



**UNIVERSIDAD
DE LA FRONTERA**

Introducción a la Programación

Taller 10. Manejo de Archivos en JAVA

Octubre 2016



Departamento de
Computación e Informática




El objetivo de este taller es brindar el conocimiento básico para la gestión de archivos en JAVA mediante la clase Files.

Es importante destacar que al ingresar rutas de directorios se deben reemplazar los backslash “\” encontrados en la dirección del directorio.

Ej:

Dirección en Windows:

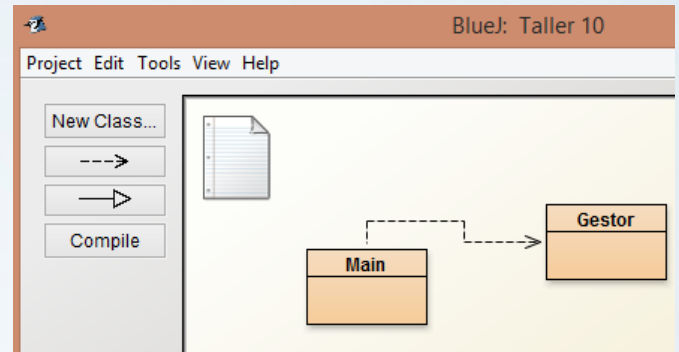
 C:\Taller 10

Dirección en JAVA:

`"C:/Taller 10"`

Actividad 1, “Gestor de Archivos”

En esta actividad crearemos un programa que permite crear nuevas carpetas, así como escribir y leer archivos de texto, para esto usaremos la clase “Files”, la cual posee gran variedad de métodos relacionados al uso de archivos.



Paso 1.1

Primero debemos crear la clase Gestor, la cual contendrá los métodos que necesitaremos. Además importamos las librerías de clases necesaria para manejar archivos, las rutas de directorios y archivos, copiado y manejo de excepciones.

```

import java.util.Scanner;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;
import java.io.IOException;
  
```

Paso 1.2

Luego será necesario crear un menú, el cual nos mostrara las opciones disponibles y nos permitirá seleccionar que acción queremos realizar mediante el uso de un *Scanner*.

```

/**
 * Debe ingresar un numero entero entre 1 y 5
 *
 */
public void menu() {
    Scanner leer = new Scanner(System.in);
    System.out.println("Seleccione la operacion a realizar:");
    System.out.println("1 - Crear directorio");
    System.out.println("2 - Crear archivo de texto");
    System.out.println("3 - Leer archivo de texto");
    System.out.println("4 - Copiar archivo");
    System.out.println("5 - Eliminar archivo");
    int opcion = leer.nextInt();
    seleccion(opcion);
}
  
```

```
private void seleccion(int o) {  
    switch (o) {  
        case 1:  
            crearDirectorio();  
            break;  
        case 2:  
            crearArchivo();  
            break;  
        case 3:  
            leerArchivo();  
            break;  
        case 4:  
            copiarArchivo();  
            break;  
        case 5:  
            eliminarArchivo();  
            break;  
    }  
}
```

Paso 1.3

Ahora crearemos el método selección, el cual a través de un *switch* llamara al método que corresponda según la opción del menú que hayamos escogido y procederemos a crear los métodos necesarios.

```
/**  
 * Los directorios deben ser ingresados de la siguiente forma:  
 *  
 * "C:/Taller 10/carpeta a"  
 *  
 */  
private void crearDirectorio() {  
    Scanner leer = new Scanner(System.in);  
    System.out.println("Ingrese ruta completa del directorio");  
    String ruta = leer.nextLine();  
    Path directorio = Paths.get(ruta);  
    if (verificarDirectorio(directorio)) {  
        System.out.println("El directorio ya existe");  
    } else {  
        try {  
            Files.createDirectories(directorio);  
        } catch (IOException e) {  
            System.out.println("El directorio no pudo ser creado");  
            menu();  
        }  
    }  
    System.out.println("El directorio fue creado");  
}
```

Paso 1.4

El método *crearDirectorio()* es el encargado de crear nuevas carpetas, para esto se debe guardar la ruta del archivo en una variable de tipo *Path*, la cual requiere del método *get(String)* de la clase *Paths* para ser inicializada, posteriormente será usado por el método *createDirectories(Path)* de la clase *Files*,

Paso 1.5

Debido a que el método anterior no puede sobrescribir carpetas, rearemos el método ***verificarDirectorio(Path)***, el cual verificara si el directorio ya existe de modo que el programa no intente volver a crearlo.

```
/**
 * Verifica si el directorio existe
 *
 */
private boolean verificarDirectorio(Path dir) {
    if (Files.exists(dir)) {
        return true;
    } else {
        return false;
    }
}
```

Paso 1.6

A continuación crearemos el método ***crearArchivo()***, al igual que el método anterior es necesario guardar la dirección en una variable ***Path***, el método ***write()*** de la clase Files lee el texto ingresado mediante un arreglo de Bytes, por lo que es necesario transformar el ***String*** a ***Bytes*** mediante el método ***getBytes()***.

```
/**
 * Los archivos deben ser ingresados de la siguiente forma:
 *
 * "C:/Taller 10/ejemplo.txt"
 *
 */
private void crearArchivo() {
    Scanner leer = new Scanner(System.in);
    System.out.println("Ingrese ruta completa del archivo");
    String ruta = leer.nextLine();
    Path archivo = Paths.get(ruta);
    System.out.println("Ingrese texto a guardar en el archivo");
    String texto = leer.nextLine();
    try {
        Files.write(archivo, texto.getBytes());
    } catch (IOException e) {
        System.out.println("El archivo no pudo ser guardado");
        menu();
    }
    System.out.println("Se ha guardado el archivo");
}
```

FINISHED?

```
/**
 * Este metodo leera un archivo de texto
 * e imprimira por consola su contenido.
 *
 */
private void leerArchivo() {
    Scanner leer = new Scanner(System.in);
    System.out.println("Ingrese ruta completa del archivo");
    String ruta = leer.nextLine();
    Path directorio = Paths.get(ruta);
    String texto = "";
    try {
        texto = new String(Files.readAllBytes(directorio));
    } catch (IOException e) {
        System.out.println("El archivo no pudo ser leído");
        menu();
    }
    System.out.println("El contenido del archivo es:\n" + texto);
}
```

Paso 1.7

Este método funciona al inverso del anterior, por lo que devolverá un arreglo de *Bytes* el cual puede transformarse a *String* fácilmente con el uso de *new String()*.

```
/**
 * Ambos archivos deben ingresarse con sus rutas completas incluida la
 * extension
 *
 * StandardCopyOption es un campo opcional, puede ser omitido. este permite
 * reemplazar el archivo de destino en caso de ya existir.
 */
private void copiarArchivo() {
    Scanner leer = new Scanner(System.in);
    System.out.println("Ingrese ruta completa del archivo original");
    String ruta = leer.nextLine();
    Path archivo = Paths.get(ruta);
    System.out.println("Ingrese ruta completa de la copia del archivo");
    String newruta = leer.nextLine();
    Path newarchivo = Paths.get(newruta);
    try {
        Files.copy(archivo, newarchivo, StandardCopyOption.REPLACE_EXISTING);
    } catch (IOException e) {
        System.out.println("El archivo no pudo ser copiado");
        menu();
    }
    System.out.println("El archivo fue copiado exitosamente");
}
```

Paso 1.8

Para este método es necesario leer 2 rutas de archivos, primero la del archivo original y posteriormente la ruta en donde se guardara la copia, ambas rutas deben incluir la extensión del archivo y es posible guardar la copia con un nombre diferente al original.

🖱 Paso 1.9

Al igual que en los métodos anteriores, es necesario leer la ruta mediante una variable *Path*, se debe considerar tanto el que el método *delete(Path)* como el *deleteIfExists(Path)* no pueden borrar archivos que se encuentren en uso, de igual forma no pueden eliminar un directorio si este contiene algún archivo.

```
/**
 * En caso de intentar eliminar un directorio,
 * este dara error a menos que este vacio
 */
private void eliminarArchivo() {
    Scanner leer = new Scanner(System.in);
    System.out.println("Ingrese ruta completa del archivo a eliminar");
    String ruta = leer.nextLine();
    Path archivo = Paths.get(ruta);
    try {
        Files.deleteIfExists(archivo);
    } catch (IOException e) {
        System.out.println("El archivo no pudo ser eliminado");
        menu();
    }
    System.out.println("El archivo fue eliminado exitosamente");
}
```

Actividad 2, Finalizacion del Programa

Para finalizar el programa, cree la clase *Main* y úsela para instanciar la clase *Gestor*.

Adicionalmente modifique el menú para agregar la opción de salida.
Y mediante el uso de métodos *String* valide las entradas de las rutas.



**UNIVERSIDAD
DE LA FRONTERA**

Curso de Introducción a la Programación

Taller 10. Manejo de Archivos en Java

UNIVERSIDAD DE LA FRONTERA

FACULTAD DE INGENIERÍA Y CIENCIAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INFORMÁTICA

Avda Francisco Salazar 01145
Temuco – Chile / casilla 54-D
Fono (56) 45 2325000 /2744219

dci.ufro.cl



Departamento de
Computación e Informática