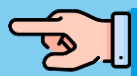
Arreglos y archivos

Escritura y lectura de archivos

{desafío} latam_

Implementar una aplicación básica de consola utilizando las buenas prácticas y convenciones para resolver un problema de baja complejidad acorde al lenguaje Java

- Unidad 1: Flujo, ciclos y métodos
- Unidad 2: Arreglos y archivos
- Unidad 3: Programación orientada a objetos
- Unidad 4: Pruebas unitarias y TDD



Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Comprender la Clase FileWriter y BufferedWriter para el uso de los métodos en la construcción y escritura de archivos.*
- *Construir Clase FileReader a través de su constructor para lectura de archivos.*
- *Aplicar la Clase BufferedReader y sus métodos para lectura de archivos.*

{desafío} latam_

¿Qué entendemos por lectura y escritura de archivos?

{desafío} latam_

/* Estructura de archivos y ficheros */

Para crear un directorio o fichero

Se debe especificar la ruta y nombre como parámetro de entrada

- Importación

```
import java.io.File;
```

- Directorio

```
File directorio = new File("src/carpeta"); // ruta donde quedará mi carpeta
```

- Fichero

```
File archivo = new File("src/carpeta/texto.txt"); // ruta donde quedará mi archivo
```

File es la clase principal para trabajar con archivos

Permite crear archivos y directorios a través de sus métodos

Método `exists()`

- Valida si el directorio o fichero que se creará existe o no dentro del proyecto.
- Retorna ***false*** cuando no existe y ***true*** cuando exista el archivo o fichero.
- Siempre se utiliza en operadores condicionales.

```
directorio.exists()
```

File es la clase principal para trabajar con archivos

Permite crear archivos y directorios a través de sus métodos

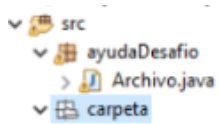
Método `mkdirs()`

- Permite crear los directorios dentro de nuestro proyecto.
- Lo debemos considerar para utilizarlo en condiciones de validaciones.

```
directorio.mkdirs()
```

Importante

Antes de crear un directorio se debe validar si este existe previamente, ya que, si no se valida, su existencia se sobreescribirá en el directorio anterior y esto puede hacer que se pierda toda la información contenida dentro de él.



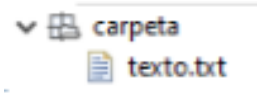
File es la clase principal para trabajar con archivos

Permite crear archivos y directorios a través de sus métodos

Método `createNewFile()`

- Permite crear un archivo físico en nuestro programa.
- Debes considerar siempre que, si el archivo ya está creado y no se realizan las validaciones de preexistencia, este sobrescribirá toda la información sobre el archivo.

```
archivo.createNewFile();
```



Ejercicio guiado

Crear directorio

Crear un directorio usando métodos de la clase File

PASO 1:
Crear un método llamado “crearDirectorio” que recibirá como parámetro el nombre del directorio.

```
import java.io.File;
public static void crearDirectorio(String nombre) {
}
```

PASO 2:
Instanciar la clase File para crear el directorio.

```
public static void crearDirectorio(String nombre) {
File directorio = new File("src/"+nombre);
}
```

Crear directorio

Crear un directorio usando métodos de la clase File

PASO 3:
Se requiere validar que el directorio no existe para poder crearlo. Si el directorio no existe, lo creamos con el método mkdir().

```
public static void crearDirectorio(String nombre) {
     File directorio = new File("src/"+nombre);
     if (directorio.exists() == false) {
          directorio.mkdir();
     }
}
```

Crear directorio

Crear un directorio usando métodos de la clase File

PASO 4:
Llamar al método crearDirectorio dentro del método main, pasándole como parámetro el nombre que le daremos al directorio.

```
public static void main(String[] args) {
     crearDirectorio("directorio");
}
```

Crear directorio

Crear un directorio usando métodos de la clase File

Solución final

```
import java.io.File;
public static void main(String[] args) {
     crearDirectorio("directorio");
}
public static void crearDirectorio(String nombre) {
     File directorio = new File("src/"+nombre);
     if (directorio.exists() == false) {
          directorio.mkdir();
     }
}
```

Ejercicio propuesto

¡Practiquemos!

Ejercicio

- Se solicita crear un método llamado “Crear fichero” que recibe como parámetro de entrada el nombre del fichero y el nombre del directorio.

- Se requiere validar que el directorio no exista para completar la creación del directorio.

/* FileWriter - BufferedWriter */

Clase FileWriter

- Usada para escribir caracteres en archivos.
- Método **`write():`** permite escribir caracteres o strings a un fichero.
- Normalmente está envuelta en objetos Writer de más alto nivel, como *BufferedWriter*.

Crear FileWriter

Para crear un FileWriter, luego de realizar la importación `import java.io.FileWriter;`, se necesita un String como ruta de archivo o una clase File.

Para este caso, y siguiendo la línea del curso, utilizaremos un File:

```
File archivo = new File("src/carpeta/texto.txt");
FileWriter fileW = new FileWriter(archivo);
```

Clase BufferedWriter

- Usada para hacer clases de bajo nivel como FileWriters de una manera más eficiente y más fácil de usar.
- Escriben relativamente grandes cantidades de información en un archivo, lo que minimiza el número de veces que las operaciones de escritura de archivos, que son operaciones más lentas, se llevan a cabo.
- Provee un método llamado **`newLine()`** creando separadores de línea específicos de la plataforma de manera automática.

Crear BufferedWriter

Para crear un FileWriter, luego de realizar la importación `import java.io.BufferedWriter;`, se necesita un objeto FileWriter:

```
File archivo = new File("src/carpeta/texto.txt");
FileWriter fileW = new FileWriter(archivo);
BufferedWriter bufferedWriter = new BufferedWriter(fileW);
```

Para escribir utilizaremos el método write("texto"), donde el texto es lo que se escribirá en el archivo: `bufferedWriter.write("texto");`

Para que se guarde la información debemos cerrar el archivo, esto hace que guarde automáticamente los cambios. Utilizaremos el método **`close()`**:

```
bufferedWriter.close();
```

/* FileReader - BufferedReader*/

/* Las Clases BufferedReader y FileReader nos permitirá leer archivos físicos de grandes volúmenes de información y manejar el trato de estos. Esto nos servirá para trabajar en aplicaciones de grandes empresas o en cualquier aplicativo donde se necesite almacenar archivos físicos */

Clase FileReader

- Usada para leer archivos de caracteres.
- Su método `read()` es usado a bajo nivel, permitiendo leer caracteres de manera singular.

Para crear un FileReader, luego de realizar la importación `import java.io.FileReader;`, se necesita un String como ruta de archivo o una clase File. Para este caso, y siguiendo la línea del curso, utilizaremos un File:

```
File archivo = new File("src/carpeta/fichero.txt");
FileReader fr = new FileReader(archivo);
```

Clase BufferedReader

- Usada para hacer clases Reader de bajo nivel como FileReader, pero de una manera más eficiente y más fácil de usar.
- Leen relativamente grandes cantidades de un archivo a la vez, y mantienen esta información en el buffer (memoria de la Java virtual machine).
- Cómo está precargada hace que la información se cargue en memoria y sea más fácil de leer y manejar.

```
import java.io.FileReader;
import java.io.BufferedReader;

File archivo = new File("src/carpeta/fichero.txt");
FileReader fr = new FileReader(archivo);
BufferedReader br = new BufferedReader (fr);
```

Ejercicio guiado

Crear y escribir en un archivo

Requerimiento

Crear un método llamado **crearFile** el cual creará un directorio físico llamado “miDirectorio”, dentro de este directorio crearemos un archivo llamado **fichero.txt**, en este archivo escribiremos texto.

Crear y escribir en un archivo

PASO 1:
Crear el directorio.

```
import java.io.*;
public static void crearFile()} // comienzo del método
     File directorio = new File("src/carpeta");
     directorio.mkdirs();
```

Crear y escribir en un archivo

PASO 2:
Crear el fichero o archivo.
Creamos el objeto File llamado objeto con la ruta.

```
File archivo = new File("src/carpeta/texto.txt");
archivo.createNewFile();
```

Crear y escribir en un archivo

PASO 3:
Crear el FileWriter y BufferedWriter.

- Creamos el objeto FileWriter con un Archivo File.
- Creamos el objeto BufferedWriter con un Archivo FileWriter

```
FileWriter fileW = new FileWriter(archivo);
BufferedWriter bufferedWriter = new BufferedWriter(fileW);
```

Crear y escribir en un archivo

PASO 4:
Escribir y cerrar el archivo.

- Utilizamos el método write para escribir en el archivo.
- Hacemos un salto de línea con el método newLine().
- Cerramos el archivo con el método close().

```
bufferedWriter.write("texto 1");
bufferedWriter.write("texto 2");
bufferedWriter.newLine();
bufferedWriter.close();
} // cierre del método
```

¿Siempre es necesario especificar la ruta para la creación de un archivo o fichero?

¿Para qué sirve el método `mkdir()`?

Próxima sesión...

- *Aplicar los métodos de la clase File.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales