DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

CARRERA: Ingeniería Electrónica	**GUÍA** No. 01	**TIEMPO ESTIMADO:** 1h y 20 min.
ASIGNATURA: Programación Orientada a Objetos	**FECHA DE ELABORACIÓN:** 02/07/2021 **SEMESTRE**: Mayo – Septiembre 2021	
TÍTULO: Persistencia de Datos en JAVA	**DOCENTE: Ing. César Osorio** **Integrantes: Brayam Guanoliquin, Steven Flores, Quishpi Jordan,Arley Camayo,David Maila**	

OBJETIVO

p.e. Implementar el intercambio de datos (lectura y escritura) entre fuentes externas (archivos y/o entrada y salida estándar) y un programa (en un lenguaje orientado a objetos).

INSTRUCCIONES

p.e.

i) Utilice como material principal, aquel indicado en clase por el docente.
ii) Utilice información consultada en Internet y conocimiento adquirido en clase.

MARCO TEÓRICO

Archivos de texto en java

Si se desea procesar datos de un archivo existente, se debe:

1. Abrir el archivo
2. Leer o introducir los datos en las variables, un elemento a la vez
3. Cerrar el archivo cuando se termine de trabajar con él

Para transferir algunos datos de ciertas variables a un archivo, se debe:

1. Abrir el archivo
2. Extraer o escribir los elementos en la secuencia requerida

3. Cerrar el archivo cuando se termine de trabajar con él

Al leer un archivo, todo lo que puede hacerse es leer el siguiente elemento. Si, por ejemplo, Quisiéramos examinar el último elemento, tendríamos que codificar un ciclo para leer cada uno de los elementos en turno, hasta llegar al elemento requerido. Para muchas tareas, es conveniente visualizar un archivo como una serie de líneas de texto, cada una compuesta por un número de caracteres y que termina con el carácter de fin de línea. Un beneficio de esta forma de trabajo es la facilidad de transferir archivos entre aplicaciones. Así se podría crear un archivo ejecutando un programa en Java y después cargarlo en un procesador de palabras, editor de texto o correo electrónico.
Los programas que utilizan archivos deben contener la instrucción:

import java.io.*;

Antes de hablar sobre cómo gestionamos los archivos debemos de conocer qué son, para ello tenemos esta sencilla explicación: un archivo es un conjunto de datos estructurados guardados en algún medio de almacenamiento que pueden ser utilizados por aplicaciones.

Está compuesto por:

Nombre: Identificación del archivo.
Extensión: Indica el tipo de archivo.

Un archivo o fichero informático es un conjunto de bits que son almacenados en un dispositivo. Un archivo es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. A los archivos informáticos se les llama así porque son los equivalentes digitales de los archivos escritos en expedientes, tarjetas, libretas, papel o microfichas del entorno de oficina tradicional.

Sentencias try-catch

• Para controlar las excepciones, se utiliza la sentencia try-catch
• Dentro del bloque try, se ejecutan las sentencias sobre las que se quiere controlar las excepciones
• Dentro del bloque catch, se muestran las sentencias que se deben ejecutar en caso de error

```
try {
   //SENTENCIAS A EJECUTAR
} catch (Exception e) {
   //SENTENCIAS QUE SE EJECUTAN SI HAY ERROR
}
```

• Como parámetro, el catch pone el tipo de excepción a controlar. Si se quieren controlar todas, se usa
Exception
• La sentencia finally es opcional, y contiene las sentencias que se van a ejecutar exista o no una excepción

Operaciones con ficheros

• **Apertura:** El programa abre el fichero y se prepara para leerlo o escribirlo. Suele "reservar" el fichero
para sí
• **Cierre:** Indica que se ha finalizado con las operaciones sobre el fichero. Libera el fichero
• **Lectura:** Lee el fichero o una de sus partes
• **Escritura:** Permite escribir en el fichero, ya sea añadiendo datos o sobrescribiendo los ya existentes
• **Ejecución:** Similar a la lectura, pero utiliza los datos del fichero para ejecutar un software
• **Creación:** Crea un nuevo fichero con un nombre, extensión y ruta determinados
• **Eliminación:** Elimina un fichero determinado

La clase File

• La clase que manipula los ficheros en Java se llama File
• Con esta clase se pueden hacer un gran número de operaciones sobre un fichero y sus propiedades, pero no se permite
leer ni escribir
• También permite obtener datos del fichero, como rutas, nombres, permisos e incluso si existe

• El resto de clases que manipulan ficheros parten de la existencia de una clase File, por lo que es la base de cualquier
operación de manipulación de ficheros

La clase File

• Además de la apertura y cierre, la clase File permite realizar un gran número de operaciones:
• Comprobar si el fichero existe
• Crear un fichero
• Borrar el fichero
• Obtener nombre, rutas (absolutas y relativas) y extensión del fichero
• Decir si el fichero es un fichero o un directorio
• Obtener el tamaño en bytes del fichero
• Consultar y cambiar los permisos del fichero
• Si es un directorio, obtener la lista de ficheros que contiene el mismo
• Crear nuevos directorios
• Renombrar fichero
• Etc
"Búfer" significa que, el software lee un gran trozo de datos del dispositivo de almacenamiento externo y lo guarda en la RAM, de tal forma que invocaciones sucesivas de los métodos que necesitan leer una pequeña cantidad de datos del dispositivo de almacenamiento de archivos puedan obtener rápidamente los datos de la RAM. Por lo tanto, un búfer actúa como un amortiguador temporal entre el dispositivo de almacenamiento y el programa.

Lectura de un fichero de texto en java

Podemos abrir un fichero de texto para leer usando la clase FileReader. Esta clase tiene métodos que nos permiten leer caracteres. Sin embargo, suele ser habitual querer las líneas completas, bien porque nos interesa la línea completa, bien para poder analizarla luego y extraer campos de ella. FileReader no contiene métodos que nos permitan leer líneas completas, pero sí BufferedReader. Afortunadamente, podemos construir un BufferedReader a partir del FileReader de la siguiente forma:

```
File archivo = new File ("C:\\archivo.txt");
FileReader fr = new FileReader (archivo);
BufferedReader br = new BufferedReader(fr);
...
String linea = br.readLine();
```

La apertura del fichero y su posterior lectura pueden lanzar excepciones que debemos capturar. Por ello, la apertura del fichero y la lectura debe meterse en un bloque try-catch.

Además, el fichero hay que cerrarlo cuando terminemos con él, tanto si todo ha ido bien como si ha habido algún error en la lectura después de haberlo abierto. Por ello, se suele poner al try-catch un bloque finally y dentro de él, el close() del fichero.

Crear archivos de texto usando FileWriter y BufferWriter

Te mostramos como crear un archivo de texto en Java con dos técnicas diferentes.

La primera estrategia es usando las clases FileWriter y BufferWriter, donde usamos el método write que te permite escribir cadenas o arreglos de caracteres.

La segunda forma es usando PrintWriter que te permite hacer más o menos lo mismo, pero de una forma más resumida y con la posibilidad de escribir otros tipos de datos sobre el archivo.

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;

public class CrearArchivo {

    public static void main(String ars[]){
        try {
            String ruta = "/ruta/filename.txt";
            String contenido = "Contenido de ejemplo";

            File file = new File(ruta);

            // Si el archivo no existe es creado
            if (!file.exists()) {
                file.createNewFile();
            }

            FileWriter fw = new FileWriter(file);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(contenido);
            bw.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

La escritura se hace en estas sentencias:

```
FileWriter fw = new FileWriter(file);
BufferedWriter bw = new BufferedWriter(fw);
bw.write(contenido);
bw.close();
```

Crear archivos usando PrintWriter

El código es el siguiente:

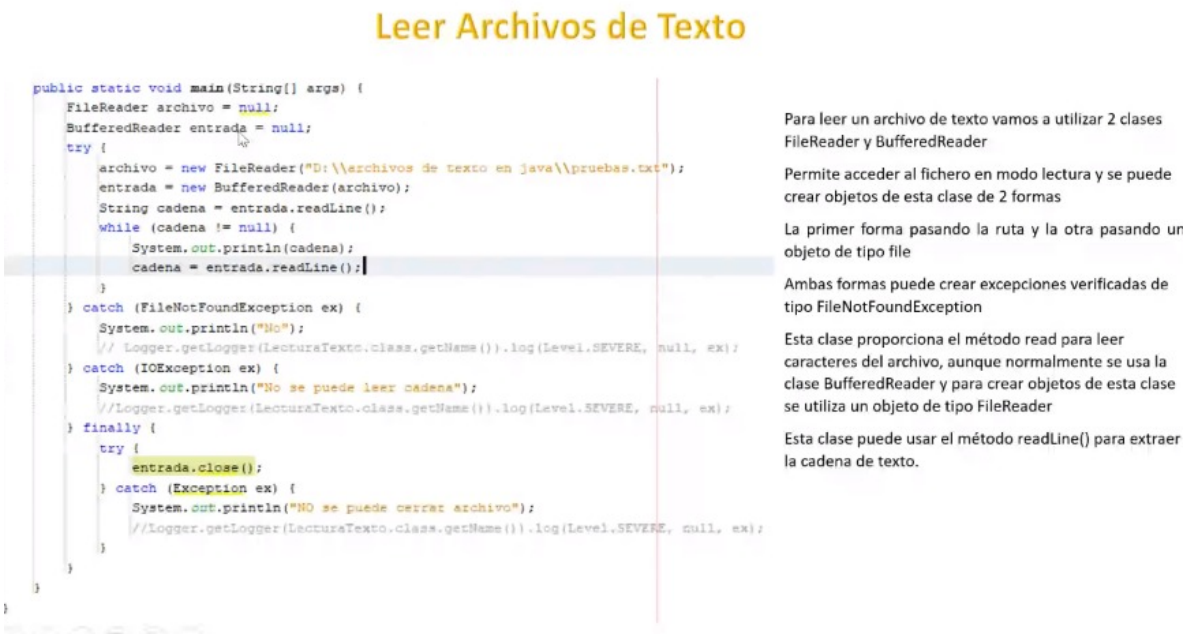
```
import java.io.PrintWriter;

public class CrearArchivo {

   public static void main(String ars[]){

       try {
```

```
            PrintWriter writer = new PrintWriter("/ruta/filename.txt", "UTF-8");
            writer.println("Primera línea");
            writer.println("Segunda línea");
            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Leer un archivo de texto

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER



En Java existen diversas formas de leer un archivo, pero una de las más sencillas es usando la clase Scanner, como te mostramos a continuación:

```
import java.io.File;

import java.util.Scanner;

public class LeerArchivo {

   public static void main(String[] args) {

       try {

           Scanner input = new Scanner(new File("/ruta/filename.txt"));
```

```
            while (input.hasNextLine()) {

                String line = input.nextLine();

                System.out.println(line);

            }

            input.close();

        } catch (Exception ex) {

            ex.printStackTrace();

        }

    }

}
```

Usando el método **hasNextLine()** puedes determinar si el archivo de texto contiene líneas que pueden ser leídas, normalmente esto se hace dentro de un ciclo en la condición de paro, para después leer el contenido de las mismas con el método **nextLine()**, cuando el método **hasNextLine()** deja de devolver **true** el ciclo de lectura debe parar. Es importante que después de leer la información que necesites del archivo uses el método **close()** que liberará el recurso para que pueda ser usado sin problemas posteriormente.

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

1. **Planteamiento del problema**
   p.e. Ejercicio

   Construya un programa que permita grabar en un archivo de texto el cálculo de la edad que tiene cada empleado de una empresa a partir de su fecha de nacimiento, la salida deberá mostrar el nombre del empleado con su edad en formato de años, meses y días, para lo cual deberá crear la siguiente interfaz:

   - Un Menú con las siguientes opciones
     - Abrir /Crear Archivo
     - Insertar Datos
     - Leer Datos
     - Cerrar Archivo
     - Salir

Conclusiones

El presente informe concluye con la resolución y la ejecución correcta del planteamiento del problema, aplicando los métodos y los procesos a seguir, es así, como el manejo de archivos dentro de Java, nos permite ayudar agilizando los diferentes transcursos dentro del programa, siendo estos la lectura y escritura sumamente importante, que como ya visualizamos dentro del marco teórico, una serie de elementos que son de suma importancia a la hora de nosotros poder lograr el objetivo deseado.

_____ _____

DOCENTE RESPONSABLE COORDINADOR DE ÁREA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

Ing. César O. Osorio A

Ing. Silvia Arévalo