Generalitat de Catalunya
Departament d'Educació
Institut Milà i Fontanals. Igualada

Dossier

Informàtica

ICB0 Desenvolupament d'aplicacions multiplataforma

M06 Accés a dades

UF1 Persistència en fitxers

Diari d'activitats

Isidre Guixà / Pere Ollé

Curs 2023/24

Milà i Fontanals IGUALADA
Unió Europea
Fons Social Europeu
ISO 9001 BUREAU VERITAS Certification

Gestió de fitxers		
Gestió del sistema de fitxers	**Projecte**	**Data**

Apartat 1.1 de DAM_M06_UF1_Fitxers_1_Fluxos&Seriacio.pdf
 	
Introducció bàsica a la gestió de fitxers. Treballat a UF5 del M03.
Classe **`File`** **-** Veure documentació oficial
- Pertany al paquet `java.io`. 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 `java.nio` 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
`exists`, `isFile`, `isDirectory`, `createNewFile`, `delete`
`getName`, `getParent`, `getAbsolutePath`, `getPath`, `lastModified`
`mkdir`, `mkdirs`, `listFiles`
- Camps a usar per aconseguir independència del S.O:
`pathSeparator`, `separator`, `pathSeparatorChar`, `separatorChar`		

Exercici: Fer un programa que comprovi l'existència de les rutes: `X:\programes\pepe.txt, C:\Windows, C:\Windows\regedit.exe, C:\fontanals.txt, D:\fontanals.txt` ➢ 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	230913_1...	13/09/23
Exercici: Fer un programa que elimini el fitxer `D:\fontanals.txt` creat en exercici anterior	230913_2...	13/09/23
Exercici: Considereu la ruta `A\B\Carpeta` on suposem que `A` no existeix. Fer un programa que intenti crear `Carpeta` usant `mkdir` i `mkdirs`. Una vegada creada, comprovar el resultat dels mètodes `getName`, `getParent`, `getAbsolutePath`, `getPath` i `lastModified` aplicats sobre la ruta indicada.	230913_3...	13/09/23
Exercici: Fer un programa que mostri el contingut de totes les carpetes i fitxers existents en el projecte.	230915_1...	15/09/23
Gestió de fitxers		
Gestió del contingut dels fitxers	**Projecte**	**Data**
A 1r curs s'ha usat les classes: - `Scanner` i `PrintStream` per a gestionar el contingut de fitxers de text - `RandomAccessFile` 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&Seriació.pdf *Fluxos orientats a dades*		15/09/23
Exercici pel dilluns, 18 de setembre **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	230918_1...	18/09/23

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.

➢ Feu un tercer programa que també recuperi la informació del fitxer generat en el programa 1 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 `230918_1...` incorpora:

- Programes P01 que genera i P02 que recupera. El programa P01 enregistrea els enters com a byte i sembla que és correcte (funciona) per què els valors "caben" 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. El programa P04 facilita 2 versions respecte a obertura/tancament de fitxer:

  v1: Intenta obrir el fitxer.
  - 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.

  v2: Tot el codi (obertura i tractament de les dades) està dins un `try... catch...` En aquest cas, abans de finalitzar el programa, cal tancar el fitxer i ho posem dins `finally`... Però en entrar en finally, pot ser:
  - Que no s'hagués obert el fitxer. No s'ha de fer el `close`!
  - Que s'hagués obert el fitxer. Cal fer el `close`!

La manera de saber-ho és preguntant si `fis==null`.

Apartat 1.3 de DAM_M06_UF1_Fitxers_1_Fluxos&Seriació.pdf

Fluxos orientats a caràcter

Exercici

Gestió de fitxers textuals utilitzant `FileReader` i `FileWriter`

Sigui ST una seqüència de dades caràcter del tipus `<L><valor><espai>` on L és una lletra identificadora del tipus de valor numèric:

`B-byte | S-short | I-int | F-float | L-long | D-double`

- 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`

  És a dir, si dins el fitxer hi ha `I22 S44 L9292 I-30 B22 S-30 I99 L8080`, la sortida haurà de ser:

```
Valors de tipus byte: 22
Valors de tipus short: -30, 40
Valors de tipus int: -30, 22
Valors de tipus long: 8080, 9292
```

230920_1...

20/09/23

Exercici EXTRA (no es resoldrà a classe – es publicarà la solució 26/09/23)

- Programa que generi fitxer binari d'enters amb els primers 100 enters estrictament positius, enregistrant els parells com a `short` i els senars com a `int`.
  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 int (no tenim per què saber quans enters té el fitxer)

Versió 1 usant `FileOutputStream` i `FileInputStream`.
Versió 2 usant `RandomAccessFile`.

230926_1...

Accés des de C/C++ a fitxers binaris generats per Java

Recordatori de funcions de C per gestió de fitxers i comparativa amb classes de Java:

		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`

En C, a més, cal usar les següents funcions:

- `fopen` per obrir fitxer, on Recordeu que cal indicar el mode del fitxer (text-binari)
- `fclose` per tancar fitxer.
- `feof` per saber si s'ha arribat al final del fitxer després d'efectuar lectura

Explicació detallada de les funcions en C, a https://cplusplus.com/reference/cstdio/
Cal incloure en el programa `#include <stdio.h>`

Exercici: Programa en C/C++ que mostri per pantalla el contingut del fitxer binari generat en el projecte `230918_1_Gestio_FileInputStream_FileOutputStream`.

22/09/23

Solució: - El projecte `230922_1` 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 `big-endian` i C/C++ usa l'ordre de bytes de l'arquitectura de la CPU, que en Intel acostuma a ser `little-endian` i, per tant, apareix un problema per recuperar valors enregistrats en ordre de bytes diferent (nostre cas) El projecte `230922_2` conté programa Java que informa de l'odre de bytes que usa la CPU. El projecte `230922_3` conté programa C que informa de l'ordre de bytes que usa la CPU. - Per recuperar valors des de C/C++ si la CPU treballa en `little-endian` (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 Exemples: - Sigui taula `t` que conté 2 bytes corresponents a `short` generat per Java i emplenada per C. Per obtenir el valor `short`: `(int)t[1] | (int)t[0]<<8` - Sigui taula `t` que conté 4 bytes corresponents a `int` generat per Java i emplenada per C. Per obtenir el valor `int`: `(int)t[3] | (int)t[2]<<8 | (int)t[1]<<16 | (int)t[0]<<24` El projecte `230922_4` conté programa C per la nostra arquitectura de màquina que recupera el contingut del fitxer generat per Java de forma correcta.		
Apartats 1.4 i 1.4.1 de DAM_M06_UF1_Fitxers_1_Fluxos&Seriació.pdf *Modificadors de fluxos - Fluxos de tipus de dades* 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 `DataOutputStream` que ha d'estar vinculat a un `FileOutputStream` i el mateix amb un `DataInputStream` vinculat amb un `FileIntputStream`. 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.		20/09/22
Exercici per dimecres, 27/09 **Gestió de f. binaris via modificadors de flux `DataInputStream/DataOutputStream`** Refer els programes del projecte `230918_1_Gestio_FileInputStream_FileOutputStream` que usen `FileInputStream` i `FileOutputStream`, utilitzant modificadors de flux.	230927_1...	27/09/23
Fitxers XML via JDOM (no és l'acrònim de *Java Document Object Model* !!!) 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` (que veurem a M03-UF5) encara que s'integra bé amb `DOM` i `SAX`: - Classe `DOMOutputter` per passar un `JDOM.Document` a `DOM.Document`. - Classe `DOMBuilder` per obtenir un `JDOM.Document` a partir de `DOM.Document`. - Classe `SAXOutputter` per passar un `JDOM.Document` a flux per ser gestionar via `SAX` 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 banda de la informació que facilita el lloc oficial, podeu trobar altres articles a la web, com per exemple: aquest, aquest, ...		25/09/23

Per instal·lar-lo, baixar la darrera versió (jdom-2.0.6.1.zip en el moment actual) i descomprimir-lo.

La descompressió genera diversos jar. Interessa:

- `jdom-2.0.6.1.jar`
- `jdom-2.0.6.1-javadoc.jar`

Per utilitzar el paquet JDOM cal tenir accés a la llibreria `jdom-2.0.6.1.jar`

Tots els exemples que acompanyen aquests materials i que utilitzin JDOM, pressuposen l'existència d'una llibreria en el Netbeans anomenada exactament `JDom 2.0.6.1`

A més, per esbrinar si un fitxer XML conté etiqueta `<!DOCTYPE` per procedir a validar, Java no disposa de cap utilitat (caldria gestionar fitxer de text... ufff) i ens podem aprofitar de la classe `FileUtils` de *Apache Commons IO*. Per utilitzar aquesta classe cal tenir accés a la llibreria `commons-io-2.x.jar`.

Els exemples que acompanyen aquesta materials i que utilitzin `Commons IO`, pressuposen l'existència d'una llibreria en el Netbeans anomenada exactament `Apache Commons IO 2.13.0` que conté la versió 2.13.0 de `Commons IO`.

Exemples XML-JDOM

1. Lectura de document XML de disc: `Ex01_ComptarXML.java`

Programa que compta quants títols, autors, anys i preus hi ha en el fitxer `books.xml` ubicat en la carpeta que conté el projecte

2. Enregistrament de document XML a disc: `Ex02_CrearXML.java`

Programa que crea document XML com `books.xml` i l'enregistra a fitxer

3. Modificació de document XML: `Ex03_ModificarXML.java`

Programa que parteix del document `books.xml` i en crea un de nou amb les següents modificacions:

1. Modificar a tots els llibres, l'atribut `category` a majúscules
2. Augmentar els preus un 2%
3. Afegir un nou element `editorial` a cada llibre
4. Eliminar l'element `year`
5. Eliminar l'atribut `lang` del títol de cada llibre
6. Afegir l'atribut `sex` a cada autor

230927_2...

27/09/23

Exercici:

Programa que calcula la població europea a l'arxiu `mondial.xml` ubicat en la carpeta que conté el projecte. Es facilita `mondial.xml` – `mondial.dtd` – `mondial.xsd` dins el ZIP.

Observacions a tenir en compte:

- Una ullada sobre alguns països ens mostra que per aconseguir la informació que se'ns demana, caldrà tenir en compte l'element `encompassed` de `country`, que ens informa del continent al què pertany el país i l'element `population`, que ens informa de la població.
- Segons DTD, un país pot tenir varis elements `encompassed` i això té sentit per què hi ha països, com Turquia, que tenen el seu territori en diversos continents, i l'atribut `percentage` 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 1.0000.000 * 60/100.
- Segons DTD, un país pot tenir varis elements `population`, 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 `measeured`), caldrà considerar aquell que tingui el valor `"census"`.
- Segons DTD, l'atribut `year` de l'element `population` no és obligatori, però segons XSD, sí

230929_1...

29/09/23

que ho és... Ens creiem el XDS... i no cal tenir en compte que `year` pugui ser `null`...		
NF2 – Persistència d'objectes en fitxers		
Recordatori de temes vinculats a l'ordenació de col·leccions en Java: - Ordre natural en una classe X: `class X implements Comparable<X>` fet que ens obliga a implementar mètode `compareTo` Exemple en el projecte: classe `NumRef`, on l'ordre natural és per referència. - Ordre alternatiu pels elements d'una classe X: Cal crear una classe `Comparator` que defineixi l'ordre entre elements. Exemples en el projecte: ➢ Classe `CompararNumRefPerNum`, que permet comparar i ordenar objectes `NumRef` pel camp `num`. ➢ Classe `CompararNumRefAmbString`, que permet comparar i ordenar objectes `NumRef` amb `String`. - Exemples: Programa `P01` ordena elements `NumRef` per `ref` (ordre natural) i per `num` (ordre alternatiu) - Inserció dicotòmica/binària en col·leccions ordenades Mètode `binarySearch` per ordre natural Exemple: Programa `P02` que fa inserció ordenada per referència d'un NumRef en una taula i en una llista que estan ordenades per referència. Primer fa cerca binària i, si es pot, insereix. - Cerca dicotòmica/binària en col·leccions ordenades d'objectes p Mètode `binarySearch` per ordre alternatiu Exemple: Programa `P03` que fa cerca de referència (`String`) en un conjunt (llista/taula) de `NumRef`. Per tant, utilitza comparador de `NumRef` amb `String`.	231002_1...	02/10/23
Jerarquia de classes `Electrodomestic` a utilitzar en les pràctiques següents Considerar el model implementat en el projecte `000000_1_JerarquiaElectrodomestic` que conté: • Classe abstracta `Electrodomestic`, amb definició de dades comunes a totes les tipologies d'electrodomèstics. És un exemple acadèmic amb problemàtiques per una aplicació real. • Classe `GammaBlanca` per gestionar electrodomèstics de gamma blanca • Classe `GammaGris` per gestionar electrodomèstics de gamma gris	000000_1...	02/10/23

Exercici per dimecres, 04/10
 	
Desenvolupar un projecte que únicament contingui la classe:
`ElectrodomesticFitxerBinariFormatManual` en
paquet `org.milaifontanals.persistence` amb utilitats:
 	
`saveListElectrodomestic(List<Electrodomestic> ll, String nomFitxer);`
per crear un fitxer de nom indicat que contingui els objectes de la llista indicada.
 	
`List<Electrodomestic>loadListElectrodomestic(String nomFitxer);`
per recuperar a partir d'un fitxer la llista d'objectes `Electrodomestic` continguda
 	
Requeriments:
 	
- El fitxer ha de ser binari
- Format intern de dades en el fitxer decidit pel programador; d'aquí el nom `FormatManual`
- Les dues utilitats han d'estar coordinades, és a dir, la utilitat `save` enregistra els electrodomèstics en un format decidit pel programador i la utilitat `load` els recupera segons el mateix format.
- No gestionar les dades automàtiques (`id`, `momentCreacio`, `momentLastModificacio`), que podríem enregistrar però no tenim manera de recuperar (no hi ha mètodes `setter`).
- No gestionar les dades estàtiques automàtiques (`qElectrodomesticsCreats` i `qElectrodomesticsVius`) que podríem enregistrar, però en el moment de recuperar, a banda que no es disposa de mètodes `setter`, no tindria sentit la seva recuperació, doncs el programa
	231004_1...	04/10/23

que invoqués la utilitat `load` podria tenir ja creats diversos electrodomèstics en memòria... - En cas d'error, han de generar `ElectrodomesticFitxerBinariFormatManualException` Desenvolupar programes que comprovin el correcte funcionament de les dues utilitats. Ubicar els programes de prova a la zona *Test* dels projectes NetBeans. **Observacions a la solució**: • Cal usar marques per informar de la tipologia dels objectes que anem a enregistrar i així disposar d'informació en recuperar la informació. En el programa solució proposat, s'ha utilitzat un caràcter `B` per indicar que el segueix un objecte `GammaBlanca` i un caràcter `G` per indicar que el segueix un objecte `GammaGris`. • Donat que no es pot enregistrar valors `null`, per les dades que poden valer `null` cal usar marca informativa a usar en la recuperació de la informació. En el programa solució proposat, pels camps que poden valer `null`, s'ha utilitzat un caràcter `N` si contenen valor `null` (i no s'enregistra cap valor) i un caràcter `S` si no contenen valor `null`, seguit del valor. • La utilitat `load`, en el moment de recuperar un objecte `GammaBlanca` o `GammaGris`, ha d'invocar el constructor de la corresponent classe. En la solució proposada, per evitar "repetir" la lectura dels camps comuns (classe `Electrodomestic`) quan toca recuperar un `GammaBlanca` i quan toca recuperar un `GammaGris`, s'ha desenvolupat el mètode `loadElectrodomestic` que s'encarrega de recuperar les dades comunes i crear l'objecte que correspongui, motiu pel qual necessita una marca per saber què ha de crear. El problema que sorgeix és que segons els constructors que faciliten les classes `GammaBlanca` i `GammaGris`, és possible que no tinguem encara les dades adequades i calgui usar uns valors falsos, com les `mides` en `GammaBlanca` i el `model` en `GammaGris`. **Atenció!** • No és un bon sistema per emmagatzemar objectes que continguin camps que s'omplen automàticament i pels que no hi ha possibilitat d'incidir en la seva creació/modificació, com són els camps `id`, `momentCreacio` i `momentLastModificacio`. • Tampoc és un bon sistema si la classe té dades estàtiques que es gestionen a partir de la creació i destrucció d'objectes (no és usual...), ja que la recuperació d'aquests objectes persistents altera el seu contingut. • Pot no interessar enregistrar/recuperar alguna dada estàtica, doncs potser en el programa que hagi d'usar la utilitat `load` ja té un valor per les dades estàtiques que vulgui mantenir.		
Serialització d'objectes - Apartat 1.5 de DAM_M06_UF1_Fitxers_1_Fluxos&Seriació.pdf Verb correcte? **seriar** o **serialitzar**? • 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. • 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. **Exemple**: Reversionar la classe `ElectrodomesticFitxerBinariFormatManual` (projecte 231004_1) per aconseguir la classe `ElectrodomesticFitxerBinariSerialitzable` amb	231006_1... 000000_2...	06/10/23

les mateixes utilitats, usant serialització. **Solució**: El projecte 231006_1 facilita les matixes utilitats que enregistren/recuperen una llista d'objectes `Electrodomestic` via serialització. Per fer-ho possible, ha calgut retocar la definició de la classe `Electrodomestic` fent-la `Serializable` (projecte 000000_2). Observar que: • Hem serialitzat una `List` sencera. En el nostre cas és possible per què la implementació usada de `List` (`ArrayList`) `implements Serializable`. , però compte... la interfície List no implementa `Serializable` i, per tant, hi pot haver classes que implementin List i no siguin serialitzables. • En cas que la llista contingui una posició `null`, la recuperació de la llista el mantindria. • Què passa amb els camps amb contingut automàtic assignat pels constructors (`id`, `momentCreacio`, `momentLastModificacio`)? Doncs que es mantenen... La serialització va "per sota de constructors" i enregistra i recupera bytes. • Què passa amb les dades estàtiques? No s'enregistren ni es recuperen. Si es vol conservar dades estàtiques cal usar serialització personalitzada.		

Java – Serialització personalitzada => Implica retocar la classe a serialitzar
 	
Revisar apartat 1.5.1. del dossier:
 	
• 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 emplena 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`
 	
Cal mantenir el prototipus indicat i dins aquests mètodes es pot invocar els mètodes `out.defaultWriteObject` i `in.defaultReadObject` si es necessita executar el procés automàtic de serialització.
 	
Exemple: Millorar projecte 231006_1... per aconseguir recuperar la dada estàtica `dteVigent`.
 	
Solució: Projecte 231009_1... idèntic a 231006_1... però basat en nova versió de la classe `Electrodomèstic` (000000_3) que incorpora serialització personalitzada.
 	
Exemple: Retocar projecte 231009_1... anterior per aconseguir que el preu es serialitzi amb un 10% d'increment.
 	
Solució: Projecte 231009_2... idèntic a 231009_1... però basat en nova versió de la classe `Electrodomèstic` (000000_4) que retoca serialització personalitzada per assolir el requeriment, on hem marcat com a `transient` el camp preu per a que la serialització automàtica no el tingui en compte, ja que en fem un tractament manual dins els mètodes `writeObject` i `readObject`.
	231009_1...
	000000_3...
	
	231009_2...
	000000_4...
		09/10/23

Retoc a la classe `Electrodomestic` per a les pràctiques següents Hem constatat que en una classe amb dades automàtiques (`id`, `momentCreacio`, `momentLastModificacio`, `qElectrodomesticsCreats`, `qElectrodomesticsVius`), la tècnica de serialització és capaç d'enregistrar-les i de recuperar-les, però altres tècniques més manuals on calgui usar constructor i mètodes `setter` per reconstruir els objectes no permeten la		

seva recuperació, doncs la classe les gestiona de manera automàtica. Per simplificar, treballarem amb una versió de `Electrodomestic` que no contingui aquestes dades automàtiques. Es facilita una reversió de la jerarquia `Electrodomestic` on: - S'ha eliminat els camps automàtics. - Es manté un camp `id` però amb gestió manual. - S'ha retocat adequadament els mètodes afectats (constructors, `getters`, `setters`, `toString`...) La nova versió de `Electrodomestic` i les seves derivades resideix en el projecte 000000_5.		

Exercici (a desenvolupar dimecres, 11 d'octubre)
 	
Desenvolupar
classe `ElectrodomesticFitxerXmlV1ViaJDom` en
paquet `org.milaifontanals.persistence` que contingui utilitats:
 	
`saveListElectrodomestic(List<Electrodomestic> ll, String nomFitxer);`
per crear un fitxer de nom indicat que contingui els objectes de la llista indicada.
 	
`List<Electrodomestic> loadListElectrodomestic(String nomFitxer);`
per recuperar a partir d'un fitxer la llista de Electrodomestics que conté
 	
Requeriments:
 	
- El fitxer ha de ser XML, validat pel fitxer **`listHomeApplianceV1.dtd`** facilitat.
- Cal usar l'API JDom
- La utilitat `save` ha de generar el fitxer de sortida amb l'etiqueta `!DOCTYPE`.
- La utilitat `load` ha de validar el fitxer d'entrada.
- El DTD inclou al final alguns requeriments de format que cal verificar.
- En cas d'error, han de generar `ElectrodomesticFitxerXmlV1ViaJDomException`
 	
Comprovar el funcionament de les dues utilitats.
 	
Observacions per gestionar números amb un determinat format (es demana coma decimal):
 	
Coneixem la classe `SimpleDateFormat` per gestionar dates amb un determinat format, usant:
`SimpleDateFormat sdf = new SimpleDateFormat (<format que interessa>);`
`sdf.format(d)`, per obtenir la representació textual de la data d en el format definit.
`sdf.parse(s)`, per obtenir la data a partir del contingut de la cadena `s`.
 	
Per gestionar números amb un determinat format, disposem de la classe `DecimalFormat`, amb mètodes similars:
`DecimalFormat df = new DecimalFormat(<format que interessi>);`
`df.format(valorNumeric)`, per obtenir la representació textual del valor numèric
`df.parse(s)`, per obtenir valor `Number` a partir del contingut de la cadena s, sobre el qual caldrà aplicar mètodes `floatValue()`, `doubleValue()`... per obtenir el valor del tipus/classe que correspongui.
 	
Per usar uns símbols concrets per al format, abans d'usar els mètodes `format` o `parse`, cal assignar-los, amb mètodes:
`DecimalFormatSymbols dfs = new DecimalFormatSymbols();`
`dfs.setDecimalSeparator(',');` // Per assignar el caràcter separador decimal
`df.setDecimalFormatSymbols(dfs);` // Per assignar el conjunt de símbols `dfs` a objecte `df` que usarem per gestionar (`format`/`parse`) els valors numèrics.
 	
Consell per quan cal formatar dates i/o números en diversos mètodes
 	
En una classe on cal usar `format`/`parse` de dates i/o números en diversos mètodes, enlloc de definir els objectes `SimpleDateFormat` i `DecimalFormat` repetidament en els diversos mètodes, és aconsellable definir-los una única vegada com a dades estàtiques de la classe i usar-
	231011_1...	11/10/23

los allà on calgui.

Observacions per gestionar camps optatius:

Quan un camp és optatiu, el DTD valida com a correcte tant si la corresponent etiqueta no existeix com si hi és i està buida. Per tant:

- En generar l'XML podem optar per generar l'element sense contingut o no generar-lo.
- En recuperar de l'XML hem de gestionar-ho correctament, tant si es troba l'element sense contingut com si no es troba l'element.

Respecte la solució proposada:

- Crea `dfs` i `df` com a estàtics i, per tant, el codi necessari per assignar la coma com a separador decimal per a `df`, cal fer-ho en un bloc inicialitzador estàtic.
- La utilitat `save` no enregistra els camps optatius (que podria enregistrar-los buits), però la utilitat `load` es preocupa de recuperació correcta de camps optatius, tant si no hi ha element com si hi és buit.

Exercici per practicar en el pont del Pilar – No es resoldrà a classe

Repetir el programa anterior de manera que l'XML gestionat validi **`listHomeApplianceV2.dtd`**.

Canvieu en el nom de les classes, la partícula V1 per V2.

Persistència XML via XmlBinding

JAXB (Java Architecture for XML Binding) permet mapar classes Java en representacions XML.

- 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

Els projectes anteriors defineixen l'especificació de l'API i per usar l'API en programes cal disposar d'alguna implementació.

- 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 (`jaxb-api.jar`) 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 `javax.xml.bind` i **javax** i **java** són noms propietat d'Oracle.

16/10/23

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.

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.

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 `javax.xml.bind` ha passat a anomenar-se `jakarta.xml.bind`.

Per ajudar a entendre aquest embolic, considerem els 3 paquets 230916_1... que es faciliten:

- `230916_1_XmlBinding_JAXB235`, que incorpora la llibreria JAXB 2.3.5
  Observem que el projecte compila i que podem executar qualsevol dels seus programes `Pxx`.
  Fixem-nos en els imports dels paquets `javax.xml.bind`.
  Malauradament l'ajuda no funciona.

- `230916_1_XmlBinding_Jakarta10EE`, que incorpora la llibreria Jakarta 10 EE API.
  En aquest cas, l'ajuda és accessible.
  Fixem-nos en els imports dels paquets `jakarta.xml.bind`.
  Observem que el projecte compila però que en executar qualsevol dels seus programes `Pxx` ens trobem l'error *Implementation of Jakarta XML Binding-API has not been found*, degut a que no incorpora la implementació.
- Per tant si volem usar l'especificació més actual de `XmlBinding` (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 jaxb-ri-4.0.3 i incorporem a NetBeans.18 la llibreria de nom `JAXB-RI 4.0.3` 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 `230916_1_XmlBinding_JAXB-RI-4.0.3` facilitat.

Documentació referent a la utilització de XmlBinding:

- Dossier de l'IOC: DAM_M06_UF1_Fitxers_3_Binding.pdf (molt espès)
- Tutorial de Java
- Dades i anotacions bàsiques:

  `@XmlRootElement`, per l'arrel de la classe
  `@XmlElement`, pels camps de la classe que es corresponguin amb elements de l'XML (per defecte)
  `@XmlAgttribute`, 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`).

  `@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).

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

  On incorporar les marques?
  - 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)

Exemples de **XmlBinding** – Projecte `231018_1_XmlBinding_Exemples`

1. Classe simple preparada per a poder fer *binding* dels seus objectes
   `Customer.java`
   Programes que en comproven el funcionament per a un objecte:
   `P01_JAXBConvertCustomerToXML.java`
   `P02_JAXBConvertXMLToCustomer.java`
   Programes que fan binding d'una col·lecció d'objectes de la classe:
   `P03_JAXBConvertSeveralCustomerToXML.java`
   `P04_JAXBConvertXMLToSeveralCustomer.java`
2. Classe que conté referències a objectes d'altres classes:
   `Product.java`
   `CustomerWithOneProduct.java`
   Programes que en comproven el funcionament:
   `P05_JAXBConvertCustomerWithOneProductToXML.java`
   `P06_JAXBConvertXMLToCustomerWithOneProduct.java`
3. Classe que conté una col·lecció d'objectes:
   `Product.java`
   `CustomerWithSeveralProductVer1.java`
   En aquest cas, els diversos productes no queden englobats dins un node (no s'utilitza `@XmlElementWrapper`)
   `CustomerWithSeveralProductVer2.java`
   En aquest cas, els diversos productes queden englobats dins un node `<products>` (s'utilitza `@XmlElementWrapper`)
   Programes que en comproven el funcionament:
   `P07_JAXBConvertCustomerWithOneProductVer1ToXML.java`
   `P08_JAXBConvertXMLToCustomerWithOneProductVer1.java`
   `P09_JAXBConvertCustomerWithOneProductVer2ToXML.java`
   `P10_JAXBConvertXMLToCustomerWithOneProductVer2.java`

18/10/23

4. Conversions especials, com afegir un caràcter o enregistrar una data en un format concret. Concretament, observem la nova versió de la classe `Customer`, que conté la `dataAlta` i volem que aquesta data s'enregistri en un format concret i també que, el camp `id`, en el fitxer XML quedi amb un prefix $.

   Per aconseguir-ho tenim dos mecanismes:

   - Fer "transient" les propietats `getter` corresponents i crear unes noves propietats `getter` 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 `CustomerAmbConversionsEspecialsVer1.java`

     Programes que en comproven el funcionament:

     `P11_JAXBConvertCustomerToXMLAmbConversionsEspecialsVer1`
     `P12_JAXBConvertXMLToCustomerAmbConversionsEspecialsVer1`

   - Marcar les propietats `getter` habituals amb l'anotació `@XmlJavaTypeAdapter` que permet indicar una classe `XmlAdapter` que específica que s'encarrega de la conversió i que hem de dissenyar.

     Observar-ho a la classe `CustomerAmbConversionsEspecialsVer2.java`

     Atenció!!!
     - Les classes `XmlAdapter` no gestionen els camps `null`.
     - Les classes `XmlAdapter` només "adapten" classes i no tipus primitius.

     Programes que en comproven el funcionament:

     `P13_JAXBConvertCustomerToXMLAmbConversionsEspecialsVer2`
     `P14_JAXBConvertXMLToCustomerAmbConversionsEspecialsVer2`

Com diferenciar els noms dels elements vinculats a una col·lecció

Suposem que en un fitxer XML tenim un node com:

```
<dada>
  <tipusA ... />
  <tipusB ... />
  <tipusA ... />
</dada>
```

on els fills "tipusA" i "tipusB" corresponen a classes derivades d'una mateixa classe "tipus", que formen part d'una col·lecció que hem de gestionar "adequadament" via JAXB.
Cal usar marcatge:

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

Particularitats de JAXB:

Pels camps `char`, JAXB els transforma en el seu codi ASCII, cosa que segurament NO es vol, doncs si tenim un camp `char` en un objecte amb valor `'A'`, JAXB el passaria a XML com `"65"`. Per tant, si cal marcar via JAXB algun camp `char`, caldrà fer-ho via accessors (`get-set`) adequats.

Exercici, per començar a casa i finalitzar divendres, 20 d'octubre a classe

Desenvolupar
classe `ElectrodomesticFitxerXmlV1ViaJAXB` en
paquet `org.milaifontanals.persistence` que contingui utilitats:

```
saveListElectrodomestic(List<Electrodomestic> ll, String nomFitxer);
```

per crear un fitxer de nom indicat que contingui els objectes de la llista indicada.

```
List<Electrodomestic> loadListElectrodomestic(String nomFitxer);
```

per recuperar a partir d'un fitxer la llista de Electrodomestics que conté

Requeriments:

- El fitxer ha de ser XML, validat pel fitxer **`listHomeApplianceV1.dtd`** facilitat.
- Cal usar l'API JAXB => Caldrà nova versió de `JerarquiaElectrodomestic` amb els corresponents retocs
- La utilitat `save` ha de generar el fitxer de sortida amb l'etiqueta `!DOCTYPE`.
- La utilitat `load` ha de validar el fitxer d'entrada.
- El DTD inclou al final alguns requeriments de format que cal verificar.
- En cas d'error, han de generar `ElectrodomesticFitxerXmlV1ViaJAXBException`

Desenvolupar programes que comprovin funcionament de les dues utilitats.

Ajudes:

- En classes abstractes no usar `@XmlRootElement` i posar-lo a les classes derivades.
- Per recuperar la llista en el mateix ordre, donat que el DTD indica que pot contenir elements `whiteRange` i `grayRange` barrejats, caldrà recuperar tots els fills de l'arrel i processar-los un a un. A més, com que en invocar el mètode `unmarshal` cal fer `cast` cap a la classe corresponent, cal saber de quina classe és l'objecte que JAXB ha recuperat.
- Per formatar les dates i els valors decimals segons requeriments, es pot crear classes `Adapter` en un paquet específic i així les podem usar en diversos projectes/classes. Així, la classe que adapta els valors decimalses pot usar en `Electrodomestic` (`preu`) i en `GammaBlanca` (`altura`, `amplada`, `fondaria`).
- Degut a que els camps `preu`, `altura`, `amplada` i `fondaria` són tipus primitius, NO podem aplicar sobre ells la classe `Adapter` i caldrà crear mètodes específics `getPreuByJaxb`, `getAlturaByJaxb`,... i els camps originals marcar-los `@XmlTransient`. Degut a això, a la propietat `@XmlPropOrder` caldrà indicar els noms dels nous camps.

000000_6...
231020_1...

20/10/23

Exercici, per començar a casa i finalitzar dilluns, 23 d'octubre a classe Repetir el programa anterior de manera que l'XML gestionat validi **listHomeApplianceV2.dtd**. **Ajudes**: - Per aconseguir crear un subnivell (elements `whiteRange` i `grayRange` dins l'element `homeAppliance`) es pot crear una classe auxiliar (`GammaBlancaAuxByJaxb` i `GammaGrisAuxByJaxb`) que englobi els camps que ha de contenir el subnivell i afegir dins la classe original un camp objecte d'aquesta classe, marcant-lo amb JAXB per a que JAXB el tingui en compte i deixant els camps originals amb `@XmlTransient`. Veure `000000_7`. - En ocasions, com en aquest exemple, objectes de diferents classes (`GammaBlanca` i `GammaGris`) estan mapats sota un mateix element XML (<homeAppliance>). En el procés de conversió d'objecte a XML (`marshall`) no hi ha problema, però en el procés contrari, de conversió d'XML a objecte, apareix un problema si el conversor (`unmarshaller`) no sap a què ha de convertir l'element. Solució: - Tenir diferents `unmarshaller` per a cadascuna de les classes; per tant, caldrà tenir diferents `JAXBContext` per a cadascuna de les classes afectades. - En el moment d'invocar el `unmarshaller` a partir del codi XML, haver detectat prèviament la classe a la que s'ha de convertir i invocar l'`unmarshaller` adequat.	000000_7... 231023_1...	23/10/23