EJERCICIOS T. 6. Entrada/Salida en Java

1. La orden `dir` sirve para visualizar el contenido de una ruta del sistema. Si se ejecuta sin parámetros, visualiza el contenido del directorio actual (en el que se encuentra), y si se pasa un parámetro, éste habrá de corresponder con la ruta de un directorio válido del sistema, cuyo contenido se quiere mostrar.

   Desarrolle su propia versión de la orden `dir`, como programa en Java que puede no recibir argumentos, en cuyo caso mostrará el contenido de su posición actual, o recibir un directorio existente que será ahora el que se visualice. De los ficheros se mostrará su nombre, un guión, y su tamaño en bytes. De los directorios se ha de visualizar su nombre, un guión, y la palabra `DIR` que denota que se trata de un directorio y no de un fichero.

   Realice las comprobaciones de error que estime oportunas (invocación del programa, etc).

2. Amplíe el ejercicio anterior para que pueda ser invocado con el modificador "`-r`", que querrá decir se ha de ejecutar de manera recursiva, esto es, cuando en el listado a mostrar aparezca un directorio, también habrá de mostrarse su contenido. Así, el programa se puede invocar de las siguientes maneras:

`# java miDir`: visualizará el contenido del directorio actual
`# java miDir ruta`: visualizará el contenido del directorio indicado en "ruta"
`# java miDir -r`: visualizará el contenido del directorio actual de manera recursiva.
`# java miDir -r ruta`: visualizará el contenido del directorio indicado en "ruta" de manera recursiva.

3. La orden `mv` sirve en algunos sistemas operativos para renombrar un archivo. Así, se le pasan como parámetros dos elementos: en primer lugar el nombre actual del fichero, y en segundo lugar el nuevo nombre que se desea que tenga. Desarrolle una aplicación en Java llamada `miMv`, que funcione como se acaba de describir la orden `mv`.

   Tenga en cuenta que, si no existe el fichero que se pasa como primer argumento, habrá de informarse al usuario. Del mismo modo, si ya existe un fichero con el nombre que se indica en el segundo argumento, se habrá de pedir consentimiento previo al usuario, de modo que si se proporciona, se sobrescribirá el contenido del fichero que ya existía con ese nombre.

4. La orden `rm` permite borrar ficheros y/o directorios en algunos sistemas operativos. A dicha orden se le pasa el nombre de un fichero, y si existe, lo borra, previa petición de confirmación, y si no existe, se lo comunica al usuario. También se puede usar con el modificador "`-r`", indicando que lo que quiere borrar no es un fichero, sino un directorio de manera recursiva (esto es, el propio directorio y todo su contenido, previa petición de confirmación). Implemente su propia versión de la orden `rm`, llamada `miRm`, que se comporte como se acaba de describir. A continuación puede ver algunos ejemplos de uso del programa, con lo que tendría que hacer en cada caso:

`# java miRm fichero`: si el fichero existe lo borra previa petición de confirmación, y si no, informa al usuario de que no existe.

`java miRm algo fichero`: se pasan dos argumentos y el primero no es "`-r`", por lo que la invocación no es correcta.
`# java miRm directorio`: se pasa un argumento y no es un fichero, sino un directorio, y no se ha especificado el modificador "`-r`", por lo que la invocación es incorrecta.
`# java miRm -r fichero`: se especifica el modificador "`-r`" y sin embargo lo que se pasa no es un directorio, sino un fichero, por lo que la invocación no es correcta.
`# java miRm -r directorio`: hace un borrado recursivo del directorio indicado (borra el directorio tras borrar todo su contenido, siempre previa petición de confirmación).

> NOTA: ponga especial cuidado en las pruebas que realiza mientras desarrolla el software, pues está desarrollando una aplicación que se encarga de borrar ficheros y directorios, y cualquier fallo en el código podría provocar que perdiese información accidentalmente.

5. Cree un programa con un menú principal en el que se dé al usuario la opción de escribir un carácter en un fichero, leer un carácter de un fichero, o salir. Haga un tratamiento de errores adecuado para informar al usuario acerca de posibles problemas ocurridos.

6. Modifique el programa anterior para que en esta ocasión lea y escriba cadenas de caracteres a petición del usuario.

7. Cree un programa que muestre un menú principal con las opciones "Nuevo cumpleaños", "Consultar cumpleaños" y "Salir".

   La primera opción pedirá un nombre de una persona y una fecha, y las almacenará en un fichero con el siguiente formato: en una línea, el nombre, y en la siguiente línea, la fecha, con el formato dia/mes/anho.

   La segunda opción mostrará un listado con los nombres y fechas de cumpleaños almacenados en el fichero.

   El menú se mostrará hasta que el usuario seleccione la opción de Salir.

   Implemente el programa haciendo uso de un método llamado `nuevoCumpleaños`, que reciba el nombre del fichero que sirve como agenda, y el cumpleaños a añadir. Dicho cumpleaños estará albergado en una estructura con los campos necesarios (nombre, día del cumpleaños, mes, y año de nacimiento).

   Implemente también un método `muestraCumpleños`, que reciba como argumento el nombre del fichero que sirve como agenda, y muestre por pantalla todos los cumpleaños albergados en dicha agenda. Cada cumpleaños se mostrará de la siguiente manera:

   ```
   El cumpleaños de Pepe es el día 14 del 5 (nació en el año 1982).
   El cumpleaños de Marta es el día 8 del 11 (nació en el año 1984).
   ...
   ```

   Realice una segunda versión del programa, en que lo único que cambie sea el formato de almacenamiento de cada cumpleaños. En este caso se guardará todo en la misma línea, de modo que figure el nombre del contacto, un espacio separador, y la fecha del cumpleaños,

con el mismo formato que el especificado anteriormente. Nótese que en esta versión el nombre del contacto no podrá ser compuesto, pues el espacio ubicado entre la primera y la segunda parte del nombre podría ser problemático, al ser el carácter considerado como separador entre nombre y fecha de cumpleaños.

8. Escriba un método que reciba como argumentos un carácter y el nombre de un fichero. El método ha de leer el fichero de texto cuyo nombre ha recibido, y mostrar por pantalla aquellas líneas del mismo que contengan el carácter indicado en el otro argumento.

9. Escriba un programa que cuente las palabras contenidas en un fichero.

10. Implemente un programa `buscaPalabras`, que reciba como argumento una palabra, y busque el número de ocurrencias de la misma en un fichero, también pasado como argumento.

11. Construye la orden `micopy`, que permita copiar un archivo (análogo a copy de MS-DOS). La sintaxis de llamada a su programa, desde la línea de órdenes, será como sigue:

```
# java  micopy <fichero_origen> <fichero_destino>
```

El programa comprobará que el usuario ha hecho un uso correcto del mismo, llamándolo con el número adecuado de argumentos. Una vez comprobada la corrección de la invocación, se copiará el fichero con los datos indicados por el usuario. En caso de producirse algún error, habrá de mostrarse por pantalla un mensaje aclaratorio al usuario.

12. Modifique el programa del juego de adivinanza de una palabra realizado en temas anteriores, para que lea las palabras a adivinar de un fichero de texto, cuyo nombre se le pasa como argumento.

13. Programe un método llamado `leeLínea`, que haga algo similar al método `readLine` de `BufferedReader`. Dicho método recibirá un `BufferedReader` abierto con un fichero asociado, y devolverá un `String` con la siguiente línea a leer de un fichero. Cuando llegue al final de fichero devolverá `null` en vez de una cadena o `String`.

14. Implemente un programa en Java que reciba dos argumentos. El primero de ellos será el nombre de un fichero de texto. El programa creará otro fichero de texto, cuyo nombre se habrá pasado como segundo argumento, cuyo contenido pasará a ser el mismo que el del primero, pero con las líneas numeradas; cada número de línea tendrá el formato "#numero_de_linea#".

15. Desarrolle una clase en Java que permita cifrar de forma sencilla un fichero de texto. El programa obtendrá de los parámetros de entrada el nombre del fichero a cifrar y el nombre del fichero de destino (que quedará cifrado). El programa pedirá por teclado una clave, correspondiente a un valor numérico entre 0 y 127, y comenzará el proceso de

cifrado con dicha clave. Para cifrar, simplemente recorrerá carácter a carácter el fichero de origen, y calculará la operación OR-exclusiva (XOR) con el valor numérico proporcionado por el usuario. El carácter resultante será el que se vuelque al fichero cifrado.

Compruebe que la función desarrollada permite cifrar y descifrar ficheros correctamente, viendo que genera texto sin sentido al proporcionarle una clave, y que es capaz de retornar al fichero original en texto plano siempre y cuando proporcione la misma clave.

16. Amplíe la agenda desarrollada en ejercicios anteriores. Se tratará de la agenda que posee un profesor con datos de sus alumnos: nombre o nick del alumno (palabra que lo identifica), número de teléfono para localizarlo, curso que estudia, y letra identificativa del grupo. En este caso el fichero no tiene por qué ser legible, por lo que podrá usar la lectura y escritura directa de objetos. Las opciones que habrá de integrar el programa son las siguientes:

    a. Inclusión de una nueva entrada en la agenda (un nuevo alumno)
    b. Búsqueda de un número de teléfono a partir del login de un usuario
    c. Consulta del número de entradas de la agenda
    d. Lectura de una entrada concreta a partir de la posición que ocupa
    e. Consultar todas las entradas de la agenda
    f. Actualizar datos de un alumno
    g. Borrar un alumno de la agenda
    s. Salir

    NOTAS
    - El nombre del fichero agenda se le pasará al programa al ser ejecutado.
    - Se hará control de errores siempre que sea posible / necesario.

17. Realice un programa en Java que gestione la escritura de los datos personales (nombre, primer apellido, segundo apellido, edad, curso, letra ó grupo y nota media) en un fichero de texto. Haga dos versiones del mismo:

    a) Fichero delimitado: habrá un carácter que sirva para separar los distintos campos que forman parte del fichero.

    b) Encolumnado: no habrá un carácter separador, sino que cada campo ocupará un tamaño fijo, completándose con espacios en blanco a la derecha.

18. Escriba un programa que solicite al usuario el nombre de un fichero y una posición dentro del mismo. El programa habrá de mostrar en pantalla el texto contenido en el fichero, desde la posición indicada hasta el primer salto de línea que encuentre.

19. Escriba un programa que lea líneas de un fichero de texto y las imprima línea a línea de forma alternativa, esto es, la primera, la última, la segunda, la penúltima, la tercera, la antepenúltima, etc.

20. Implemente un programa en Java que redirija la salida estándar de error a un fichero. Se solicitará al usuario el nombre del fichero que albergará los mensajes de error. Fuerce

algún tipo de fallo para comprobar que, efectivamente, los mensajes van a parar al fichero indicado y no a pantalla, que por defecto es la salida estándar de error.

21. Escriba diferentes programas de prueba que le permitan practicar con los diversos aspectos relacionados con el manejo de ficheros en Java.