***Manipulación Básica de archivos***

***Clase: File***

***Metodos:***

* ***File(nombreCompleto):*** Es uno de los constructores de la clase. Permite crear una instancia a partir del nombre completo del archivo o directorio que se quiere representar.
* ***File(director, nombreCorto):*** Este constructor permite crear una instancia a partir del objeto de la clase File que representa el directorio en el que debe estar el archivo y su nombre corto.
* ***length():*** Retorna el número de bytes que un archivo contiene, si el archivo no existe, retorna 0.
* ***getAbsolutePath():*** Retorna el nombre completo del archivo o directorio incluyendo su ruta.
* ***getName():*** Retorna el nombre corto del archivo o directorio.
* ***lastModified():*** Retorna un valor tipo log, que representa el momento en el cual el archivo o directorio fue modificado por última vez , si el archivo no existe retorna 0.
* ***createNewFile():*** Crea un archivo vacío con el nombre especificado por el objeto File, solo si un archivo con ese nombre no existe previamente. Este método retorna verdadero si el archivo fue creado y false en caso contrario.
* ***listFiles():*** Retorna un arreglo de objetos de la clase File que contiene todos los elementos (directorios y archivos) presentes en el directorio. Si el objeto no es un directorio, este método retorna null.
* ***isDirectory():*** indica si el objeto representa un directorio.
* ***isfile():*** Indica si el objeto representa un archivo.
* ***(CONSTANTE PUBLICA)File.separator:*** Representa el carácter que en el nombre de un archivo separa los directorios de la ruta y el nombre corto del archivo. (“/”, “\”)

***Lectura de archivos***

***File:*** Objeto de la representación abstracta del archivo

***FileReader:*** Permite manejar un archivo como una fuente de caracteres

***BufferedReader:*** Permite manejar un archivo como una secuencia de líneas (String)

***Metodos(FileReader):***

* ***FileReader(File):*** Este constructor nos permite crear una instancia dela clase, haciendo referencia a archivo representado por el objeto de la clase file que recibe como parámetro.
* ***Close():*** Este método nos permite cerrar el flujo de entrad de caracteres desde el archivo, anulando la relación que existía con el archivo físico.

***Metodos(BufferedReader):***

* ***BufferedReader(FileReader):*** Este constructor nos permite crear una instancia de la clase, asociándola con la fuente de entrada de caracteres que recibe como parámetro.
* ***Close():*** Permite cerrar la relación con la fuente de caracteres.
* ***readLine():*** Lee la siguiente línea del archivo y la retorna en una cadena de caracteres (String). Si ya ha llegado al final del archivo, este método retorna null. La cadena que retorna no trae al final los caracteres de cambio de línea que se almacenan en el archivo.

***Manipulación de cadenas de caracteres***

***Metodos de la clase String:***

* ***trim():*** Elimina los blancos del comienzo y del final de la cadena de texto y retorna el resultado.
* ***toLowerCase():*** Convierte una cadena a minúsculas y retorna el resultado.
* ***toUpperCase():*** convierte una cadena a mayúsculas y retorna el resultado.
* ***indexOf(Caracter):*** Retorna la posición en la cadena de la primera aparición (de izq. A der.) del carácter que llega como parámetro. Si no aparece, el método retorna -1.
* ***substring(inicio, fin):*** Retorna una cadena que contiene los caracteres que se encuentran en las posiciones inicio y fin-1, si cualquiera de los dos limites es invalido este método lanza la excepción ***IndexOutOfBoundsException***. Si solo hay un parámetro, se toma la cadena que va desde dicho punto hasta el final de la cadena.
* ***startsWith(prefijo):*** Indica si la cadena comienza con el prefijo que llega como parámetro .
* ***endsWith(sufijo):*** Indica si la cadena termina con el sufijo que llega como parámetro .
* ***split(separador):*** Parte la cadena en un arreglo de palabras, las cuales se encuentran separadas en la cadena por un carácter que llega como parámetro. Este método es mucho ms general y poderoso de lo aquí explicado, pero para el uso que le vamos a dar es suficiente con esta explicación.
* ***replace(car1, car2):*** Retorna una cadena en la que se ha reemplazado todas las ocurrencias del car1 por el car2.

***Escritura de archivos***

Para escribir una cadena en un archivo se usa la clase ***PrintWriter***. Un objeto de esta clase representa un flujo de salida de caracteres o cadenas el cual está conectado físicamente a un archivo en la memoria secundaria del computador.

***Metodos de PrintWriter:***

* ***PrintWrite(File):*** crea un nuevo flujo de salida con el archivo descrito por el objeto File que llega como parámetro. Si dicho archivo ya existe borra su contenido y se prepara para escribir en el. Si el archivo no existe, lo crea . En el caso de que no exista o no pueda crearlo o encontrarlo el método lanza la excepción ***FileNotFoundException***.
* ***print(cadena):*** Escribe en el flujo de salida la cadena de caracteres que recibe como parámetro.
* ***println(cadena):*** Escribe en el flujo de salida la cadena de caracteres que recibe como parámetro y le escribe al final los caracteres necesario para marcar el final de una línea((“\n”) o (“\r”)).
* ***close():*** Cierra la conexión del flujo de salida con el archivo. Mientras esta conexión no se cierre, no es posible abrir un flujo de lectura desde el archivo. Al terminar la ejecución de cualquier programa en java, se invoca este método para todos los flujos de escritura que continúen abiertos.
* ***flush():*** Este método obliga al flujo a llevar al archivo todo lo que está en memoria.

***Persistencia por serializción.***

***(pag 180 del libro)***

Proceso de recuperación a partir de un archivo. Participan las clases ***ObjectImputStream*** y ***FileImputStream.*** Los objetos de la clase ***ObjectImputStream*** ven el archivo como una fuente secuencial de entrada de objetos, cada uno de los cuale se lee con el método ***readObject().***

También contaremos con las clases ***ObjectOutputStream*** y ***FileOutputStream***. Con el método ***writeObject()*** de la clase ***ObjectOutputStream*** enviamos objetos completos por el flujo de salida

***ObjectOutputStream:*** Crea un flujo de datos desde dentro del programa hacia afuera (medio de almacenamiento, ordenador remoto)

* ***writeObject():*** Crea el objeto para lanzarlo como una sucesión de bytes

***ObjectInputStream:*** Crea un flujo de datos desde fuera (medio de almacenamiento, ordenador remoto) hasta dentro del programa

* ***readObject():*** Lee la sucesión de datos que entra por el inputStream