Tarea online

Título de la tarea: Estructuras de datos externas (ficheros).

Unidad: 08

Ciclo formativo y módulo: DAM/DAW, Programación.

Curso académico: 2025/2026

¿Qué contenidos o resultados de aprendizaje trabajaremos?

Los resultados de aprendizaje que, de forma parcial, se van a trabajar con esta tarea serán:

- RA 5: Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

1.- Descripción de la tarea

Caso práctico

Ministerio de Educación. Uso educativo-nc. Elaboración propia.

La consultora *SweetWaters* Ltd. ha ganado la licitación de un proyecto a nivel nacional y uno de los puntos críticos es la próxima fecha de entrega. **Ada, la jefa de proyecto,** ha encargado la realización de las funciones de persistencia de datos a **Juan** y **María**.

Estas funciones de persistencia engloban cuatro funcionalidades que deberán realizar Juan y María: la gestión de un sistema de gestión de la plantilla docente de un instituto y la serialización de objetos. A raíz del poco tiempo de desarrollo que tienen surgen las dudas en el equipo de desarrollo. Y **María**, que ha visto a **Juan** muy nervioso tras el encargo de **Ada**, intenta tranquilizar a **Juan**:

—**Juan**, no te preocupes, no sabemos cómo hacerlo, pero podemos descubrir el cómo ;).

—Venga, no te agobies, recuerda que somos un equipo. Juntos avanzamos. —insiste **María**.

¿Qué te pedimos que hagas?

El objetivo fundamental de esta tarea consiste en trabajar con archivos, de manera que habrá que recuperar información desde disco, trabajar con ella y generar nuevos datos que serán también almacenados de manera externa a vuestros programas. Para ello utilizaremos los conceptos y herramientas que has aprendido sobre **serialización y persistencia**.

Se desarrollará un único ejercicio en un solo paquete, donde habrá solo una clase con su método `main()`, en un **único proyecto** NetBeans. En el apartado de información adicional tienes el proyecto base a partir del cual deberás construir la solución al ejercicio.

Ejercicio 1: Sistema de Gestión de la Plantilla Docente de un Instituto

La dirección de un centro educativo necesita automatizar la gestión de su profesorado. Actualmente, disponen de un archivo de texto plano con la información en bruto y desean un sistema en Java que sea capaz de procesar estos datos, generar informes formateados para impresión y realizar copias de seguridad de alta eficiencia mediante persistencia de objetos (serialización).

profesores

El alumno deberá implementar la lógica necesaria para realizar las siguientes operaciones de Entrada/Salida (I/O):

1. Lectura y Procesamiento del archivo de texto proporcionado

Se deberá leer el archivo de texto **ListadoProfesores.txt** ubicado en la carpeta /recursos/. Este archivo contiene una línea por profesor con los campos separados por punto y coma (;).

2. Generación del archivo de texto con la información leída de los profesores en el formato especificado.

Se debe generar un nuevo archivo de texto llamado **Instituto.txt** en un formato especificado.

3. Persistencia Binaria (Serialización)

Para garantizar la integridad de los datos y su rápida recuperación, el sistema debe:

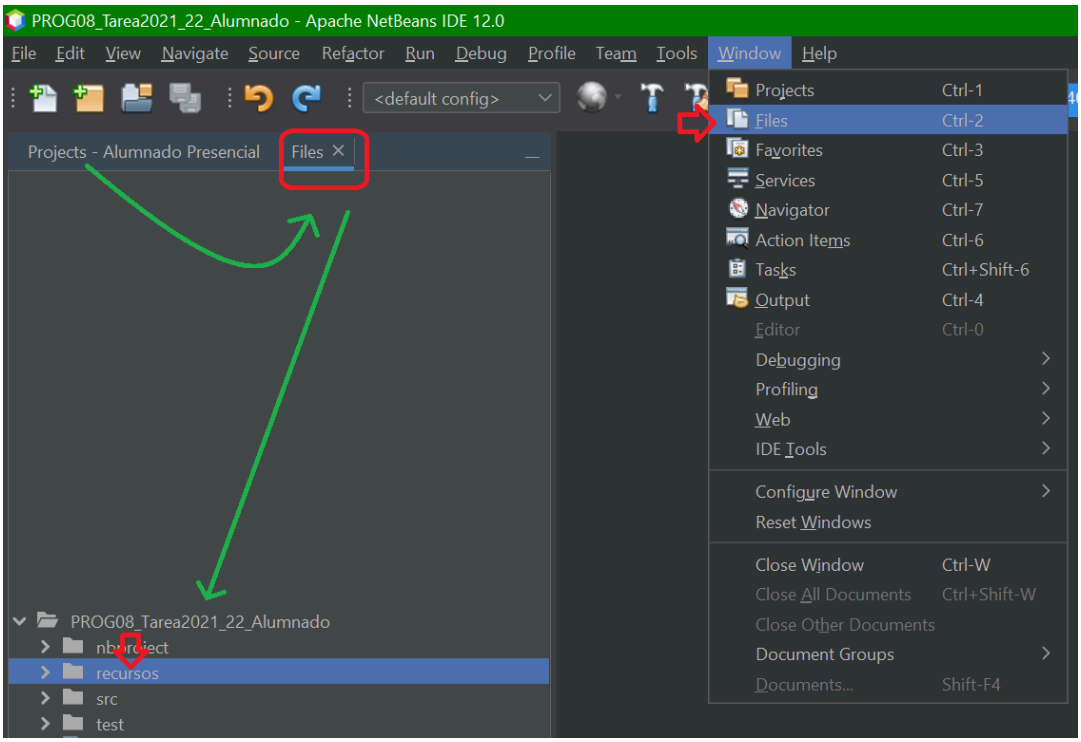
- Exportar: Guardar el objeto Instituto completo en un archivo binario llamado **Instituto.dat.**
- Importar: Leer el archivo **Instituto.dat** recién creado, reconstruir el objeto en memoria y mostrar su contenido por consola para verificar que la información es idéntica.

Parte 1. Lectura/Escritura de Profesores en ficheros de texto

En esta primera parte del ejercicio se va a generar un registro para el instituto con todos los profesores disponibles a partir de un **listado de profesores.** El listado se encuentra en el archivo de texto **`ListadoProfesores.txt`**.

El registro de datos que se debe generar, y que contendrá todos los profesores del archivo de texto, se almacenará en un segundo archivo que se llamará **`Instituto.txt`**.

Se proporciona una carpeta llamada **`recursos`** que se utilizará para el tratamiento de los ficheros de esta actividad y de las posteriores. En *NetBeans*, habitualmente se trabaja en la ventana ***Proyectos*** (**`Window > Projects Ctrl-1`**). Sin embargo, la carpeta **`recursos`** está visible desde la ventana ***Archivos*** (**`Window > Files Ctrl-2`**). En la siguiente imagen se muestra la carpeta y la interfaz de *Netbeans* correspondiente:



Germán Roche. *Interfaz de NetBeans: carpeta Recursos* (GNU/GPL)

Ten en cuenta que para poder acceder a esta ubicación, debemos hacer uso de la llamada al sistema **`System.getProperty("user.dir")`**, que nos devuelve la ruta absoluta a nuestro proyecto actual. De esta forma, como puedes observar en los ejercicios (**`main`**) que se aportan en el proyecto base, cada archivo se accede desde la carpeta **`recursos`** concatenada a dicha ruta del proyecto base:

```
String rutaProfesores = System.getProperty("user.dir") + "/recursos/ListadoProfesores.txt";
```

En el proyecto base que se entrega encontrarás dos clases creadas **`Instituto`** y **`Profesor`**. No debes modificar nada de estas clases en esta parte del ejercicio pero si debes hacer uso de ellas para resolver los apartados correspondientes de la tarea.

Esta será la secuencia ordenada de pasos que deberás seguir para la resolución del ejercicio:

1. Se crea un objeto de tipo **`Instituto`** para albergar cada uno de los profesores.
2. Se lee el archivo de texto **`ListadoProfesores.txt`**, que contiene **un profesor por línea.** Cada dato dentro del profesor se separa por el carácter **";"**.
3. Se extraen los datos de cada uno de los profesores: *nombre, especialidad, fecha de incorporación, departamento y listado de módulos.*
4. Se extraen los módulos de manera individual y se insertan en una lista. Cada módulo se separa por el carácter **",".**
5. Para cada profesor generamos un objeto de tipo **`Profesor`** con los datos extraídos en los puntos anteriores.
6. Una vez creado el objeto, se añade al objeto **`Instituto`**.
7. Cuando todos los profesores hayan sido incluidos en el **`Instituto`**, se genera su representación textual mediante el método **`toString()`** y se escribe cada profesor en el archivo de texto **`Instituto.txt`**, siguiendo el formato que se indica en el ejemplo de ejecución.

Formato y estructura de los archivos

A continuación, tienes una descripción detallada del formato de los archivos de texto con los que debes trabajar.

El archivo **`ListadoProfesores.txt`** contiene el listado de todos los profesores posibles.

Se trata de un archivo de texto con una línea por profesor. Aquí puedes observar su contenido:

```
Juan Martinez;Matemáticas;2015-07-15;Ciencias;Álgebra,Geometría,Trigonometría,Cá
Laura Sánchez;Filosofía;2019-09-30;Humanidades;Ética,Metafísica,Historia de la f
Pedro Gómez;Informática;2025-04-12;Tecnología;Programación,Redes,Seguridad infor
```

El formato del archivo **`Instituto.txt`** que se ha de generar es el siguiente:

```
********************************************************************************
LISTADO DE PROFESORES
********************************************************************************
```

```
NOMBRE DEL PROFESOR:Juan Martinez
ESPECIALIDAD:Matemáticas
FECHA DE INCORPORACIÓN:2015-07-15
DEPARTAMENTO:Ciencias
MÓDULOS:[Álgebra, Geometría, Trigonometría, Cálculo.]
********************************************************************************
NOMBRE DEL PROFESOR:Laura Sánchez
ESPECIALIDAD:Filosofía
FECHA DE INCORPORACIÓN:2019-09-30
DEPARTAMENTO:Humanidades
MÓDULOS:[Ética, Metafísica, Historia de la filosofía, Pensamiento crítico.]
********************************************************************************
NOMBRE DEL PROFESOR:Pedro Gómez
ESPECIALIDAD:Informática
FECHA DE INCORPORACIÓN:2025-04-12
DEPARTAMENTO:Tecnología
MÓDULOS:[Programación, Redes, Seguridad informática, Sistemas operativos.]
********************************************************************************
```

Parte 2. Persistencia de Objetos mediante Serialización en Java

Una vez realizada la primera parte del ejercicio, en esta segunda parte se debe implementar un mecanismo de **Serialización** para guardar el estado completo del objeto `Instituto` en un archivo físico y, posteriormente, ser capaz de recuperarlo (**deserializarlo**) para restaurar la información.

Para esta parte del ejercicio, deberás realizar las modificaciones oportunas sobre las clases `instituto` y `Profesor` que se aportan en el proyecto base, para que nos permitan realizar los procesos de **Serialización** y **Deserialización**.

Para realizar esta parte debemos implementar dos fases:

Fase A: Persistencia de Datos (Escritura)

En esta fase se deberá desarrollar una implementación para volcar el objeto `Instituto` en un archivo binario llamado `Instituto.dat`.

El proceso debe garantizar que, independientemente de si la operación tiene éxito o falla, los recursos del sistema queden liberados correctamente.

Es fundamental que el programa informe al usuario mediante la consola si el archivo se generó con éxito. Se debe prever y capturar cualquier error de entrada/salida que pueda interrumpir el proceso.

Fase B: Recuperación de Datos (Lectura)

En esta fase se implementará la lógica necesaria para leer el archivo `Instituto.dat`. generado previamente y se reconstruirá el objeto original en una variable objeto de tipo `Instituto`.

Tratamiento de Errores: El código debe ser capaz de identificar y responder de forma personalizada ante situaciones críticas de error:

1. Que el archivo de datos no se encuentre en la ruta especificada.
2. Que ocurra un error inesperado durante la transferencia de datos.
3. Que el sistema no reconozca el tipo de objeto que se está intentando importar.

Validación y Salida

Si el objeto se ha recuperado correctamente, se debe mostrar por pantalla toda su información detallada (haciendo uso del método de visualización por defecto de la clase).

El programa debe imprimir un mensaje final indicando que todos los procesos han concluido y los flujos de datos están cerrados.

Ejemplos de ejecución

A continuación se muestra una ejecución de cómo quedaría la salida por pantalla después de ejecutar el código de la `Parte 2`, sirve para probar que la implementación

correspondiente a la **Serialización** y **Deserialización** ha sido correcta.

A continuación, podéis observar el detalle de cada una de las acciones que se realizan.

```
Archivo serializable 'Instituto.dat' generado correctamente.

Archivo serializable leído correctamente.

Contenido del Instituto leído desde archivo serializable:

#Juan Martinez;Matemáticas;2015-07-15;Ciencias;[Álgebra, Geometría, Trigonometrí
#Laura Sánchez;Filosofía;2019-09-30;Humanidades;[Ética, Metafísica, Historia de
#Pedro Gómez;Informática;2025-04-12;Tecnología;[Programación, Redes, Seguridad i

Archivos serializados cerrados y procesamiento finalizado.

Fin del programa.
```

Y a continuación se presenta cómo quedaría la salida por consola de la ejecución completa de nuestra aplicación:

```
Abriendo archivo de profesores...
Cerrando archivo de profesores...

Abriendo archivo del instituto...
#Juan Martinez;Matemáticas;2015-07-15;Ciencias;[Álgebra, Geometría, Trigonometrí
#Laura Sánchez;Filosofía;2019-09-30;Humanidades;[Ética, Metafísica, Historia de
#Pedro Gómez;Informática;2025-04-12;Tecnología;[Programación, Redes, Seguridad i

Cerrando archivo del Instituto...

Archivos de texto cerrados y procesamiento finalizado
------------------------------------------------------

Archivo serializable 'Instituto.dat' generado correctamente.

Archivo serializable leído correctamente.

Contenido del Instituto leído desde archivo serializable:

#Juan Martinez;Matemáticas;2015-07-15;Ciencias;[Álgebra, Geometría, Trigonometrí
#Laura Sánchez;Filosofía;2019-09-30;Humanidades;[Ética, Metafísica, Historia de
#Pedro Gómez;Informática;2025-04-12;Tecnología;[Programación, Redes, Seguridad i

Archivos serializados cerrados y procesamiento finalizado.
```

```
Fin del programa.
```

2.- Información de interés

Recursos necesarios y recomendaciones

Recursos para la tarea:

mohamed Hassan (Licencia Pixabay)

- Se proporciona un proyecto base sobre el que debes trabajar: ProyectoBase (zip - 594,64 KB). En él tendrás incluido el paquetes **`ejercicio1`**, correspondiente a las las actividades con las que debes trabajar. **Es obligatorio utilizar este proyecto con sus clases y programas** .
- En la carpeta **`recursos`** se encuentra el archivo de texto con el listado de profesores.

Como consejos:

- **Indenta** tu código de forma adecuada. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "*formatear*" o "*embellecer*" y pulsa **`Alt+Mayús.+F`**).
- Procura respetar las convenciones de **nombrado de identificadores** indicadas en la unidad 1. Recuerda que en NetBeans puedes renombrar rápidamente una variable, un método o una clase, situando el cursor encima del nombre y pulsando **`Ctrl+R`**.

Indicaciones de entrega

Una vez realizada la tarea, el envío se realizará a través de la plataforma. Renombra el proyecto NetBeans (y la carpeta que lo contiene) siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG_Tarea08

A continuación, comprime la carpeta del proyecto NetBeans en un fichero **`.zip`** y súbelo en el área de entrega de la tarea también con ese nombre.

3.- Evaluación de la tarea

Criterios de evaluación implicados

Del RA5 (*Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases*):

- a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.
- b) Se han aplicado formatos en la visualización de la información.
- c) Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.
- d) Se han utilizado ficheros para almacenar y recuperar información.
- e) Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

Peggy_Marco (Pixabay License)

¿Cómo valoramos y puntuamos tu tarea?

Cada uno de los CE cubiertos por esta tarea será evaluado a través de los criterios de corrección y puntuación indicados detalladamente en la rúbrica de la tarea que puedes consultar en la plataforma.

Rúbrica de la tarea			
(RA5.a) Ejercicio1 Parte 1. Se ha utilizado la consola para realizar operaciones de entrada y salida de información.			
No se realiza ninguna operación de entrada/salida en consola. **0 puntos**	Se realizan operaciones mínimas e insuficientes para entender el flujo del programa. **0.25 puntos**	Las operaciones de entrada/salida permiten seguir el programa, aunque carecen de formato o claridad. **0.5 puntos**	Las operaciones de entrada/salida son claras, bien organizadas y ayudan al usuario a comprender el flujo del programa. **1 punto**
(RA5.b) Ejercicio1 Parte 1. Se han aplicado formatos en la visualización de la información.			

No se aplica ningún formato al archivo de salida instituto.txt **0 puntos**	Se aplica formato al archivo instituto.txt pero no tal cual se muestra en la descripción de la tarea. **0.5 puntos**	Se genera correctamente el archivo instituto.txt con el formato requerido. **1 punto**

(RA5.c) Ejercicio1 Parte 1. Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.

No se emplean clases adecuadas para entrada/salida de ficheros. **0 puntos**	Se usan clases básicas para entrada/salida, pero con errores o sin aprovechar sus funcionalidades. **0.25 puntos**	Se emplean clases estándar de Java como FileReader y FileWriter, aunque no de forma óptima. **0.75 puntos**	Se emplean correctamente clases para entrada/salida de datos. **1 punto**

(RA5.d) Ejercicio1 Parte 1. Se han utilizado ficheros para almacenar y recuperar información.

No se utiliza almacenamiento o recuperación de datos en ficheros **0 puntos**	Solo se realiza una de las operaciones: almacenar o recuperar datos **0.3 puntos**	Se almacenan y recuperan datos correctamente. Se genera correctamente el fichero instituto.txt. **1 punto**

(RA5.e) Ejercicio1 Parte 1. Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

No se implementen métodos de acceso a ficheros. **0 puntos**	Se implementan métodos de acceso a ficheros pero con errores. No se lee correctamente del fichero ListadoProfesores.txt. **0.15 puntos**	Se implementan métodos de acceso a ficheros, se lee correctamente del fichero ListadoProfesores.txt pero, no se generan correctamente los objetos ni se añaden al objeto instituto. **0.4 puntos**	Se implementan métodos de manera correcta para acceder al contenido de los ficheros. **1 punto**

(RA5.a,RA5.c) Ejercicio1 Parte 2. Se accede correctamente al fichero binario para serializar y deserializar.

No se emplean clases adecuadas para entrada/salida de ficheros. **0 puntos**	Se usan clases básicas para entrada/salida, pero con errores o sin aprovechar sus funcionalidades. **0.5 puntos**	Se emplean clases de Java para serializar y deserializar como ObjectInputStream y ObjectOutputStream, aunque no de forma óptima. **1 punto**	Se emplean correctamente clases para entrada/salida de datos. **1.5 puntos**

(RA5.d) Ejercicio1 Parte 2. Se han utilizado ficheros para almacenar y recuperar información.		
No se emplea la clase XStream para entrada/salida de ficheros. **0 puntos**	Solo se realiza una de las operaciones con el archivo binario instituto.dat: almacenar o recuperar datos **0.75 puntos**	Se almacenan y recuperan datos correctamente. Se genera correctamente el fichero instituto.dat. **1.5 puntos**

(RA5.e) Ejercicio1 Parte 2. Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.			
No se implementen métodos de acceso a ficheros. **0 puntos**	Se implementan métodos de acceso a ficheros binarios pero con errores. No se lee correctamente del fichero instituto.dat **0.15 puntos**	Se implementan métodos de acceso a ficheros binarios, se escribe correctamente en el fichero instituto.dat pero, no se reconstruye correctamente el objeto instituto. **0.4 puntos**	Se implementan métodos de manera correcta para acceder y recuperar al contenido del fichero instituto.dat de manera correcta. **1 punto**

(RA5.b) Ejercicio1 Parte 2. Se han aplicado formatos en la visualización de la información.		
No se muestra la información recuperada del archivo del archivo binario instituto.dat. **0 puntos**	Se muestra la información recuperada del archivo del archivo binario instituto.dat pero no se utiliza el método “toString()” de la clase Instituto. **0.5 puntos**	Se muestra la información recuperada del archivo del archivo binario instituto.dat correctamente. **1 punto**

Aquí tienes algunas cuestiones generales que podrán penalizarte en la evaluación de los distintos apartados de la tarea. Tenlos muy en cuenta al resolverlos y comprobar su funcionamiento:

- **Las clases y los programas de prueba deben compilar.** Cualquier funcionalidad pedida en el enunciado debe poderse probar y ha de funcionar correctamente de acuerdo a las especificaciones del enunciado. **Si la clase o el programa que la prueba ni siquiera compila, el ejercicio podría considerarse nulo y su puntuación podría llegar a ser 0.**
- **La ejecución de los programas de prueba que usan la clase no debe romperse**. Si eso sucede es porque la clase no ha sido implementada correctamente con respecto a los requisitos indicados en el enunciado.
- **Se utilizan estructuras de salto incondicional para el control del flujo**, o bien herramientas como `return`, `System.exit()` o cualquier otro mecanismo que no sea el fin de la ejecución natural del programa.
- **Se declaran identificadores que no cumplen con el convenio sobre "asignación de nombres a identificadores"** establecido para el lenguaje Java para constantes, variables,

métodos, clases, etc.
- **Se declaran identificadores que no tienen nombres significativos o descriptivos** que representen de alguna manera la información que están almacenando para que el código quede lo más claro, legible y autodocumentado posible.
- **No se observa una corrección ortográfica y gramatical,** así como la coherencia en las expresiones lingüísticas, tanto en los comentarios, en el código como en los textos de los mensajes que aparezcan en pantalla para pedir información de entrada al usuario, mostrar resultados de salida o indicar situaciones de error. Deben evitarse mensajes de entrada de datos, salida de resultados inapropiados y/o error, descontextualizados, insuficientes o incorrectos.
- **El código no está correctamente indentado**. Es fundamental para poder observar e intuir rápidamente, y de forma visual, la estructura de los programas. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "formatear" o "embellecer" y **pulsa Alt+Mayús.+F**).

A partir de ahí, se calculará la nota de cada CE normalizando sobre una escala **de 0 a 10** teniendo en cuenta la puntuación obtenida en cada punto de control o elemento de evaluación y su peso en la tarea para cada CE.

Se valorará especialmente en cada uno de los apartados:

- que la respuesta sea personal y no copiada de Internet o de otras fuentes;
- que sea coherente con el resto de respuestas y con los conceptos de la unidad;
- que muestre que se comprenden adecuadamente los conceptos tratados.

IMPORTANTE: Motivos de devolución de la tarea o de calificación mínima

La tarea obtendrá automáticamente una calificación de **cero** si:

- No se entrega **un único proyecto** que incluya los dos ejercicios que se piden en la tarea utilizando el IDE NetBeans.
- Alguno de los programas presenta **errores de compilación** impidiendo su ejecución y prueba.
- **No se respeta el proyecto base** que se entrega al alumno para realizar la tarea.
- Se modifican las clases `Instituto` y `Profesor` de manera no requerida. Sólo se podrán modificar para la parte 2 del ejercicio y de forma adecuada.
- Se implementan más archivos fuente además del proporcionado en el proyecto base (`Ejercicio1.java`).
- Se utilizan **contenidos** en alguno de los ejercicios del proyecto **que no se han estudiado** en la unidad actual o anteriores a la misma.
- Se presenta un proyecto total o parcialmente copiado de otro alumno.

Recordatorio: Si una tarea obtiene calificación cero por alguno de los motivos previamente mencionados, será devuelta al alumno. Este tendrá derecho a una segunda entrega únicamente en los siguientes casos:

- Que se trate de la primera vez que presenta la tarea.
- Que la fecha de entrega inicial haya sido al menos una semana antes de la fecha límite establecida para dicha tarea.