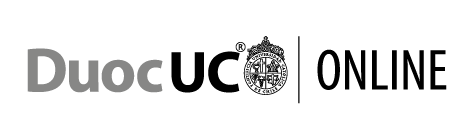
Rectángulo: esquinas redondeadas 1, Forma

Guía de aprendizaje

Entradas y Salidas (I/O) y Manejo de Archivos

Exp 2 – Semana 6

Desarrollo Orientado a Objetos I (PRY2202)

**Facilitador disciplinar:** Alberto Campos

**Asesor par:** Verónica Tramenelli

# Índice

[Introducción a la semana 3](#_Toc158628123)

[Resultado de aprendizaje 4](#_Toc158628124)

[Conceptos relevantes 4](#_Toc158628125)

[Preguntas activadoras 5](#_Toc158628126)

[Actividad 5](#_Toc158628127)

[Entradas y Salidas (I/O) y Manejo de Archivos 6](#_Toc158628128)

[Integración de archivos externos 8](#_Toc158628129)

[Tipos de archivos externos en Java y sus formatos 11](#_Toc158628130)

[Configuración del entorno de desarrollo en java 13](#_Toc158628131)

[Almacenar los datos utilizados en un archivo externo 15](#_Toc158628132)

[Importación de bibliotecas para manejar diferentes formatos de archivos en Java 18](#_Toc158628133)

[Lectura de archivos CSV en Java 23](#_Toc158628135)

[Lectura de un archivo XML utilizando la API DOM o la biblioteca DOM4J en Java 26](#_Toc158628136)

[Lectura de un archivo JSON 34](#_Toc158628137)

[Cómo combinar los diferentes métodos de lectura de archivos para integrar archivos externos en una aplicación Java 37](#_Toc158628138)

[Videos 40](#_Toc158628139)

[Dato 40](#_Toc158628140)

[Cierre de la semana 41](#_Toc158628141)

[Referencias 42](#_Toc158628142)

[Lecturas de la semana 42](#_Toc158628143)

[Apuntes 43](#_Toc158628144)

# Introducción a la semana

En esta guía, aprenderás sobre los conceptos de Entradas y Salidas (I/O), el Manejo de Archivos y la integración de archivos externos en tus proyectos.

Además, conocerás los diferentes tipos de archivos externos en Java, sus formatos y cómo configurar de manera efectiva tu entorno de desarrollo para maximizar tu eficiencia. Desde almacenar datos en archivos externos hasta la importación de bibliotecas para manejar diversos formatos, cada paso te acercará más a comprender la complejidad y la versatilidad de Java en la gestión de archivos.

También aprenderás a leer archivos CSV, a manipular archivos XML utilizando API DOM o la biblioteca DOM4J, y lo más importante: cómo combinar estos métodos para integrar archivos externos en aplicaciones Java de manera sinérgica y poderosa.

# Resultado de aprendizaje

El estudiante será capaz de:

**RA2.** Desarrolla programas Java aplicando conceptos avanzados de la programación orientada a objetos para dar respuesta a requerimientos previamente definidos.

**Indicador de logro:**

**IL4.** Establece medios de control para los eventos del usuario mediante la utilización de excepciones en un programa Java.

**IL5.** Utiliza diversas colecciones que permitan el almacenamiento, la organización y gestión de datos y objetos en un programa Java.

**IL6.** Integra archivos externos de distintos formatos en el desarrollo de un programa Java.

# Conceptos relevantes

|  |  |  |
| --- | --- | --- |
| Archivo TXT | Biblioteca o Libreria | FileWriter |
| Archivo CSV | JDK | BufferedReader |
| Archivo XML | FileReader | BufferedWriter |

# Preguntas activadoras

* ¿Cómo se obtiene la lista de archivos de un directorio en Java?
* ¿Qué clases usan más para trabajar con archivos en Java?
* ¿Qué es el objeto ‘Runtime’ en Java y para qué se utiliza?

# Actividad

### Descripción de la actividad

En esta Sexta semana realizarás una actividad sumativa individual por encargo llamada "Aplicando Entradas y Salidas (I/O) y Manejo de Archivos". A partir del proyecto iniciado en la semana 4, debes demostrar el control y validación de todos los requerimientos funcionales, no funcionales y reglas de negocio (Try/Catch), a través de la integración de archivos externos de distintos formatos y operaciones Input-Output. Además, deberás definir el mejor tipo de colección y sus funciones para el caso planteado, así también poder almacenar los datos utilizados en un archivo externo para utilizarlo en una nueva ejecución de su programa.

# Entradas y Salidas (I/O) y Manejo de Archivos

En Java, la Entrada y Salida (I/O) se refiere a cómo se manejan las operaciones de lectura y escritura de datos desde y hacia diferentes fuentes, como archivos, consolas, redes, entre otros. El paquete java.io proporciona clases y métodos para realizar operaciones de entrada y salida (I/O) de datos.

Estas clases se utilizan para trabajar con flujos de datos, archivos, entrada desde el teclado, salida a la consola, y más. El paquete java.io es fundamental para interactuar con archivos y realizar operaciones de entrada/salida en Java.

### Principales clases y conceptos en Java I/O:

1. Clases de flujo de datos (‘Streams’ o ‘InputStream’ y ‘OutputStream’ – I/O): representan secuencias de datos para lectura y escritura de bytes.
2. Clases de flujo de caracteres (‘Readers’ o ‘Reader’ y ‘Writers’ o ‘Writer’): proporcionan operaciones de lectura y escritura de caracteres, encapsulando los streams de bytes en caracteres.
3. Clases para archivos (‘File’, ‘FileReader’, ‘FileWriter’, ‘FileInputStream’, FileOutputStream’): permiten interactuar con archivos en el sistema de archivos.
4. Clases de buffer (‘BufferedReader’, ‘BufferedWriter’, ‘BufferedInputStream’, ‘BufferedOutputStream’): mejoran la eficiencia de lectura y escritura al almacenar datos en memoria intermedia (buffer) antes de interactuar con el flujo subyacente.
5. Clases para la manipulación de objetos (‘ObjectInputStream’, ‘ObjectOutputStream’): permiten la lectura y escritura de objetos Java completos, facilitando la persistencia de datos y la comunicación entre procesos.
6. Clases de flujo de datos (‘InputStream’ y ‘OutputStream’): Son las clases base para los flujos de entrada y salida de bytes respectivamente. Proporcionan métodos para leer y escribir datos en forma de bytes.

Estas clases proporcionan una amplia gama de funcionalidades para realizar operaciones de entrada y salida de datos en Java, ya sea desde archivos, flujos de red, flujos de memoria o cualquier otra **fuente/salida** de datos. Además, el paquete **java.io** es fundamental para trabajar con archivos y flujos de datos en aplicaciones Java.

Figura 1

*Cómo crear un archivo, escribir datos en él y luego leer su contenido*

*Nota*. Ejemplo de cómo crear un archivo, escribir datos en él y luego leer su contenido. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En el ejemplo, se utilizan las clases ‘FileReader’, ‘FileWriter’, ufferedReader’, y ‘IOException’ para manejar posibles excepciones durante las operaciones de lectura y escritura de archivos.

## Integración de archivos externos

La integración de archivos externos en el desarrollo de aplicaciones Java es de gran importancia pues permite acceder y utilizar recursos y funcionalidades que se encuentran fuera del código fuente de la aplicación, lo que facilita la reutilización de código, mejora la modularidad y promueve la separación de responsabilidades.

En Java, la integración de archivos externos en el desarrollo de aplicaciones es fundamental para manipular datos, configuraciones y recursos. A continuación, veamos cómo se integran y manejan comúnmente los archivos externos en aplicaciones Java:

### Lectura y escritura de archivos:

Java proporciona clases en el paquete ‘java.io’ y ‘java.nio.file’ para leer y escribir archivos. Puedes utilizar estas clases para abrir, leer, escribir y cerrar archivos de diferentes tipos, como archivos de texto, binarios o de configuración.

Figura 2

*Lectura y escritura de archivos de texto*

*Nota*. Lectura y escritura de archivos de texto. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

### Carga de recursos:

Java también permite cargar recursos como imágenes, archivos de configuración, archivos de propiedades, etc., desde el sistema de archivos o desde el ‘classpath’ de la aplicación. Esto es útil para recursos que forman parte de la aplicación y que deben ser accesibles durante su ejecución.

Figura 3

*Carga de un archivo de configuración*

*Nota*. Carga de un archivo de configuración. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

### Utilización de bases de datos:

Además de trabajar con archivos, las aplicaciones Java pueden integrarse con bases de datos utilizando JDBC (Java Database Connectivity) para realizar operaciones CRUD (‘Create’, ‘Read’, ‘Update’, ‘Delete’) en bases de datos relacionales.

### Consideraciones de manejo de errores:

Es importante manejar las excepciones (‘IOException’, ‘FileNotFoundException’, etc.) correctamente al manipular archivos en Java para garantizar un manejo adecuado de recursos y una aplicación más robusta.

La integración de archivos externos en aplicaciones Java requiere un manejo cuidadoso de recursos, seguridad y manejo de errores para garantizar un funcionamiento confiable y seguro de la aplicación.

## Tipos de archivos externos en Java y sus formatos

En Java, puedes trabajar con varios tipos de archivos externos que contienen datos en distintos formatos. Aquí tienes una lista de algunos tipos comunes de archivos externos y sus formatos asociados:

1. **Archivo CSV (Comma-Separated Values):** almacena datos tabulares en forma de texto plano, donde cada línea corresponde a una fila y los campos están separados por comas.
2. **Archivo JSON (JavaScript Object Notation):** almacena datos estructurados en formato legible por humanos utilizando la sintaxis de JavaScript, utilizado para representar objetos y estructuras de datos.
3. **Archivo XML (eXtensible Markup Language):** almacena datos estructurados en un formato legible por humanos utilizando etiquetas jerárquicas, comúnmente utilizado para intercambio de datos entre diferentes sistemas.
4. **Archivos Binarios:** almacenan datos en un formato binario, que no es legible directamente por humanos y generalmente se utilizan para almacenar datos complejos o estructuras específicas de manera eficiente.
5. **Archivos de Bases de Datos (SQLite, MySQL, etc.):** almacenan datos en estructuras de bases de datos, permitiendo realizar operaciones CRUD sobre los datos.

Cada tipo de archivo tiene su propio formato específico y su estructura de datos asociada. En Java, puedes utilizar las clases del paquete **java.io** y otras bibliotecas (como **Gson** para JSON, **Jackson** para XML, **OpenCSV** para CSV, entre otros) para leer, escribir y manipular datos en estos diferentes formatos de archivo externo.

## Configuración del entorno de desarrollo en java

La configuración del entorno de desarrollo en Java implica preparar el entorno para escribir, compilar, depurar y ejecutar aplicaciones Java. Aquí hay pasos básicos para configurar un entorno de desarrollo:

|  |  |
| --- | --- |
| **PASO 1** | **Instala el Kit de Desarrollo de Java (JDK):** descarga e instala el JDK adecuado para tu sistema operativo desde el sitio web de Oracle o de la fuente oficial de distribución de JDK. Asegúrate de configurar las variables de entorno JAVA\_HOME y PATH para apuntar al directorio de instalación del JDK. |
| **PASO 2** | **Instala un Entorno de Desarrollo Integrado (IDE):** puedes elegir entre varias opciones de IDE:   * Eclipse: un IDE popular con una amplia gama de características y soporte para Java. * IntelliJ IDEA: ofrece una experiencia de desarrollo Java robusta y avanzada. * NetBeans: un entorno de desarrollo que soporta múltiples lenguajes además de Java.   Descarga e instala el IDE que prefieras y configúralo según tus preferencias. |
| **PASO 3** | **Configura el IDE:**   * Configuración del JDK en el IDE: asegúrate de que el IDE esté configurado para utilizar el JDK que has instalado. * Configuración del proyecto: define la estructura del proyecto, establece las rutas de compilación y los ajustes específicos del proyecto. |
| **PASO 4** | **Configura las dependencias:** si estás utilizando dependencias externas o bibliotecas, puedes gestionarlas con herramientas como Maven o Gradle. Configura los archivos de configuración (pom.xml en Maven, build.gradle en Gradle) para incluir las dependencias necesarias. |
| **PASO 5** | **Realiza pruebas y ajustes:**   * Configuración del depurador: aprende a usar las herramientas de depuración del IDE para encontrar errores y depurar tu código. * Configuración del sistema de compilación: familiarízate con cómo compilar y ejecutar tus aplicaciones Java dentro del IDE. |
| **PASO 6** | **Actualizaciones y mantenimiento:** mantén actualizado tu entorno de desarrollo, incluyendo el JDK, el IDE y cualquier otra herramienta que utilices. Instala las actualizaciones de seguridad y parches según sea necesario. |

Siguiendo estos pasos, podrás configurar un entorno de desarrollo Java sólido que te permitirá escribir, depurar y ejecutar aplicaciones Java de manera efectiva y eficiente. Además, la configuración puede variar ligeramente dependiendo del IDE y las herramientas específicas que utilices para tu desarrollo.

## Almacenar los datos utilizados en un archivo externo

En Java, existen dos formas principales de almacenar archivos externos:

|  |  |
| --- | --- |
| **Utilizando la clase ‘File’** | **Utilizando la clase ‘InputStream’ y ‘OutputStream’** |
| La clase ‘File’ proporciona una representación de un archivo en el sistema de archivos. Se puede utilizar para crear, eliminar, mover, y leer o escribir datos en un archivo. | Las clases ‘InputStream’ y ‘OutputStream’ proporcionan una interfaz para leer y escribir datos en un flujo. Se pueden utilizar para leer o escribir datos en un archivo de forma secuencial. |

### Almacenamiento de archivos utilizando la clase ‘File’

Para almacenar archivos externos utilizando la clase ‘File’, se pueden utilizar los siguientes métodos:

* ‘createNewFile()’: crea un nuevo archivo si no existe.
* ‘delete()’: elimina un archivo existente.
* ‘renameTo()’: renombra un archivo.
* ‘exists()’: comprueba si existe un archivo.
* ‘length()’: devuelve la longitud de un archivo.
* ‘canRead()’: comprueba si se puede leer un archivo.
* ‘canWrite()’: comprueba si se puede escribir en un archivo.

Figura 4

*Cómo crear un nuevo archivo de texto*

*Nota*. Ejemplo de cómo crear un nuevo archivo de texto. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Este código creará un nuevo archivo de texto llamado **"myfile.txt"** en el directorio actual.

### Almacenamiento de archivos utilizando las clases ‘InputStream’ y ‘OutputStream’

Para almacenar archivos externos utilizando las clases **‘InputStream’** y ‘**OutputStream’,** se pueden utilizar los siguientes métodos:

* ‘read()’: lee un byte del flujo.
* ‘read(byte[] b)’: lee un número de bytes especificado en el array b.
* ‘write(byte[] b)’: escribe un número de bytes especificado en el array b en el flujo.
* ‘close()’: cierra el flujo.

Por ejemplo, el siguiente código muestra cómo escribir datos en un archivo de texto:

Figura 5

*Ejemplo de cómo escribir datos en un archivo de texto*

*Nota*. Cómo escribir datos en un archivo de texto. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Este código escribirá los datos "Estos son datos para almacenar en un archivo de texto” en el archivo **‘myfile.txt’.**

### Elección de la forma de almacenamiento

La forma de almacenamiento de archivos externos que se elija dependerá de las necesidades específicas de la aplicación.

* Si se necesita un control completo sobre el archivo, como, por ejemplo, la posibilidad de crear, eliminar, mover, o cambiar el tamaño del archivo, se recomienda utilizar la clase ‘File’.
* Si se necesita leer o escribir datos en un archivo de forma secuencial, se recomienda utilizar las clases ‘InputStream’ y ‘OutputStream’.

## Importación de bibliotecas para manejar diferentes formatos de archivos en Java

Para manejar diferentes formatos de archivos en Java, es necesario importar las bibliotecas adecuadas. A continuación, se muestra cómo importar algunas bibliotecas comunes para trabajar con diferentes formatos de archivos:

* **Archivos de texto:** Puedes importar la clase ‘java.io.BufferedReader’ para leer archivos de texto línea por línea, y la clase ‘java.io.’ para escribir en archivos de texto.

Figura 6

*Cómo importar archivos de texto*

*Nota*. Ejemplo de cómo importar archivos de texto. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

* **Archivos CSV (valores separados por comas):** puedes importar la biblioteca ‘opencsv’. A continuación, te mostraremos cómo importar la clase ‘com.opencsv.CSVReader’ para leer archivos CSV.

Figura 7

*Cómo importar archivos CSV*

*Nota*. Ejemplo de cómo importar archivos CSV. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

* **Archivos JSON:** puedes importar la biblioteca ‘org.json’. Veamos un ejemplo de cómo importar la clase ‘org.json.JSONObject’ para manejar objetos JSON.

Figura 8

*Cómo importar archivos JSON*

*Nota*. Ejemplo de cómo importar archivos JSON. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

* **Archivos XML:** puedes importar la biblioteca ‘javax.xml.parsers’. Veamos un ejemplo de cómo importar la clase ‘javax.xml.parsers.DocumentBuilder’ para analizar archivos XML.

Figura 9

*Cómo importar archivos XML*

*Nota*. Ejemplo de cómo importar archivos XML The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Importante

Recuerda que estas son solo algunas de las bibliotecas comunes utilizadas para manejar diferentes formatos de archivos en Java. Dependiendo de tus necesidades específicas, es posible que necesites importar otras bibliotecas adicionales.

Para **importar las bibliotecas** necesarias para manejar diferentes formatos de archivos en Java, puedes utilizar las siguientes instrucciones de importación:

Figura 10

*Instrucciones de importación*

*Nota*. Muestra de instrucciones de importación. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Estas importaciones te permitirán utilizar las clases y métodos necesarios para trabajar con archivos en Java. Además, puedes utilizar estas bibliotecas para leer y escribir archivos en diferentes formatos. Por ejemplo, para leer un archivo de texto, puedes utilizar la clase ‘Scanner’ y la clase ´FileReader’ de la siguiente manera:

Figura 11

*Uso de ‘Scanner’ y ´FileReader’ para leer un archivo de texto*

*Nota*. Ejemplo de uso de ‘scanner’ y ´FileReader’ para leer un archivo de texto. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Y para escribir en un archivo de texto, puedes utilizar la clase ‘FileWriter’ de la siguiente manera:

Figura 12

*Cómo escribir en un archivo de texto usando la clase ‘FileWriter’*

*Nota*. Ejemplo de cómo escribir en un archivo de texto usando la clase ‘FileWriter’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Recuerda reemplazar **"archivo.txt"** con la ruta y el nombre del archivo que deseas leer o escribir. Además, ten en cuenta que es importante manejar las excepciones que puedan ocurrir al trabajar con archivos, por lo que se utiliza un bloque ‘try-catch’ para capturar y manejar cualquier excepción que pueda ocurrir.

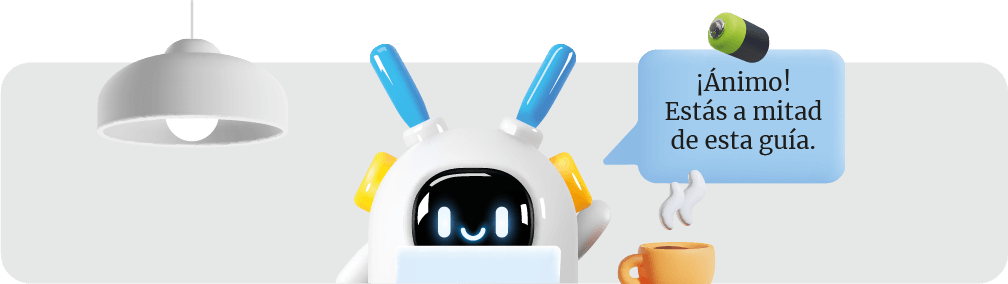
# Links de Interés

* A continuación, te invitamos a leer el texto “Gestión de archivos en Java” escrito por Javier Morales, donde podrás profundizar sobre este tema. Visita este contenido en el siguiente enlace:

<http://jmoral.es/blog/IO-java>

* Además, te recomendamos estudiar el documento “Entrada y salida en Java”. Ficheros” de Jorge Sánchez. Descárgalo en el siguiente enlace:

https://jorgesanchez.net/manuales/viejos/fpr/fpr1009.pdf



## Lectura de archivos CSV en Java

Para leer archivos Comma Separated Values (CSV o en español, valores separados por comas) en Java, existen varias opciones y bibliotecas disponibles. Una de las bibliotecas más populares es ‘OpenCSV’, que proporciona una forma sencilla de leer y escribir archivos CSV en Java. Los archivos CSV son un formato comúnmente utilizado para almacenar datos tabulares en texto plano, donde cada línea del archivo representa una fila y los valores están separados por comas (u otro delimitador, como punto y coma).

La lectura de archivos CSV en Java es una tarea común en el desarrollo de aplicaciones que manejan datos tabulares. ‘OpenCSV’, como mencioné anteriormente, es una biblioteca popular que facilita la lectura de archivos CSV en Java.

Los pasos básicos para leer un archivo CSV utilizando ‘OpenCSV’ en Java son:

1. **Agrega la dependencia:**

Asegúrate de incluir la biblioteca ‘OpenCSV’ en tu proyecto. Puedes hacerlo mediante una herramienta de gestión de dependencias como Maven o Gradle, o simplemente descargando el archivo JAR y añadiéndolo al classpath de tu proyecto.

Figura 13

*Ejemplo de Maven*

*Nota*. Ejemplo del uso de Maven. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Código para la lectura de archivos CSV:**

Aquí tienes un ejemplo básico de cómo puedes utilizar OpenCSV para leer un archivo CSV:

Figura 14

*Uso de ‘OpenCSV’*

*Nota*. OpenCSV. The Apache Software Foundation. (2023). Apache NetBeans (16). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, el código utiliza un ‘CSVReader’ para leer el archivo CSV línea por línea. Cada línea se almacena en un array (‘nextLine’) donde cada elemento del array representa un valor en esa fila.

1. **Manejo de excepciones:**

Aquí tienes un ejemplo básico de cómo puedes utilizar ‘OpenCSV’ para leer un archivo CSV línea por línea:

Figura 15

*Uso de ‘OpenCSV’ para leer un archivo CSV*

*Nota*. Ejemplo de ‘OpenCSV’ para leer un archivo CSV. The Apache Software Foundation. (2023). Apache NetBeans (16). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, se utiliza la clase ‘CSVReader’ de la biblioteca ‘OpenCSV’ para leer el archivo CSV línea por línea. Cada línea se almacena en un arreglo de cadenas ‘(String[])’, donde cada elemento del arreglo representa un valor en el archivo CSV.

Ten en cuenta que se utiliza un bloque ‘try-with-resources’ para garantizar que el recurso (‘CSVReader’) se cierre correctamente al finalizar la lectura. Además, se maneja la excepción ‘IOException’ que puede ocurrir durante la lectura del archivo.

## Lectura de un archivo XML utilizando la API DOM o la biblioteca DOM4J en Java



Para leer archivos **XML** en Java, puede utilizar la **API DOM** **(Document Object Model)** o la biblioteca **DOM4J**. A continuación, te mostraré cómo leer un archivo XML utilizando ambas opciones:

Los pasos básicos para leer un archivo XML que utiliza la API DOM en Java son:

1. **Crear un DocumentBuilder:**

Utiliza la fábrica ‘DocumentBuilderFactory’ para obtener un ‘DocumentBuilder’. Este objeto será responsable de parsear el archivo XML y construir la representación en memoria del documento XML.

Figura 16

*Creación de ‘DocumentBuilder’*

*Nota*. Ejemplo de ‘DocumentBuilder’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Parsear el archivo XML:**

Utiliza el ‘DocumentBuilder’ para parsear el archivo XML y obtener un objeto ‘Document’, que es la representación en memoria del documento XML. El proceso de parseo implica analizar la estructura del documento XML para extraer la información contenida en elementos y atributos. En Java, la API DOM o SAX se utiliza para parsear archivos XML.

Figura 17

*Parseo de XML*

Nota. Ejemplo de parseo XML. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Puedes manejar ‘IOException’ y ‘SAXException’ que pueden ocurrir durante el proceso de *parsing.*

1. **Acceder a los elementos del documento:**

Una vez que tienes el objeto ‘Document’, puedes acceder a los elementos del documento, navegar por el árbol y obtener la información que necesitas. Utiliza el método ‘getDocumentElement()’ del objeto ‘Document‘ para obtener el elemento raíz del archivo XML.

Por ejemplo, para obtener una lista de elementos con un cierto nombre:

Figura 18

*Acceder a ‘Document’*

*Nota*. Ejemplo de cómo acceder a ‘Document’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Puedes iterar sobre ‘nodeList’ para procesar cada elemento.

1. **Acceder a los atributos y contenido:**

Puedes obtener los atributos de un elemento y acceder a su contenido. Utiliza los métodos ‘getAttribute()’ o ‘getAttributeNode()’ del objeto ‘Element’ para obtener los atributos de los elementos que contienen los datos.

Figura 19

*Acceso a atributos y contenido*

*Nota*. Ejemplo de acceso a atributos y contenido. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Manejar excepciones:**

Ten en cuenta que debes manejar las excepciones que puedan ocurrir durante el proceso de *parsing* y manipulación del documento, como ‘IOException’, ‘SAXException’ y ‘DOMException’.

Figura 20

*Excepciones ‘IOException’, ‘SAXException’ y ‘DOMException’*

*Nota*. Ejemplo de excepciones ‘IOException’, ‘SAXException’, y ‘DOMException’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Estos son los pasos básicos para leer un archivo XML utilizando la API DOM en Java. Ten en cuenta que existen otras alternativas en Java para trabajar con XML, como SAX (Simple API for XML) y StAX (Streaming API for XML), que pueden ser más eficientes para archivos grandes, pero la API DOM es más conveniente cuando necesitas acceder y manipular diferentes partes del documento de manera arbitraria.

Los pasos básicos para leer un archivo XML utilizando la biblioteca DOM4J en Java son:

1. **Crear una instancia de ‘SAXReader’:**

Inicia creando un objeto ‘SAXReader’, el cual se empleará para la lectura de nodos en el documento XML.

Figura 21

*Objeto ‘SAXReader’*

*Nota*. Ejemplo de Objeto SAXReader. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Cargar el archivo XML:**

Utiliza el método ‘read()’ del objeto SAXReader para cargar el contenido del archivo XML.

Figura 22

*Método ‘read()’*

*Nota*. Ejemplo de método ‘read()’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Asegúrate de manejar la excepción ‘DocumentException’, que puede ocurrir si hay problemas durante la lectura del archivo XML.

1. **Obtener elementos específicos del XML:**

Utiliza los métodos ‘selectNodes()’ o ‘selectSingleNode()’ del objeto ‘Document’ para extraer los elementos deseados del archivo XML.

Figura 23

*Método ‘selectNodes()’*

*Nota*. Ejemplo de método ‘selectNodes()’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

O, para obtener un único nodo:

Figura 24

*Método ‘selectSingleNode()’*

*Nota*. Ejemplo de método ‘selectSingleNode()’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Estos pasos te proporcionan una estructura básica para crear un objeto ‘SAXReader’, cargar un archivo XML y luego acceder a los elementos deseados del documento utilizando la biblioteca DOM4J en Java. Recuerda ajustar y ampliar el código según las necesidades específicas de tu aplicación y la estructura del archivo XML que estás manejando. Recuerda también reemplazar **"ruta\_del\_archivo.xml"** con la ruta real de tu archivo XML.

## Lectura de un archivo JSON

Para realizar la lectura de archivos JSON en Java, una opción común es emplear la biblioteca ‘json-simple’. Asegúrate de sustituir "ruta\_del\_archivo.json" con la ubicación exacta de tu archivo JSON.

Los pasos básicos para leer un archivo JSON utilizando la biblioteca ‘JSON-simple’ en Java son:

1. **Agrega la dependencia de ‘json-simple’ a tu proyecto:**

Ejemplo para Maven:

Figura 25

*Dependencia de ‘json-simple’*

*Nota*. Ejemplo de dependencia de ‘json-simple’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Crea un archivo Java y utiliza el siguiente código:**

Figura 26

*Lectura de archivo JSON*

*Nota*. Ejemplo de lectura de archivo JSON. The Apache Software Foundation. (2023). *Apache NetBeans (16). [*Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Observaciones importantes:**

Este código presupone que la estructura de tu archivo JSON incluye un objeto principal con un array denominado "PlatoFuerte". Asegúrate de ajustar el código según la estructura específica de tu JSON.

Importante

No olvides gestionar adecuadamente las excepciones (tal como se hace en el ejemplo) para manejar posibles errores durante la lectura y análisis del archivo JSON. Además, previo a la ejecución del código, asegúrate de haber incorporado la biblioteca json-simple en tu proyecto.

## Cómo combinar los diferentes métodos de lectura de archivos para integrar archivos externos en una aplicación Java

Para integrar archivos externos en una aplicación Java, puedes combinar diferentes métodos de lectura de archivos. Aquí hay una forma de hacerlo:

1. **Usar la clase ‘File’:** puedes utilizar la clase ‘File’ de Java para representar un archivo en el sistema de archivos. Puedes crear una instancia de la clase ‘File’ pasando la ruta del archivo como argumento al constructor. Luego, puedes utilizar métodos como ‘exists()’ para verificar si el archivo existe y ‘isFile()’ para verificar si es un archivo válido.

Figura 27

*Uso de clase ‘File’*

*Nota*. Ejemplo de uso de clase ‘File’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Usar la clase ‘BufferedReader’:** la clase ‘BufferedReader’ te permite leer el contenido de un archivo línea por línea. Puedes crear una instancia de ‘BufferedReader’ pasar una instancia de ‘FileReader’ como argumento al constructor. Luego, puedes utilizar el método ‘readLine()’ para leer cada línea del archivo.

Figura 28

*Uso de clase ‘BufferedReader’*

*Nota.* Uso de clase ‘BufferedReader’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. **Lectura de archivos utilizando la clase ‘Scanner’:** la clase ‘Scanner’ proporciona métodos convenientes para leer diferentes tipos de datos de un archivo. Puedes utilizar ‘Scanner’ para leer archivos de texto y analizar su contenido.

Figura 29

*Uso de clase ‘Scanner’*

*Nota*. Uso de clase ‘Scanner’. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

# Videos

A continuación, te invitamos a explorar cómo acceder a la información almacenada en un fichero de texto externo:

* Métodos de lectura de archivos para integrar archivos externos en una aplicación Java: <https://youtu.be/etQN4EfYN7k>
* Métodos de escritura de archivos para integrar archivos externos en una aplicación Java: <https://youtu.be/E0H4OzW2_1Y>

# Dato

El nombre "Java" fue inspirado en el amor de los creadores por el café. El nombre hace referencia a una especie de café, el café arábigo, originario de la región de Yemen. Esta elección se debió a que el equipo de desarrollo apreciaba mucho el café y el área de descanso de Sun Microsystems (la compañía detrás de Java en sus inicios) solía tener una máquina de café espresso. Por eso, el nombre "Java" representa no solo el lenguaje de programación, sino también el gusto del equipo por el café.

# Cierre de la semana

El manejo de entradas y salidas (I/O) en Java es un componente esencial de la programación que permite a los programas interactuar con el mundo exterior a través de archivos. Estos procesos abarcan la lectura y escritura de datos en diversos formatos, como CSV, XML y JSON, y requieren conocimientos de cómo configurar el entorno de desarrollo y cómo importar y utilizar bibliotecas específicas para la manipulación eficiente de archivos. Estas habilidades son fundamentales para la creación de aplicaciones Java que sean robustas, seguras y capaces de manejar la persistencia de datos.



# Referencias

* Duhnnae. (2017). *Tutorial de java: Crear leer borrar archivo* [Video] YouTube. <https://www.youtube.com/watch?v=RgqeekuSObA>
* Morales, J. (2016). *Gestión de archivos en Java.* [enlace] <http://jmoral.es/blog/IO-java>
* Pildorasinformaticas. (2015). *Curso Java. Streams I. Accediendo a ficheros. Lectura. Vídeo 152* [Video] YouTube. <https://youtu.be/etQN4EfYN7k>
* Pildorasinformaticas. (2015). *Curso Java. Streams II. Accediendo a ficheros Escritura. Vídeo 153* [Video] YouTube. <https://youtu.be/E0H4OzW2_1Y>
* Sánchez, J. (2010). *Entrada y salida en Java. Ficheros.* [enlace] <https://jorgesanchez.net/manuales/viejos/fpr/fpr1009.pdf>

# Lecturas de la semana

**Capítulo 12:** Interfaz y clases abstractas

M. Hernández B. (2023). Programación orientada a objetos en Java. Buenas prácticas. Ediciones de la U. <https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a51968-Programacion/243/> Páginas 253 a 265

# Apuntes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.