	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS II

Grupo: 5

Número de Práctica(s): Guía práctica de estudio 10: Archivos

Integrante(s): MURRIETA VILLEGAS ALFONSO |VALDESPINO MENDIETA JOAQUÍN

Núm. De Equipo de cómputo empleado: 38

Semestre : 2019 - 1

Fecha de entrega: 23 DE OCTUBRE DE 2018

Observaciones:

CALIFICACIÓN: _____

ARCHIVOS

OBJETIVOS DE LA PRÁCTICA

- Trabajar con operaciones, tipo de acceso y organización de TDA archivo en algún lenguaje de progrmación.

INTRODUCCIÓN

El sistema operativo hace una abstracción de las propiedades físicas de los dispositivos de almacenamiento secundario para definir una unidad lógica de almacenamiento, el archivo.

Un archivo es una colección de información relacionada con un nombre que se guarda en memoria secundaria y comúnmente representan programas y datos. En general es una secuencia de bits, bytes, líneas o registros cuyo significado es definido por el creador y usuario del archivo. Se puede visualizar como espacios de direcciones lógicas contiguas.

ATRIBUTOS DE UN ARCHIVOS

Además del nombre un archivo tiene otros atributos que varían de un sistema operativo a otro, en general son:

- **Nombre:** Nombre simbólico visible al usuario.
- **Tipo**: Esta información es necesaria cuando se pueden trabajar diferentes tipos.
- **Ubicación**: Es un apuntador a un dispositivo y a la ubicación del archivo en el dispositivo
- **Tamaño**. Se tiene el tamaño actual (en bytes, palabras o bloques) y posiblemente el tamaño máximo permitido.
- **Protección**. Información del control de acceso, que determina quién puede leer, escribir, ejecutar, etc., el archivo
- **Hora, fecha e identificación del usuario**. Estos datos pueden ser útiles para la protección de seguridad y control de uso.

OPERACIONES SOBRE ARCHIVOS

Un archivo es un tipo de dato abstracto (TDA) que se manipula por medio de una interfaz ya sea por el SO o por algún programa. Al ser un TDA se tienen operaciones que se realizan sobre él. Las principales son.

• **Creación del archivo:** Para crear un archivo primero, es necesario encontrar espacio para el mismo

• **Escribir en un archivo:** Para escribir en un archivo, se debe realizar una llamada al sistema especificando el nombre y la información que se escribirá. Con el nombre se localiza la ubicación así, el sistema mantendrá un apuntador de escritura a la ubicación en el archivo, donde va a tener lugar a la siguiente escritura y este debe actualizarse siempre que ocurra una escritura.

• **Lectura del archivo:** Para leer un archivo, se realiza una llamada al sistema especificando el nombre y dónde debe colocarse (dentro de la memoria) el siguiente bloque del archivo. Se explora el directorio para hallar la entrada asociada y el sistema necesita mantener un apuntador de lectura que haga referencia a la ubicación dentro del archivo en la que tendrá lugar la siguiente lectura. Una vez que la lectura se completó, se actualiza el apuntador de lectura

• **Borrar un archivo:** Para borrar un archivo, se explora el directorio en busca del archivo indicado. Una vez encontrada la entrada de directorio asociada, se libera todo el espacio del archivo y se borra la entrada del directorio.

Dado que en las operaciones descritas se realiza primero una búsqueda en el directorio para encontrar la entrada asociada con el archivo, muchos sistemas requieren que se realice una función de apertura (open()) antes de utilizar por primera vez el archivo, así cuando se solicita una operación sobre un archivo, se utiliza un índice (descriptor de archivo) en la llamada tabla de archivos abiertos que contiene información de todos los archivos abiertos , de manera que no se requiere de una búsqueda. Cuando el archivo deja de utilizarse será cerrado por un proceso y el S. O. eliminará la entrada correspondiente en la tabla de archivos abiertos.

Es importante mencionar que en la función de apertura de un archivo se toma el nombre de este y explora el directorio, copiando la entrada del directorio correspondiente en la tabla de archivos abiertos. Esta función además de recibir el nombre del archivo también acepta información acerca del modo de acceso: creación, solo lectura, escritura, lectura-escritura, agregar etc.

TIPOS ARCHIVOS

Existen archivos de diferentes tipos: cada uno podría ser un documento de texto, un binario ejecutable, un archivo de audio o video, etc. Una de las estrategias principales para que el sistema operativo reconozca el tipo de un archivo es la extensión. Cada nombre de archivo se divide en dos porciones, empleando como elemento separador al punto: el nombre del archivo y su extensión. hola.txt, programa.exe.

ESTRUCTURA DE ARCHIVOS

Los sistemas de disco normalmente tienen un tamaño de bloque bien definido y determinado por el tamaño de un sector. Todas las operaciones de entrada y salida en disco se realizan en unidades de un bloque (registro físico) y todos los bloques tienen un mismo tamaño. Es poco probable que el tamaño del registro físico corresponda exactamente a la longitud del registro lógico deseado el cual puede variar de longitud. Para resolver este problema se utiliza el llamado empaquetamiento de registros lógicos en bloques físicos.
El tamaño del registro lógico, el tamaño del bloque físico y la técnica de empaquetamiento determinan cuántos registros lógicos se almacenan en cada bloque físico.

MÉTODOS DE ACCESO

Los archivos almacenan información y cuando se requiere utilizarla es necesario tener acceso a ella y leerla en la memoria de la computadora.

El método más simple de acceso a la información es el secuencial donde la información se procesa en orden, un registro después de otro. Una operación de lectura lee la siguiente porción del archivo e incrementa automáticamente un apuntador de archivo, que controla la ubicación de E/S. De forma similar, la operación de escritura añade información al final del archivo y hace que el apuntador avance hasta el final de los datos recién escritos (el nuevo final del archivo). Los archivos pueden reinicializarse para situar el apuntador al principio de los mismos, y en algunos sistemas, puede que el programa sea capaz de saltar hacia adelante o hacia atrás n registros, para algún cierto valor n.

Otro método es el acceso directo, relativo u aleatorio surge con la llegada de los dispositivos de acceso directo como los discos magnéticos; en este método de acceso el archivo se considera como un conjunto

de registros, cada uno de los cuales puede ser un byte. Se puede acceder al mismo desordenadamente moviendo el apuntador de acceso al archivo a uno u otro registro. Esta forma de acceso se basa en un modelo de archivo almacenado en disco, ya que se asume que el dispositivo se puede mover aleatoriamente entre los distintos bloques que componen el archivo.

DIRECTORIOS

Algunos sistemas almacenan millones de archivos en terabytes de disco. Para gestionar todos esos datos, se necesitan que se organicen y esa organización implica el uso de directorios. La forma más simple de un sistema de directorios es tener un directorio que contenga todos los archivos.

Tener un solo nivel es adecuado para aplicaciones dedicadas simples, pero para los usuarios modernos con miles de archivos, sería imposible encontrar algo si todos los archivos estuvieran en un solo directorio.

Lo que se usa es una **jerarquía**. Cada directorio puede contener un número arbitrario de archivos, y también puede contener otros directorios (subdirectorios). Los otros directorios pueden contener todavía más archivos y directorios, y así sucesivamente, construyéndose una estructura en árbol en la que un directorio raíz puede contener cualquier número de niveles de otros directorios y archivos.

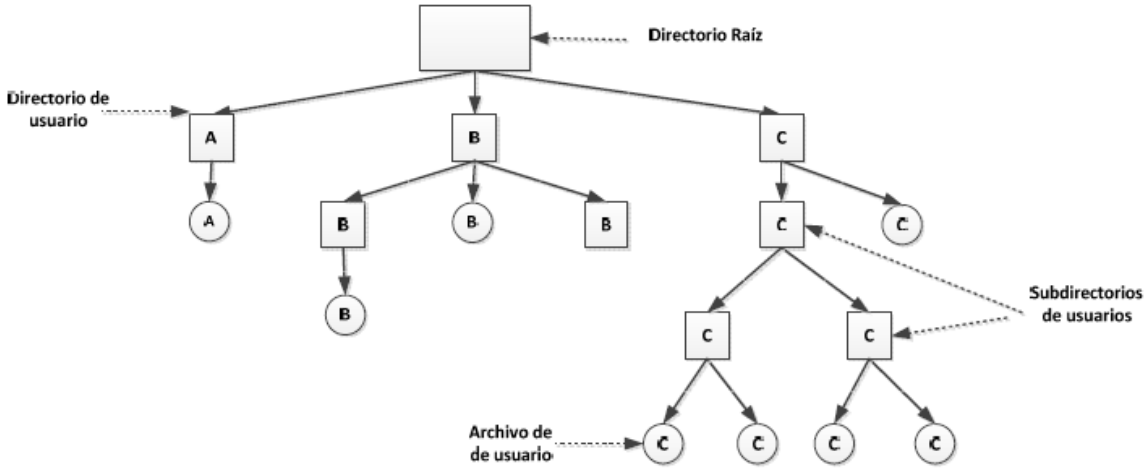

Imagen 1: Diagrama de un árbol donde realmente los nodos son archivos y directorios de un usuario.

Cuando el sistema de archivos está organizado como un árbol de directorios, se necesita cierta forma de especificar los nombres de los archivos. Por lo general se utilizan dos métodos distintos. En el primer método, cada archivo recibe un nombre de ruta absoluto que consiste en la ruta desde el directorio raíz al archivo. Por ejemplo, en las siguientes rutas el directorio raíz contiene un subdirectorio llamado usr que a su vez contiene un subdirectorio ast, el cual contiene el archivo mailbox (Ejemplos abajo)

Windows: \usr\ast\mailbox
UNIX: /usr/ast/mailbox
MULTICS: >usr>ast>mailbox

El otro tipo de nombre es el nombre de ruta relativa. Éste se utiliza en conjunto con el concepto del directorio de trabajo (también llamado directorio actual). Un usuario puede designar un directorio como el directorio de trabajo actual, en cuyo caso todos los nombres de las rutas que no empiecen en el

directorio raíz se toman en forma relativa al directorio de trabajo. Por ejemplo, si el directorio de trabajo actual es /usr/ast, entonces el archivo cuya ruta absoluta sea /usr/ast/mailbox se puede referenciar simplemente como mailbox.

Actividad 1. Ejercicios del Manual

Para este apartado se pidió codificar y aprender el manejos y aspecto acerca de archivos, así como sus respectivas operaciones, tipo de acceso y organizaciones lógicas.

NOTA: A continuación, se muestra una tabla con algunos comandos convenientes:

r	Modo de apertura	Ubicación del puntero
r	Solo lectura	Al inicio del archivo
rb	Solo lectura en modo binario	Al inicio del archivo
r+	Lectura y escritura	Al inicio del archivo
rb+	Lectura y escritura en modo binario	Al inicio del archivo
w	Solo escritura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
wb	Solo escritura en modo binario. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
w+	Escritura y lectura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
wb+	Escritura y lectura en modo binario. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
a	Añadido (agregar contenido). Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
ab	Añadido en modo binario (agregar contenido). Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
a+	Añadido (agregar contenido) y lectura. Crea el archivo si éste no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
ab+	Añadido (agregar contenido) y lectura en modo binario. Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo

SALIDA DEL PROGRAMA

A continuación, se muestran la salida del programa empleando el código dado en la práctica y a su vez los métodos que se tuvieron que llevar a cabo

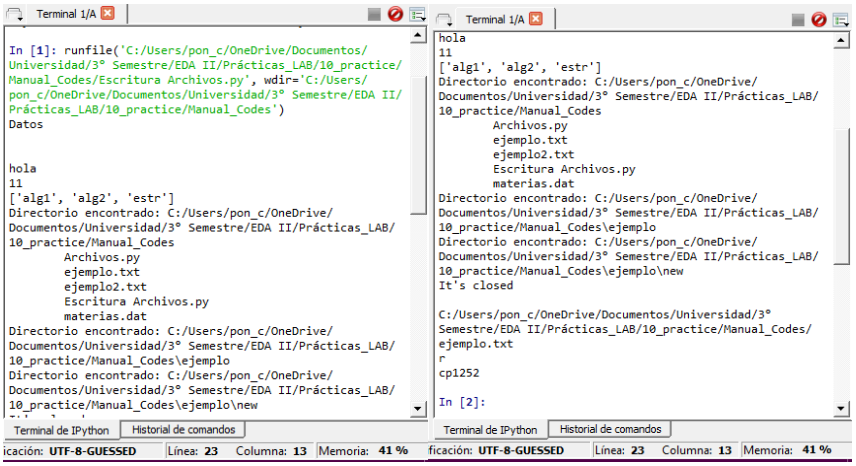

Imagen 2: Salidas del programa donde se puede ver como se desglosaron y crearon los archivos con su respectivo contenido, también se puede ver la salida acerca de la creación de la carpeta.

Evidencia de la creación de los 2 documentos hechos mediante los programas, además se puede ver la carpeta ejemplo la cual también fue creada mediante el código previo.

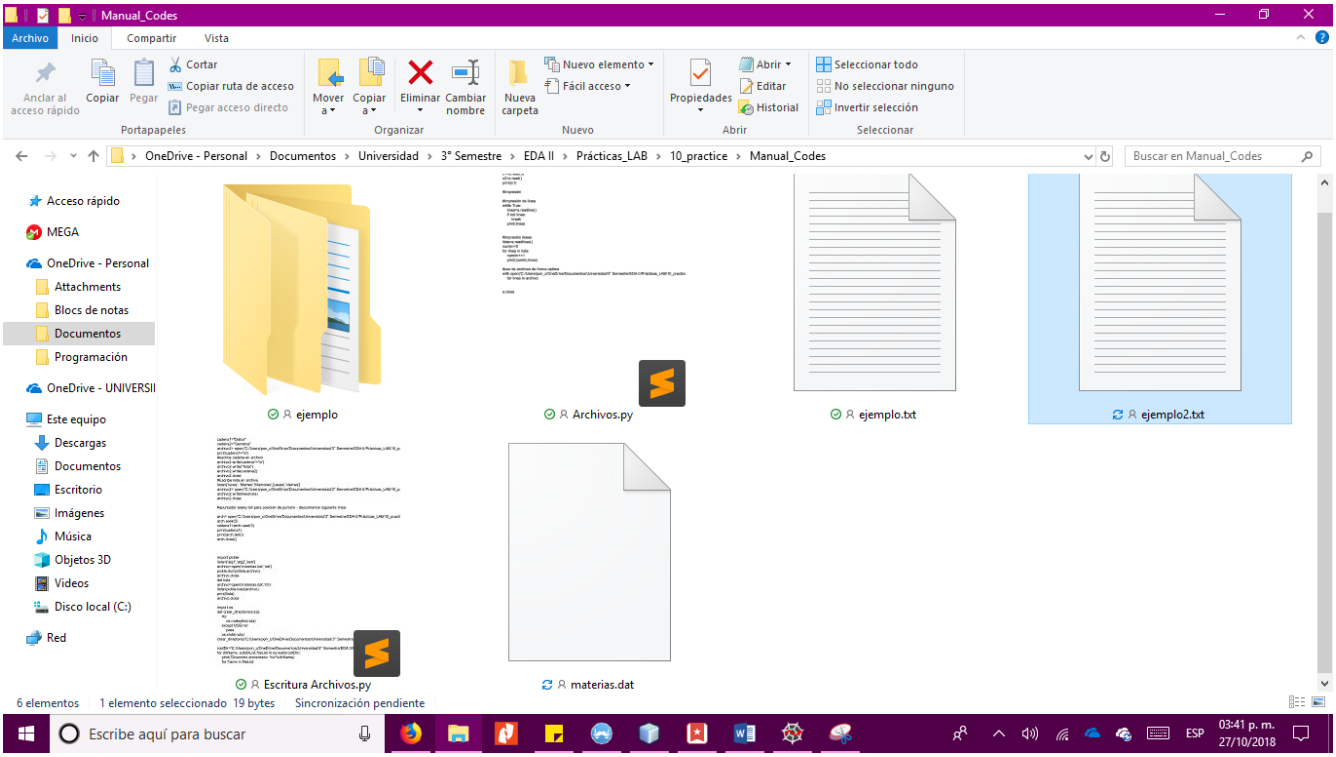

Imagen 2: Captura de pantalla donde se puede corroborar la creación de archivos y carpeta.

Actividad 0. Búsqueda de Bibliotecas de manejo de archivos en Java

Como apartado previo, se pidió buscar y realizar una lista de las bibliotecas y métodos correspondientes del manejo de archivos en el lenguaje Java, a continuación, se muestra la correspondiente lista:

BIBLIOTECA FILE

La instancia de la clase file crea un archivo. Con esto existen varios constructores para creación de objetos file.

Constructores:

- File f= new File("c:/carpeta/datos.txt");
- File f= new File ("fichero", "datos.txt");

Métodos de File:

- Boolean createNewFile(): Da a la creación de un fichero asociado al archivo File , esta función devuelve verdadero si se ha podido crear. Para la creación no debe de existir el fichero. Este necesita de la excepción IOException.

- Boolean delete(): Elimina un fichero o directorio. Si el directorio debe estar vacío. Devuleve true si se ha eliminado.

- Boolean exists(): Devuelve verdadero si el directorio o archivo existe.

- Boolean mkdir(): Crea un directorio, devuelve verdadero si se creó.

- String[] list(): Devuelve un array de strings con el nombre de los archivos y directorios que contiene el direcotrio indicado en el objeto File

- String getName(): Devuelve el nombre

CLASES FILE WRITER / BUFFEREDWRITER / PRINTWRITER

FileWriter tiene como función principal escribir datos en un archivo, con ayuda con la dirección o el nombre de este.

BufferedWriter Es un objeto que reserva mal memoria para que se guarde la información antes de su escritura en el archivo

PrintWriter Es un objeto que se utiliza para escribir directamente sobre un archivo de texto.

CLASES Y MÉTODOS USADOS

Para el desarrollo de esta práctica se emplearon las siguientes clases y métodos para el manejo de archivos:

JFileChooser

Es una clase destinada al apartado gráfico de archivos en java, lo único que se empleó de esta clase fue la instanciación para cada uno de los apartados que se pidieron en la práctica.

FileOutputStream y FileInputStream

A diferencia de las clases previas, el manejo en el caso concreto de estas dos clases es a través del manejo de bytes, realmente debemos considerar que en el flujo de datos de archivos a datos dentro de nuestro programa podemos hacerlo ya sea a través de datos del tipo cadena o a través de datos del tipo byte.

Actividad 1. Ejercicios del laboratorio | Bibliotecas en Java

Para este apartado simplemente se creó un proyecto en NetBeans donde la clase principal "Murrieta,Valdepino_practica10" la cual a través de su método main fuera la encargada de llamar a través de instancias o directamente mediante métodos las distintas clases que a continuación se describirán:

NOTA: Para el desarrollo de esta práctica y para aplicar conocimientos de la materia POO fue por lo que se decidió hacer mediante una interfaz gráfica el manejo y creación de cada uno de los casos del programa que se pidió en la práctica 10.

Clase "Murrieta_Valdespino_Practica10"

Es la clase encargada de llevar a cabo la instanciación de la clase ventana y el uso de dos métodos (Ver en captura inferior), para hacer visible la ventana y a su vez la ubicación al centro de la pantalla de esta.

```
public class Murrieta_Valdespino_Practica10 {
    public static void main(String[] args) {
        ventana ventana1 = new ventana();
        ventana1.setVisible(true);
        ventana1.setLocationRelativeTo(null);
    }
}
```

Actividad 2. Menú del programa

Como se mencionó anteriormente, para hacer el menú del programa se decidió hacer mediante una interfaz gráfica (Ver imagen 3) donde los elementos que se emplearon serán descritos a continuación:

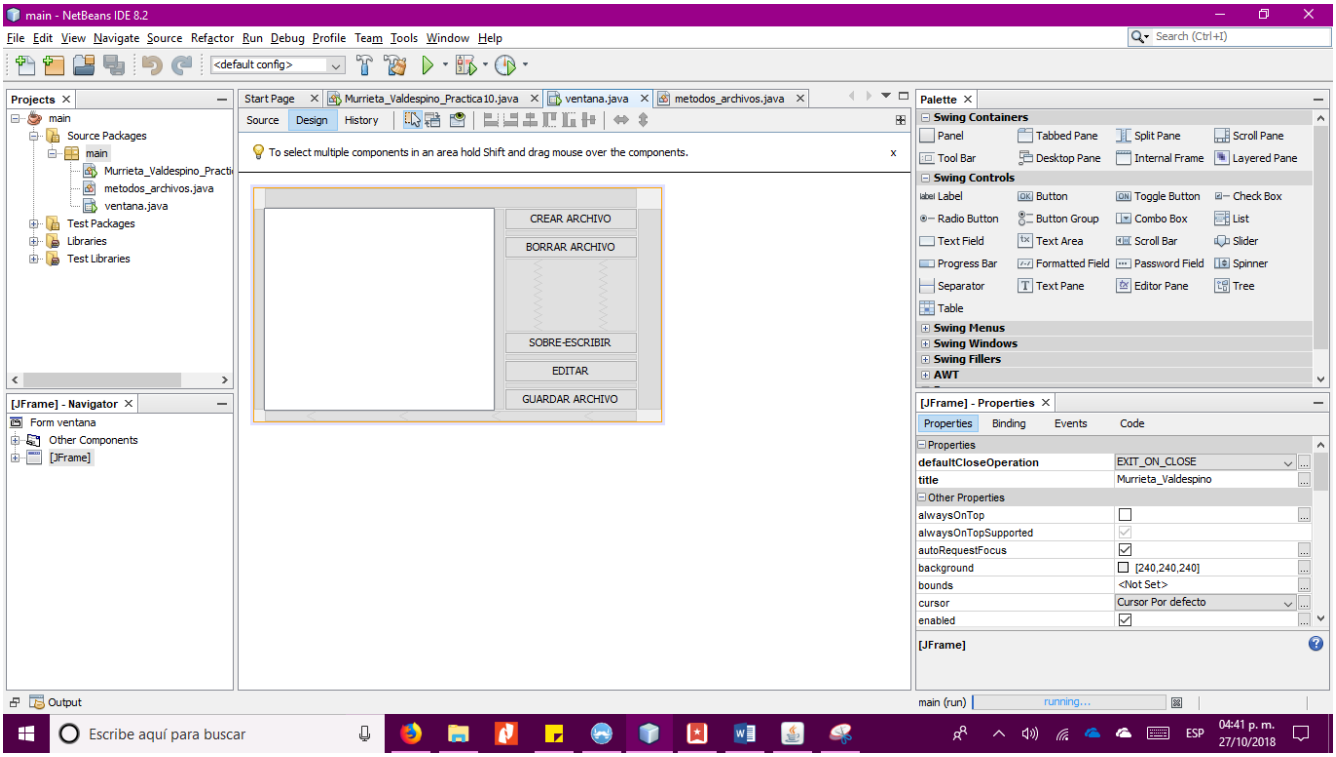

Imagen 3: Captura de pantalla donde se puede la creación y asignación de cada elemento a la ventana principal.

Para desarrollar cada uno de los apartados siguientes se tuvo que designar cada uno a un botón de la ventana, sin embargo, cabe destacar que para el desarrollo del apartado 4 y 5 se necesitó de un campo de texto para poder tanto desplegar el contenido del archivo (El apartado de edición de archivos) y para poder escribir en el archivo (Para el apartado de edición y sobre-escritura).

Actividad 3. Creación de un archivo.

Para este apartado lo primero que se realizó fue la declaración del método encargado al momento de dar click sobre el botón de creación de archivos, dentro de este a través de distintas condiciones if es como se pudo hacer toda la lógica de este apartado.

En el primer if lo que se condiciona es que se debe dar click en el botón de crear archivo para poder hacer todo el proceso, posteriormente lo que se realizó fue que una vez ingresada la cadena donde esta debía tener como últimos elementos "txt" ya que de esta forma se podría crear un archivo de este estilo, ahí es donde entra la segunda condición if si todo lo anterior es correcto se despliega una pantalla mediante "JOptionPane.showMessageDialog()" donde se afirma que se ha creado el archivo, en caso contrario se despliega un mensaje mediante la misma clase y método solamente con el mensaje de que el contenido debe guardarse en un txt.

NOTA: La creación verdadera del archivo se realizó a través de la llamada del método "createFile" el cual es uno de los 3 métodos ubicados en la clase "métodos_archivos", este método "createFile " es del tipo cadena el cual a través de lo que se ingrese es como se determina si se crea o no un archivo (Esto a través del flujo de control de datos mediante try y catch). Dentro de este método lo único que se hace es el uso de la clase FileOutputStream para la creación del archivo.

SALIDA DEL PROGRAMA

A continuación, se muestran las salidas del programa de los distintos casos que se pueden dar:

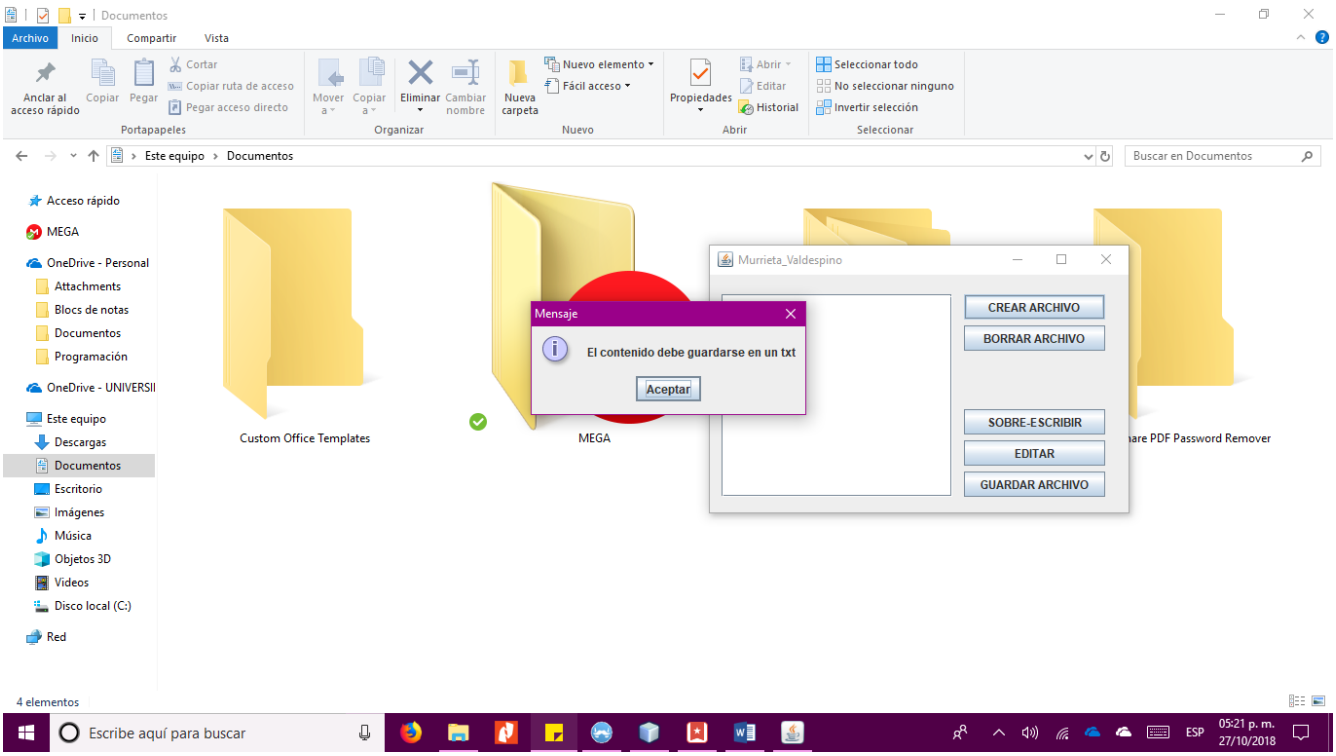

Imagen 4: Captura de pantalla donde se muestra el error en caso de que al crear un archivo no se le coloque la extensión ".txt" es por ello se despliega el mensaje.

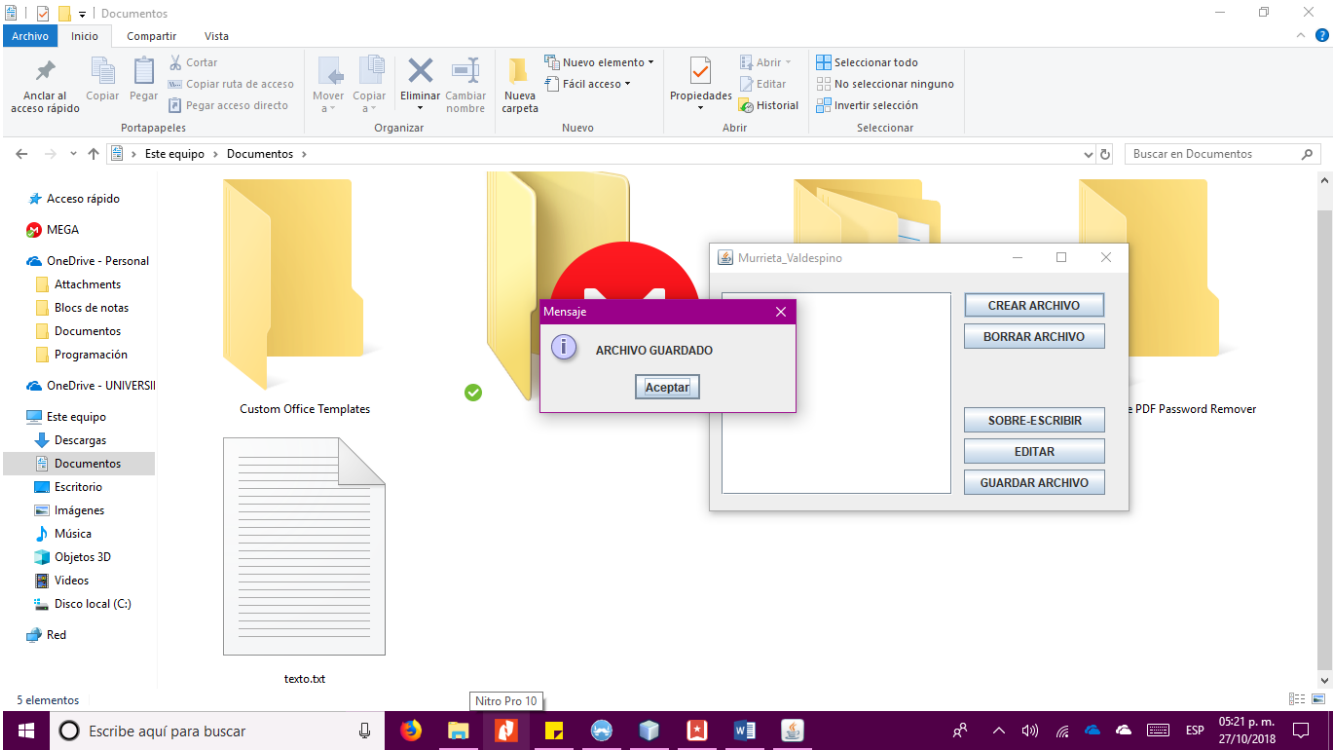

Imagen 5: Captura de pantalla donde se muestra la creación del archivo y el despliegue de una ventana afirmando la creación del archivo.

Actividad 4 /5. Sobre escritura de un archivo y Edición de un archivo

Consideraciones: Para estos 2 apartados lo que se decidió fue realmente partir desde el caso general para de esa forma facilitar la programación de estos casos.

Lo que realmente se realizó fue un apartado (Botón Sobre escribir) dedicado a la lectura de un archivo y de esta forma directamente borrar el contenido de este, para posteriormente desplegar la opción de colocar el nuevo texto en el campo de texto de la ventana, por último, simplemente se debía ingresar en el botón guardar y de esa forma directamente guardar el contenido en el mismo archivo o en otro (Opción del usuario).

Por otro lado, se realizó un apartado (Botón Editar) dedicado a la lectura de un archivo y de esta forma directamente mostrar el contenido de este en el campo de texto de la venta, por último, simplemente se debía ingresar en el botón guardar y de esa forma directamente guardar el contenido en el mismo archivo o en otro (Opción del usuario).

Partiendo de lo anterior, a continuación, se describirán cada uno de los 2 casos a través de los métodos que se utilizaron para llevarlos a cabo.

CASO DE SOBRE ESCRITURA

Para este apartado lo primero que se realizó fue la declaración del método encargado al momento de dar click sobre el botón de sobre escribir.

Dentro de este método lo que se desarrolló fueron los diversos casos que se podían suscitar al sobre escribir un archivo, el primero se dio a través de la primera condición if donde se condición a que se diera en el botón sobre escribir, si era el caso a través nuevamente del método *getSelectedFile* es como se seleccionaría el archivo a editar, posteriormente a través de otra condición y empleando el método *canRead* es como se determinaba si era posible poder escribir o no en el archivo y por último y através de otra condición if donde a través del método *endsWith* es como se determinaba si el archivo era un txt o no, una vez dentro de todos estas condiciones a través del uso de la variable tipo string denominada clear es como se borraba el contenido del archivo (Obviamente empleando el método de la clase métodos_archivos, "saveFile").

NOTA: Realmente lo que aquí se realizó fue un pseudo borrado ya que lo que lo que sucede es que al momento de leer el archivo se le escribe directamente una cadena en blanco o vacía la cual obviamente "borra" el contenido del archivo

Por último, en caso de que no se seleccione un archivo del tipo txt se despliega un mensaje indicando que se seleccione un archivo txt

Respecto al apartado de guardado de contenido realmente lo que se hace es que el contenido que quiera escribir el usuario debe ser colocado en el campo de texto de la venta, una vez que quiera guardar ese contenido debe dar click en el botón Guardar Archivo el cual a continuación será descrito:

El método dedicado al apartado de guardar archivo es prácticamente el mismo que el de crear archivo solamente que dentro de este hay un pequeño apartado que es el dedicado a incrustar el contenido del campo de texto dentro del archivo

Este apartado se llevó a través del método getText el cual directamente obtiene las líneas escritas en el campo de texto y las guarda en una cadena que posteriormente a través del uso del método saveFile es como se lleva a cabo el guardado del contenido.

CASO DE EDICIÓN DE ARCHIVO

Para este apartado lo primero que se realizó fue la declaración del método encargado al momento de dar click sobre el botón de editar archivo.

Dentro de este método lo que se desarrolló fueron los diversos casos que se podían suscitar al sobre escribir un archivo, el primero se dio a través de la primera condición if donde se condición a que se diera en el botón sobre escribir, si era el caso a través nuevamente del método *getSelectedFile* es como se seleccionaría el archivo a editar, posteriormente a través de otra condición y empleando el método *canRead* es como se determinaba si era posible poder escribir o no en el archivo y por último y através de otra condición if donde a través del método *endsWith* es como se determinaba si el archivo era un txt o no, una vez dentro de todos estas condiciones a través del uso de la variable tipo string denominada contenido y mediante el método openFile es como se pudo obtener el contenido del archivo leído.

NOTA: EL método openFile es un método ubicado en la clase metodos_archivos el cual a través de la apertura de un archivo mediante la clase FileInputStream y a través de la conversión de su contenido en bytes a caracteres y posteriormente a una cadena es como se hace la lectura y obtención del contenido de un archivo.

Una vez obtenido el contenido, simplemente a través del método setText es como se hace el despliegue del contenido del archivo en el campo de Texto. Por último, en caso de que no se seleccione un archivo del tipo txt se despliega un mensaje indicando que se seleccione un archivo txt

Respecto al apartado de guardado de contenido realmente lo que se hace es que el contenido que quiera escribir el usuario debe ser colocado en el campo de texto de la venta, una vez que quiera guardar ese contenido debe dar click en el botón Guardar Archivo el cual fue descrito previamente.

SALIDAS DEL PROGRAMA

A continuación, se muestran las salidas del programa de los distintos casos que se pueden dar tanto en la edición como sobre escritura de un archivo:

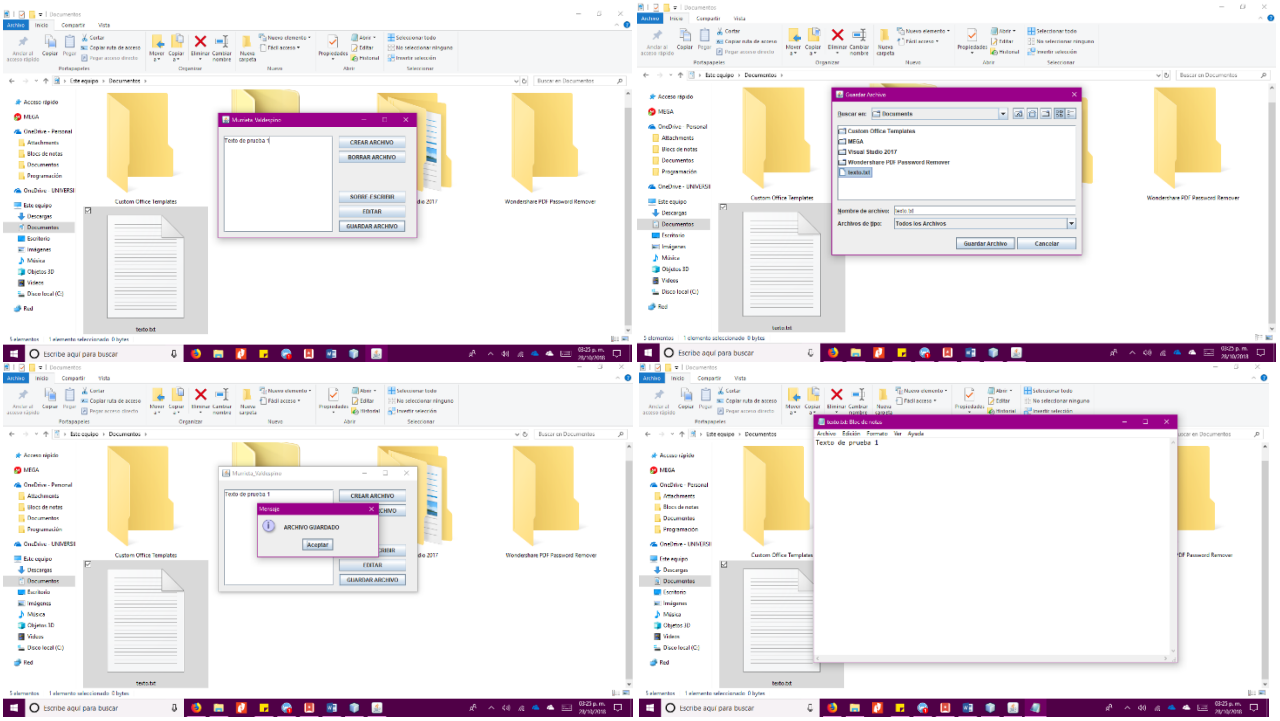

Imagen 6: Conjunto de imágenes donde se muestra como un archivo vacío fue editado mediante la opción de editar archivo, posteriormente en las 2 imágenes de abajo se muestra la salida y pantalla de que se ha realizado el guardado del contenido.

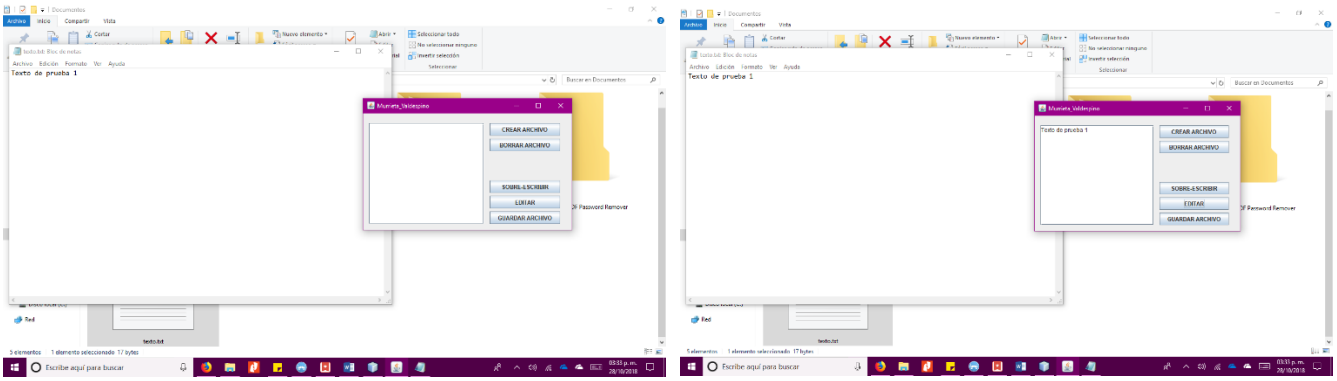

Imagen 7: Conjunto de imágenes donde se muestra la opción de edición de un archivo (En la imagen del lado derecho se aprecia el contenido del archivo).

Actividad 6. Eliminación de un archivo.

Para este apartado lo primero que se realizó fue la declaración del método encargado al momento de dar click sobre el botón de borrar archivos. Dentro de este método lo único que se utilizó fue el método de getSelectedFile para seleccionar el archivo correspondiente y por último el método delete() el cual es el encargado directo de borrar un archivo.

NOTAS: Realmente este apartado es muy sencillo debido a que solamente se emplean métodos tanto de la clase JFileChooser como de la clase JOptionPane, para este apartado realmente no se tuvo que programar más que las condiciones y casos de la creación del archivo.

SALIDA DEL PROGRAMA

A continuación, se muestran las salidas del programa de los distintos casos que se pueden dar:

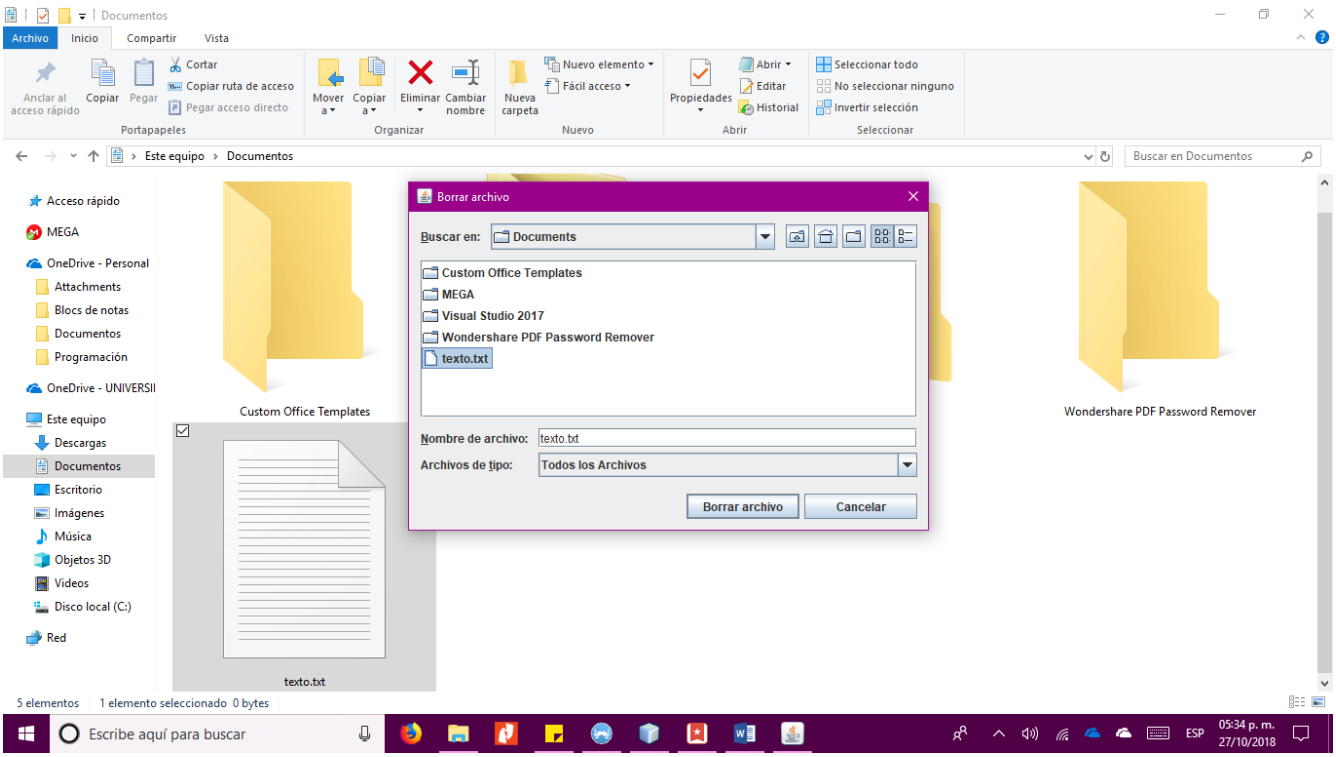

Imagen 8: Captura de pantalla donde se muestra la selección de un archivo para su eliminación

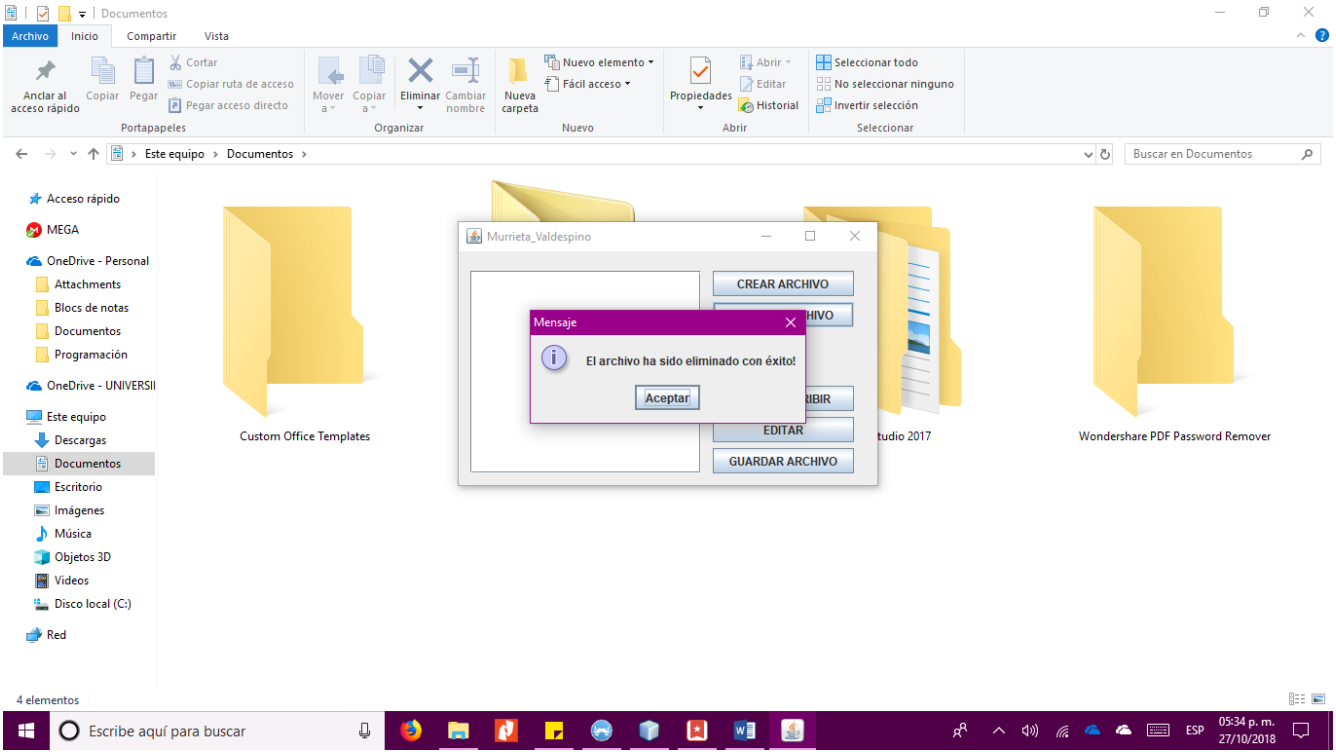

Imagen 9: Captura de pantalla donde se muestra la eliminación del archivo y a su vez el despliegue de que el archivo ha sido eliminado.

CONCLUSIONES

MURRIETA VILLEGAS ALFONSO

El uso de archivos en el siglo actual es sin duda una de las actividades cotidianas más comunes y necesarias, realmente el pensar que un archivo es un tipo de dato abstracto es una de las cosas que nos hace reflexionar acerca de que tanto realmente sabemos acerca de lo que nos rodea o usamos en nuestra vida. Esta práctica además de ser probablemente la más fácil de todo el semestre realmente resultó incluso agradable de hacer sobre todo porque se decidió emplear conceptos y herramientas vistas en otra materia como fue el caso de POO.

Además, y como parte realmente importante, se debe destacar que en verdad el manejo de una interfaz y sobre todo el uso de la clase FileChooser realmente facilitó el desarrollo de cada uno de los apartados para esta práctica sobre todo porque de esta forma se pudo omitir toda la parte tediosa de pedir la dirección al usuario, etc.

Por último, es necesario destacar que el conocer y explorar bibliotecas hechas es realmente necesario para poder desarrollar nuestras prácticas de una forma más fácil sobre todo porque por ejemplo cuando se desarrolló el proyecto nunca empleamos clases como FileInputStream o FileOutputStream y en esta práctica esas dos clases facilitó totalmente el manejo del contenido tanto al momento de leer como de manipular el contenido de los archivos.

VALDESPINO MENDIETA JOAQUÍN

En conclusión se han completado los objetivos, los archivos son parte natural de un programa más grande de los que realizamos comúnmente en las prácticas, ya que una gran ventaja de ellos es que pueden almacenar información a largo plazo sin necesidad de mantener en ejecución un programa, la utilidad de

un archivo depende del objetivo o de la tarea que se desee realizar a si mismo que su operación, como se ha logrado observar hay distintas bibliotecas que manejan archivos, cada una se diferencia en algunos aspectos, como en las operaciones básicas como abrir, leer y escribir, la manera en que se debe codificar para que funcione de manera correcta, cabe recalcar que ya se había usado el manejo de archivos con anterioridad lo que permitió un desarrollo un tanto más práctico y de retroalimentación sobre este tema.

REFERENCIAS

Mark J. Guzdial, Barbara Ericson. (2015). *Introducción a la computación y programación con Python*. 3ra Edición. Madrid España.

Bradley N. Miller y David L. Ranum, Franklin, Beedle & Associates. (2011). Problem Solving with Algorithms and Data Structures using Python. Segunda Edición.

Elba Karen Saenz García. (2017). *Manual de Prácticas del laboratorio de Estructuras de Datos y algoritmos 1I*. UNAM, Facultad de Ingeniería.