educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 1 - Java Analyst

Módulo 10

educación IT
CODE YOUR FUTURE

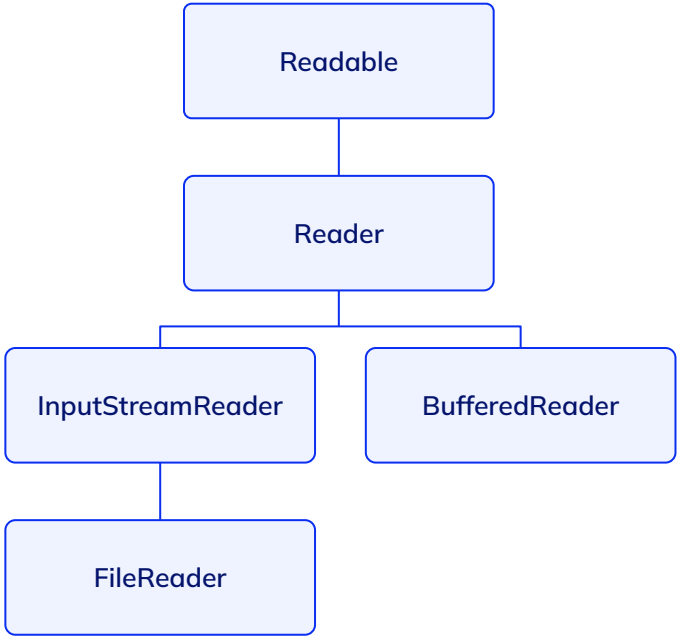
Java IO

Readable

La interfaz **`Readable`** proporciona el **método de lectura.**

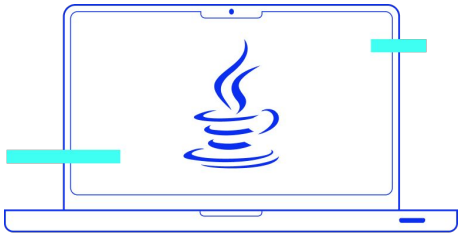
La clase abstracta **`Reader`** proporciona una implementación esquelética de la interfaz `Readable` y, simplemente, agrega implementaciones para los métodos **`read`** y **`close`.**

La clase **`InputStreamReader`** es un puente entre los flujos de bytes y los flujos de caracteres: **lee bytes y los decodifica** en caracteres en su representación **ASCII** a través de un número entero que proporciona la clase **`FileReader`**.



Métodos

Tipo	Método	Descripción
int	`getEncoding()`	Devuelve el nombre de la codificación de caracteres que utiliza esta secuencia.
void	`close()`	Cierra este flujo de entrada y libera los recursos del sistema asociados con el flujo.
int	`read()`	Lee el siguiente byte de datos del flujo de entrada.



File reader

```
try (FileReader leerFichero = new FileReader(archivoLectura)) {
    int c;

    while ((c = leerFichero.read()) != -1) {
        char caracter = (char) c;
        System.out.print(caracter);
    }

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

Nuestra oferta de formación se encuentra orientada al desarrollo profesional buscando potenciar las habilidades personales de los alumnos. Nos adaptamos rápidamente a los cambios que exigen las tecnologías de IT, para generar constantemente productos de valor, novedosos y de alta calidad.

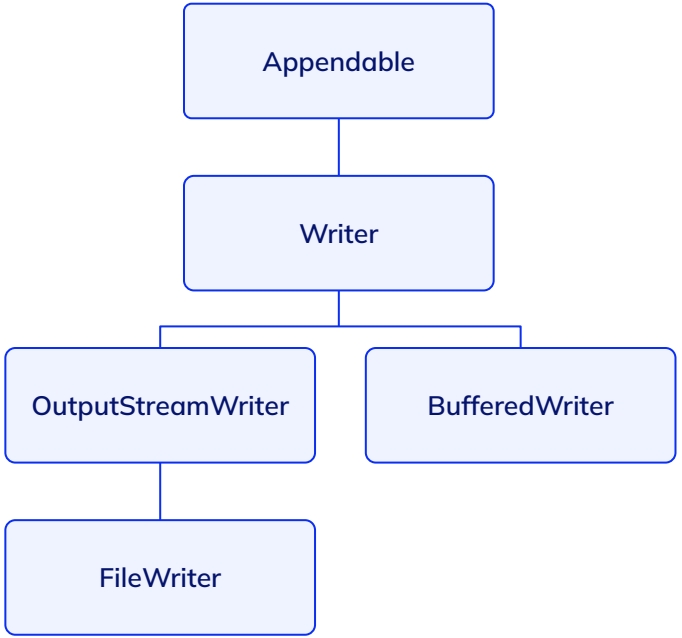
Conocemos el "inside tecnológico" de las compañías y sabemos qué tecnologías y metodologías se ajustan mejor a las necesidades del mercado. Las transmitimos con un alto nivel académico y un fuerte enfoque en la rápida adaptación del alumno al ambiente laboral.

Appendable

La interfaz **`Appendable`** proporciona el **método de escritura.**

La clase abstracta **`Writer`** proporciona una implementación esquelética de la interfaz **`Appendable`** y, simplemente, agrega implementaciones para los métodos **`write`** y **`close`**.

La clase **`OutputStreamWriter`** funciona como un puente entre los flujos de bytes, para **escribir los datos como caracteres nos proporcionan la clase `FileWriter`.**



Métodos

Tipo	Método	Descripción
void	`close()`	Cierra este flujo de entrada y libera los recursos del sistema asociados con el flujo.
void	`write(int b)`	Escribe el byte especificado del flujo de salida.

File writer

```
try (FileWriter escribirFichero = new FileWriter(archivoEscritura)) {

    for (int i = 0; i < PARRAFO.length(); i++) {
        escribirFichero.write(PARRAFO.charAt(i));
    }

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

Búfer

Es un espacio de memoria en el que se **almacenan datos de manera temporal,** normalmente para un único uso. Se utiliza principalmente para evitar que el programa o recurso que los requiere se quede sin datos durante una transferencia de datos.

BufferedInputStream

```
try (BufferedInputStream archivoBinario = new BufferedInputStream(new FileInputStream(fichero))) {
    byte[] bytesArchivo = archivoBinario.readAllBytes();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

BufferedOutputStream

```
try (BufferedOutputStream archivoBinario = new BufferedOutputStream(new FileOutputStream(fichero))) {
    archivoBinario.write(bytes);
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

BufferedReader

Lee texto de un flujo de entrada de caracteres, almacena caracteres en el búfer para proporcionar una lectura eficiente de caracteres:

Tipo	Método	Descripción
void	`close()`	Cierra este flujo de entrada y libera los recursos del sistema asociados con el flujo.
Int	`read()`	Lee el siguiente byte de datos del flujo de entrada.
String	`readLine()`	Lee una línea de texto.
boolean	`ready()`	Indica si el flujo está listo para leerse.

Ejemplo

```
try (BufferedReader leerFichero = new BufferedReader(new FileReader(archivoLectura))) {
    String mensaje = null;

    if (leerFichero.ready()) {
        while ((mensaje = leerFichero.readLine()) != null) {
            System.out.println(mensaje);
        }
    }

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

Nuestra oferta de formación se encuentra orientada al desarrollo profesional buscando potenciar las habilidades personales de los alumnos. Nos adaptamos rápidamente a los cambios que exigen las tecnologías de IT, para generar constantemente productos de valor, novedosos y de alta calidad.

Conocemos el "inside tecnológico" de las compañías y sabemos qué tecnologías y metodologías se ajustan mejor a las necesidades del mercado. Las transmitimos con un alto nivel académico y un fuerte enfoque en la rápida adaptación del alumno al ambiente laboral.

BufferedWriter

Escribe texto de un flujo de entrada de caracteres, almacena caracteres en búfer para proporcionar una escritura eficiente.

Tipo	Método	Descripción
void	`close()`	Cierra este flujo de entrada y libera los recursos del sistema asociados con el flujo.
void	`write(int b)`	Escribe el byte especificado del flujo de salida.
void	`write(String s)`	Escribe una cadena de caracteres
void	`newLine()`	Escribe un salto de línea.

Ejemplo

```
try (BufferedWriter escribirFichero = new BufferedWriter(new FileWriter(archivoEscritura))) {

    for (String linea : PARRAFO) {
        escribirFichero.write(linea);
        escribirFichero.newLine();
    }

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

¡Sigamos trabajando!