



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:*

TISTA GARCÍA EDGAR

*Asignatura:*

PROGRAMACIÓN ORIENTADA A OBJETOS

*Grupo:*

3

*No de Práctica(s):*

PRÁCTICA #11 – MANEJO DE ARCHIVOS

*Integrante(s):*

CARRANZA OCHOA JOSE DAVID  
CARRILLO CERVANTES IVETTE ALEJANDRA

*No. de Equipo de  
cómputo empleado:*

TRABAJO EN CASA

*No. de Lista o Brigada:*

07, 08

*Semestre:*

2022 - 1

*Fecha de entrega:*

08 DICIEMBRE 2021

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

## PRÁCTICA #11 – Manejo de Archivos

**Objetivo.** 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).

### ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

#### Ejemplos de la guía


- **Escribe**

Es una clase que tiene como objetivo, probar los siguientes métodos de la clase File:

- ✓ Exists(). Comprueba si un archivo existe, retorna un valor booleano el cual indica si dicho archivo existe.
- ✓ createNewFile(). Tiene como función crear un archivo nuevo con el nombre que lo indique el objeto File.

Para implementar el funcionamiento de cada uno de estos métodos, se creó una clase llamada “Escribe”, en la cual se creo un archivo tipo File con un nombre en específico “archivo.txt”, este objeto manda a llamar a los métodos mencionados anteriormente y posterior a ello se imprimen estos, dando la siguiente salida del programa:

```
run:
false
true
true
BUILD SUCCESSFUL (total time: 0 seconds)
```

 archivo.txt

Cabe destacar que el primer valor booleano corresponde a la existencia del archivo.txt (al momento de instanciar el objeto File, todavía no se crea); mientras que los siguientes dos valores booleanos, corresponden al momento de llamar al método “createNewFile” y “exists”, los cuales se implementan después de crear dicho archivo.

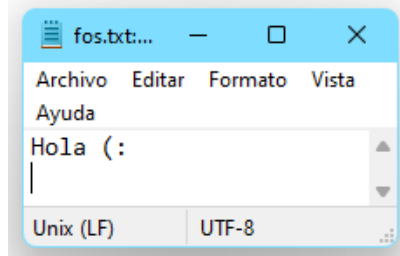
- **Clase FileOutputStream**

Dentro de esta clase en particular, se tiene un objeto de la clase “FileOuputStream” quien, tras recordar las clases teóricas, recordaremos que estos en particular trabajan sobre archivos binarios. A raíz de esto, el programa trabaja con un arreglo de caracteres quienes tendrán una longitud máxima de 81 por arreglo.

Lo que simulamos dentro del código es una lectura dentro de un flujo de bytes que se encuentra asociado a lo que es un arreglo de “byte”; sobre este mismo, por medio de “read” se establece la cantidad de bytes disponibles, quienes serán expresados como un flujo de datos.

Con ayuda de un canal de estos bytes, la información se transmite entre lo que representa la entrada desde el teclado para posteriormente, escribir sobre un archivo de texto, la secuencia de bytes obtenida.

```
run:
Escribir texto a guardar en el archivo
Hola (:
BUILD SUCCESSFUL (total time: 12 seconds)
```



- **Clase FileInputStream**

```
run:
Error: java.io.FileNotFoundException: hola.txt (El sistema no puede encontrar el archivo especificado)
BUILD SUCCESSFUL (total time: 0 seconds)
```

- **Clase FileReader**

Dentro de tal clase, se tiene ahora el principio para la lectura de archivos de texto donde observamos que se trabaja con “BufferedReader” quien a través de lo que se conoce cómo buffer, almacenará dentro de su canal el archivo del tipo “FileReader” que proporcionará las características del archivo trabajado como lo es su nombre.

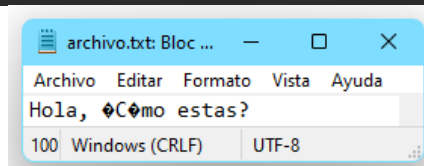
Observando el cómo trabaja este último mencionado, requiere una cadena que será inicializada por su constructor, que, tras ser convertido a un flujo de bytes entendible por Java, podrá acceder a las propiedades de los archivos cómo lo son las líneas presentes en el archivo.

Con ayuda de un ciclo “while”, se puede recoger las líneas del archivo previamente descrito, mostrando su contenido gracias a la expresión intrínseca de nuestro buffer, siendo esta la lectura de cadenas.

```
run:
El texto contenido en el archivo leer.txt es:
Hola, hola, holaaaaaaa! (:
BUILD SUCCESSFUL (total time: 0 seconds)
```

- **Clase FileWriter**

```
run:
Escribir texto:
Hola, ¿Cómo estas?
BUILD SUCCESSFUL (total time: 10 seconds)
```



- **LeeTecladoBR**

Dentro de esta clase, se maneja nuevamente “BufferedReader” con el fin de poder leer una secuencia de caracteres por medio del buffer.

Aquí, se presenta la opción de ingresar desde el teclado aquella cadena de información que será expresada; teniendo entonces, una herramienta que contribuye además del “Scanner” para la entrada de datos desde el teclado.

Cabe mencionar que se está trabajando de forma que el flujo de datos acceda directamente desde un objeto lector, quien, a su vez, deberá ser apoyado por “InputStreamReader”. Tal manejo a nivel de bytes permite que se trabaje de manera más eficiente con la entrada de la información

```
run:
Escribir el texto deseado:
Texto
El texto escrito fue: Texto
BUILD SUCCESSFUL (total time: 4 seconds)
```

- **LeeTecladoCompleto**

Para este programa se usa la clase StringTokenizer, la cual permite separar una cadena de texto por palabras (espacios) o por algún otro carácter. En el caso de este ejemplo, tiene como objetivo separar una cadena de texto dada por espacios.

```
run:
Escribir texto:
Ya casi es Navidaaaaad (:

El texto separado por espacios es:
Ya
casi
es
Navidaaaaad
(:
BUILD SUCCESSFUL (total time: 20 seconds)
```

## ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Para implementar cada ejercicio que se pidió en esta práctica, se realizó un menú con las siguientes opciones, con el fin de que la implementación del programa fuera más eficiente:

```
ARCHIVOS
1. Archivo Texto
2. Archivo Binario
3. Archivo Objeto
4. Salir
-> Elige una opción:
```

### Ejercicio 1. Archivos de texto plano

En este primer ejercicio tiene la función de realizar diferentes acciones sobre un archivo de texto plano, como es la creación, la escritura y la lectura de este, para ello se realizó un submenú con las siguientes opciones:

```
ARCHIVO TEXTO
1. Crear archivo
2. Ver contenido de archivo
3. Editar archivo
4. Salir
-> Elige una opción:
```

Para la primera opción, se realizó una clase con los métodos correspondientes para cada opción:

Para crear el archivo, lo primero que se realizó fue realizar una estructura try catch, la cual tiene como objetivo pedirle al usuario el nombre que desea que tendrá el archivo y con base a dicho nombre, instanciar un objeto de tipo PrintWriter con el nombre del archivo nuevo que se generará, con el objetivo de crearlo, recordemos que este método es de escritura tiene la función, como su nombre lo indica, de escribir sobre un archivo de texto; sin embargo, al no tener ningún archivo, lo crea con el nombre que el usuario decida, posterior a ello lo debe de cerrar.

Para el segundo método, leer, se realizó también un bloque try catch, el cual tiene como propósito leer la información de un archivo de texto en específico indicado por el usuario, para ello se creó un objeto de la clase FileReader para que este pueda leer el archivo indicado. Cabe recalcar que para leer el archivo se realizó un ciclo de repetición el cual será encargado de leer línea por línea de dicho archivo y posterior a ello imprimirla.

Finalmente, para el tercer y último método, "editarArchivo", tiene como objetivo modificar un archivo de texto, ya sea sobrescribirlo o agregar texto al final de este, según la opción de decida al usuario se manda al método con un valor booleano como parámetro, el cual posteriormente indicará si el

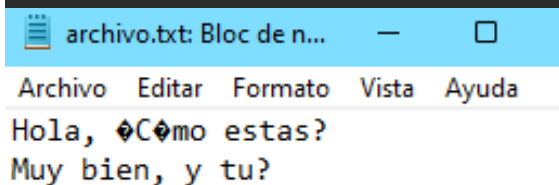
archivo se sobrescribirá o se agregará texto al final de dicho archivo. Cabe destacar que este método utiliza la clase `PrintWriter`, con el fin de escribir, o bien, sobrescribir en un archivo en específico.

La salida del programa es la siguiente:

```
. _ . _ . _ . _ . _ . _ .  
  
      CREAR ARCHIVO  
Introduzca el nombre del nuevo archivo: (con extensión)      Hi.txt  
. _ . _ . _ . _ . _ . _ .
```

```
. _ . _ . _ . _ . _ . _ .  
  
      LEER ARCHIVO  
Introduzca el nombre del archivo: (con extensión)      archivo.txt  
Hola, ¿Cómo estas?  
. _ . _ . _ . _ . _ . _ .
```

```
. _ . _ . _ . _ . _ . _ .  
  
      EDITAR ARCHIVO  
1. Agregar texto al final  
2. Sobreescribir el archivo  
-> Elige una opción:      1  
Introduce la frase que quieras agregar:      Muy bien, y tu?  
Introduzca el nombre de un archivo existente:      archivo.txt  
. _ . _ . _ . _ . _ . _ .
```



archivo.txt: Bloc de n... — □

Archivo Editar Formato Vista Ayuda

Hola, ¿Cómo estas?

Muy bien, y tu?

## Ejercicio 2. Archivos Binarios

Este segundo ejercicio tiene como finalidad convertir un archivo de texto plano a un archivo binario y viceversa, para que la implementación del programa se realizó el siguiente menú de opciones:

```
• ____ • ____ • ____ • ____ • ____ • ____ • ____ •  
  
ARCHIVO BINARIO  
  
1. Convertir texto a binario  
2. Convertir de binario a texto  
3. Salir  
-> Elige una opción:
```

Este breve submenú ejemplifica lo que el programa debe de realizar, comenzando con la conversión de binario de texto, donde se trabaja directamente con un archivo “.dat” este será expresado en su formato de bytes quien posteriormente, para cada iteración de la cadena o cadenas proporcionadas, se trabajará con la escritura en un archivo de texto de tal información.

Teniendo entonces el caso donde el archivo se procesa como lista para ser pasada al método “crearArchivoTexto” quien ejecutará la conversión para este tipo de archivos.

```
• ____ • ____ • ____ • ____ • ____ • ____ • ____ •  
  
DE TEXTO A BINARIO  
  
• ____ • ____ • ____ • ____ • ____ • ____ • ____ •  
Introduzca el nombre del archivo de texto: (con extensión)      archivo.txt  
Hola, ¿Cómo estas?  
  
• ____ • ____ • ____ • ____ • ____ • ____ • ____ •  
  
Nombre del archivo binario a crear (sin extensión):           holaComoEstas  
• ____ • ____ • ____ • ____ • ____ • ____ • ____ •
```

Como se logra apreciar, la mayor dificultad se presentó al manejar flujos de bytes para reconvertir el archivo de binario a texto, por lo que aún necesitamos profundizar más en esta cuestión.

### Ejercicio 3. Archivos de Objetos

El ejercicio en cuestión nos solicita crear una serie de alumnos que cuenten con ciertos atributos, quienes, al ser inicializados y almacenados con algún valor, deberán ser almacenados en una lista de “Alumnos”; esta lista deberá ser expresada en un archivo, el cual deberá contener la información y podrá ser modificado.

Para esto, el primer paso es realizar nuestra clase “Alumno” quien contiene los atributos del mismo como lo es su nombre y una serie de asignaturas con las que será definido.

Tras esto, nuestra clase “ArchivoObjeto” es quien realiza el trabajo más importante; dentro de esta clase existe un submenú para el usuario donde se definen los casos para crear un nuevo archivo, observar el contenido de uno ya existente o bien, editarlo.

Para nuestro primer caso, se tiene una llamada a “crearArchivo” donde es necesario establecer una lista de alumnos para poder trabajar sobre ella, antes de poder establecer las características internas del archivo, se deberá inicializar los valores de los alumnos que se consideren necesarios; por lo que, “crearAlumno” realiza esta labor retornando un “Alumno”.

Ahora, para agregar cada “Alumno” declarado, se podrá acceder a este por medio de un nuevo archivo para que se almacene en memoria secundaria, hablamos pues de archivos de objetos.

Su escritura viene dada por un bucle “for” quien recorre desde 0 hasta el valor máximo de “Alumnos” presentes en la lista, para cada “Alumno”, será ingresado a un archivo, donde en un paso previo, se ha inicializado para su apertura a datos.

La escritura viene dada entonces por “writeObject” que nos proporcione manera intuitiva la escritura en archivos de objetos, por lo que, en cada iteración, un nuevo alumno será plasmado.

Al observar la ruta donde se ha destinado (justo en la carpeta del proyecto), se observa que la información contenida no es nada coherente, ya que estos se almacenan en formato binario y por lo tanto; su decodificación necesitará un proceso inverso.

El proceso de decodificación resulta un tanto especial, aquí se tiene el método “leerArchivo” quien, cómo se vio en los ejemplos de la guía, podrá ser leído al trabajar con las clases complementarias de “OutputStream”, hablamos de “InputStream”.

Dentro del constructor de “ObjectInputStream” es necesario contar con un objeto del tipo “Scanner” que asocie nuestro archivo con el objeto “FileInputStream”, tras ser establecida la lectura del mismo, empleamos el método “readObject” quien trabaja con una secuencia de bytes para nuestro archivo de objetos de “Alumnos”.

Para nuestra listade archivos, se recorre a la lista principal poseedora de la información del archivo, pero ya en formato de “Alumnos”, aquí se establece por medio del método “display” la impresión en pantalla de los atributos para cada elemento de la lista, considerando que para el momento en que no se encuentren más de estos objetos por recorrer, nuestro ciclo finalizará.

Cabe mencionar que una de nuestras limitaciones para este tipo de ejecuciones se presenta para



casos donde el archivo de objetos solo cuenta con un objeto o bien se trata de un archivo vacío, por lo que se deberá tomar a consideración para antes de ejecutar el código.

```
· _ _ · _ _ · _ _ · _ _ · _ _ · _ _ ·  
  
    CREAR ARCHIVO  
¿Cuantos alumnos quieres crear?  2  
  
· _ _ · _ _ · _ _ · _ _ · _ _ · _ _ ·  
  
Nombre: Alejandra  
Apellido: Carrillo  
  
Asignaturas:  
Ingresa la clave de la asignatura 1:  123  
Ingresa la clave de la asignatura 2:  456  
Ingresa la clave de la asignatura 3:  789  
  
· _ _ · _ _ · _ _ · _ _ · _ _ · _ _ ·  
· _ _ · _ _ · _ _ · _ _ · _ _ · _ _ ·  
  
Nombre: David  
Apellido: Carranza  
  
Asignaturas:  
Ingresa la clave de la asignatura 1:  789  
Ingresa la clave de la asignatura 2:  456  
Ingresa la clave de la asignatura 3:  123  
[Alumno{nombre=Alejandra, apellido=Carrillo, asig  
Introduzca el nombre del archivo:  alumnos.txt
```

```
· _ _ · _ _ · _ _ · _ _ · _ _ · _ _ ·  
  
    LEER ARCHIVO  
Introduzca el nombre del archivo:  alumnos.txt  
· . . . . . . . . . . . . . . . . . . .  
    Alumno 1  
  
Nombre: Alejandra  
Apellido: Carrillo  
Asignaturas:  
[123, 456, 789]  
· . . . . . . . . . . . . . . . . . . .  
    Alumno 2  
  
Nombre: David  
Apellido: Carranza  
Asignaturas:  
[789, 456, 123]
```

Pasando entonces al tercer caso, las modificaciones que pueden realizarse dentro del archivo de objetos son: agregar un nuevo “Alumno” o bien eliminarlo.

La inserción de un nuevo “Alumno” describe la recopilación de la información de nuestro archivo hacía una lista del mismo tipo, para posteriormente agregar en susodicha lista al nuevo alumno que se haya mandado crear.

Con esto, se debe de recorrer el bucle nuevamente para expresar por separado los valores de cada “Alumno”.

La eliminación supone entonces un mecanismo un tanto más sencillo, donde se ha de solicitar el valor del alumno a eliminar, y este será suprimido de la lista principal de “Alumnos” en la que ya se cuenta; añadiendo nuevamente los valores dentro del archivo establecido.

## **Conclusiones**

Carrillo Cervantes Ivette Alejandra

Los objetivos de esta práctica se cumplieron, ya que implementamos intercambio de datos entre fuentes externas, en este caso los archivos de texto, binarios, así como de objetos.

En un principio nos costó un poco de trabajo como plantear las clases o bien, los métodos correspondientes para cada tipo de archivo, ya que al no haber trabajado antes con los archivos de objetos andábamos un poco perdidos; sin embargo, al discutir las diferentes opciones y al basarnos en el código proporcionado por el profesor, fue mucho más entendible el uso de este tipo de archivos, así como de los demás, ya que al investigar un tipo de archivos en específico, llegábamos a investigar otras cosas y poco a poco fue más entendible estos conceptos.

Personalmente, considero que el hecho de realizar esta práctica en equipos fue de gran ayuda para entender el tema, ya que pudimos explicarnos entre nosotros si no entendíamos en cierto punto de la práctica.

Considero que los ejercicios propuestos fueron muy bien planteados para comprender el tema de una manera exitosa, y a pesar de que nos faltó un ejercicio que fue el de archivos de texto a binarios, considero que el aprendizaje fue muy bueno.

**Carranza Ochoa José David**

Dentro de la práctica se consideraron aspectos importantes para el desarrollo de la misma, trabajando con archivos desde el punto de vista de la programación orientada a objetos.

Trabajando en conjunto con mi compañera, pudimos distinguir la diferencia fundamental entre los archivos como TDA o desde este caso, trabajando con canales de bytes que comunican el acceso a los archivos.

La lectura y escritura de los mismos significó una mejor comprensión de la parte teórica, quien, durante esta práctica, pudo ser expresada de mejor forma al trabajar desde una gran perspectiva de tipos de archivos, comenzando desde los de texto plano hasta llegar a su representación en bytes para expresar binarios o de objetos.

A pesar de que no todos los ejercicios resultaron como se esperaba, se consideró un buen desempeño de los mismos, considerando como pieza fundamental el trabajo en equipo para mejora del objetivo.

Quisiera agregar que si bien se tuvo una amplia gama de ejercicios, pudiera parecer que los últimos son más

complejos al trabajarlos desde esta perspectiva, por lo que, en la parte teórica se podría profundizar más.