Escriure els nostres objectes en disc

El tractament de fitxers, ens serveix per poder donar persistència als nostres programes. Un cop gravades al disc les dades creades per un programa, podem finalitzar la seva execució, amb la seguretat de que les dades importants presents a la memòria han estat convenientment preservades. Posteriorment, quan tornem a executar el programa, podrem recuperar les dades que tenim emmagatzemades al disc i recuperar l'estat del programa previ a la seva finalització.

Probablement, les dades que més ens interessarà conservar al disc, son els objectes que ens creem a partir de les nostres classes. Imagineu-vos una base de dades d'empleats, on hem introduït les dades de tots els empleats de la nostra organització. Aquest procés pot ser considerablement lent, i no el podem fer cada cop que executem el programa. Seria interessant poder desar aquesta informació al disc, i recuperar-la quan vulguem. Com podem fer això és el que anem a comentar en aquest document.

Exemple de programa per desar un vector d'objectes al disc

A continuació, es mostra un programa que crea un vector d'objectes de la classe **Persona** i el desa al disc de dues formes diferents, com un únic objecte, el vector en si, o com una sèrie d'objectes **Persona**. Posteriorment, el mateix programa recupera els objectes del disc (de les dues formes també).

```
1  /*          Programa per provar de guardar els nostres objectes en disc.
2              Realitzat per:   José Javier Faro Rodríguez
3     Gravarem un vector de classe Persona de dues formes diferents:
4       * Com un únic objecte que és el vector en si (una única escriptura al disc)
5       * Com n objectes Persona (tantes escriptures com objectes tingui el vector)
6     Les classes que guardem han d'implementar la interfície Serializable
7  */
8  import java.io.Serializable;        // Hem realitzat únicament els import que fem servir
9  import java.io.File;
10 import java.io.EOFException;
11 import java.io.FileInputStream;
12 import java.io.FileOutputStream;
13 import java.io.ObjectOutputStream;
14 import java.io.FileNotFoundException;
15 import java.io.ObjectInputStream;
16 import java.util.Scanner;
17
18 // Classe que gravarem al disc.  (només afegir: implements Serializable)
19 class Persona implements Serializable {
       private String nom;
       private int edat;
       private char sexe;       // 'H'-Home  o 'D'-Dona
23     private long dni;        // Número sense la lletra
24
25     public long getDni() {  // Faltarien procediments de propietat i altres mètodes
26         return dni;
27     }
28     public void setDni(long dni) {
29         this.dni = dni;
30     }
31     public Persona(String nom, int edat, char sexe, long dni) {
32         this.nom = nom;
33         this.edat = edat;
34         this.sexe = sexe;
35         this.dni = dni;
36     }
37     @Override
       public String toString() {
39         return "Persona{" + "nom=" + nom + ", edat=" + edat + ", sexe=" +
40                 (sexe=='H'? "Home":"Dona") + ", dni=" + dni + '}';
41     }
42 }
43
44 public class FileObjects {
```

```
45      // numPersones, indica quantes persones a l'organització (a l'exemple només 5)
46      private static int numPersones = 5;
47      // Creem un vector d'objectes de la classe Persona (amb 2 objectes més buits = null)
48      private static Persona [] vPersona = new Persona[7];    // Posarem 5 persones només
49      // Fitxer que emmagatzema el vector de persones com un únic objecte
50      private static String fitxerPersonal = "C:/Java/Fitxers/Personal.bin";
51      // Fitxer que emmagatzema el vector de persones com objectes Persona individuals
52      private static String fitxerIndividuals = "C:\\Java\\Fitxers\\Persones.bin";
53
54      // Funció que espera una resposta S o N de part de l'usuari, escriu la pregunta i
55      // no acaba fins que l'usuari respón de forma adequada  (Si, S, N, No, etc...)
56      private static boolean siNo (String pregunta)
57      {
58          Scanner entrada = new Scanner(System.in);
59          char car;
60          do
61          {
62              System.out.print(pregunta+" [S/N] ? ");
63              car = entrada.next().toUpperCase().charAt(0);
64              if (car != 'N' && car != 'S')
65                  System.out.println("S'esperava una resposta S/N ...");
66          }
67          while (car != 'N' && car != 'S');
68          return car == 'S';
69      }
70
71      private static boolean escriuVectorPersonesDeCop(String nomFitxer, Persona [] vp){
72          // Escriu al fitxer un vector sencer de persones (un objecte). Sobreescriu el fitxer
73          try{
74              // Si el fitxer existeix, pregunta si el volem reescriure
75              File fitxer = new File(nomFitxer);
76              if (fitxer.exists())
77                  if (!siNo("El fitxer ja existeix.  Vol reescriure'l?"))
78                      return false;    // No s'ha escrit el fitxer
79          }catch (Exception e){
80              System.out.println("Error accedint al sistema de fitxers");
81              System.out.println(e.getMessage());
82              return false;           // No s'ha escrit el fitxer
83          }
84          // Si estem aquí, el fitxer no existeix o be el volem reescriure
85          try (FileOutputStream fo = new FileOutputStream(nomFitxer, false);
86              ObjectOutputStream obo = new ObjectOutputStream(fo)){
87              // Escrivim totes les persones del vector com un únic objecte (vector)
88              obo.writeObject(vp);
89              return true;        // Indiquem que el procés s'ha completat correctament
90          }catch (Exception e){
91              System.out.println("Error en escriure el vector de persones al fitxer");
92              System.out.println(e.getMessage());
93              return false;       // Indiquem que el procés no s'ha completat correctament
94          }
95      }
96
97      private static Persona[] llegeixVectorPersones(String nomFitxer){
98          // Llegeix totes les persones del fitxer com un únic objecte (un vector)
99          try (FileInputStream fi = new FileInputStream(nomFitxer);
100             ObjectInputStream obi = new ObjectInputStream(fi)) {
101             // Llegim totes les persones de cop, i les retornem directament
102             // El vector llegit es crearà de la mateixa longitud que es va
103             // escriure, amb 2 elements null al final (7 persones de capacitat).
104             return (Persona []) obi.readObject();
105         }catch (FileNotFoundException e) {
106             System.out.println("Fitxer no trobat: " + nomFitxer);
107         } catch (EOFException e) {
108             System.out.println("Hem arribat al final del fitxer");
109             System.out.println(e.getMessage());
110         } catch (ClassNotFoundException e) {
```

```
111             System.out.println("Format de fitxer incompatible");
112         } catch (Exception e) {
113             System.out.println("Error accedint al fitxer: " + nomFitxer);
114             System.out.println(e.getMessage());
115         }
116         return null;
117     }
118 
119     private static boolean escriuVectorPersonesIndividuament(String nomFitxer, Persona [] vp){
120         // Escriu al fitxer un vector sencer de persones, però una a una. Sobreescriu aquest
121         try{
122             File fitxer = new File(nomFitxer);
123             if (fitxer.exists())
124                 if (!siNo("El fitxer ja existeix.  Vol reescriure'l?"))
125                     return false;   // No s'ha escrit el fitxer
126         }catch (Exception e){
127             System.out.println("Error accedint al sistema de fitxers");
128             System.out.println(e.getMessage());
129             return false;           // No s'ha escrit el fitxer
130         }
131         // Si estem aquí, el fitxer no existeix o el volem reescriure
132         try (FileOutputStream fo = new FileOutputStream(nomFitxer, false);
133             ObjectOutputStream obo = new ObjectOutputStream(fo)){
134             // Escrivim les persones del vector una a una al fitxer, però no els null
135             // Només escrivim les 5 persones reals que hi ha al vector
136             for (int n=0; n<numPersones; n++)
137                 obo.writeObject(vp[n]);
138             return true;
139         }catch (Exception e){
140             System.out.println("Error en escriure les persones al fitxer");
141             System.out.println(e.getMessage());
142             return false;
143         }
144     }
145 
146     // Funció per llegir el vector de persones una a una del fitxer
147     private static Persona[] llegeixPersonesUnaAUna(String nomFitxer){
148         Persona [] vPerson = new Persona[8];    // Declarem del màxim esperat (abans 7)
149         // Llegeix les persones del vector individualment.  No s'han escrit els null
150         try (FileInputStream fi = new FileInputStream(nomFitxer);
151             ObjectInputStream obi = new ObjectInputStream(fi)) {
152             numPersones = 0;    // Indicarà el número de persones que hem llegit del fitxer
153             // Es provocarà una excepció, només hi ha 5 persones al fitxer, mirem de llegir 8
154             while (numPersones < 8){
155                 // Llegim cada persona individualment e incrementem el nombre de llegits
156                 vPerson[numPersones] = (Persona) obi.readObject();
157                 numPersones++;      // Incrementem el nombre de persones llegides
158             }
159             return vPerson;     // La funció retornaria el vector llegit (no ho farà aquí)
160         }catch (FileNotFoundException e) {
161             System.out.println("Fitxer no trobat: " + nomFitxer);
162         } catch (EOFException e) {
163             // El fitxer no té tots els elements esperats (en té 5 únicament)
164             System.out.println("Final del fitxer, contenia " + numPersones + " persones");
165             // Aquí retornem el vector que hem declarat de 8 persones, amb únicament 5 elements
166             return vPerson;
167         } catch (ClassNotFoundException e) {
168             System.out.println("Format de fitxer incompatible");
169         } catch (Exception e) {
170             System.out.println("Error accedint al fitxer: " + nomFitxer);
171             System.out.println(e.getMessage());
172         }
173         return null;
174     }
175     private static void ompleVector(Persona [] vOmplir){
176         // Cridem al constructor per a les 5 persones de l'organització
```

```
177             vOmplir[0] = new Persona("Felipe",28,'H',11111111);
178             vOmplir[1] = new Persona("Julian",32,'H',22222222);
179             vOmplir[2] = new Persona("Ana",35,'D',33333333);
180             vOmplir[3] = new Persona("Angela",56,'D',44444444);
181             vOmplir[4] = new Persona("Maria",23,'D',55555555);
182         }
183         public static void main(String[] args) {
184             // Omplim el vector de persones per treballar-ho
185             ompleVector(vPersona);  // Només 5 persones, el vector té longitud 7 (2 null)
186             // Creem el vector on llegirem les persones (Podria ser el mateix)
187             Persona [] vLlegit;     // Vector de persones que llegirem (es farà de longitud 7)
188             // Escrivim al disc tot el vector de persones de cop
189             if (escriuVectorPersonesDeCop(fitxerPersonal,vPersona)){
190                 // Si l'escriptura del vector s'ha realitzat correctament, ho indica i el llegeix
191                 System.out.println("Escrites un total de " + numPersones + " persones al vector");
192                 System.out.println("El vector té tanamateix longitud 7, escrivim també els 2 null");
193                 // Llegim del disc tot el vector de persones de cop, quedarà amb longitud 7
194                 vLlegit = llegeixVectorPersones(fitxerPersonal);
195                 // Escrivim les persones del vector, fixeu-vos que hi ha 2 null
196                 numPersones = 0;        // Per comptar les persones que hem llegit del vector
197                 for (int n=0; n < vLlegit.length; n++){
198                     System.out.println(vLlegit[n]);
199                     if (vLlegit[n] == null && numPersones == 0)
200                         numPersones = n;
201                 }
202                 System.out.println("Llegides un total de " + numPersones + " persones del disc");
203             }
204             // El mateix, però escrivint les persones una a una al fitxer
205             if (escriuVectorPersonesIndividuament(fitxerIndividuals,vPersona)){
206                 // Si s'han escrit les persones al vector correctament, ho indiquem
207                 System.out.println("Escrites un total de " + numPersones + " persones al vector");
208                 System.out.println("El vector tenia longitud de 8, no hem escrit els 3 null");
209                 // Aquí, el vector té longitud 8 (persones màximes que esperem llegir del fitxer)
210                 vLlegit = llegeixPersonesUnaAUna(fitxerIndividuals);
211                 // Escrivim les persones del vector, fixeu-vos que hi ha 3 null al final
212                 numPersones = 0;        // Per comptar les persones que hem llegit del vector
213                 for (int n=0; n < vLlegit.length; n++){
214                     System.out.println(vLlegit[n]);
215                     if (vLlegit[n] == null && numPersones == 0)
216                         numPersones = n;
217                 }
218             }
219         }
220     }
```

Si executem aquest codi, obtindrem els missatges que s'indiquen a continuació:

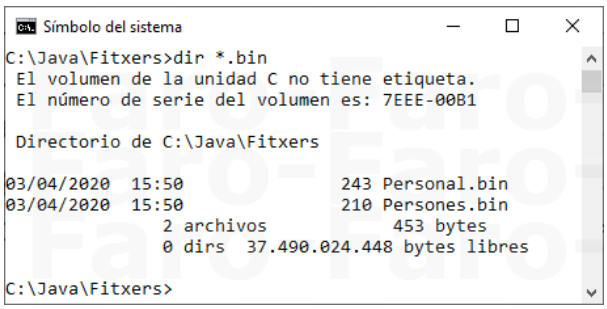
Escriptura del vector de 7 elements (5 plens) al disc com un únic objecte

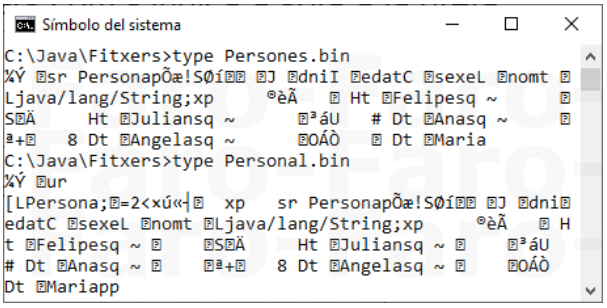
```
run:
Escrites un total de 5 persones al vector
El vector té tanamateix longitud 7, escrivim també els 2 null
Persona{nom=Felipe, edat=28, sexe=Home, dni=11111111}
Persona{nom=Julian, edat=32, sexe=Home, dni=22222222}
Persona{nom=Ana, edat=35, sexe=Dona, dni=33333333}
Persona{nom=Angela, edat=56, sexe=Dona, dni=44444444}
Persona{nom=Maria, edat=23, sexe=Dona, dni=55555555}
null
null
Llegides un total de 5 persones del disc
```

Escriptura del vector de 8 elements (5 plens) al disc com 5 objectes separats

```
Escrites un total de 5 persones al vector
El vector tenia longitud de 8, no hem escrit els 3 null
Final del fitxer, contenia 5 persones
Persona{nom=Felipe, edat=28, sexe=Home, dni=11111111}
Persona{nom=Julian, edat=32, sexe=Home, dni=22222222}
Persona{nom=Ana, edat=35, sexe=Dona, dni=33333333}
Persona{nom=Angela, edat=56, sexe=Dona, dni=44444444}
Persona{nom=Maria, edat=23, sexe=Dona, dni=55555555}
null
null
null
BUILD SUCCESSFUL (total time: 0 seconds)
```

Hem escrit al disc els dos vectors d'objectes de la classe **Persona**. El fitxer **Personal.bin**, manté un vector de **7** components, de les quals **2** no apunten a cap objecte (el seu valor és **null**). L'altre fitxer que escrivim des del programa és **Persones.bin**, que manté **5** objectes independents de la classe **Persona**. Podem veure els dos fitxers a continuació. Son fitxers binaris (tot i que l'extensió no caldria que fos **.bin**). Això ho podem constatar si fem un **type** dels fitxers com s'ha fet a sota a la dreta.





El contingut dels fitxers, evidentment no és exactament el mateix. **Persones.bin** manté els **5** objectes separadament, i la seva grandària és lleugerament inferior. Noteu que si el fitxer manté **5** objectes, i ocupa **210** bytes, podria donar-nos la sensació de que cada objecte ocupa al disc **210/5 = 42 bytes**, però això només és cert en mitjana, ja que no tots els objectes del fitxer ocuparan necessàriament el mateix espai (el nom de la persona per exemple, pot tenir llargàries diferents d'un objecte a un altre). Això fa que no puguem indexar aquest fitxer com si fos un vector per accedir aleatòriament a una persona determinada (cosa que podríem fer si tots els objectes tinguessin la mateixa grandària al disc.

Amb aquest programa prou exemplificador, donem per acabat el tema de llegir i escriure els nostres objectes al disc. Com veieu, els fitxers amb els que estem treballant son seqüencials, i llegim i escrivim tots els objectes del vector al disc. Un cop recuperat el vector de persones des del disc (per qualsevol dels dos mètodes comentats), podem afegir o treure elements en aquest vector (que evidentment, en un cas real tindrà molts més elements), i un cop actualitzat, tornar a desar-ho al disc.

Al document "**Manejo-básico-de-archivos-en-java.pdf**" que es pot descarregar de:
http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/4-manejo-bai81sico-de-archivos-en-java.pdf
a partir de la pàgina **19**, comenta com podem fer servir els fitxers binaris d'accés directe o aleatori (classe **RandomAccessFile** per emmagatzemar els nostres objectes. Evidentment, això passa per aconseguir una codificació on tots els objectes ocupin el mateix espai al disc. És una lectura molt interessant, però queda fora dels objectius d'aquest curs.