



ACCESO A DATOS

UD2. Manejo de archivos

Práctica 4



Reconocimiento – No Comercial: permite el uso de la obra siempre que no sea con fines comerciales y se reconozca la autoría de su creador. Se autoriza la reproducción realizada bajo demanda para alumnos cursando Formación Profesional en el **IES Ciudad Escolar**.

Contenido

Objetivos:	2
Materiales y recursos:	2
Desarrollo de la actividad:	3
Enunciado	3
Entrega:	5
Criterios de evaluación:	5

UD2. Manejo de archivos

Práctica 4

Objetivos:

- Poner en práctica los recursos estudiados en clase para el tratamiento de ficheros JSON desde aplicaciones JAVA.
- Comprender y entender los fundamentos de procesamiento en JAVA para la interacción sobre ficheros de texto, JSON y XML.
- Asimilar y poner en práctica las buenas prácticas de programación (evitando “hardcodes” y la gestión ineficiente de recursos).

Materiales y recursos:

- Ejercicios realizados en clase incluidos en la unidad didáctica.
- Credenciales de la cuenta personal de Github miembro de la organización @DAM2-AccesoDatos
- Ordenador del/a alumno/a.
- Software: Entorno portable VSCode+JDK+Git+Maven.
- Fichero XML con los detalles de un hospital (*hospital_2526.xml*).
- Fichero DTD validador del xml previo (*hospital_2526.dtd*)
- Fichero XSD validador del xml nuevo a generar (*nuevo_hospital.xsd*)
- Fichero de texto con los nuevos médicos que se incorporan este año (*nuevos_medicos.txt*).

Desarrollo de la actividad:

A continuación, se da un enunciado con una serie de actividades que el alumno deberá llevar a cabo codificando un programa en JAVA y haciendo uso de la API Jackson databind para JSON y de la API org.w3c DOM para XML.

Enunciado

1. Descarga y descomprime el fichero **AADD-ent_portable-v8.zip** compartido por el profesor en la carpeta de usuario del alumno (ej. C:\Users\alumno). Se trata de un entorno de desarrollo **portable** e **independiente** de la máquina: VSCode + JDK + Maven + Git.

Nota: NO USAR CARPETAS CON ACENTOS, EÑES NI ESPACIOS en la ruta donde descomprimir el fichero. Os dará error durante el uso.

2. Entra en la carpeta descomprimida y ejecuta **launch.cmd** para arrancar el entorno.

Nota: NUNCA EJECUTANDO DIRECTAMENTE "CODE.EXE".

3. Acepta la invitación de la tarea de GitHub Classroom que tendrás en tu correo electrónico. Esto creará un nuevo repo privado para el alumno en la organización (<https://github.com/DAM2-AccesoDatos/XXX.git>).

Nota: Te pedirá que te autentiques con tu usuario y contraseña de GitHub. Hazlo desde el navegador (browser) para autorizar el clonado y posterior subida.

4. Fija usuario y email en el repo de trabajo:

```
git config user.name "XXXX"  
git config user.email "YYYY"
```

Nota: XXXX debe ser tu nombre y primer apellido. **YYYY** debe ser tu correo electrónico (el que vinculaste a tu cuenta de GitHub)

5. Clonar el repositorio privado creado para ti en la organización:
6. Lee detenidamente el readme.md del repositorio clonado pues ahí encontrarás, entre otras cosas, la rúbrica de evaluación.
7. Desarrolla un programa en JAVA haciendo uso de las clases disponibles en el repo clonado que cumpla con **los siguientes requisitos**:

Nota: No olvides ir haciendo *commit* conforme vayas avanzando en el desarrollo.

Debe procesar el fichero xml del hospital (hospital_2526.xml) y el fichero de texto con los nuevos médicos en plantilla (nuevo_medicos.txt). A partir de ellos deberá crear **dos nuevos ficheros**:

- a. `hospital_<ciudad>_<nomAlumno>_<ape1Alumno>.xml` que ha de validarse contra el acuerdo de interfaz proporcionado (`hospital_2526.xsd`)

Ej: hospital_Zaragoza_jose_sala.xml

- b. `hospital_<ciudad>_<nomAlumno>_<ape1Alumno>.json`.

Ej: hospital_Zaragoza_jose_sala.json

Nota1: Si no se cumple el formato, las pruebas unitarias (UTs) no se completarán satisfactoriamente.

Nota2: El nombre de la ciudad debe recuperarse del XML, nunca hardcodearla.

Nota3: En ambos ficheros deben constar TODOS los médicos, es decir, los del XML original más los disponibles en el fichero de texto

Ejemplo de xml de salida:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Hospital xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="hospital_2526.xsd">
  <nombre>Hospital General Central</nombre>
  <ciudad>Zaragoza</ciudad>
  <Medico Apellido="Sanz Díaz" Nombre="Luisa" especialidad="Nefrología" id="M04" />
  <Medico Apellido="Pérez López" Nombre="Juan" especialidad="Neurología" id="M03" />
  <Medico Apellido="Pin Delgado" Nombre="Pedro" especialidad="Traumatología" id="M05" />
  <Medico Apellido="Martínez Ruiz" Nombre="Carlos" especialidad="Pediatria" id="M002" />
  <Medico Apellido="Pérez Gómez" Nombre="Laura" especialidad="Cardiología" id="M001" />
  <Paciente Apellido="López Sánchez" Nombre="María" id="P100" seguro="Si" />
  <Paciente Apellido="García Torres" Nombre="Javier" id="P101" seguro="No" />
  <Citas>
    <Cita estado="Confirmada" fecha="2025-10-05" id="C001">
      <PacienteRef id="P100" />
      <MedicoRef id="M001" />
      <Motivo>Chequeo cardiológico anual</Motivo>
    </Cita>
    <Cita estado="Pendiente" fecha="2025-10-07" id="C002">
      <PacienteRef id="P101" />
      <MedicoRef id="M002" />
      <Motivo>Revisión pediátrica</Motivo>
    </Cita>
  </Citas>
</Hospital>
```

Ejemplo del fichero Json de salida:

```
{
  "nombre" : "Hospital General Central",
  "medicos" : [ {
    "id" : "M04",
    "especialidad" : "Nefrología",
    "Nombre" : "Luisa",
    "Apellido" : "Sanz Díaz"
  }, {
    "id" : "M03",
    "especialidad" : "Neurología",
    "Nombre" : "Juan",
    "Apellido" : "Pérez López"
  }, {
    "id" : "M05",
    "especialidad" : "Traumatología",
    "Nombre" : "Pedro",
    "Apellido" : "Pin Delgado"
  }, {
    "id" : "M002",
    "especialidad" : "Pediatría",
    "Nombre" : "Carlos",
    "Apellido" : "Martínez Ruiz"
  }, {
    "id" : "M001",
    "especialidad" : "Cardiología",
    "Nombre" : "Laura",
    "Apellido" : "Pérez Gómez"
  } ]
}
```

Entrega:

Cada alumno/a subirá el código al repositorio privado remoto creado con la asignación de GitHub Classroom el proyecto.

Nota: no es necesario subir el fichero json ni xml de salida, puesto que el profesor durante la evaluación, ejecutará el fat jar que genera maven para probar la funcionalidad con los requisitos indicados.

Criterios de evaluación:

- No hay errores de compilación.
- Cumple con los requisitos funcionales de la práctica.
- Se adelanta y gestiona posibles errores provocados por el usuario controlando número de parámetros, tipos, etc.

- Se entrega toda la información requerida, en el plazo establecido y con la nomenclatura y formato exigido.
- Se hace uso de las **buenas prácticas de programación** (evitando hardcodes, parametrizando la aplicación, optimizando el uso de recursos, modularizando funcionalidades en métodos y clases, reutilizando los recursos siempre que sea posible, organizando en clases la información y funcionalidades...)
- Se gestionan correctamente las posibles **excepciones** que se puedan disparar en el programa evitando su propagación desde el método main.
- Se añaden suficientes **trazas** (con nivel **error**, **warn**, **info**, **debug** o **trace**) a sus programas para facilitar futuras correcciones o cambios de alcance en la funcionalidad de los mismos.

La calificación de dicha práctica se tendrá en consideración para la evaluación de los siguientes **Resultados de Aprendizaje y Criterios de Evaluación**:

RA1. Desarrolla aplicaciones que gestionan información almacenada en ficheros identificando el campo de aplicación de los mismos y utilizando clases específicas.

- a) Se han utilizado clases para la gestión de ficheros y directorios.
- c) Se han utilizado clases para recuperar información almacenada en ficheros.
- d) Se han utilizado clases para almacenar información en ficheros.
- e) Se han utilizado clases para realizar conversiones entre diferentes formatos de ficheros.
- f) Se han previsto y gestionado las excepciones.
- g) Se han probado y documentado las aplicaciones desarrolladas.