

LENGUAJE DE PROGRAMACIÓN I



ARCHIVOS

Docente: Isabel Alvarez

AGENDA

- Entrada / Salida
- I/O Streams
- Byte / Characters / Buffered / Data / Object (Streams)
- Manejo de Archivos y Directorios
- Escaneo y Formateo
- Serialización
- Lectura / Escritura de Archivos XML

Entrada / Salida



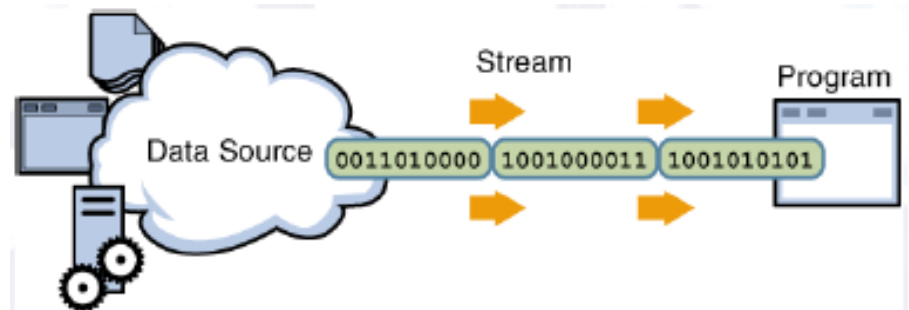
Entrada/Salida : I/O Streams

Los programas acceden a datos externos y estos se recuperan a partir de un **origen de entrada** y se guardan a través de un **destino de salida**.

Java llama flujo (Stream) a esta abstracción y la implementa con clases del paquete **java.io**

Un programa usa un **input stream** para leer datos desde una fuente, un item a la vez.

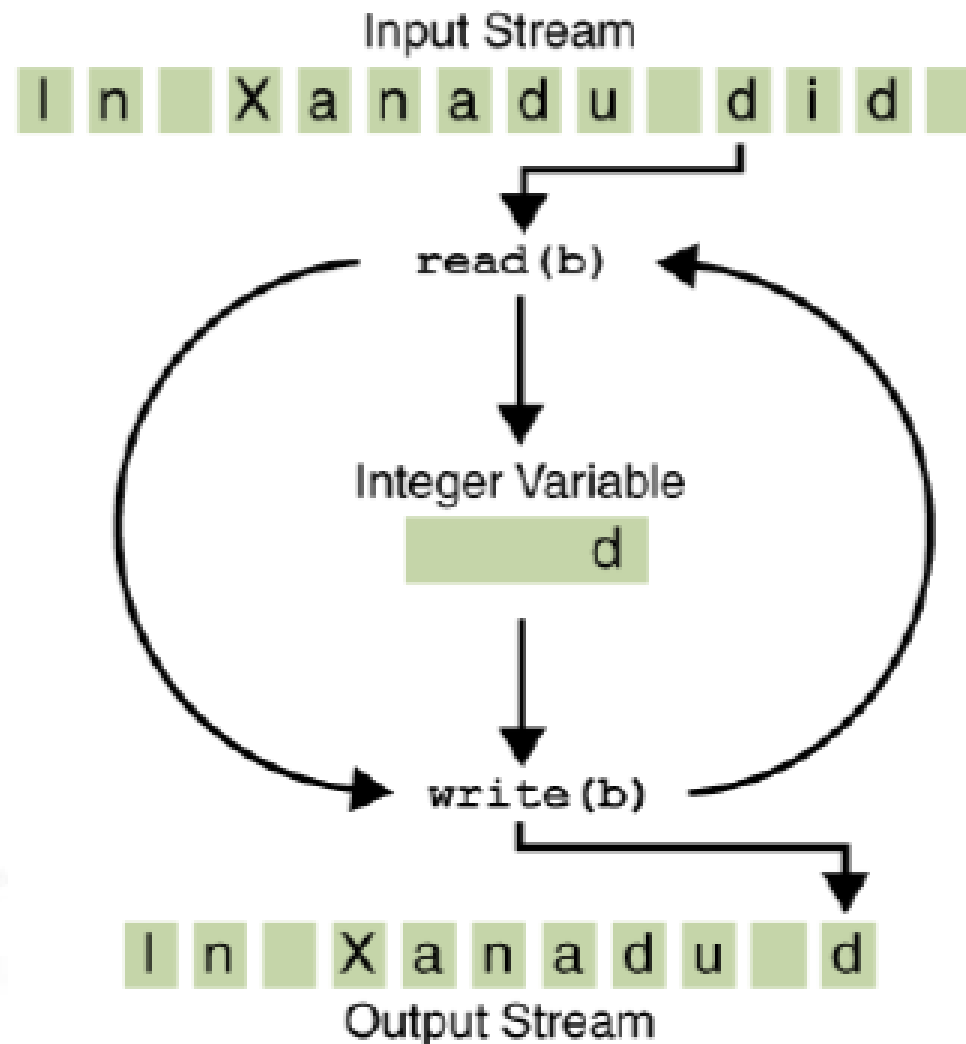
Un programa usa un **output stream** para escribir datos a una fuente, un item a la vez.



FileInputStream/FileOutputStream

<i>Filehero Binario</i>	<i>Filehero Texto</i>
FileInputStream	FileReader
BufferedInputStream	BufferedReader
FileOutputStream	FileWriter
BufferedOutputStream	BufferedWriter
	PrintWriter * (Java 5)

Lectura/Escritura byte a byte



Lectura/Escritura byte a byte

```
FileInputStream in = null;
FileOutputStream out = null;
try {
    //Flujo de entrada
    in = new FileInputStream(archivoOrigen);
    //Flujo de salida
    out = new FileOutputStream(archivoDestino);
    int b=0;
    while((b=in.read())!= -1){
        out.write(b);
    }
} catch (FileNotFoundException ex) {
    System.out.println("Archivo no encontrado: "+ex.getMessage());
} catch (IOException ex1) {
    System.out.println("Archivo no encontrado: "+ex1.getMessage());
}finally{
    try {
        in.close();
        out.close();
    } catch (IOException ex) {
        System.out.println("No puedo cerrar el descriptor de archivos "
            + ex.getMessage());
    }
}
```

Estructura

Ejercicio

Ejercicio

```

1 package itq.edu.ec;
2
3 import ...6 lines
4
5
6
7
8
9
10 public class Byte_Byte {
11
12     public static void main(String[] args) {
13         //Instancio la clase actual, para uso de metodos
14         Byte_Byte flujo = new Byte_Byte();
15         //Ruta de archivo origen y archivo destino
16         String archivoOrigen = "C:\\Users\\Choppy\\Documents\\ejr1_archivos.txt";
17         String archivoDestino = "C:\\Users\\Choppy\\Documents\\copia_ejr1_archivos.txt";
18         //Referencio al metodo creado, envio dos parámetros
19         flujo.copiaByteToByte(archivoOrigen, archivoDestino);
20     }
21
22     private void copiaByteToByte(String archivoOrigen, String archivoDestino){
23         //IMPORTAR: java.io.FileInputStream
24         FileInputStream in = null;
25         //IMPORTAR: java.io.FileOutputStream
26         FileOutputStream out = null;
27
28         try { //try-catch NO SE COPIA, se importa
29             //IMPORTAR: Surround Statement with try-catch
30             in = new FileInputStream(archivoOrigen);
31             out = new FileOutputStream(archivoDestino); //sobre escribe lo que hay
32             //out = new FileOutputStream(archivoDestino,true); //escribe despues de lo que hay
33             int b = 0;
34             //IMPORTAR: Add catch Clause
35             while ((b = in.read()) != -1) {
36                 out.write(b);
37             }
38             System.out.println("Proceso de copia realizado con exito !!! ");
39
40         } catch (FileNotFoundException ex) {
41             //Logger.getLogger(Byte_Byte.class.getName()).log(Level.SEVERE, null, ex);
42             System.out.println("Archivo no encontrado" + ex.getMessage());
43         } catch (IOException ex) {
44             //Logger.getLogger(Byte_Byte.class.getName()).log(Level.SEVERE, null, ex);
45             System.out.println("Error de lectura de bytes" + ex.getMessage());
46         } finally {
47             try { //NO SE COPIA, se importa
48                 //IMPORTAR: Surround Block with try-catch
49                 in.close();
50                 out.close();
51             } catch (IOException ex) {
52                 System.out.println("No se pudo cerrar los flujos I/O" + ex.getMessage());
53             }
54         }
55     }
56 }

```


Archivos

La forma de interactuar con los sistemas de archivos locales se realiza a través de la clase **File**, esta clase proporciona muchas utilidades relacionadas con archivos y con la obtención de información básica sobre esos archivos.

Clase File → Paquete java.io

Clase File

La clase File tiene 3 constructores.

- `File(String path);`
- `File(String path, String name);`
- `File(File dir, String name);`

El 1º toma un String recibe el nombre de un archivo, incluyendo su ruta, en forma de cadena.

El 2º toma por separado la ruta y el nombre del archivo

El 3º similar al primero excepto que el primer parámetro que indica el directorio del archivo es del tipo File en lugar del tipo String.

Métodos Clase File

Algunos métodos de la clase File:

<i>MÉTODO</i>	<i>DESCRIPCIÓN</i>
<code>objetoFile.getName();</code>	Devuelve el nombre del fichero.
<code>objetoFile.getPath();</code>	Devuelve la ruta del fichero.
<code>objetoFile.exists();</code>	Devuelve true si existe el fichero.
<code>objetoFile.canWrite();</code>	Devuelve true si se puede escribir.
<code>objetoFile.canRead();</code>	Devuelve true si se puede leer.
<code>objetoFile.isDirectory();</code>	Devuelve true si es un directorio.
<code>objetoFile.isFile();</code>	Devuelve true si es un fichero.
<code>objetoFile.lastModified();</code>	Devuelve la fecha de la última modificación.
<code>objetoFile.length();</code>	Devuelve la longitud del fichero.

```
public class ClaseFile {
```

```
    public static void main(String[] args) {
        ClaseFile p = new ClaseFile();
        String archivo = "C:\\\\Users\\\\Choppy\\\\Documents\\\\NetBeansProjects\\\\LengProgra_1\\\\archivoFile.txt";
        p.impressionDatosArchivo(archivo);
    }

    private void impresionDatosArchivo(String archivo) {
        //IMPORTAR: java.util.Calendar;
        Calendar hoy = Calendar.getInstance();
        //IMPORTAR: java.text.SimpleDateFormat;
        SimpleDateFormat sdf = new SimpleDateFormat("EEEEEE,dd-MMMM-yyyy HH:mm:ss:SSS");
        //IMPORTAR: java.io.File;
        File file = new File(archivo);

        if (file != null && file.exists()) { //Si existe el archivo lo lee
            System.out.println("\nEl archivo (" + file.getName() + ") existe.. ");
            System.out.println("Directorio : " + file.getParent());
            hoy.setTimeInMillis(file.lastModified());
            System.out.println("Modificado : " + sdf.format(hoy.getTime()));
            System.out.println("Tamaño : " + ((double) file.length() / 1024 + " Kb"); //esta en byte lo paso a kb
            if (file.canRead()) {
                System.out.println("(" + file.getName() + ") tiene permisos de lectura");
            } else {
                System.out.println("(" + file.getName() + ") no tiene permisos de lectura");
            }

            if (file.canWrite()) {
                System.out.println("(" + file.getName() + ") tiene permisos de escritura");
            } else {
                System.out.println("(" + file.getName() + ") no tiene permisos de escritura");
            }

            if (file.canExecute()) {
                System.out.println("(" + file.getName() + ") tiene permisos de ejecucion");
            } else {
                System.out.println("(" + file.getName() + ") no tiene permisos de ejecucion");
            }
        } else { //Si NO existe el archivo lo crea
            try { //NO COPIO EL TRY, no importo
                //IMPORTAR: Surround Block with try-catch
                System.out.println("\nArchivo (" + file.getName() + ") no existe, procediendo a crearlo...");
                file.createNewFile();
                String nuevoArchivo = file.getPath();
                //Metodo recursivo
                impresionDatosArchivo(nuevoArchivo);
            } catch (IOException ex) {
                System.out.println("No se pudo crear el archivo" + ex.getMessage());
            }
        }
    }
}
```

Ejercicio

Consulta

Consultar el formato para fechas en java

Listado de Archivos en Directorio

```
public class ArchivosEnDirectorio {
    //Variable global
    private static final String dir_inicial = "C:\\\\Documents\\NetBeansProjects\\LengProgra_1\\Directorio";

    public static void main(String[] args) {
        ArchivosEnDirectorio d = new ArchivosEnDirectorio();
        d.imprimirContenido(ArchivosEnDirectorio.dir_inicial);
    }

    private void imprimirContenido(String dirInicial) {
        //IMPORTAR: java.io.File
        File directorio = new File(dirInicial);
        //Si es directorio, va al contenido
        if (directorio != null && directorio.exists() && directorio.isDirectory())
        {
            File[] archivos = directorio.listFiles();
            if (archivos != null && archivos.length > 0) { //Si tiene archivos trae datos
                System.out.println("\nDirectorio : " + directorio.getAbsolutePath());
                for (File tmpArchivo : archivos) {
                    if (tmpArchivo != null && tmpArchivo.isDirectory()) //subdirectorio
                    {
                        //Recurividad
                        imprimirContenido(tmpArchivo.getPath());
                    } else {
                        if (tmpArchivo != null && tmpArchivo.isFile()) //archivo
                        {
                            System.out.println("Archivo : " + tmpArchivo.getName() + "; Tamaño "
                                + ((double) tmpArchivo.length()) / 1024 + " kb");
                        }
                    }
                }
            }
            else { //Si el directorio no tiene archivos
                try {
                    System.out.println("\nDirectorio .- " + directorio.getCanonicalPath() + " \nNO tiene archivos");
                } catch (IOException ex) {
                    System.out.println(ex.getMessage());
                }
            }
        } else { //solo es archivo
            System.out.println("\n" + directorio.getName() + "; " + ((double) directorio.length()) / 1024 + " kb");
        }
    }
}
```

Buffered

Si usamos sólo `FileInputStream`, `FileOutputStream`, `FileReader` o `FileWriter`, cada vez que hagamos una lectura/escritura (con pocos caracteres), se accederá cada vez al disco duro, haciendo un proceso costoso y lento.

Los `BufferedReader`, `BufferedInputStream`, `BufferedWriter` y `BufferedOutputStream` añaden un buffer intermedio.

Cuando leamos o escribamos, esta clase controlará de forma eficiente los accesos a disco, haciendo que el programa corra más rápido. La diferencia se notará más grande sea el archivo a leer o escribir.

API java.io

java.io Class	Extends From	Key Constructor(s) Arguments	Key Methods
File	Object	File, String String String, String	createNewFile() delete() exists() isDirectory() isFile() list() mkdir() renameTo()
FileWriter	Writer	File String	close() flush() write()
BufferedWriter	Writer	Writer	close() flush() newLine() write()
PrintWriter	Writer	File (as of Java 5) String (as of Java 5) OutputStream Writer	close() flush() format()* printf()* print(), println() write()
FileReader	Reader	File String	read()
BufferedReader	Reader	Reader	read() readLine()

DataOutputStream/ DataInputStream

- Los objetos DataInputStream se comportan como los FileInputStream.
- Los streams de datos pueden leer cualquiera de las variables de tipo nativo, como floats, ints o chars.
- Generalmente se utilizan DataInputStream con archivos binarios.

Ejercicio Escritura

```
public class EscrituraDataStream {

    public static void main(String[] args) {
        //IMPORTAR: java.io.FileOutputStream;
        FileOutputStream fos = null;
        //IMPORTAR: java.io.DataOutputStream;
        DataOutputStream dos = null;
        try {
            fos = new FileOutputStream("C:\\Users\\Choppy\\Documents\\pedido.txt");
            dos = new DataOutputStream(fos);
            String[] productos = {"Resma de papel", "Lapices", "Boligrafos", "Carteras", "Mesas"};
            int[] unidades = {5, 7, 12, 8, 30};
            double[] precio = {35.0, 40.5, 8.77, 2.01, 3.99};

            for (int i = 0; i < productos.length; i++) {
                dos.writeChars(productos[i]);
                dos.writeChar('\n');
                dos.writeInt(unidades[i]);
                dos.writeChar('\t');
                dos.writeDouble(precio[i]);
            }
            System.out.println("Escritura de datos realizado");
        } catch (FileNotFoundException ex) {
            Logger.getLogger(EscrituraDataStream.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(EscrituraDataStream.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            try {
                dos.close();
                fos.close();
            } catch (IOException ex) {
            }
        }
    }
}
```

Ejercicio Lectura

```
public class LecturaDataStream {
```

```
    public static void main(String[] args) {
```

```
        FileInputStream fis = null;//IMPORTAR: java.io.FileInputStream
```

```
        DataInputStream dis = null;//IMPORTAR: java.io.DataInputStream
```

```
        BufferedReader br = null;//IMPORTAR: java.io.BufferedReader;
```

```
        String producto;
```

```
        int unidad;
```

```
        double precio;
```

```
        double total = 0.0;
```

```
        try {
```

```
            fis = new FileInputStream("C:\\Users\\Choppy\\Documents\\pedido.txt");
```

```
            dis = new DataInputStream(fis);
```

```
            //br = new BufferedReader(fis);
```

```
            while ((producto = dis.readLine()) != null) {
```

```
                //dis.readChar();
```

```
                unidad = dis.readInt();
```

```
                dis.readChar();
```

```
                precio = dis.readDouble();
```

```
                System.out.println("has pedido: " + unidad + " " + producto + " a " + precio + " dólares");
```

```
                total += precio * unidad;
```

```
            }
```

```
            System.out.println("Costo total del pedido:" + total + " dólares");
```

```
        } catch (FileNotFoundException ex) {
```

```
            Logger.getLogger(LecturaDataStream.class.getName()).log(Level.SEVERE, null, ex);
```

```
        } catch (IOException ex) {
```

```
            Logger.getLogger(LecturaDataStream.class.getName()).log(Level.SEVERE, null, ex);
```

```
        } finally {
```

```
            try {
```

```
                fis.close();
```

```
                dis.close();
```

```
            } catch (IOException ex) {
```

```
                Logger.getLogger(LecturaDataStream.class.getName()).log(Level.SEVERE, null, ex);
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

Escaneo y Formateo

El programar con I/O a menudo involucra trasladar la información a y desde un formato específico. Para esto en java se cuenta con 2 API's:

- Scanner API, que rompe la entrada a tokens individuales de datos.
- Formatting API que transforma la información a un formato específico.

Class Scanner

La clase Scanner se introdujo a partir de java 5 y tiene utilidades como separar o romper su dato de entrada en subcadenas o “tokens” y además convertir estos datos en tipos primitivos como enteros, flotantes, etc.

```
List<Double> numeros = new ArrayList<Double>();
File f = new File("C:\\recurso\\numeros.txt");
try {
    Scanner sc = new Scanner(f);
    sc.useDelimiter(" ");
    while(sc.hasNextDouble()){
        //while(sc.hasNext()){
            Double d = new Double(sc.nextDouble());
            numeros.add(d);
        }
    for (Double numero : numeros) {
        System.out.println("El numero es: "+numero.doubleValue());
    }
} catch (FileNotFoundException ex) {
    System.out.println("No existe el archivo"+ ex.getMessage());
}
```

19.77 14.28 2251,00 14,86

Class PrintWriter

La clase PrintWriter desde la versión de java 1.5 recibe en su constructor objetos de tipo File, permitiendo que se puedan escribir datos formateados a los archivos directamente.

```
try {
    PrintWriter pr = new PrintWriter("C:\\recurso\\ejemplo_printwriter.txt");
    pr.print("Hola mundo");
    pr.print('\t');
    pr.print(29.89F);
    pr.print('\t');
    pr.print(true);
    pr.print('\n');
    pr.println(135);
    pr.close();
} catch (FileNotFoundException ex) {
    System.out.println("No existe el archivo..." + ex.getMessage());
}
```

Consulta

Consultar clases de Scanner y PrintWriter

Class Formatter

Clase de formateo multi-propósito de Java, se encuentra en el paquete `java.util`. Convierte datos binarios en formas legibles. Tiene algunos constructors.

Formatter()
Formatter(Local loc)
Formatter(String filename) throws FileNotFoundException
Formatter(OutputStream outstrm)
Formatter(PrintStream outstrm)


```

public class ClaseFormatter {

    public static void main(String[] args) {
        //IMPORTAR: java.util.Formatter;
        Formatter fmtCon = new Formatter(System.out);
        //Impresion en consola
        fmtCon.format("Este es un número negativo: %(.3f\n", -123.34);
        fmtCon.format("%8s %8s\n", "Value", "Square");
        for (int i = 1; i < 5; i++) {
            fmtCon.format("%8s %8s\n", i, i * i);
        }
        Formatter fmtFile = null;
        try {
            //Guarda datos en archivo
            fmtFile = new Formatter("C:\\Users\\Choppy\\Documents\\test.txt");
            fmtFile.format("Este es un número negativo: %(.2f\n\n", -123.34);
            fmtFile.format("%8s %8s\n", "Value", "Square");
            for (int i = 1; i < 20; i++) {
                fmtFile.format("%8s %8s\n", i, i * i);
            }
            System.out.println("\nDatos guardados correctamente");
        } catch (FileNotFoundException ex) {
            System.out.println("No se puede abrir el archivo " + ex.getMessage());
        } finally {
            fmtFile.close();
        }
    }
}

```

Resumen Manejo de Archivos

Clase	Constructores	Métodos	Excepciones
File	File(String s)		
		createNewFile()	IOException
		renameTo(File f)	
FileReader	FileReader(String s) FileReader(File f)		FileNotFoundException
		read() close()	IOException
FileWriter	FileWriter(String s) FileWriter (File f)		FileNotFoundException
		write() flush() close()	IOException
BufferedReader	BufferedReader(FileReader fr)		
		readLine() close()	IOException
BufferedWriter	BufferedWriter(FileWriter w)		
		write() newLine() close()	IOException

Serialización (Object Stream)

La serialización (o marshalling en inglés) es un proceso de codificación de un Objeto (programación orientada a objetos) en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de **bytes** o en un formato humanamente más legible.

Es un mecanismo ampliamente usado para transportar objetos a través de una red, para hacer **persistente** un objeto en un archivo o base de datos, o para distribuir objetos idénticos a varias aplicaciones o localizaciones.

Serialización (Object Stream)

Para que un programa pueda convertir un objeto en un montón de bytes y pueda luego recuperarlo, el objeto necesita ser Serializable. Al poder convertir el objeto a bytes, ese objeto se puede enviar a través de red, guardarlo en un fichero, y después reconstruirlo al otra lado de la red, leerlo del fichero.

Para que un objeto sea serializable basta con que implemente la interfaz **Serializable**.

Serialización (Parte 1-3)

```
//IMPLEMENTACION SERIALIZABLE
public class Gato implements Serializable{

    private String nombre;
    private String color;
    private int edad;

    //METODOS GET Y SET
    public String getNombre() { ...3 lines }

    public void setNombre(String nombre) { ...3 lines }

    public String getColor() { ...3 lines }

    public void setColor(String color) { ...3 lines }

    public int getEdad() { ...3 lines }

    public void setEdad(int edad) { ...3 lines }

    //CONSTRUCTORES
    public Gato(String nombre, String color, int edad) { ...5 lines }

    //STRING
    @Override
    public String toString() { ...3 lines }
}
```

Serialización (Parte 2-3)

```
public class GatoSerializable {
    public static void main(String[] args) {

        try { //NO COPIAR-IMPORTAR
            Gato minino=new Gato("sambito", "cafe", 10);
            //IMPORTAR: java.io.FileOutputStream
            FileOutputStream fos=null;
            //IMPORTAR: java.io.ObjectOutputStream;
            ObjectOutputStream oos=null;

            fos=new FileOutputStream("C:\\Users\\LengProgra_1\\gato.bin");
            oos = new ObjectOutputStream(fos);

            oos.writeObject(minino);
            oos.close();
            fos.close();
            //IMPORTAR: Surround Block with try-catch
        } catch (FileNotFoundException ex) {
            Logger.getLogger(GatoSerializable.class.getName()).log(Level.SEVERE, null, ex);
            //IMPORTAR: Add catch Clause
        } catch (IOException ex) {
            Logger.getLogger(GatoSerializable.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Serialización (Parte 3-3)

```
public class GatoDeserializable {
    public static void main(String[] args) {
        try {
            FileInputStream fis=null;//IMPORTAR: java.io.FileInputStream
            ObjectInputStream ois =null;//IMPORTAR: java.io.ObjectInputStream

            fis=new FileInputStream("C:\\Users\\LengProgra_1\\gato.bin");
            ois=new ObjectInputStream(fis);

            Gato gatitoRecuperado= (Gato)ois.readObject();
            System.out.println("Nombre del gato : " + gatitoRecuperado.getNombre());
            System.out.println("Color del gato : " + gatitoRecuperado.getColor());
            System.out.println("Edad del gato : " + gatitoRecuperado.getEdad());

            ois.close();
            fis.close();
        } catch (FileNotFoundException ex) {
            Logger.getLogger(GatoDeserializable.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(GatoDeserializable.class.getName()).log(Level.SEVERE, null, ex);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(GatoDeserializable.class.getName()).log(Level.SEVERE, null, ex);
        }finally{
        }
    }
}
```

Manejo de Archivos XML

JAXP - Java API for XML Processing.

JAXP es el API que provee Java para el procesamiento de archivos XML.

Contiene tres interfaces de parseo básico:

- DOM (Document Object Model)
- SAX (Simple API for XML)
- StAX (Streaming API for XML)

Manejo de Archivos XML

- **DOM (Document Object Model)**

Todo el documento es leído en memoria como una estructura de árbol para acceso aleatorio por la aplicación llamante.

- **SAX (Simple API for XML)**

Basadas en eventos, la aplicación se registra para recibir eventos según las entidades se encuentran en el documento de origen.

- **StAX (Streaming API for XML)**

Es un término medio entre DOM y SAX.

Manejo de Archivos XML

XPath (XML Path Language)

Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML.

XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML. XPath fue creado para su uso en el estándar XSLT, en el que se usa para seleccionar y examinar la estructura del documento de entrada de la transformación.

Se puede decir que Xpath es una especie de SQL para archivos XML.

XPath

Seleccionando Nodos.

Xpath usa "path expressions", para seleccionar Nodos en un documento XML. Siendo las siguientes las más comunes:

Expresión	Descripción
<i>nodename</i>	Selecciona todos los nodos con el nombre "nombre ingresado".
/	Selecciona desde el nodo raíz.
//	Selecciona los nodos en el documento del nodo actual que coinciden con la selección sin importar dónde se encuentren.
.	Selecciona el nodo actual.
..	Selecciona el padre del nodo actual.
@	Selecciona atributos.

XPath

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

Expresión Path	Resultado
bookstore	Selecciona todos los nodos con el nombre "bookstore".
/bookstore	Selecciona el elemento raíz bookstore. Nota: si la ruta comienza con (/) siempre representa una ruta absoluta a un elemento.
bookstore/book	Selecciona todos los elementos book que son hijos de bookstore.
//book	Selecciona todos los elementos book sin importar dónde se encuentren en el documento.
bookstore//book	Selecciona todos los elementos book que son descendientes del elemento bookstore, sin importar dónde se encuentren bajo el elemento bookstore
//@lang	Selecciona todos los atributos que se denominan lang

XPath

Predicados: Son usados para encontrar un nodo específico o un nodo con un valor específico. Se los incluye en [].

Expresión Path	Resultado
/bookstore/book[1]	Selecciona el primer elemento book que es el hijo del elemento bookstore. Nota: En IE 5,6,7,8,9 el primer nodo es [0], pero según W3C, es [1]. Para resolver este problema en IE, establezca SelectionLanguage en XPath.
/bookstore/book[last()]	Selecciona el último elemento book que es el hijo del elemento bookstore.
/bookstore/book[last()-1]	Selecciona el último, pero un elemento book, que es el elemento secundario de bookstore.
/bookstore/book[position()<3]	Selecciona los dos primeros elementos book, que son hijos del elemento de bookstore.
//title[@lang]	Selecciona todos los elementos cuyo título contiene el atributo lang
//title[@lang='eng']	Selecciona todos los elementos cuyo título tienen el atributo lang con un valor de 'eng'
/bookstore/book[price>35.00]	Selecciona todos los elementos book que contiene bookstore cuyo elemento precio tiene un valor superior a 35.00
/bookstore/book[price>35.00]/title	Selecciona todos los elementos de bookstore tomando en cuenta el título, precio con un valor mayor a 35.00

XPath

Seleccionado nodos desconocidos.

Xpath wildcards pueden ser usados para seleccionar elementos XML desconocidos.

Wildcard	Descripción
<code>*</code>	Coincide con cualquier elemento del nodo
<code>@*</code>	Coincide con cualquier atributo del nodo
<code>node()</code>	Coincide con cualquier tipo del nodo

Ejemplo:

Path Expression	Result
<code>/bookstore/*</code>	Selecciona todos los nodos secundarios del elemento de bookstore.
<code>//*</code>	Selecciona todos los elementos del documento.
<code>//title[@*]</code>	Selecciona todos los elementos del título que tengan algún atributo.

Ejercicio Lectura (Parte 1-3)

IMPORTAR: librería dom4j-full.jar y recursos .xml

```
public class LecturaXml {  
  
    public static void main(String[] args) {  
        LecturaXml x = new LecturaXml();  
        String contenidoXml;  
        try {  
            contenidoXml = x.obtenerStringXml("C:\\Users\\LengProgra_1\\Archivos\\recursos\\orden.xml");  
            x.leerXml(contenidoXml);  
        } catch (IOException ex) {  
  
        }  
    }  
  
    private String obtenerStringXml(String archivoXml) throws IOException { ...20 lines }  
  
    private void leerXml(String contenidoXml) { ...50 lines }  
}
```

Ejercicio Lectura (Parte 2-3)

```
private String obtenerStringXml(String archivoXml) throws IOException {
    FileReader fr = null;
    BufferedReader br = null;
    String contenidoXml = "";
    try {
        fr = new FileReader(archivoXml);
        br = new BufferedReader(fr);
        String linea = "";
        while ((linea = br.readLine()) != null) {
            contenidoXml += "\n" + linea;
            //System.out.println(contenidoXml);
        }
    } catch (FileNotFoundException ex) {

    }

    finally {
        fr.close();
        br.close();
    }

    return contenidoXml;
}
```


Ejercicio Lectura (Parte 3-3)

```
private void leerXml(String contenidoXml) {
    Document doc = null;
    try {
        doc = DocumentHelper.parseText(contenidoXml);
        Node nodoCliente = doc.selectSingleNode("//Orden/Cliente");
        String nombreCliente = nodoCliente.getText();
        String cedulaCliente = nodoCliente.valueOf("@cedula");
        System.out.println("Nombre Cliente: " + nombreCliente + ", cédula: " + cedulaCliente);

        List<Node> nodosProductos = doc.selectNodes("//Orden/Producto");
        for (Node nodoProducto : nodosProductos) {
            System.out.println("\nNombre producto: " + (nodoProducto.selectSingleNode("Nombre")).getText());
            System.out.println("Codigo producto: " + (nodoProducto.selectSingleNode("Codigo")).getText());
            System.out.println("Cantidad producto: " + (nodoProducto.selectSingleNode("Cantidad")).getText());
            System.out.println("Precio producto: " + (nodoProducto.selectSingleNode("Precio")).getText());
            System.out.println("Moneda precio producto: " + (nodoProducto.selectSingleNode("Precio")).valueOf("@moneda"));
            if (nodoProducto.selectSingleNode("Descuento") != null) {
                System.out.println("Descuento Producto: " + (nodoProducto.selectSingleNode("Descuento")).getText());
            }
            System.out.println(">> ENTREGA");
            if (nodoProducto.selectSingleNode("Destinatario") != null) {
                System.out.println("Destinatario: " + (nodoProducto.selectSingleNode("Entrega/Destinatarario")).getText());
            }
            System.out.println("Calle: " + (nodoProducto.selectSingleNode("Entrega/Calle")).getText());
            System.out.println("Ciudad: " + (nodoProducto.selectSingleNode("Entrega/Ciudad")).getText());
            System.out.println("Provincia: " + (nodoProducto.selectSingleNode("Entrega/Provincia")).getText());
            System.out.println("Codpostal: " + (nodoProducto.selectSingleNode("Entrega/Codpostal")).getText());

            System.out.println(">> MENSAJE REGALO");
            if (nodoProducto.selectSingleNode("Mensajeregalo") != null) {
                System.out.println("Mensaje Regalo: " + (nodoProducto.selectSingleNode("Mensajeregalo")).getText());
            }
            System.out.println("=====");
        }
    } catch (DocumentException ex) {
        Logger.getLogger(LecturaXml.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Ejercicio Lectura (impresión)

Nombre Cliente: René Segovia, cédula: 1712950532

Nombre producto: Reloj de Pared

Codigo producto: 244

Cantidad producto: 12

Precio producto: 21.95

Moneda precio producto: USD

>> ENTREGA

Calle: Cesar Zea N53-262

Ciudad: Quito

Provincia: Pichincha

Codpostal: 02882

>> MENSAJE REGALO

= = = = =

Nombre producto: Brasalette

Codigo producto: 258

Cantidad producto: 1

Precio producto: 144.95

Moneda precio producto: USD

Descuento Producto: .10

>> ENTREGA

Calle: 271 Old Homestead Way

Ciudad: Guayaquil

Provincia: Guayas

Codpostal: 02895

>> MENSAJE REGALO

Mensaje Regalo:

Feliz cumpleaños mi amor!

Ejercicio Escritura (Parte 1-2)

```
public class EscrituraXml {

    public static void main(String[] args) {
        EscrituraXml x = new EscrituraXml();
        Document documento = x.crearDocumentoXml();
        try {
            System.out.println(x.documentoToString(documento));
        } catch (IOException ex) {
            System.out.println("Problemas con el documento" + ex.getMessage());
        }
    }

    private Document crearDocumentoXml() {
        Document documento = DocumentHelper.createDocument();
        Element peliculas = documento.addElement("peliculas");
        Element pelicula1 = peliculas.addElement("pelicula");
        pelicula1.addElement("titulo").addAttribute("idioma", "ES").addText("Salvando al soldado Ryan");
        pelicula1.addElement("director").addText("Steven Spielberg");
        pelicula1.addElement("año").addText("2000");
        Element actores1 = pelicula1.addElement("actores");
        actores1.addElement("actor").addAttribute("tipo", "principal").addText("Tom Hans");
        actores1.addElement("actor").addAttribute("tipo", "reparto").addText("Matt Daemon");




        Element pelicula2 = peliculas.addElement("pelicula");
        pelicula2.addElement("titulo").addAttribute("idioma", "ES").addText("Tacones lejanos");
        pelicula2.addElement("director").addText("Pedro Almodovar");
        pelicula2.addElement("año").addText("1991");
        Element actores2 = pelicula2.addElement("actores");
        actores2.addElement("actor").addAttribute("tipo", "principal").addText("Victoria Abril");
        actores2.addElement("actor").addAttribute("tipo", "reparto").addText("Miguel Bose");

        return documento;
    }

    private String documentoToString(Document documento) throws IOException {...9 lines }
}
```

Ejercicio Escritura (Parte 2-2)

```
private String documentoToString(Document documento) throws IOException {  
    OutputFormat outFormat = new OutputFormat();  
    outFormat.setEncoding("ISO-8859-1");  
    StringWriter writer = new StringWriter();  
    XMLWriter out = new HTMLWriter(writer, outFormat);  
    out.write(documento);  
  
    return writer.toString();  
}
```

Usages	Output - Archivos (run) %
	run:
	<películas><película><título idioma="ES">Salvando al soldado Ryan</título><director>Steven Spielberg</director><año>2000</año><actores><a
	BUILD SUCCESSFUL (total time: 0 seconds)



Copiar en un archivo plano, su extensión debe ser **.xml**