



Informàtica

ICB0 Desenvolupament d'aplicacions multiplataforma

M06 Accés a dades

UF1 Persistència en fitxers

Diari d'activitats

Isidre Guixà

Curs 2024/25



Gestió de fitxers		
<i>Gestió del sistema de fitxers</i>	Projecte	Data
Apartat 1.1 de DAM M06 UF1 Fitxers 1 Fluxos&Seriación.pdf <i>Introducció bàsica a la gestió de fitxers. Treballat a UF5 del M03.</i> Classe File - Veure documentació oficial <ul style="list-style-type: none"> - Pertany al paquet <code>java.io</code>. És un paquet que existeix des de la primera versió de Java. - Donat que al llarg del temps la gestió de fitxers ha evolucionat, Java 7 va publicar el paquet <code>java.nio</code> que conté classes més sofisticades que faciliten funcionalitats més adequades per a certes necessitats. En principi no les veurem, per cal ser-ne conscient. - Mètodes interessants: Constructors <code>exists, isFile, isDirectory, createNewFile, delete</code> <code>getName, getParent, getAbsolutePath, getPath, lastModified</code> <code>mkdir, mkdirs, listFiles</code> - Camps a usar per aconseguir independència del S.O: <code>pathSeparator, separator, pathSeparatorChar, separatorChar</code> 		
Exercici: Fer un programa que comprovi l'existència de les rutes: <code>X:\programes\pepe.txt, C:\Windows, C:\Windows\regedit.exe, C:\fontanals.txt, D:\fontanals.txt</code> ➤ En cas d'existir, informi si es tracta d'un fitxer o una carpeta. ➤ En cas de no existir, intenti crear-la com a fitxer i informi del resultat	240916_1...	16/09/24
Exercici per dimecres, 18/09/2024 Fer un programa que elimini el fitxer <code>D:\fontanals.txt</code> creat en exercici anterior	240918_1...	18/09/24
Exercici per dimecres, 18/09/2024 Considereu la ruta <code>A\B\Carpeta</code> on suposem que <code>A</code> no existeix. Fer un programa que intenti crear <code>Carpeta</code> usant <code>mkdir</code> i <code>mkdirs</code> . Una vegada creada, comprovar el resultat dels mètodes <code>getName, getParent, getAbsolutePath, getPath</code> i <code>lastModified</code> aplicats sobre la ruta indicada.	240918_2...	18/09/24
Exercici per dimecres, 18/09/2024 Fer un programa que mostri el contingut de totes les carpetes i fitxers existents en el projecte. Cal fer un procés recursiu: Versió P01: Fa recorregut per la carpeta arrel, mostrant els seus elements: si és un fitxer, el mostra i si és una carpeta, invoca el mètode de forma recursiva. D'aquesta manera és "complicat" veure el contingut d'una carpeta que conté fitxers i carpetes. És fa necessari algun mecanisme de tabulació per veure la jerarquia. Versió P02 (similar a <code>dir /s</code> de MS-DOS): Fa recorregut per la carpeta arrel mostrant tot el seu contingut (fitxers i carpetes) i, posteriorment, per cada carpeta invoca el mètode de forma recursiva. Facilita veure el contingut de cada carpeta.	240918_3...	18/09/24
Gestió de fitxers		
<i>Gestió del contingut dels fitxers</i>	Projecte	Data
A 1r curs s'ha usat les classes: <ul style="list-style-type: none"> - <code>Scanner</code> i <code>PrintStream</code> per a gestionar el contingut de fitxers de text - <code>RandomAccessFile</code> per a gestionar el contingut de fitxers binaris En aquesta UF coneixerem altres classes per a la gestió del contingut de fitxers		
Apartat 1.2 de DAM M06 UF1 Fitxers 1 Fluxos&Seriación.pdf <i>Fluxos orientats a dades (binaris)</i>		18/09/24



Exercici per dijous, 19/09/2024

Gestió de fitxers binaris utilitzant `FileOutputStream` i `FileInputStream`

- Programa que generi fitxer binari d'enters amb els primers 100 enters estrictament positius... I posterior programa que n'efectuï la lectura (sense saber la quantitat d'enters que conté)

En enregistrar els valors binaris cal efectuar les conversions adequades a taules de bytes en funció del tipus de dada!!!

Per conversions entre dades i bytes, donar una ullada a la classe `ByteBuffer`, que serveix de pont per passar de tipus primitius a `byte[]` i a l'inrevés. Mètodes de `ByteBuffer` a tenir en compte:

- `ByteBuffer.allocate` per crear un `ByteBuffer` amb una quantitat determinada de bytes.
 - `array` per passar el contingut d'un `ByteBuffer` a una taula de mateixa dimensió
 - `ByteBuffer.wrap` per crear un `ByteBuffer` a partir d'una taula
 - `putInt`, `putLong`, `put...` per emplenar un `ByteBuffer` amb una dada de tipus adequat
 - `getInt`, `getLong`, `get...` per recuperar d'un `ByteBuffer` una dada de tipus adequat.
- Repetiu els programes (generació i recuperació) usant la classe `RandomAccessFile` que és de més alt nivell, doncs té mètodes per llegir i escriure dades sense haver de fer conversions a bytes i facilita la feina del programador.

El projecte 240919_1... incorpora:

- Programes P01 que genera i P02 que recupera. El programa P01 enregistra els enters com a byte i sembla que és correcte (funciona) per què els valors "cabem" en un byte (estan entre 0 i 255), però no fa exactament el què es demanava, doncs no enregistra "enters" sinó "bytes". El programa P02 de lectura funciona per què també fa la "trampa" de llegir bytes. Proveu d'enregistrar els enters entre 201 i 300 i recuperar-los. També serien 100 enters però no quedarien ben enregistrats i en recuperar-los observem que recupera del 201 al 255 i a continuació del 0 al 44.
- A més, si es revisa l'ocupació del fitxer en bytes, ens dirà que són 100 quan hauria de ser 400.

En enregistrar els valors binaris cal efectuar les conversions adequades a taules de bytes en funció del tipus de dada!!!

- Programes P03 que genera i P04 que recupera correctament.
- Programes P05 que genera i P06 que recupera usant la classe `RandomAccessFile` usada a 1r i que és de més alt nivell i facilita la feina del programador.

Es pot comprovar que el fitxer generat amb P03 o P05 es pot recuperar indistintament amb P04 o P06. És a dir, si es genera amb `FileOutputStream` no s'està obligat a recuperar amb `FileInputStream` i si es genera amb `RandomAccessFile` no s'està obligat a recuperar amb `RandomAccessFile`.

ALERTA

- El constructor `FileOutputStream` usat (només 1 paràmetre) recrea el fitxer en cas d'existir. Si es vol afegir, cal usar constructor amb 2n paràmetre `append`.
- En canvi, `RandomAccessFile` no té la possibilitat d'indicar afegir ni tampoc recrear el fitxer, sinó que sobreescriu al damunt, de manera que si s'escriu menys informació que l'existent inicialment, el fitxer contindrà la nova informació seguida del residu de la informació original. Per això, en el programa P05, prèviament es comprova l'existència del fitxer i, en cas d'existir, l'elimina (avortant si no ho aconsegueix).
- És fonamental tancar els fitxers quan ja no es necessita treballar amb ells i, en tot cas, abans de finalitzar el programa.

240919_1...

19/9/24



<p>- El programa P03 facilita 2 versions respecte a obertura/tancament de fitxer:</p> <p>v1: Intenta obrir el fitxer.</p> <ul style="list-style-type: none">- Si no pot, avisa i el programa finalitza i no arribarà a intentar tancar el fitxer.- Si pot, el programa continua i abans de finalitzar, tanca el fitxer. <p>v2: Tot el codi (obertura i tractament de les dades) està dins un <code>try... catch...</code>. En aquest cas, abans de finalitzar el programa, cal tancar el fitxer i ho posem dins <code>finally...</code>. Però en entrar en <code>finally</code>, pot ser:</p> <ul style="list-style-type: none">- Que no s'hagués obert el fitxer. No s'ha de fer el <code>close</code>!- Que s'hagués obert el fitxer. Cal fer el <code>close</code>! <p>La manera de saber-ho és preguntant si <code>fis==null</code>.</p>																						
<p>Apartat 1.3 de DAM M06 UF1 Fitxers 1 Fluxos&Seriació.pdf</p> <p><i>Fluxos orientats a caràcter (text)</i></p> <p>Exercici per dimecres, 25/09/24</p> <p>Gestió de fitxers textuais utilitzant <code>FileReader</code> i <code>FileWriter</code></p> <p>Segui ST una seqüència de dades caràcter del tipus <code><L><valor><espai></code> on L és una lletra identificadora del tipus de valor numèric:</p> <p>B-byte S-short I-int F-float L-long D-double</p> <ul style="list-style-type: none">➤ Programa que demani a l'usuari la introducció d'un caràcter (B)yte, (S)hort, (I)nteger, (L)ong, (F)loat, (D)ouble o \$ per finalitzar i, segons el caràcter introduït, demani el corresponent valor, per tal de generar un fitxer de text amb una seqüència ST indicada.➤ Programa que recuperi la informació existent en un fitxer de text amb una seqüència ST com a contingut i mostri per pantalla, per a cada tipus de dades trobada en el fitxer: Valors de tipus ... : els valors ordenats ascendent <p>És a dir, si dins el fitxer hi ha I22 S44 L9292 I-30 B22 S-30 I99 L8080, la sortida haurà de ser:</p> <p>Valors de tipus byte: 22 Valors de tipus short: -30, 40 Valors de tipus int: -30, 22 Valors de tipus long: 8080, 9292</p>			240925_1...	23/09/24 25/09/24																		
<p>Exercici EXTRA (no es resoldrà a classe – es publicarà la solució 26/09/24)</p> <ul style="list-style-type: none">➤ Programa que generi fitxer binari d'enters amb els primers 100 enters estrictament positius, enregistrant els parells com a <code>short</code> i els senars com a <code>int</code>. Alerta: El fitxer haurà d'ocupar 300 bytes (50 parells x 2 + 50 senars x 4)➤ Programa que llegeixi enters d'un fitxer binari amb contingut segons norma anterior, sabent que comença per valor <code>int</code> (no tenim per què saber quans enters té el fitxer) <p>Versió 1 usant <code>FileOutputStream</code> i <code>FileInputStream</code>. Versió 2 usant <code>RandomAccessFile</code>.</p>			240926_1...																			
<p>Accés des de C/C++ a fitxers binaris generats per Java</p> <p>Recordatori de funcions de C per gestió de fitxers i comparativa amb classes de Java:</p> <table><tr><td></td><td></td><td>Java (algunes classes)</td><td>C (algunes funcions)</td></tr><tr><td rowspan="2">Fitxers Text</td><td>Lectura</td><td>Scanner FileReader</td><td>fscanf, fgetc, fgets</td></tr><tr><td>Escriptura</td><td>PrintStream FileWriter</td><td>fprintf, fputc, fputs</td></tr><tr><td rowspan="2">Fitxers Binaris</td><td>Lectura</td><td>RandomAccessFile FileInputStream</td><td>fread</td></tr><tr><td>Escriptura</td><td>RandomAccessFile FileOutputStream</td><td>fwrite</td></tr></table>					Java (algunes classes)	C (algunes funcions)	Fitxers Text	Lectura	Scanner FileReader	fscanf, fgetc, fgets	Escriptura	PrintStream FileWriter	fprintf, fputc, fputs	Fitxers Binaris	Lectura	RandomAccessFile FileInputStream	fread	Escriptura	RandomAccessFile FileOutputStream	fwrite		25/09/24
		Java (algunes classes)	C (algunes funcions)																			
Fitxers Text	Lectura	Scanner FileReader	fscanf, fgetc, fgets																			
	Escriptura	PrintStream FileWriter	fprintf, fputc, fputs																			
Fitxers Binaris	Lectura	RandomAccessFile FileInputStream	fread																			
	Escriptura	RandomAccessFile FileOutputStream	fwrite																			



<p>En C, a més, cal usar les següents funcions:</p> <ul style="list-style-type: none"> - <code>fopen</code> per obrir fitxer, on recordeu que cal indicar el mode del fitxer (text-binari) - <code>fclose</code> per tancar fitxer. - <code>feof</code> per saber si s'ha arribat al final del fitxer després d'efectuar lectura <p>Explicació detallada de les funcions en C, a https://cplusplus.com/reference/cstdio/ Cal incloure en el programa <code>#include <stdio.h></code></p> <p>Exercici: Programa en C/C++ que mostri per pantalla el contingut del fitxer binari generat pels programes P03/P05 del projecte 240919_1_Gestio_FileInputStream_FileOutputStream.</p> <p>Solució:</p> <ul style="list-style-type: none"> - El projecte 240925_2 mostra un intent de solució que sembla correcte però... el resultat no és correcte. Per què? - La causa rau en que Java i C enregistren/gestionen els bytes en ordre diferent, problema que es coneix en anglès com Endianness. Més informació: https://ca.wikipedia.org/wiki/Ordre_dels_bytes https://howtodoinjava.com/java/basics/little-endian-big-endian/ https://cs-fundamentals.com/tech-interview/c/c-program-to-check-little-and-big-endian-architecture - En conclusió, Java sempre usa <code>big-endian</code> i C/C++ usa l'ordre de bytes de l'arquitectura de la CPU, que en Intel acostuma a ser <code>little-endian</code> i, per tant, apareix un problema per recuperar valors enregistrats en ordre de bytes diferent (nostre cas) <p>El projecte 240925_3 conté programa Java que informa de l'ordre de bytes que usa la CPU. El projecte 240925_4 conté programa C que informa de l'ordre de bytes que usa la CPU.</p> <ul style="list-style-type: none"> - Per recuperar valors des de C/C++ si la CPU treballa en <code>little-endian</code> (nostre cas) caldrà llegir els bytes corresponent al valor en una taula i invertir-los per aconseguir el valor correcte i per fer-ho hem d'usar operadors a nivell de bit: https://en.wikipedia.org/wiki/Bitwise_operations_in_C <p>En aquesta pàgina trobem mètode per convertir <code>big-endian</code> a <code>little-endian</code> en C:</p> <pre>// Unsigned 16 bit conversion: swapped = (num>>8) (num<<8); // Unsigned 32-bit conversion: swapped = ((num>>24)&0xff) // move byte 3 to byte 0 ((num<<8)&0xff0000) // move byte 1 to byte 2 ((num>>8)&0xff00) // move byte 2 to byte 1 ((num<<24)&0xff000000); // move byte 0 to byte 3</pre> <p>El projecte 240925_5 conté programa C per la nostra arquitectura (<code>little-endian</code>) de màquina que recupera el contingut del fitxer generat per Java de forma correcta.</p>	
<p>Apartats 1.4 i 1.4.1 de DAM_M06_UF1_Fitxers_1_Fluxos&Seriació.pdf</p> <p><i>Modificadors de fluxos - Fluxos de tipus de dades</i></p> <p>Per utilitzar un modificador de flux, igualment cal tenir el flux original creat. És a dir, per enregistrar en un fitxer binari, podem usar un <code>DataOutputStream</code> que ha d'estar vinculat a un <code>FileOutputStream</code> i el mateix amb un <code>DataInputStream</code> vinculat amb un <code>FileInputStream</code>. Però no és necessari tenir les referències als fluxos originals, ja que en tancar el modificador de flux, també es tanca el flux originals vinculat.</p>	
<p>Exercici per dijous, 26/09</p> <p>Gestió de f. binaris via modificadors de flux <code>DataInputStream/DataOutputStream</code></p> <p>Refer P03 i P04 del projecte 240919_1_Gestio_FileInputStream_FileOutputStream que usen <code>FileInputStream</code> i <code>FileOutputStream</code>, utilitzant modificadors de flux.</p>	<p>240926_2...</p> <p>26/09/24</p>

<p>Fitxers XML via JDOM (no és l'acrònim de <i>Java Document Object Model</i> !!!)</p> <p>JDOM és, simplement, una representació de Java d'un document XML. JDOM proporciona una forma de representar aquest document per a una lectura, manipulació i escriptura fàcils i eficients. Té un API senzill, és lleuger i ràpid, i està optimitzat per al programador Java. És una alternativa a DOM i SAX existents dins JDK (DOM s'usa a M03-UF5) i s'integra bé amb DOM i SAX gràcies a:</p> <ul style="list-style-type: none"> - Classe <code>DOMOutputter</code> per passar un <code>JDOM.Document</code> a <code>DOM.Document</code>. - Classe <code>DOMBuilder</code> per obtenir un <code>JDOM.Document</code> a partir de <code>DOM.Document</code>. - Classe <code>SAXOutputter</code> per passar un <code>JDOM.Document</code> a flux per ser gestionat via SAX <p>JDOM va ser creat en el 2000 per Jason Hunter i Brett McLaughlin. No forma part del JDK i cal baixar-lo del lloc oficial JDOM (a data 26/09/2024 no accessible). A banda de la informació que facilita el lloc oficial, podeu trobar altres articles a la web, com per exemple: aquest, aquest, ... i també es pot trobar el codi a GitHub.</p> <p>Per instal·lar-lo, baixar darrera versió (jdom-2.0.6.1.zip en el moment actual) i que trobareu al Classroom.</p> <p>La descompressió genera diversos jar. Interessa:</p> <ul style="list-style-type: none"> - <code>jdom-2.0.6.1.jar</code> - <code>jdom-2.0.6.1-javadoc.jar</code> <p>Per utilitzar el paquet JDOM cal tenir accés a la llibreria <code>jdom-2.0.6.1.jar</code></p> <p>Tots els exemples que acompanyen aquests materials i que utilitzin JDOM, pressuposen l'existència d'una llibreria en el Netbeans anomenada exactament <code>JDom 2.0.6.1</code></p> <p>A més, per esbrinar si un fitxer XML conté etiqueta <code><!DOCTYPE</code> per procedir a validar, Java no disposa de cap utilitat (caldrà gestionar fitxer de text... uff) i ens podem aprofitar de la classe <code>FileUtils</code> de Apache Commons IO. La darrera versió (a data 26/09/2024) és la 2.17.0. Per utilitzar la classe <code>FileUtils</code> cal tenir accés a la llibreria <code>commons-io-2.x.jar</code>.</p> <p>Els exemples que acompanyen aquesta materials i que utilitzin Commons IO, pressuposen l'existència d'una llibreria en el Netbeans anomenada exactament <code>Apache Commons IO 2.17.0</code> contenint la versió 2.17.0.</p>		26/09/23
<p>Exemples XML-JDOM basats en fitxer <code>books.xml</code> que trobareu a l'arrel del ZIP de l'assignatura</p> <ol style="list-style-type: none"> 1. Lectura de document XML de disc: <code>Ex01_ComptarXML.java</code> <p>Programa que compta quants títols, autors, anys i preus hi ha en el fitxer <code>books.xml</code> ubicat en la carpeta que conté el projecte</p> <ol style="list-style-type: none"> 2. Enregistrament de document XML a disc: <code>Ex02_CrearXML.java</code> <p>Programa que crea document XML com <code>books.xml</code> i l'enregistra a fitxer</p> <ol style="list-style-type: none"> 3. Modificació de document XML: <code>Ex03_ModificarXML.java</code> <p>Programa que parteix del document <code>books.xml</code> i en crea un de nou amb les següents modificacions:</p> <ol style="list-style-type: none"> 1. Modificar a tots els llibres, l'atribut <code>category</code> a majúscules 2. Augmentar els preus un 2% 3. Afegir un nou element <code>editorial</code> a cada llibre 4. Eliminar l'element <code>year</code> 5. Eliminar l'atribut <code>lang</code> del títol de cada llibre 6. Afegir l'atribut <code>sex</code> a cada autor 	240926_3...	26/09/24 30/09/24
<p>Exercici per dimecres, 2 d'octubre</p> <p>Programa que calcula la població europea a l'arxiu <code>mondial.xml</code> ubicat en la carpeta que conté el projecte. Es facilita <code>mondial.xml</code> – <code>mondial.dtd</code> – <code>mondial.xsd</code> dins el ZIP.</p> <p>Observacions a tenir en compte:</p>	241002_1...	02/10/24



<ul style="list-style-type: none"> - Una ullada sobre alguns països ens mostra que per aconseguir la informació que se'ns demana, caldrà tenir en compte l'element <code>encompassed de country</code>, que ens informa del continent al què pertany el país i l'element <code>population</code>, que ens informa de la població. - Segons DTD, un país pot tenir varis elements <code>encompassed</code> i això té sentit per què hi ha països, com Turquia, que tenen el seu territori en diversos continents, i l'atribut <code>percentage</code> indica quina part del territori està en cada continent. Caldrà usar aquest percentatge per calcular quina part de la població del país correspon al continent pel que estem calculant la població. És a dir, si un país te 1.000.000 d'habitants i té un 60% de territori a Europa, considerarem que el número d'habitants que estan a Europa és $1.000.000 * 60/100$. - Segons DTD, un país pot tenir varis elements <code>population</code>, per donar informació de la població en diversos anys. Caldrà considerar la població de l'any més actual i, en cas que per un mateix any hi hagi diverses mesures (atribut <code>measured</code>), caldrà considerar aquell que tingui el valor "census". - Segons DTD, l'atribut <code>year</code> de l'element <code>population</code> no és obligatori, però segons XSD, sí que ho és... Ens creiem el XDS... i no cal tenir en compte que <code>year</code> pugui ser null... 		
NF2 – Persistència d'objectes en fitxers		
<p>Presentació de les classes a fer persistents</p> <p>Considerar el model implementat en el projecte <code>000000_1_Model</code> ideat per gestionar una biblioteca. i que consta de:</p> <ul style="list-style-type: none"> • Classe abstracta <code>Fitxa</code>, amb definició de dades comunes a totes les tipologies de fitxa que pot tenir una biblioteca. • Classe <code>FitxaLlibre</code> amb la definició de les dades específiques de llibre. • Classe <code>FitxaRevista</code> amb la definició de les dades específiques de revista. • Classe <code>FitxaException</code> per gestionar les excepcions sobre objectes <code>Fitxa</code>. • Interfície <code>IBiblioteca</code> que defineix les operacions habituals en una biblioteca • Classe abstracta <code>Biblioteca</code> que defineix el contingut que es gestiona dins la biblioteca <ul style="list-style-type: none"> ◦ Nom ◦ Col·lecció d'objectes <code>Fitxa</code> amb referència i l'ordre únics i permanentment ordenada per referència. <p>i que implementa <code>IBiblioteca</code> i que a més, conté constructor, fet que obliga que les classes derivades tinguin obligatòriament un constructor.</p> <ul style="list-style-type: none"> • Classe <code>BibliotecaException</code> per gestionar les excepcions sobre <code>Biblioteca</code>. <p>El projecte <code>000000_2_BibliotecaAL</code> és una implementació de <code>Biblioteca</code> utilitzant una <code>ArrayList</code> per gestionar la col·lecció de fitxes.</p> <p>El projecte <code>000000_3_BibliotecaTS</code> és una implementació de <code>Biblioteca</code> utilitzant un <code>TreeSet</code> per gestionar la col·lecció de fitxes.</p> <p>El projecte <code>000000_4_ExempleProgramaBiblioteca</code> conté un programa exemple de gestió d'una biblioteca, preparat amb dues configuracions d'execució:</p> <ul style="list-style-type: none"> - <code>BibliotecaAL</code> que utilitza la implementació <code>Biblioteca</code> del projecte <code>000000_2...</code> - <code>BibliotecaTS</code> que utilitza la implementació <code>Biblioteca</code> del projecte <code>000000_3...</code> <p>Observar que el programa <code>Proves</code> és únic i, en funció de paràmetre, ha de poder instanciar <code>BibliotecaAL</code> o <code>BibliotecaTS</code> segons paràmetre i, aquestes classes no poden estar en un <code>!import</code>, doncs en una biblioteca real on usessin el programa, usarien una implementació concreta (<code>BibliotecaAL</code> o <code>BibliotecaTS</code> o alguna altra) i en aquella instal·lació només hi hauria la implementació necessària. Per tant, es necessita que <code>Proves</code> carregui la implementació de <code>Biblioteca</code> que correspongui segons paràmetre i es fa necessari l'ús de:</p> <ul style="list-style-type: none"> - <code>Class.forName(nomClasse).newInstance()</code>, si invoquem constructor sense paràmetres de la classe a carregar. - <code>Constructor c = Class.forName(nomClasse).getConstructor(llistaTipusParàmetres);</code> <code>c.newInstance(llistaValorsParàmetres)</code>, si invoquem constructor amb paràmetres 	<p>000000_1... 000000_2... 000000_3... 000000_4...</p> <p>0AAAAA_1 0AAAAA_2 0AAAAA_3 0AAAAA_4</p>	<p>02/10/24 03/10/24</p>



<p>Els projectes 0AAAAA_* són equivalents als 000000_* anteriors però sense la interfície IBiblioteca. En conseqüència:</p> <ul style="list-style-type: none"> - La classe abstracta Biblioteca incorpora com a abstractes, tots els mètodes que definia la interfície IBiblioteca, per obligar a que les derivades de Biblioteca implementin els mètodes. - L'únic projecte 0AAAAA_* que canvia respecte 000000_* és 0AAAAA_1_Model, on desapareix IBiblioteca i canvia Biblioteca. Els altres projectes són idèntics en el codi; només es diferencien en el projecte Model que han d'incorporar com a llibreria, que en 000000_2/3/4 és 000000_1_Model, mentre que en 0AAAAA_2/3/4 és 0AAAAA_1_Model. <p>Pregunta: Per què definir la interfície IBiblioteca si ho podem fer sense ella?</p>		
<p>Exercici per dilluns, 7 d'octubre: Utilitats per fer persistent una Biblioteca en fitxer binari</p> <p>Desenvolupar la classe BPFitxerBinariManual dins paquet org.milaifontanals.biblioteca.persistence amb utilitats:</p> <pre>public static void saveBiblioteca(IBiblioteca b, String nomFitxer) public static IBiblioteca loadBiblioteca(String nomFitxer)</pre> <p>Requeriments:</p> <ul style="list-style-type: none"> - No es pot usar la classe RandomAccessFile per gestionar el fitxer binari. - Enregistrem les dades binàries d'una Biblioteca seguint la seqüència: <ol style="list-style-type: none"> 1. Nom de la biblioteca 2. Comptador per al número d'ordre (autonumèric) a assignar a les noves fitxes. 3. Seqüència de fitxes on per cada fitxa: <ol style="list-style-type: none"> 1. Marca per identificar el tipus de fitxa: <ul style="list-style-type: none"> - Caràcter 'L' per llibre - Caràcter 'R' per revista 2. Referència 3. Títol 4. EsDeixa 5. Ordre 6. DataCreació 7. DataModificació 8. Les dades específiques de llibre o revista <ul style="list-style-type: none"> - Si llibre: Editorial seguit de ISBN - Si revista: Any seguit del número de revista - Pels camps que poden tenir valor nul, com l'editorial de FitxaLlibre, enregistrem el caràcter N per indicar nul i en cas que continguin valor, prèviament inserim el caràcter * i posteriorment el corresponent valor. <p>Desenvolupar programes que en comprovin el funcionament.</p> <ul style="list-style-type: none"> - En el moment de recuperar una IBiblioteca... i donat que tenim diverses possibilitats de Biblioteca... quina Biblioteca (AL o TS o qualsevol altra que es pugui crear) cal crear? Decidim enregistrar, dins el fitxer binari, com a primera dada, el nom de la classe IBiblioteca. - Aquesta decisió impossibilita usar aquestes utilitats per canviar de tipus de Biblioteca entre enregistrament-càrrega i viceversa. 	241007_1...	07/10/24 09/10/24
<p>Disseny de capa persistència BPFitxerBinariManual</p> <p>Les utilitats anteriors permeten carregar una IBiblioteca des de fitxer binari a memòria i descarregar una IBiblioteca de memòria a fitxer binari.</p> <p>Una aplicació (per exemple el programa Proves del darrer projecte) que vulgui usar aquestes utilitats, invocarà els seus mètodes amb els paràmetres requerits per cada mètode.</p> <p>Suposem que interessa fer persistent una IBiblioteca en un SGBDR de manera que l'aplicació no necessiti modificació. Haurem de desenvolupar els mètodes saveBiblioteca i loadBiblioteca però en aquest cas, els paràmetres haurien de ser diferents (url, usuari, contrasenya) i, per tant, el codi de l'aplicació (crida als mètodes) hauria de canviar.</p>	241009_1...	09/10/24 10/10/24



Per evitar això, interessa que els mètodes `saveBiblioteca` i `loadBiblioteca` no hagin de tenir paràmetres diferents en funció de l'origen de dades.

La solució es dissenyar una classe que contingui els mètodes que interactuen amb l'origen de dades (fitxer, BDR, BDOR, BDOO,...) i que guardi en el seu interior les dades necessàries per interactuar amb l'origen de dades (`rutaFitxer` en cas de fitxer, `url+usuari+contrasenya` en cas de BD,...). Aquestes classes s'anomenen capes de persistència.

Evolucionem el projecte `BPFitxerBinariManual` que conté dues utilitats per gestionar la persistència d'una `IBiblioteca` en fitxer binari cap una capa de persistència. Per aconseguir-ho:

- La classe ha de contenir el nom del fitxer on guardar la `IBiblioteca`.
- Ja que es tracta d'una classe que conté dades, podem aprofitar per a que contingui el nom de la classe `Biblioteca` a gestionar (`BibliotecaAL`, `BibliotecaTS` o qualsevol altra implementació que pugui aparèixer en un futur). D'aquesta manera, no caldrà que el nom de la classe `Biblioteca` quedi enregistrada dins el fitxer binari i això permetrà que una `BibliotecaAL` s'enregistri en fitxer i es recuperi en una `BibliotecaTS`.
- La capa de persistència ha de tenir algun(s) constructor(s) que s'encarreguen de recuperar, des d'un fitxer de configuració, les dades necessàries per accedir a l'origen de dades.
- Podem tenir un constructor sense paràmetres que s'encarregui de recuperar les dades a partir d'un fitxer de configuració amb nom per defecte.
- Podem tenir un constructor amb un paràmetre que correspongui al nom del fitxer de configuració a usar.
- Cal definir i documentar quina és l'estructura i contingut del fitxer de configuració.
- Els mètodes `saveBiblioteca` i `loadBiblioteca` deixen de ser estàtics i passen a ser mètodes de classe.
- Totes les excepcions que es generen en una CP acostumen a ser d'una classe que acompanya la CP, de manera similar a com els mètodes de JDBC generen `SQLException` o com els mètodes de l'API JDom generen `JDOMException`.

El projecte `BPFitxerBinariManualCP` és una proposta de capa de persistència:

- Classe `BPFitxerBinariManual` és la capa de persistència
- Conté 2 dades: `nomFitxer` i `nomClasseBiblioteca`
- Conté constructor sense paràmetres que carrega les dades des de fitxer de configuració anomenat `BPFitxerBinariManual.properties`.
- Conté constructor amb 1 paràmetre que carrega les dades des de fitxer de configuració de nom indicat pel paràmetre.
- Fitxer de configuració textual amb propietats `fitxer` i `classeBiblioteca` en format adequat per ser carregat amb el mètode `load` de la classe `Properties` de Java.
- Classe `BPFitxerBinariManualException` per generar totes les excepcions.

Per comprovar funcionament, disposem com a test:

- `P01_EnregistrarBiblioteca` que rep per paràmetre el nom del fitxer de configuració a usar i amb 2 programes de prova:
 - ✓ `P01_EnregistrarBibliotecaAL` que usa fitxer `PerBibliotecaAL.properties`
 - ✓ `P01_EnregistrarBibliotecaTS` que usa fitxer `PerBibliotecaTS.properties`
- `P02_RecuperarBiblioteca` que rep per paràmetre el nom del fitxer de configuració a usar i amb 2 programes de prova:
 - ✓ `P02_RecuperarBibliotecaAL` que usa fitxer `PerBibliotecaAL.properties`
 - ✓ `P02_RecuperarBibliotecaTS` que usa fitxer `PerBibliotecaTS.properties`
- En els dos fitxers de configuració, la propietat `fitxer` fa referència al mateix fitxer (podria ser diferent), fet que demostra que la capa permet enregistrar una `BibliotecaAL` i recuperar una `BibliotecaTS` i viceversa.

<p>Exercici per dilluns, 14 d'octubre: Disseny capa persistència BPFitxerXmlJDom</p> <p>Desenvolupar la CP BPFitxerXmlJDom per gestionar una IBiblioteca en fitxer XML, dins paquet <code>org.milaifontanals.biblioteca.persistence</code> amb mateixos mètodes que els desenvolupats en BPFitxerBinariManualCP tenint en compte:</p> <ul style="list-style-type: none">- Cal usar l'API JDom- El fitxer XML ha de validar el fitxer <code>library.dtd</code> facilitat.- Cal seguir els requeriments indicats al final de <code>library.dtd</code>. <p>Es facilita també un fitxer <code>librarySkeleton.xml</code> per ajudar a la comprensió.</p> <p>Comproveu-ne el funcionament amb programes de proves.</p>	241014_1...	14/10/24		
<p>Persistència XML via XmlBinding</p> <p>JAXB (Java Architecture for XML Binding) permet mapar classes Java en representacions XML.</p> <ul style="list-style-type: none">- Versió 1.0, especificada en el projecte JSR 31. (4/03/2003). Incorporada en Java6.- Versions 2.x (2.0-...-2.3), especificades en el projecte JSR 222. La 2.3 en 19/09/2017 <p>Els projectes anteriors defineixen l'especificació de l'API i per usar l'API en programes cal disposar d'alguna implementació.</p> <ul style="list-style-type: none">- JDK6-7-8 incorporaven especificació i implementació.- JDK9+ SE no incorpora JAXB i cal incorporar llibreries (especificació i implementació).- NetBeans ha anat incorporant alguna implementació. NetBeans-18 incorpora la llibreria JAXB 2.3.5 però hi ha algun problema d'entesa entre NetBeans-18 i el javadoc de JAXB 2.3.5 que provoca que l'ajuda no es pugui invocar mentre s'està desenvolupant.- Si observem els jars que incorpora la llibreria JAXB 2.3.5, observarem que hi ha l'API (<code>jaxb-api.jar</code>) i altres jars que corresponen a la implementació, necessaris per executar els programes.- Si fem una ullada a la documentació de l'API de JAXB 2.3.5, veurem que el nom dels seus paquets comença per <code>javax.xml.bind</code> i javax i java són noms propietat d'Oracle. <p>Per altra banda, Java EE (plataforma de programació Java per desenvolupar i executar programari escrit amb el llenguatge Java amb una arquitectura distribuïda amb nivells, basada en components de programari, tot plegat executant-se en un servidor d'aplicacions) propietat d'Oracle s'ha quedat en la versió 8.</p> <p>Java EE (propietat d'Oracle) ha evolucionat cap a Jakarta EE (Open Source). En aquest enllaç es pot veure totes les especificacions que incorpora Jakarta EE i una d'elles és Jakarta XML Binding, que ha partit de la versió 2.3 per evolucionar fins la 4.0 incorporada en Jakarta EE 10.</p> <p>NetBeans-18 incorpora Jakarta EE 8-9-10 API, les quals incorporen especificacions 2.3-3.0-4.0 de JAXB. Però només incorpora l'especificació i no aporta cap implementació. Per altra banda, el salt de Java a Jakarta implica haver de canviar, com a mínim, el prefix dels paquets que s'usaven en JAXB 2.3-, ja que <code>javax.xml.bind</code> ha passat a anomenar-se <code>jakarta.xml.bind</code>.</p> <p>Per ajudar a entendre aquest embolic, considerem els 3 paquets 230916_1... que es faciliten:</p> <ul style="list-style-type: none">- 241014_1_XmlBinding_JAXB235, que incorpora la llibreria JAXB 2.3.5 Observem que el projecte compila i que podem executar el programa Main. Fixem-nos en els imports dels paquets <code>javax.xml.bind</code>. Malauradament l'ajuda no funciona.- 241014_1_XmlBinding_JakartaEE10, que incorpora la llibreria Jakarta 10 EE API. En aquest cas, l'ajuda és accessible. Fixem-nos en els imports dels paquets <code>jakarta.xml.bind</code>. Observem que el projecte compila però que en executar el programa Main ens trobem l'error <i>Implementation of Jakarta XML Binding-API has not been found</i>, degut a que no incorpora la implementació.- Per tant si volem usar l'especificació més actual de <code>XmlBinding</code> (4.0) incorporada en Jakarta 10 EE API, ens cal trobar una implementació. Eclipse Foundation desenvolupa la implementació jaxb-ri. En els moments de redactar aquest dossier, ens descarreguem la darrera versió <code>jaxb-ri-4.0.5</code> i incorporem a				14/10/24 16/10/24



NetBeans.18 la llibreria de nom JAXB-RI 4.0.5 amb els jars de la carpeta mod del zip descarregat i el javadoc de Jakarta EE 10 (RutaNetBeans-18\netbeans\enterprise\docs\jakartaee10-doc-api.jar). És el projecte 241014_1_XmlBinding_JAXB-RI-4.0.5 facilitat.

Documentació referent a la utilització de XmlBinding:

- Dossier de l'IOC: DAM_M06_UF1_Fitxers_3_Binding.pdf (molt espès)
- [Tutorial de Java](#)
- Conceptes bàsics:

Anotació `@XmlRootElement` per l'arrel de la classe

Anotació `@XmlElement` pels camps de la classe que es corresponguin amb elements de l'XML (per defecte)

Anotació `@XmlAttribute` pels camps de la classe que es corresponguin amb atributs de l'XML

La classe ha de tenir forçosament un constructor sense paràmetres, malgrat no faci res i sigui `private`, però si la classe s'ha de poder derivar, cal que sigui `protected`, per a que les classes derivades també puguin tenir constructor sense paràmetres que invoqui el de la classe base.

Si la classe no incorpora cap constructor, ja n'hi ha prou amb el constructor sense paràmetres que incorpora Java.

El contingut mapat entre Java i XML es pot definir a nivell de camps o a nivell de propietats (`get`) o combinant ambdós tipus de possibilitats. Això es defineix amb una anotació a la classe (`@XmlAccessorType`) que admet diversos valors, molt ben explicats amb exemples [aquí](#):

- `XmlAccessType.PUBLIC_MEMBER` (per defecte), que mapa: camps públics, camps amb anotacions, tots els accessors (`getter` i `setter`)
- `XmlAccessType.PROPERTY`, que mapa: camps amb anotacions, tots els accessors
- `XmlAccessType.FIELD`, que mapa: tots els camps, accessors amb anotacions
- `XmlAccessType.NONE`, que mapa: camps amb anotacions, accessors amb anotacions

Les anotacions en els accessors s'acostumen a posar abans del `getter`, que hi ha de ser obligatòriament. Si la classe no disposa de `set` coherent amb el `get`, la recuperació de dades d'un XML no emplena el corresponent camp.

Molt recomanable introduir dins `XMLElement/XMLAttribute` l'atribut `required` (per defecte `false`).

`@XmlID` pels atributs que siguin ID.

`@XmlTransient` per indicar que un camp o propietat no vol ser mapat.

`@XmlElementWrapper` per a indicar que una col·lecció de dades (`Array`, `Vector`, `List`,...) vagi tancada en un node (si no s'hi indica, no es genera un node pare).

Cal usar `@XmlElement` si es vol algun marcatge específic per cada objecte de la col·lecció.

A més, si els elements de la col·lecció són de diferents classes i es vol marcatge diferenciat, caldrà usar `@XmlElements` per indicar la marca que es desitja per als objectes de cada classe derivada. És a dir:

```
@XmlElementWrapper(name = "nomCampCol·lecció")
@XmlElements({
    @XmlElement(name="tipusA", type=nomDeLaClasseCorrespoentATipusA.class),
    @XmlElement(name="tipusB", type=nomDeLaClasseCorrespoentATipusB.class),
    ...
})
```

Camps/accessors `static` no són gestionats per JAXB. Si cal, crear `private get/set` que els gestionin.



<p>On incorporar les marques?</p> <ul style="list-style-type: none"> - Si es disposa del codi de les classes, a les pròpies classes (habitual i fàcil, opció que utilitzarem) - Si no es disposa del codi de les classes, es pot crear un model derivat del model a "marcar" i incorporar les marques en el nou model derivat (no és fàcil i no ho farem) <p>JAXB tradueix els camps <code>char</code> pel seu codi ASCII, cosa que segurament NO es vol, doncs si tenim un camp <code>char</code> en un objecte amb valor 'A', JAXB el passaria a XML com "65". Per tant, si cal marcar via JAXB algun camp <code>char</code>, cal fer-ho via accessors (get-set) adequats.</p>	
<p>Exemples de XmlBinding – Projecte 241016_1_XmlBinding_Exemples</p> <ol style="list-style-type: none"> 1. Classe simple preparada per a poder fer <i>binding</i> dels seus objectes <code>Customer.java</code> Programes que en comproven el funcionament per a un objecte: <code>P01_JAXBConvertCustomerToXML.java</code> <code>P02_JAXBConvertXMLToCustomer.java</code> Programes que fan binding d'una col·lecció d'objectes de la classe: <code>P03_JAXBConvertSeveralCustomerToXML.java</code> <code>P04_JAXBConvertXMLToSeveralCustomer.java</code> 2. Classe que conté referències a objectes d'altres classes: <code>Product.java</code> <code>CustomerWithOneProduct.java</code> Programes que en comproven el funcionament: <code>P05_JAXBConvertCustomerWithOneProductToXML.java</code> <code>P06_JAXBConvertXMLToCustomerWithOneProduct.java</code> 3. Classe que conté una col·lecció d'objectes: <code>Product.java</code> <code>CustomerWithSeveralProductVer1.java</code> En aquest cas, els diversos productes no queden englobats dins un node (no s'utilitza <code>@XmlElementWrapper</code>) <code>CustomerWithSeveralProductVer2.java</code> En aquest cas, els diversos productes queden englobats dins un node <code><products></code> (s'utilitza <code>@XmlElementWrapper</code>) Programes que en comproven el funcionament: <code>P07_JAXBConvertCustomerWithOneProductVer1ToXML.java</code> <code>P08_JAXBConvertXMLToCustomerWithOneProductVer1.java</code> <code>P09_JAXBConvertCustomerWithOneProductVer2ToXML.java</code> <code>P10_JAXBConvertXMLToCustomerWithOneProductVer2.java</code> 4. Conversions especials, com afegir un caràcter o enregistrar una data en un format concret. Concretament, observem la nova versió de la classe <code>Customer</code>, que conté la data <code>dataAlta</code> i volem que aquesta data s'enregistri en un format concret i també que, el camp <code>id</code>, en el fitxer XML quedi amb un caràcter \$ al davant. Per aconseguir-ho tenim dos mecanismes: <ul style="list-style-type: none"> - Fer "transient" les propietats <code>getter</code> corresponents i crear unes noves propietats <code>getter</code> que s'encarreguin d'efectuar la conversió (portaran el marcatge JAXB). Aquestes propietats que podríem anomenar "auxiliars", seran privades per a que un programador no les pugui usar, però seran visibles per JAXB Observar-ho a la classe <code>CustomerAmbConversionsEspecialsVer1.java</code> Programes que en comproven el funcionament: <code>P11_JAXBConvertCustomerToXMLAmbConversionsEspecialsVer1</code> <code>P12_JAXBConvertXMLToCustomerAmbConversionsEspecialsVer1</code> - Marcar les propietats <code>getter</code> habituals amb l'anotació <code>@XmlJavaTypeAdapter</code> que permet indicar una classe <code>XmlAdapter</code> que especifica que s'encarrega de la conversió i que hem de dissenyar. Observar-ho a la classe <code>CustomerAmbConversionsEspecialsVer2.java</code> 	<p>16/10/24</p>



<p>Atenció!!!</p> <ul style="list-style-type: none"> o Les classes XmlAdapter no gestionen els camps null. o Les classes XmlAdapter només “adapten” classes i no tipus primitius. <p>Programes que en comproven el funcionament:</p> <p>P13_JAXBConvertCustomerToXMLAmbConversionsEspecialsVer2 P14_JAXBConvertXMLToCustomerAmbConversionsEspecialsVer2</p>	
<p>Exercici per dijous, 17 d'octubre: Disseny capa persistència BPFitxerXmlJaxb</p> <p>Ha de tenir els mateixos mètodes que la capa BPFitxerXmlJDom però usant JAXB i els mateixos requeriments relatius al DTD a validar.</p> <p>Retocar projectes 000000_* a 0BBBBB_* ja que cal retocar adequadament les classes del model Fitxa, FitxaLlibre, FitxaRevista i Biblioteca per poder usar XMLBinding.</p> <p>Pautes per la solució:</p> <ul style="list-style-type: none"> - En 0BBBBB_1_Model cal marcar les classes Fitxa, FitxaLlibre, FitxaRevista i Biblioteca. <ul style="list-style-type: none"> ➤ Per a que tot compili, cal incorporar la llibreria JAXB en el projecte. ➤ Importantíssim: Cal incorporar constructor sense paràmetres a totes les classes amb marques JAXB. Del contrari, compila però hi hauria error en temps de <i>Marshall</i> o <i>Unmarshall</i>. - En 0BBBBB_2_BibliotecaAL hem marcat la classe BibliotecaAL. <ul style="list-style-type: none"> ➤ Pel fet de ser derivada de Biblioteca, no “hereta” les marques de Biblioteca. ➤ Necessita la llibreria JAXB en el projecte. - En 0BBBBB_3_BibliotecaTS hem marcat la classe BibliotecaTS. <ul style="list-style-type: none"> ➤ Pel fet de ser derivada de Biblioteca, no “hereta” les marques de Biblioteca. ➤ Necessita la llibreria JAXB en el projecte. - En 241017_1_Biblioteca... tenim la capa demanada: <ul style="list-style-type: none"> ➤ En el constructor, aprofitem per crear els objectes Marshaller i Unmarshaller, de manera que no calgui crear-los dins els mètodes save i load cada vegada que s'invoquen ➤ En la configuració del marshaller incorporarem: <pre>marshaller.setProperty(Marshaller.JAXB_FRAGMENT, true); /* Propietat JAXB_FRAGMENT a true, desactiva les capçaleres automàtiques de JAXB. Si no existeix aquesta propietat o està a FALSE, en fer "marshall" afegeix: <?xml version="1.0" encoding="UTF-8" standalone="yes"?> que incorpora standalone="yes" i provoca que un XML formatat no sigui validable via DTD. A més, ens cal desactivar-ho si volem incorporar etiqueta !DOCTYPE dins XML, ja que haurà d'estar entre la capçalera anterior i l'arrel del document i cal ficar-la manualment */</pre> ➤ En la configuració del unmarshaller incorporarem: <pre>System.setProperty("javax.xml.accessExternalDTD", "all"); /* Propietat per a que pugui accedir al fitxer DTD, tot i que no fa validació JAXB només permet validació a partir d'schema... NO a partir de DTD. En cas de voler validació via DTD, podríem aplicar la validació de DOM sobre el contingut a "unmarshallar"... */</pre> ➤ El mètode save, per aconseguir incorporar l'etiqueta !DOCTYPE, com que s'ha d'incorporar manualment, caldrà seguir la següent seqüència: <ul style="list-style-type: none"> - Crear fitxer XML manualment, amb FileWriter - Incorporar capçalera <?xml version="1.0" encoding="UTF-8"?> i etiqueta !DOCTYPE manualment. - Usar l'objecte Marshaller per enviar la traducció de la biblioteca dins el fitxer. - Tancar el fitxer ➤ Necessita la llibreria JAXB en el projecte. 	<p>0BBBBB_*... 241017_1...</p> <p>17/10/24</p>
<p>Serialització d'objectes - Apartat 1.5 de DAM_M06_UF1_Fitxers_1_Fluxos&Seriació.pdf</p> <p>Verb correcte? seriar o serialitzar?</p> <ul style="list-style-type: none"> • El dossier d'IOC usa seriació com acció de seriar. • El diccionari de l'Institut d'Estudis Catalans (DIEC) reconeix seriar però no serialitzar. • El Centre de Terminologia (termcat) reconeix serialitzar però no seriar. 	<p>0CCCCC_1.. 241021_1...</p> <p>21/10/24</p>



- En anglès es parla de **serialization** i en castellà de **serialización**.
- Les universitats catalanes utilitzen majoritàriament el mot **serialització**. També nosaltres!

Petit resum d'apartat 1.5 del dossier: Java – Serialització automàtica

- Els objectes a serialitzar han de pertànyer a classe que implementi `Serializable` i tots els objectes incorporats a la seva definició també han de ser de classes serialitzables.
- Moltes col·leccions d'objectes són serialitzables i, per tant, podem serialitzar directament tota la col·lecció sense haver de fer un procés repetitiu, serialitzant objecte a objecte.
- El programa que recupera la informació serialitzada ha de ser coneixedor de la definició de les classes dels objectes serialitzats i de la mateixa versió. És a dir, si hem serialitzat objectes d'una classe X, per recuperar-los ha de tenir accés a la mateixa versió de la classe X.

Petit resum d'apartat 1.5.1 del dossier: Java – Serialització personalitzada

- Possibilitat de “marcar” camps amb `transient` per a no ser serialitzats, p. ex. Contrasenya
Si, per exemple, no ens interessa emmagatzemar el preu dels electrodomèstics, caldrà retocar la definició d'aquest camp a la classe, deixant-lo com:

```
private transient float preu;
```

La recuperació NO emplea aquest camp, que en ser numèric tipus primitiu, quedarà emplenat amb valor 0, malgrat pogués tenir assignat un valor per defecte.
- Possibilitat d'indicar “com” serialitzar, programant els mètodes següents dins la classe:

```
private void writeObject(ObjectOutputStream out)
    throws IOException
private void readObject(ObjectInputStream in)
    throws IOException, ClassNotFoundException
```

Sí... com es veu... són `private` i són usats pel procés de serialització
- Els camps `static` no són serialitzats. Si volem enregistrar-los, caldrà indicar “com” serialitzar.

Exercici: Disseny capa persistència BPFitxerBinariSerialitzat

Ha de tenir els mateixos mètodes que la capa `BPFitxerBinariManual` però usant serialització.

Solució:

- Ha calgut retocar les classes fent-les `Serializable` i, per això, hem evolucionat les classes (`IBiblioteca` i `Fitxa`) dels projectes `000000_x` a projectes `000000_x` i la capa de persistència del projecte `241021_1` usa els projectes `000000_x`.
- El procés de serialització, si serialitza un objecte de la classe X, en recuperar-lo, és obligatori recuperar un objecte de la classe X. Això cal tenir-ho en compte en la nostra situació, ja que si apliquem la serialització a un objecte `BibliotecaAL`, en recuperar-lo, estem recuperant un objecte `BibliotecaAL`. És a dir, utilitzant aquesta tècnica no és possible haver serialitzat un objecte `BibliotecaAL` i recuperar-lo com a `BibliotecaTS` o a l'inrevés.
- Degut a això, el fitxer de configuració de la capa no necessita incorporar el nom de la classe `Biblioteca` a recuperar. Tampoc és necessari disposar de 2 programes de recuperació, doncs el procés recuperarà l'objecte de la classe que s'hagi enregistrat.
- La serialització per part de Java de `BibliotecaAL` (que conté una `ArrayList`) i `BibliotecaTS` (que conté un `TreeSet`) és possible per què les dues col·leccions implementen `Serializable`.
- En cas que la col·lecció contingui algun `null` (no és el cas) la recuperació el mantindria.
- Si comprovem serialització havent indicat únicament “implements `Serializable`” dins `Fitxa` (també ho seran `FitxaLlibre` i `FitxaRevista`) i dins `Biblioteca` (també ho seran `BibliotecaAL` i `BibliotecaTS`) observarem que NO recupera el comptador de fitxes el qual es recupera... A ZERO!!! Això és degut a que el comptador de `Fitxa` és una dada `static` i la serialització automàtica no contempla aquestes dades. Per solucionar-ho, cal dir-li a Java com ha de serialitzar els objectes `Biblioteca`, incorporant en aquesta classe la implementació dels mètodes `writeObject` i `readObject`.