Algoritmos y Estructura de Datos

PROCESAMIENTO DE TEXTOS

DOCENTE: MARX DANLY LEÓN TRUJILLO
FACULTAD DE ING. INDUSTRIAL Y SISTEMAS
E.P. INGENIERÍA DE SISTEMAS

PROCESAMIENTO DE TEXTOS

Un fichero de texto está formado por secuencias de caracteres, organizados en líneas de igual o distinta longitud. Todos los datos que aparecen en estos ficheros están formados por caracteres.

CREAR Y ESCRIBIR EN FICHEROS DE TEXTO EN JAVA

Para escribir en un fichero de texto utilizaremos dos clases: ***FileWriter y PrintWriter***.

La clase ***FileWriter*** permite tener acceso al fichero en modo escritura. Para crear objetos ***FileWriter*** podemos utilizar los constructores:

```
FileWriter(String path)
FileWriter(File objetoFile);
```

El fichero se crea y si ya existe su contenido se pierde.

PROCESAMIENTO DE TEXTOS

Si lo que necesitamos es abrir un fichero de texto existente sin perder su contenido y añadir más contenido al final utilizaremos los constructores:

```
FileWriter(String path, boolean append)
FileWriter(File objetoFile, boolean append)
```

Si el parámetro ***append*** es **true** significa que los datos se van a añadir a los existentes. Si es false los datos existentes se pierden. La clase **FileWriter** proporciona el método ***write()*** para escribir cadenas de caracteres aunque lo normal es utilizar esta clase junto con la clase **PrintWriter** para facilitar la escritura. La clase **PrintWriter** permite escribir caracteres en el fichero de la misma forma que en la pantalla. Un objeto **PrintWriter** se crea a partir de un objeto **FileWriter**.

PROCESAMIENTO DE TEXTOS

Ejemplo:

```
FileWriter fw = new FileWriter("c:/ficheros/datos.txt");
PrintWriter salida = new PrintWriter(fw);
```

A partir de Java 5 se puede crear un objeto PrintWriter directamente a partir de un objeto File o de la ruta:

```
PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt");
```

PROCESAMIENTO DE TEXTOS

Ejemplo:

```
FileWriter fw = new FileWriter("c:/ficheros/datos.txt");
PrintWriter salida = new PrintWriter(fw);
```

A partir de Java 5 se puede crear un objeto PrintWriter directamente a partir de un objeto File o de la ruta:

```
PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt");
```

En este caso, si el fichero no existe se crea. Si no se puede crear un archivo con ese nombre o si ocurre algún error se lanza una excepción FileNotFoundException. Una vez creado el objeto podemos utilizar ***print(), println() y printf()*** para escribir en el fichero como si fuese en pantalla.

PROCESAMIENTO DE TEXTOS

Ejemplo 1 de escritura de un fichero de texto:

Programa Java que lee texto por teclado y lo escribe en un fichero de texto llamado datos.txt. El proceso consiste en leer una línea de texto por teclado y escribirla en el fichero. Este proceso se repite hasta que se introduce por teclado la cadena FIN. La cadena FIN que indica el final de lectura no se debe escribir en el fichero.

PROCESAMIENTO DE TEXTOS

Ejemplo 1 de escritura de un fichero de texto:

```
public class ProcTexto {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(System.in);
        String cadena;
        try (PrintWriter salida = new PrintWriter("D:\\CARPETA DOCENTE\\UNHEVAL\\2023\\6. AI
            System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
            cadena = sc.nextLine();                                    //se introduce por teclado u
            while (!cadena.equalsIgnoreCase("FIN")) {
                salida.println(cadena);                                //se escribe la cadena en el
                cadena = sc.nextLine();                                //se introduce por teclado u
            }
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Ejemplo 2 de escritura de un fichero de texto:

Programa que leer por teclado líneas de texto y las añade al final del ficheros datos.txt. Para resolverlo vamos a modificar el programa anterior para que añada texto al fichero datos.txt, es decir, al volver a ejecutar el programa el contenido anterior del fichero no se pierde y el contenido nuevo se añade al final.

PROCESAMIENTO DE TEXTOS

Ejemplo 2 de escritura de un fichero de texto:

```
public class ProcTexto_1 {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(System.in);
         String cadena;

         try (FileWriter fw = new FileWriter("D:\\CARPETA DOCENTE\\UNHEVAL\\2023\\6. ALGORI
              PrintWriter salida = new PrintWriter(fw)) {
              System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
              cadena = sc.nextLine();
              while (!cadena.equalsIgnoreCase("FIN")) {
                     salida.println(cadena);
                     cadena = sc.nextLine();
              }
         } catch (IOException ex) {
              System.out.println(ex.getMessage());
         }
    }
}
```

PROCESAMIENTO DE TEXTOS

LECTURA DE FICHEROS DE TEXTO EN JAVA

Para leer en un fichero de texto utilizaremos dos clases: ***FileReader y BufferedReader.***

La clase ***FileReader*** permite tener acceso al fichero en modo lectura. Para crear objetos FileReader podemos utilizar los constructores:

```
FileReader(String ruta)
FileReader(File objetoFile);
```

PROCESAMIENTO DE TEXTOS

Ambos lanzan una excepción **FileNotFoundException** si el fichero no existe. La clase **FileReader** proporciona el método ***read()*** para leer caracteres del fichero aunque lo normal es realizar la lectura mediante la clase **BufferedReader**.

Para leer utilizando la clase **BufferedReader** se debe crear un objeto **BufferedReader** a partir de un objeto **FileReader**:

```
FileReader fr = new FileReader("c:/ficheros/datos.txt");
BufferedReader entrada = new BufferedReader (fr);
```

PROCESAMIENTO DE TEXTOS

Una vez creado el objeto **BufferedReader** podemos utilizar:

- El método ***readLine()*** para leer líneas de texto del fichero (String). Este método devuelve null cuando no hay más líneas para leer.

- El método ***read()*** para leer carácter a carácter. Devuelve un entero que representa el código Unicode del carácter leído. Devuelve -1 si no hay más caracteres.

Ambos métodos lanzan una excepción **IOException** si ocurre un error de lectura.

Ejemplo 1 de lectura de un fichero de texto:

Programa Java que lee el contenido del fichero datos.txt creado en el ejemplo anterior y lo muestra por pantalla. El proceso consiste en leer una línea del fichero y mostrarla por pantalla.

El proceso se repite hasta que se llegue al final del fichero y no hayan más líneas que leer. Cuando esto ocurre el método ***readLine()*** devuelve null.

PROCESAMIENTO DE TEXTOS

```java
public class LeerTexto {
    public static void main(String[] args) throws FileNotFoundException {
        String nombreArchivo = "D:\\CARPETA DOCENTE\\UNHEVAL\\2023\\6. ALGORITMOS Y ESTRUCTURA DE DA
        try {
            // Abre el archivo para lectura usando FileReader y BufferedReader
            FileReader archivoReader = new FileReader(nombreArchivo);
            BufferedReader bufferedReader = new BufferedReader(archivoReader);

            String linea;
            // Lee el archivo línea por línea hasta llegar al final
            while ((linea = bufferedReader.readLine()) != null) {
                System.out.println(linea); // Imprime la línea leída
            }

            // Cierra el BufferedReader y el FileReader
            bufferedReader.close();
            archivoReader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Ejemplo 2 de lectura de un fichero de texto:

Mostrar por pantalla el contenido del fichero de texto datos.txt pero en este caso lo vamos a leer carácter a carácter. El proceso consiste en leer un carácter del fichero y mostrarlo por pantalla. Este proceso se repite hasta que no queden más caracteres que leer en el fichero, es decir, hasta que se alcance el final del fichero. En este caso el método ***read()*** devuelve -1.

PROCESAMIENTO DE TEXTOS

```java
public class LeerTexto1 {
    public static void main(String[] args) throws FileNotFoundException {
        String nombreArchivo = "D:\\CARPETA DOCENTE\\UNHEVAL\\2023\\6. ALGORITMOS Y ESTRUCTURA DE DAT

        try {
            // Abre el archivo para lectura usando FileReader y BufferedReader
            FileReader archivoReader = new FileReader(nombreArchivo);
            BufferedReader bufferedReader = new BufferedReader(archivoReader);

            int caracter;
            // Lee el archivo caracter por caracter
            while ((caracter = bufferedReader.read()) != -1) {
                char caracterComoChar = (char) caracter;
                System.out.print(caracterComoChar); // Imprime el caracter leído
            }

            // Cierra el BufferedReader y el FileReader
            bufferedReader.close();
            archivoReader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

PROCESAMIENTO DE TEXTOS

LECTURA DE FICHEROS DE TEXTO CON SCANNER

A partir de Java 5 se puede leer un fichero de texto utilizando la clase Scanner igual que si leyéramos por teclado.

Para ello se le pasa al constructor de Scanner el objeto File asociado al fichero. Esta operación lanza una excepción **FileNotFoundException**.

Ejemplo de lectura de un fichero de texto con Scanner:

Programa que lee línea a línea el contenido del fichero datos.txt utilizando la clase Scanner. Se utiliza el método ***hasNext()*** de Scanner para saber si quedan más datos que leer en el fichero. Este método devuelve false si se ha llegado al final del fichero y true en caso contrario.

PROCESAMIENTO DE TEXTOS

```java
package cadenas7;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class LeerTextoScan {
    public static void main(String[] args) throws FileNotFoundException {
        File f = new File("D:\\CARPETA DOCENTE\\UNHEVAL\\2023\\6. ALGORITMOS Y ESTRUCTURA DE D
         String cadena;
         try (Scanner entrada = new Scanner(f)) {
             while (entrada.hasNext()) {        //mientras no se alcance el final del fichero
                 cadena = entrada.nextLine();   //se lee una línea del fichero
                 System.out.println(cadena);    //se muestra por pantalla
             }
         } catch (FileNotFoundException e) {
             System.out.println(e.getMessage());
         }
    }
}
```