesario llamar al constructor del FileWriter con true como segundo parámetro y usar la clase PrintWriter como se indica: pw= new PrintWriter(new BufferedWriter(new FileWriter(archivo, true))); o bien new BufferedWriter(new FileWriter(archivo, true));

    Ojo, para escribir en un objeto de la clase PrintWriter usamos print() o println(). El método close() no genera excepciones.

6. A partir de este archivo, diseña una aplicación que busque cadenas de texto en el mismo. La salida debe ser como la de la imagen:

```
Buscando la palabra: textos
*********************************
Encontrada en línea 4: galería de textos y los mezcló de tal manera que logró hacer u
Encontrada en línea 14: más o menos normal de las letras, al contrario de usar textos
Encontrada en línea 15: Estos textos hacen parecerlo un español que se puede leer. Mu
BUILD SUCCESSFUL (total time: 0 seconds)
```

7. Diseña un programa que realice una gestión de los datos de una serie de alumnos almacenados en este archivo. El archivo contiene por cada alumno: un número que lo identifica, su nombre y diez notas (double). La gestión consistirá en programar una aplicación que incluya las opciones:
    a. Obtener las notas de un alumno identificado mediante su número.
    b. Obtener la media de las notas de un alumno identificado mediante su número.
    c. Obtener la mejor nota de un alumno identificado mediante su número.
    d. Mostrar número de alumno, nombre y media de sus notas (de todos los alumnos).

Una pista. La clase Arrays tiene un método llamado copyOfRange, que devuelve un array a partir de otro array, desde una posición concreta hasta otra.

8. Diseña una aplicación Java que trabaje con objetos de la clase Producto. De cada producto se guardará un id, nombre, precio y proveedor. La aplicación permitirá insertar productos, eliminar, consultar y listar productos. Para almacenar los productos utiliza un ArrayList y sobre esta estructura realiza las inserciones, eliminaciones, búsquedas y listados.
   Al finalizar la aplicación, guarda los objetos en un fichero binario y al empezar carga los objetos desde el fichero al ArrayList de forma que no se pierdan los objetos creados en ejecuciones anteriores. **Puedes optar por guardar los objetos en el archivo de forma individual o bien guardar el ArrayList entero que es más práctico.**

   Controla que no puedan existir dos productos con la misma id.
   Clases:
   - Producto.
   - GestionProductos. Contiene el ArrayList y los métodos para añadir, eliminar, consultar (buscar por id) y devolver el ArrayList. Esta clase también debe incluir los métodos para escribir/leer el arrayList en/del archivo.
   - Menu.
   - Aplicación. Mostrar el menú y el bucle de tratamiento del menú.

   Consejos para este ejercicio:
- primero resuelve sin archivos
- cuando compruebes que funciona, añade la opción de leer datos al empezar y guardar datos al finalizar.

9. Diseña un programa Java que permita gestionar las opciones del menú siguiente, a partir del archivo de texto datosProductos.txt. Como puedes comprobar cada línea del fichero tiene datos de un producto: código, nombre, año y unidades vendidas ese año. Existen datos de varios años.
   Opciones que debes programar:
   - Consultar producto x id.
   - Productos de los que se han vendido más (suma) de XXX unidades a lo largo de los años.
   - Ventas de un año (de todos los productos). Devuelve un arrayList con las unidades vendidas de cada producto en ese año.
   - Generar fichero de texto con las ventas de un año (de todos los productos). Igual que el anterior pero generando un archivo de texto.

10. Vamos a listar los archivos y directorios contenidos en una carpeta. Para ello creamos una carpeta llamada “archivos” en el directorio del usuario con el que estamos conectados. Crea varios archivos en esa carpeta. Las clases que usaremos son Paths y Files, además de la interfaz Path.

```
Contenido de la carpeta: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos

Archivo: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos/VentasAnio_2015.txt
Archivo: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos/datosProductos.txt
Directorio: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos/directorio
Archivo: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos/directorio/datosProductos2.txt
Directorio: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos/directorio/otra
Archivo: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio10/archivos/directorio/otra/datosProductos3.txt
```

11. En esta ocasión mostraremos sólo los archivos y su tamaño. Pero con valor 1 en profundidad.

```
Contenido de la carpeta: /home/jose/eclipse-workspace/GestionArchivos/Ejercicio11/archivos

/home/jose/eclipse-workspace/GestionArchivos/Ejercicio11/archivos/VentasAnio_2015.txt 13449 bytes.
/home/jose/eclipse-workspace/GestionArchivos/Ejercicio11/archivos/datosProductos.txt 16085 bytes.
```

12. Diseña una aplicación Java que sirva para copiar archivos entre dos rutas origen y destino. Crea una clase con un método estático que reciba esas dos rutas (dos Path) y copie los archivos de una a otra. Puedes acceder a los archivos individualmente tal y como vimos en el ejercicio anterior. Si el archivo existe en el destino, reemplazarlo, utiliza:
    Files.copy(archivoOrigen, archivoDestino, StandardCopyOption.REPLACE_EXISTING);

    Utilizar 1 como nivel de profundidad.

    En caso de que alguna de las dos rutas no sea un directorio, genera una “excepción propia” que avise de la situación.

13. En este caso trataremos de fusionar varios archivos de texto en uno solo. Aquí os dejo dos archivos donde se encuentran repartidos los nombre de los países del mundo. La idea es generar un archivo que los contenga todos.
    Intenta incluir:
    - fusionar varios archivos, todos con la misma extensión
    - filtro por extensión de archivo.
    - Java stream
    - parámetro: ruta origen de la información, extensión de los archivos y nombre del archivo destino.

14. Ahora ampliaremos la funcionalidad del ejercicio 6, que buscaba una cadena en las líneas de un archivo, haciendo que busque en todos los archivos de un directorio. Aquí os dejo el directorio con los archivos donde se debe buscar. La salida debe ser algo parecido a esta.

```
Buscando SARA en alumno01.txt
**********************************
Encontrado en línea 30: 2647631;SOBRADO PANIAGUA, SARA
Encontrado en línea 70: 46938891;CORDON HORNILLOS, SARA
Encontrado en línea 75: 47035769;MUÑOZ MUÑOZ, SARA
Encontrado en línea 79: 47044073;LOPEZ ALAEZ, SARAY
Encontrado en línea 140: 50218486;GONZALEZ MARTIN, SARA
Encontrado en línea 167: 50755331;HERNANDEZ PEREZ, SARA
Encontrado en línea 232: 53422257;HERRAEZ PEREZ, SARA
Encontrado en línea 282: 6573699;GARCIA LOBIT, SARA
Encontrado en línea 303: 76123032;RIO ALVAREZ, SARA DEL

Buscando SARA en alumno03.txt
**********************************
Encontrado en línea 1: 2647631;SOBRADO PANIAGUA, SARA
Encontrado en línea 41: 46938891;CORDON HORNILLOS, SARA
Encontrado en línea 46: 47035769;MUÑOZ MUÑOZ, SARA

Buscando SARA en alumno02.txt
**********************************
Encontrado en línea 17: 50218486;GONZALEZ MARTIN, SARA
Encontrado en línea 44: 50755331;HERNANDEZ PEREZ, SARA
```

15. En este ejercicio debes leer una línea del archivo “codificado.txt” y mostrarla en pantalla correctamente. La información que contiene este archivo está encriptada por lo que al abrirlo su contenido es ilegible.
    Para desencriptar esta información debes tener en cuenta **cómo se creó dicho archivo encriptado:**
    - se sumaron los números enteros contenidos en el archivo “archivoNumeros.txt”.
    - el resultado de esta suma se dividió entre 23 y se tomó el resto.
    - el valor resultante (un entero) se sumó al valor Ascii de cada carácter de la cadena original, y así se guardó en el archivo “codificado.txt”.

    Los archivos de este ejercicio puedes encontrarlos aquí.

16. El archivo “mymoviedb.tsb” contiene datos de una serie de películas: fecha de lanzamiento, título, resumen, popularidad (en miles o millones con separador “.”), votos, puntuación media, idioma original, géneros a los que está asociada (separados por comas “,”) y url del cartel. Todos estos datos utilizan como delimitador un salto de tabulación (“\\t”).

    La idea es que una aplicación Java lea este archivo y cargue los datos en memoria para poder:
    - generar un archivo binario con las películas.
    - generar un archivo binario con las pelis de un año concreto.
    - generar un archivo de texto con las películas de un género o varios.
    - obtener las “n” películas con la puntuación media más alta.