

Objectius: <ul style="list-style-type: none">- Conèixer com llegir i escriure objectes XML amb Java
Instruccions: <ul style="list-style-type: none">- Responen a l'espai de cada pregunta, si ho feu amb diapositives enganxeu la diapositiva en aquest mateix espai.- Es valorarà la presentació i els comentaris al codi
Criteris d'avaluació: <ul style="list-style-type: none">- Cada exercici té la mateixa puntuació
Entrega: <ul style="list-style-type: none">- Un arxiu .zip anomenat: PRx.y-NomCognom.zip<ul style="list-style-type: none">• PRx.y correspon al codi de la pràctica, per exemple PR1.1• NomCognom correspon al nom i primer cognom de cada participant- L'arxiu .zip conte:<ul style="list-style-type: none">• Aquest document emplenat en format .pdf anomenat memoria.pdf• Els arxius necessaris per fer anar la pràctica

Nom i Cognom: [Joel Berzal](#)

URL repositori GitHub: <https://github.com/joelberzalgithub/PR1.4-BerzalJoel>

Materials:

- Necessiteu una eina per programar en JAVA
- Feu servir Google per buscar els tutorials que us serveixin millor

Tasques, a cada exercici feu l'explicació i captures que cregueu convenientes

Preparació

Si no l'has creat, crea un 'Main.java' amb un menú per cridar cada una de les classes amb funció "main" d'aquesta activitat (consulta anteriors activitats)

```

1 import java.io.IOException;
2 import java.util.*;
3
4 public class Main {
5
6     static Scanner in = new Scanner(System.in); // System.in és global
7
8     public static void main(String[] args) throws InterruptedException, IOException {
9
10        boolean running = true;
11
12        while (running) {
13
14            System.out.println("Escull una opció:\n\n0) PR140Main\n1) PR141Main\n2) PR142Main\n100) Sortir\n");
15
16            try {
17
18                int opció = Integer.valueOf(llegirLinia("Opció: "));
19
20                switch (opció) {
21                    case 0: PR140Main.main(args);
22                        break;
23                    case 1: PR141Main.main(args);
24                        break;
25                    case 2: PR142Main.main(args);
26                        break;
27                    case 100: running = false;
28                        break;
29                    default: System.out.println("\nOpció fora del rang!");
30                        break;
31                }
32
33                System.out.println("\n*****\n");
34
35            } catch (Exception e) {
36                System.out.println("\nOpció no numérica!\n*****\n");
37            }
38        }
39    }
40
41    public static String llegirLinia (String text) {
42        System.out.print(text);
43        return in.nextLine();
44    }
45 }

```

Exercici 0 (2.5 punts)

- Crea un programa anomenat PR140Main.java.
- El programa ha de llegir el contingut del fitxer persones.xml (Veure [Arxiu "persones.xml"](#)).
- Mostra les dades llegides en una sortida per pantalla amb format de columnes alineades. Amb "columnes alineades" ens referim a que les dades de cada camp han d'aparèixer sota la seva respectiva capçalera en una presentació organitzada i fàcil de llegir.

```

1 import java.io.File;
2
3 import javax.xml.parsers.DocumentBuilderFactory;
4
5 import org.w3c.dom.Document;
6 import org.w3c.dom.Element;
7 import org.w3c.dom.Node;
8 import org.w3c.dom.NodeList;
9
10 public class PR140Main {
11
12     public static void main(String[] args) {
13
14         try {
15
16             // Crear la carpeta 'data' si no existeix
17
18             File dir = new File("data");
19
20             if (!dir.exists()) {
21                 if (!dir.mkdirs()) {
22                     System.out.println("Error en la creacio de la carpeta 'data'");
23                 }
24             }
25
26             // Analitza el fitxer XML
27
28             Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(new File(System.getProperty("user.dir") + "/data/persones.xml"));
29
30             // Obté una llista de tots els elements persona "persona" del document
31
32             NodeList listPersones = doc.getElementsByTagName("persona");
33
34             // Crea la part superior de la taula
35
36             System.out.println("");
37             System.out.println(String.format("%-20s %-20s %-10s %-20s\n", "Nom", "Cognom", "Edat", "Ciutat"));
38             System.out.println("-----\n");
39
40             // Bucle for per recórrer la llista de persones
41
42             for (int i = 0; i < listPersones.getLength(); i++) {
43                 // Obté la persona actual
44                 Node nodePersona = listPersones.item(i);
45                 // Comprova si la persona actual és un element
46                 if (nodePersona.getNodeType() == Node.ELEMENT_NODE) {
47                     // Converteix l'element actual en una persona
48                     Element elm = (Element) nodePersona;
49                     // **Obté el nom de la persona**
50                     String nom = elm.getElementsByTagName("nom").item(0).getTextContent();
51                     // **Obté el cognom de la persona**
52                     String cognom = elm.getElementsByTagName("cognom").item(0).getTextContent();
53                     // **Obté l'edat de la persona**
54                     String edat = elm.getElementsByTagName("edat").item(0).getTextContent();
55                     // **Obté el nom de la ciutat on viu la persona**
56                     String ciutat = elm.getElementsByTagName("ciutat").item(0).getTextContent();
57                     // Dona format a les dades obtingudes i les imprimeix per pantalla
58                     System.out.println(String.format("%-20s %-20s %-10s %-20s\n", nom, cognom, edat, ciutat));
59                 }
60             }
61
62         } catch (Exception e) {
63             e.printStackTrace();
64         }
65     }
66 }

```

Exercici 1 (2.5 punts)

Desenvolupa un programa en Java anomenat PR141Main.java que sigui capaç de generar un fitxer XML denominat “biblioteca.xml” amb el contingut que es mostra a l’annex [Arxiu “biblioteca.xml”](#).

```

14 public class PR141Main {
15
16     public static void main(String[] args) {
17
18         try {
19
20             // Crear la carpeta 'data' si no existeix
21
22             File dir = new File("data");
23
24             if (!dir.exists()) {
25                 if (!dir.mkdirs()) {
26                     System.out.println("Error en la creacio de la carpeta 'data'");
27                 }
28             }
29
30             // Crea un nou document XML
31
32             Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
33
34             // Crea l'element root del document XML
35
36             Element elmRoot = doc.createElement("biblioteca");
37
38             // Afegeix l'element root al document XML
39
40             doc.appendChild(elmRoot);
41
42             // Crea l'element "llibre", li afegeix l'atribut "id" i afegeix aquest element a l'element root
43
44             Element elmLlibre = doc.createElement("llibre");
45             elmLlibre.setAttribute("id", "001");
46             elmRoot.appendChild(elmLlibre);
47
48             // Crea l'element "titol" i l'afegeix a l'element "llibre"
49
50             Element elmTitol = doc.createElement("titol");
51             elmTitol.appendChild(doc.createTextNode("El viatge dels venturons"));
52             elmLlibre.appendChild(elmTitol);
53
54             // Crea l'element "autor" i l'afegeix a l'element "llibre"
55
56             Element elmAutor = doc.createElement("autor");
57             elmAutor.appendChild(doc.createTextNode("Joan Pla"));
58             elmLlibre.appendChild(elmAutor);
59
60             // Crea l'element "anyPublicacio" i l'afegeix a l'element "llibre"
61
62             Element elmAnyPublicacio = doc.createElement("anyPublicacio");
63             elmAnyPublicacio.appendChild(doc.createTextNode("1998"));
64             elmLlibre.appendChild(elmAnyPublicacio);
65
66             // Crea l'element "editorial" i l'afegeix a l'element "llibre"
67
68             Element elmEditorial = doc.createElement("editorial");
69             elmEditorial.appendChild(doc.createTextNode("Edicions Mar"));
70             elmLlibre.appendChild(elmEditorial);
71
72             // Crea l'element "genre" i l'afegeix a l'element "llibre"
73
74             Element elmGenre = doc.createElement("genre");
75             elmGenre.appendChild(doc.createTextNode("Aventura"));
76             elmLlibre.appendChild(elmGenre);
77
78             // Crea l'element "pagines" i l'afegeix a l'element "llibre"
79
80             Element elmPagines = doc.createElement("pagines");
81             elmPagines.appendChild(doc.createTextNode("320"));
82             elmLlibre.appendChild(elmPagines);
83
84             // Crea l'element "disponible" i l'afegeix a l'element "llibre"
85
86             Element elmDisponible = doc.createElement("disponible");
87             elmDisponible.appendChild(doc.createTextNode("true"));
88             elmLlibre.appendChild(elmDisponible);
89
90             // Crea un transformador XSLT
91
92             Transformer transformer = TransformerFactory.newInstance().newTransformer();
93
94             // Estableix la propietat OMIT_XML_DECLARATION a "no" per no ometre la declaració XML del document XML resultant
95
96             transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
97
98             // Estableix la propietat INDENT a "yes" per indentar el document XML resultant
99
100             transformer.setOutputProperty(OutputKeys.INDENT, "yes");
101
102             // Crea una instància de DOMSource a partir del document XML
103
104             DOMSource source = new DOMSource(doc);
105
106             // Crea una instància de StreamResult a partir del camí del fitxer XML
107
108             StreamResult result = new StreamResult(new FileOutputStream(new File(System.getProperty("user.dir") + "/data/biblioteca.xml")));
109
110             // Transforma el document XML especificat per source i escriu el document XML resultant a l'objecte especificat per result
111
112             transformer.transform(source, result);
113
114         } catch (Exception e) {
115             e.printStackTrace();
116         }
117     }
118 }

```

Exercici 2 (5 punts)

Crea un programa Java anomenat 'PR142Main.java' per gestionar l'arxiu XML [Arxiu "cursos.xml"](#) mostrat a l'annex.

Considera:

- Cada curs té:
 - Un codi identificatiu.
 - Un tutor.
 - Una llista d'alumnes.
 - Almenys un mòdul.
- Cada mòdul consta de:
 - Un codi identificatiu.
 - Un títol.
 - Almenys un professor.
 - Diverses unitats formatives.

Funcions del programa:

- Llistar ids de cursos, tutors i total d'alumnes.
- Mostrar ids i títols dels mòduls a partir d'un id de curs.
- Llistar alumnes d'un curs.
- Afegir un alumne a un curs.
- Eliminar un alumne d'un curs.

Nota: Utilitza **XPath** per navegar per l'arbre XML.

```
26 public class PR142Main {
27
28     static Scanner in = new Scanner(System.in); // System.in és global
29
30     public static void main(String[] args) throws InterruptedException, IOException {
31
32         try {
33
34             // Analitza el fitxer XML i crea un document XML
35
36             Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(new File(System.getProperty("user.dir") + "/data/cursos.xml"));
37
38             // Crea un objecte XPath
39
40             XPath xPath = XPathFactory.newInstance().newXPath();
41
42             // Crea el menú principal
43
44             boolean running = true;
45
46             while (running) {
47
48                 System.out.println("\n*****\n\nEscull una opció:\n\n0) Llistar ids de cursos" +
49                     "\n1) Llistar tutors\n2) Llistar total d'alumnes\n3) Mostrar ids i títols dels mòduls\n4) Llistar alumnes d'un curs" +
50                     "\n5) Afegir un alumne a un curs\n6) Eliminar un alumne d'un curs\n100) Tornar enrere\n");
51
52             }
```

```

51
52     try {
53
54         int opcio = Integer.valueOf(llegirLinia("Opcio: "));
55
56         switch (opcio) {
57             case 0: llistarIdCursos(doc, xPath);
58                     break;
59             case 1: llistarTutors(doc, xPath);
60                     break;
61             case 2: llistarTotalAlumnes(doc, xPath);
62                     break;
63             case 3: mostrarModuls(doc, xPath);
64                     break;
65             case 4: llistarAlumnes(doc, xPath);
66                     break;
67             case 5: afegirAlumne(doc, xPath);
68                     break;
69             case 6: eliminarAlumne(doc, xPath);
70                     break;
71             case 100: running = false;
72                     break;
73             default: System.out.println("\nOpcio fora del rang!");
74                     break;
75         }
76     } catch (Exception e) {
77         System.out.println("\nOpcio no numerica!");
78     }
79 }
80 }
81 } catch (Exception e) {
82     e.printStackTrace();
83 }
84 }
85
86 public static String llegirLinia (String text) {
87     System.out.print(text);
88     return in.nextLine();
89 }
90
91 public static void llistarIdCursos(Document doc, XPath xPath) {
92
93     try {
94
95         // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes
96
97         NodeList listCursos = (NodeList) xPath.compile("/cursos/curs/@id").evaluate(doc, XPathConstants.NODESET);
98
99         // Imprimeix l'id de tots els cursos
100
101         System.out.println("");
102
103         for (int i = 0; i < listCursos.getLength(); i++) {
104             System.out.println("Curs " + (i+1) + ": " + listCursos.item(i).getNodeValue());
105         }
106     } catch (XPathExpressionException e) {
107         e.printStackTrace();
108     }
109 }
110
111
112 public static void llistarTutors(Document doc, XPath xPath) {
113
114     try {
115
116         // Avaluem una expressió XPath que selecciona els tutors i obtenim una llista de nodes
117
118         NodeList listTutors = (NodeList) xPath.compile("/cursos/curs/tutor").evaluate(doc, XPathConstants.NODESET);
119
120         // Imprimeix els tutors
121
122         System.out.println("");
123
124         for (int i = 0; i < listTutors.getLength(); i++) {
125             System.out.println("Tutor " + (i+1) + ": " + listTutors.item(i).getTextContent());
126         }
127     } catch (XPathExpressionException e) {
128         e.printStackTrace();
129     }
130 }
131
132
133 public static void llistarTotalAlumnes(Document doc, XPath xPath) {
134
135     try {
136
137         // Avaluem una expressió XPath que selecciona els alumnes i obtenim una llista de nodes
138
139         NodeList listAlumnes = (NodeList) xPath.compile("/cursos/curs/alumnes/alumne").evaluate(doc, XPathConstants.NODESET);
140
141         // Imprimeix el total d'alumnes
142
143         System.out.println("\nAlumnes totals: " + listAlumnes.getLength());
144     } catch (XPathExpressionException e) {
145         e.printStackTrace();
146     }
147 }
148
149

```

```

150 public static void mostrarModuls(Document doc, XPath xPath) {
151
152     try {
153
154         Scanner sc = new Scanner(System.in);
155         boolean idValid = false;
156         String idCurs = "";
157
158         while (!idValid) {
159
160             // Demanem l'id d'un curs
161
162             System.out.print("\nEscriu l'id d'un curs: ");
163             idCurs = sc.next();
164
165             // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes
166
167             NodeList listCursos = (NodeList) xPath.compile("/cursos/curs[@id]").evaluate(doc, XPathConstants.NODESET);
168
169             // Comprovem si l'id introduït existeix dins del fitxer XML
170
171             for (int i = 0; i < listCursos.getLength(); i++) {
172                 if (idCurs.equals(listCursos.item(i).getNodeValue())) {
173                     idValid = true;
174                 }
175             }
176
177             if (!idValid) {
178                 System.out.println("El curs amb id " + idCurs + " no existeix!");
179             }
180         }
181
182         // Avaluem una expressió XPath que selecciona l'id dels mòduls i obtenim una llista de nodes
183
184         NodeList listIdModuls = (NodeList) xPath.compile("/cursos/curs[@id='" + idCurs + "']/modules/modul[@id]").evaluate(doc, XPathConstants.NODESET);
185
186         // Avaluem una expressió XPath que selecciona el títol dels mòduls i obtenim una llista de nodes
187
188         NodeList listTitolModuls = (NodeList) xPath.compile("/cursos/curs[@id='" + idCurs + "']/modules/modul/titol").evaluate(doc, XPathConstants.NODESET);
189
190         // Imprimeix tant l'id com el títol dels mòduls
191
192         System.out.println("");
193
194         for (int i = 0; i < listIdModuls.getLength(); i++) {
195             System.out.println("Mòdul " + (i+1) + ": [Id = " + listIdModuls.item(i).getNodeValue() + ", Títol = " + listTitolModuls.item(i).getTextContent() + "]");
196         }
197
198     } catch (XPathExpressionException e) {
199         e.printStackTrace();
200     }
201 }
202
203 public static void llistarAlumnes(Document doc, XPath xPath) {
204
205     try {
206
207         Scanner sc = new Scanner(System.in);
208         boolean idValid = false;
209         String idCurs = "";
210
211         while (!idValid) {
212
213             // Demanem l'id d'un curs
214
215             System.out.print("\nEscriu l'id d'un curs: ");
216             idCurs = sc.next();
217
218             // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes
219
220             NodeList listCursos = (NodeList) xPath.compile("/cursos/curs[@id]").evaluate(doc, XPathConstants.NODESET);
221
222             // Comprovem si l'id introduït existeix dins del fitxer XML
223
224             for (int i = 0; i < listCursos.getLength(); i++) {
225                 if (idCurs.equals(listCursos.item(i).getNodeValue())) {
226                     idValid = true;
227                 }
228             }
229
230             if (!idValid) {
231                 System.out.println("El curs amb id " + idCurs + " no existeix!");
232             }
233         }
234
235         // Avaluem una expressió XPath que selecciona els alumnes i obtenim una llista de nodes
236
237         NodeList listAlumnes = (NodeList) xPath.compile("/cursos/curs[@id='" + idCurs + "']/alumnes/alumne").evaluate(doc, XPathConstants.NODESET);
238
239         // Imprimeix els alumnes
240
241         System.out.println("");
242
243         for (int i = 0; i < listAlumnes.getLength(); i++) {
244             System.out.println("Alumne " + (i+1) + ": " + listAlumnes.item(i).getTextContent());
245         }
246
247     } catch (XPathExpressionException e) {
248         e.printStackTrace();
249     }
250 }
251

```

```

252 public static void afegirAlumne(Document doc, XPath xPath) {
253
254     try {
255
256         Scanner sc = new Scanner(System.in);
257         boolean idValid = false;
258         String idCurs = "";
259
260         while (!idValid) {
261
262             // Demanem l'id d'un curs
263
264             System.out.print("\nEscriu l'id d'un curs: ");
265             idCurs = sc.next();
266
267             // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes
268
269             NodeList listCursos = (NodeList) xPath.compile("/cursos/curs[@id]").evaluate(doc, XPathConstants.NODESET);
270
271             // Comprovem si l'id introduït existeix dins del fitxer XML
272
273             for (int i = 0; i < listCursos.getLength(); i++) {
274                 if (idCurs.equals(listCursos.item(i).getNodeValue())) {
275                     idValid = true;
276                 }
277             }
278
279             if (!idValid) {
280                 System.out.println("El curs amb id " + idCurs + " no existeix!");
281             }
282         }
283
284         // Demanem el nom d'un nou alumne
285
286         System.out.print("\nEscriu el nom del nou alumne: ");
287         String nomAlumne = sc.next();
288
289         // Avaluem una expressió XPath que selecciona els alumnes d'un curs
290
291         XPathExpression expression = xPath.compile("/cursos/curs[@id='" + idCurs + "']/alumnes");
292
293         // Obtenim un element de la expressió
294
295         Element elAlumne = (Element) expression.evaluate(doc, XPathConstants.NODE);
296
297         // Obtenim un element d'un alumne
298
299         Element elmNouAlumne = doc.createElement("alumne");
300
301         // Afegim a l'alumne el nom que s'ha demanat anteriorment
302
303         elmNouAlumne.setTextContent(nomAlumne);
304
305         // Inserim l'alumne nou dins dels alumnes
306
307         elAlumne.appendChild(elmNouAlumne);
308
309         // Crea un transformador XSLT
310
311         Transformer transformer = TransformerFactory.newInstance().newTransformer();
312
313         // Estableix la propietat OMIT_XML_DECLARATION a "no" per no ometre la declaració XML del document XML resultant
314
315         transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
316
317         // Estableix la propietat INDENT a "yes" per indentar el document XML resultant
318
319         transformer.setOutputProperty(OutputKeys.INDENT, "yes");
320
321         // Crea una instància de DOMSource a partir del document XML
322
323         DOMSource source = new DOMSource(doc);
324
325         // Crea una instància de StreamResult a partir del camí del fitxer XML
326
327         StreamResult result = new StreamResult(new FileOutputStream(new File(System.getProperty("user.dir") + "/data/cursos.xml")));
328
329         // Transforma el document XML especificat per source i escriu el document XML resultant a l'objecte especificat per result
330
331         transformer.transform(source, result);
332
333     } catch (XPathExpressionException e) {
334         e.printStackTrace();
335     }
336     } catch (FileNotFoundException e) {
337         e.printStackTrace();
338     }
339     } catch (TransformerException e) {
340         e.printStackTrace();
341     }
342 }
343

```



```

344 public static void eliminarAlumne(Document doc, XPath xPath) {
345
346     try {
347
348         Scanner sc = new Scanner(System.in);
349         boolean idValid = false;
350         String idCurs = "";
351
352         while (!idValid) {
353
354             // Demanem l'id d'un curs
355
356             System.out.print("\nEscriu l'id d'un curs: ");
357             idCurs = sc.next();
358
359             // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes
360
361             NodeList listCursos = (NodeList) xPath.compile("/cursos/curs[@id]").evaluate(doc, XPathConstants.NODESET);
362
363             // Comprovem si l'id introduït existeix dins del fitxer XML
364
365             for (int i = 0; i < listCursos.getLength(); i++) {
366                 if (idCurs.equals(listCursos.item(i).getNodeValue())) {
367                     idValid = true;
368                 }
369             }
370
371             if (!idValid) {
372                 System.out.println("El curs amb id " + idCurs + " no existeix!");
373             }
374         }
375
376         // Demanem el nom d'un nou alumne
377
378         System.out.print("\nEscriu el nom del nou alumne: ");
379         String nomAlumne = sc.next();
380
381         // Avaluem una expressió XPath que selecciona els alumnes d'un curs
382
383         XPathExpression expression = xPath.compile("/cursos/curs[@id='" + idCurs + "']/alumnes/alumne[text()=' " + nomAlumne + "']");
384
385         // Obtenim un node de la expressió
386
387         Node nodeAlumne = (Node) expression.evaluate(doc, XPathConstants.NODE);
388
389         // Eliminem l'alumne seleccionat
390
391         if (nodeAlumne != null) {
392
393             nodeAlumne.getParentNode().removeChild(nodeAlumne);
394
395             // Crea un transformador XSLT
396
397             Transformer transformer = TransformerFactory.newInstance().newTransformer();
398
399             // Estableix la propietat OMIT_XML_DECLARATION a "no" per no ometre la declaració XML del document XML resultant
400
401             transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
402
403             // Estableix la propietat INDENT a "yes" per indentar el document XML resultant
404
405             transformer.setOutputProperty(OutputKeys.INDENT, "yes");
406
407             // Crea una instància de DOMSource a partir del document XML
408
409             DOMSource source = new DOMSource(doc);
410
411             // Crea una instància de StreamResult a partir del camí del fitxer XML
412
413             StreamResult result = new StreamResult(new FileOutputStream(new File(System.getProperty("user.dir") + "/data/cursos.xml")));
414
415             // Transforma el document XML especificat per source i escriu el document XML resultant a l'objecte especificat per result
416
417             transformer.transform(source, result);
418         }
419
420     } catch (XPathExpressionException e) {
421         e.printStackTrace();
422     } catch (FileNotFoundException e) {
423         e.printStackTrace();
424     } catch (TransformerException e) {
425         e.printStackTrace();
426     }
427 }
428 }

```