Objectius: - Conèixer com llegir i escriure objectes XML amb Java
Instruccions: - Responeu 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: - Cada exercici té la mateixa puntuació
Entrega: - Un arxiu .zip anomenat: **PRx.y-NomCognom.zip** ● 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: ● 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 convenients

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)

```java
import java.io.IOException;
import java.util.*;

public class Main {

    static Scanner in = new Scanner(System.in); // System.in és global

    public static void main(String[] args) throws InterruptedException, IOException {

        boolean running = true;

        while (running) {

            System.out.println("Escull una opcio:\n\n0) PR140Main\n1) PR141Main\n2) PR142Main\n100) Sortir\n");

            try {

                int opcio = Integer.valueOf(llegirLinia("Opcio: "));

                switch (opcio) {
                    case 0: PR140Main.main(args);
                            break;
                    case 1: PR141Main.main(args);
                            break;
                    case 2: PR142Main.main(args);
                            break;
                    case 100: running = false;
                            break;
                    default: System.out.println("\nOpcio fora del rang!");
                            break;
                }

                System.out.println("\n************************************************************\n");

            } catch (Exception e) {
                System.out.println("\nOpcio no numerica!\n\n************************************************************\n");
            }
        }
    }

    public static String llegirLinia (String text) {
        System.out.print(text);
        return in.nextLine();
    }
}
```

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.

```java
import java.io.File;

import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class PR140Main {

    public static void main(String[] args) {

        try {

            // Crear la carpeta 'data' si no existeix

            File dir = new File("data");

            if (!dir.exists()) {
                if (!dir.mkdirs()) {
                    System.out.println("Error en la creacio de la carpeta 'data'");
                }
            }

            // Analitza el fitxer XML

            Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(new File(System.getProperty("user.dir") + "/data/persones.xml"));

            // Obté una llista de tots els elements persona "persona" del document

            NodeList listPersones = doc.getElementsByTagName("persona");

            // Crea la part superior de la taula

            System.out.println("");
            System.out.println(String.format("%-20s %-20s %-10s %-20s\n", "Nom", "Cognom", "Edat", "Ciutat"));
            System.out.println("-----------------------------------------------------------------\n");

            // Bucle for per recòrrer la llista de persones

            for (int i = 0; i < listPersones.getLength(); i++) {
                // Obté la persona actual
                Node nodePersona = listPersones.item(i);
                // Comprova si la persona actual és un element
                if (nodePersona.getNodeType() == Node.ELEMENT_NODE) {
                    // Converteix l'element actual en una persona
                    Element elm = (Element) nodePersona;
                    // **Obté el nom de la persona**
                    String nom = elm.getElementsByTagName("nom").item(0).getTextContent();
                    // **Obté el cognom de la persona**
                    String cognom = elm.getElementsByTagName("cognom").item(0).getTextContent();
                    // **Obté l'edat de la persona**
                    String edat = elm.getElementsByTagName("edat").item(0).getTextContent();
                    // **Obté el nom de la ciutat on viu la persona**
                    String ciutat = elm.getElementsByTagName("ciutat").item(0).getTextContent();
                    // Dona format a les dades obtingudes i les imprimeix per pantalla
                    System.out.println(String.format("%-20s %-20s %-10s %-20s\n", nom, cognom, edat, ciutat));
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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”.

```java
public class PR141Main {

    public static void main(String[] args) {

        try {

            // Crear la carpeta 'data' si no existeix

            File dir = new File("data");

            if (!dir.exists()) {
                if (!dir.mkdirs()) {
                    System.out.println("Error en la creacio de la carpeta 'data'");
                }
            }

            // Crea un nou document XML

            Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();

            // Crea l'element root del document XML

            Element elmRoot = doc.createElement("biblioteca");

            // Afegeix l'element root al document XML

            doc.appendChild(elmRoot);

            // Crea l'element "llibre", li afegeix l'atribut "id" i afegeix aquest element a l'element root

            Element elmLlibre = doc.createElement("llibre");
            elmLlibre.setAttribute("id", "001");
            elmRoot.appendChild(elmLlibre);

            // Crea l'element "titol" i l'afegeix a l'element "llibre"

            Element elmTitol = doc.createElement("titol");
            elmTitol.appendChild(doc.createTextNode("El viatge dels venturons"));
            elmLlibre.appendChild(elmTitol);

            // Crea l'element "autor" i l'afegeix a l'element "llibre"

            Element elmAutor = doc.createElement("autor");
            elmAutor.appendChild(doc.createTextNode("Joan Pla"));
            elmLlibre.appendChild(elmAutor);

            // Crea l'element "anyPublicacio" i l'afegeix a l'element "llibre"

            Element elmAnyPublicacio = doc.createElement("anyPublicacio");
            elmAnyPublicacio.appendChild(doc.createTextNode("1998"));
            elmLlibre.appendChild(elmAnyPublicacio);

            // Crea l'element "editorial" i l'afegeix a l'element "llibre"

            Element elmEditorial = doc.createElement("editorial");
            elmEditorial.appendChild(doc.createTextNode("Edicions Mar"));
            elmLlibre.appendChild(elmEditorial);

            // Crea l'element "genere" i l'afegeix a l'element "llibre"

            Element elmGenere = doc.createElement("genere");
            elmGenere.appendChild(doc.createTextNode("Aventura"));
            elmLlibre.appendChild(elmGenere);

            // Crea l'element "pagines" i l'afegeix a l'element "llibre"

            Element elmPagines = doc.createElement("pagines");
            elmPagines.appendChild(doc.createTextNode("320"));
            elmLlibre.appendChild(elmPagines);

            // Crea l'element "disponible" i l'afegeix a l'element "llibre"

            Element elmDisponible = doc.createElement("disponible");
            elmDisponible.appendChild(doc.createTextNode("true"));
            elmLlibre.appendChild(elmDisponible);

            // Crea un transformador XSLT

            Transformer transformer = TransformerFactory.newInstance().newTransformer();

            // Estableix la propietat OMIT_XML_DECLARATION a "no" per no ometre la declaració XML del document XML resultant

            transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");

            // Estableix la propietat INDENT a "yes" per indentar el document XML resultant

            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            // Crea una instància de DOMSource a partir del document XML

            DOMSource source = new DOMSource(doc);

            // Crea una instància de StreamResult a partir del camí del fitxer XML

            StreamResult result = new StreamResult(new FileOutputStream(new File(System.getProperty("user.dir") + "/data/biblioteca.xml")));

            // Transforma el document XML especificat per source i escriu el document XML resultant a l'objecte especificat per result

            transformer.transform(source, result);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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 opcio:\n\n0) Llistar ids de cursos" +
49                                    "\n1) Llistar tutors\n2) Llistar total d'alumnes\n3) Mostrar ids i titols dels moduls\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
```

```java

            try {

                int opcio = Integer.valueOf(llegirLinia("Opcio: "));

                switch (opcio) {
                    case 0: llistarIdCursos(doc, xPath);
                            break;
                    case 1: llistarTutors(doc, xPath);
                            break;
                    case 2: llistarTotalAlumnes(doc, xPath);
                            break;
                    case 3: mostrarModuls(doc, xPath);
                            break;
                    case 4: llistarAlumnes(doc, xPath);
                            break;
                    case 5: afegirAlumne(doc, xPath);
                            break;
                    case 6: eliminarAlumne(doc, xPath);
                            break;
                    case 100: running = false;
                            break;
                    default: System.out.println("\nOpcio fora del rang!");
                            break;
                }

            } catch (Exception e) {
                System.out.println("\nOpcio no numerica!");
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static String llegirLinia (String text) {
    System.out.print(text);
    return in.nextLine();
}

public static void llistarIdCursos(Document doc, XPath xPath) {

    try {

        // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes

        NodeList listCursos = (NodeList) xPath.compile("/cursos/curs/@id").evaluate(doc, XPathConstants.NODESET);

        // Imprimeix l'id de tots els cursos

        System.out.println("");

        for (int i = 0; i < listCursos.getLength(); i++) {
            System.out.println("Curs " + (i+1) + ": " + listCursos.item(i).getNodeValue());
        }

    } catch (XPathExpressionException e) {
        e.printStackTrace();
    }
}

public static void llistarTutors(Document doc, XPath xPath) {

    try {

        // Avaluem una expressió XPath que selecciona els tutors i obtenim una llista de nodes

        NodeList listTutors = (NodeList) xPath.compile("/cursos/curs/tutor").evaluate(doc, XPathConstants.NODESET);

        // Imprimeix els tutors

        System.out.println("");

        for (int i = 0; i < listTutors.getLength(); i++) {
            System.out.println("Tutor " + (i+1) + ": " + listTutors.item(i).getTextContent());
        }

    } catch (XPathExpressionException e) {
        e.printStackTrace();
    }
}

public static void llistarTotalAlumnes(Document doc, XPath xPath) {

    try {

        // Avaluem una expressió XPath que selecciona els alumnes i obtenim una llista de nodes

        NodeList listAlumnes = (NodeList) xPath.compile("/cursos/curs/alumnes/alumne").evaluate(doc, XPathConstants.NODESET);

        // Imprimeix el total d'alumnes

        System.out.println("\nAlumnes totals: " + listAlumnes.getLength());

    } catch (XPathExpressionException e) {
        e.printStackTrace();
    }
}

```

```java
public static void mostrarModuls(Document doc, XPath xPath) {

    try {

        Scanner sc = new Scanner(System.in);
        boolean idValid = false;
        String idCurs = "";

        while (!idValid) {

            // Demanem l'id d'un curs

            System.out.print("\nEscriu l'id d'un curs: ");
            idCurs = sc.next();

            // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes

            NodeList listCursos = (NodeList) xPath.compile("/cursos/curs/@id").evaluate(doc, XPathConstants.NODESET);

            // Comprovem si l'id introduït existeix dins del fitxer XML

            for (int i = 0; i < listCursos.getLength(); i++) {
                if (idCurs.equals(listCursos.item(i).getNodeValue())) {
                    idValid = true;
                }
            }

            if (!idValid) {
                System.out.println("El curs amb id " + idCurs + " no existeix!");
            }
        }

        // Avaluem una expressió XPath que selecciona l'id dels mòduls i obtenim una llista de nodes

        NodeList listIdModuls = (NodeList) xPath.compile("/cursos/curs[@id='" + idCurs + "']/moduls/modul/@id").evaluate(doc, XPathConstants.NODESET);

        // Avaluem una expressió XPath que selecciona el títol dels mòduls i obtenim una llista de nodes

        NodeList listTitolModuls = (NodeList) xPath.compile("/cursos/curs[@id='" + idCurs + "']/moduls/modul/titol").evaluate(doc, XPathConstants.NODE

        // Imprimeix tant l'id com el títol dels mòduls

        System.out.println("");

        for (int i = 0; i < listIdModuls.getLength(); i++) {
            System.out.println("Modul " + (i+1) + ": [Id = " + listIdModuls.item(i).getNodeValue() + ", Titol = " + listTitolModuls.item(i).getTextCon
        }

    } catch (XPathExpressionException e) {
        e.printStackTrace();
    }
}

public static void llistarAlumnes(Document doc, XPath xPath) {

    try {

        Scanner sc = new Scanner(System.in);
        boolean idValid = false;
        String idCurs = "";

        while (!idValid) {

            // Demanem l'id d'un curs

            System.out.print("\nEscriu l'id d'un curs: ");
            idCurs = sc.next();

            // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes

            NodeList listCursos = (NodeList) xPath.compile("/cursos/curs/@id").evaluate(doc, XPathConstants.NODESET);

            // Comprovem si l'id introduït existeix dins del fitxer XML

            for (int i = 0; i < listCursos.getLength(); i++) {
                if (idCurs.equals(listCursos.item(i).getNodeValue())) {
                    idValid = true;
                }
            }

            if (!idValid) {
                System.out.println("El curs amb id " + idCurs + " no existeix!");
            }
        }

        // Avaluem una expressió XPath que selecciona els alumnes i obtenim una llista de nodes

        NodeList listAlumnes = (NodeList) xPath.compile("/cursos/curs[@id='" + idCurs + "']/alumnes/alumne").evaluate(doc, XPathConstants.NODESET);

        // Imprimeix els alumnes

        System.out.println("");

        for (int i = 0; i < listAlumnes.getLength(); i++) {
            System.out.println("Alumne " + (i+1) + ": " + listAlumnes.item(i).getTextContent());
        }

    } catch (XPathExpressionException e) {
        e.printStackTrace();
    }
}
```

```java
public static void afegirAlumne(Document doc, XPath xPath) {

    try {

        Scanner sc = new Scanner(System.in);
        boolean idValid = false;
        String idCurs = "";

        while (!idValid) {

            // Demanem l'id d'un curs

            System.out.print("\nEscriu l'id d'un curs: ");
            idCurs = sc.next();

            // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes

            NodeList listCursos = (NodeList) xPath.compile("/cursos/curs/@id").evaluate(doc, XPathConstants.NODESET);

            // Comprovem si l'id introduït existeix dins del fitxer XML

            for (int i = 0; i < listCursos.getLength(); i++) {
                if (idCurs.equals(listCursos.item(i).getNodeValue())) {
                    idValid = true;
                }
            }

            if (!idValid) {
                System.out.println("El curs amb id " + idCurs + " no existeix!");
            }
        }

        // Demanem el nom d'un nou alumne

        System.out.print("\nEscriu el nom del nou alumne: ");
        String nomAlumne = sc.next();

        // Avaluem una expressió XPath que selecciona els alumnes d'un curs

        XPathExpression expression = xPath.compile("/cursos/curs[@id='" + idCurs + "']/alumnes");

        // Obtenim un element de la expressió

        Element elmAlumne = (Element) expression.evaluate(doc,  XPathConstants.NODE);

        // Obtenim un element d'un alumne

        Element elmNouAlumne = doc.createElement("alumne");

        // Afegim a l'alumne el nom que s'ha demanat anteriorment

        elmNouAlumne.setTextContent(nomAlumne);

        // Inserim l'alumne nou dins dels alumnes

        elmAlumne.appendChild(elmNouAlumne);

        // Crea un transformador XSLT

        Transformer transformer = TransformerFactory.newInstance().newTransformer();

        // Estableix la propietat OMIT_XML_DECLARATION a "no" per no ometre la declaració XML del document XML resultant

        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");

        // Estableix la propietat INDENT a "yes" per indentar el document XML resultant

        transformer.setOutputProperty(OutputKeys.INDENT, "yes");

        // Crea una instància de DOMSource a partir del document XML

        DOMSource source = new DOMSource(doc);

        // Crea una instància de StreamResult a partir del camí del fitxer XML

        StreamResult result = new StreamResult(new FileOutputStream(new File(System.getProperty("user.dir") + "/data/cursos.xml")));

        // Transforma el document XML especificat per source i escriu el document XML resultant a l'objecte especificat per result

        transformer.transform(source, result);

    } catch (XPathExpressionException e) {
        e.printStackTrace();

    } catch (FileNotFoundException e) {
        e.printStackTrace();

    } catch (TransformerException e) {
        e.printStackTrace();
    }
}
```

```java
    public static void eliminarAlumne(Document doc, XPath xPath) {

        try {

            Scanner sc = new Scanner(System.in);
            boolean idValid = false;
            String idCurs = "";

            while (!idValid) {

                // Demanem l'id d'un curs

                System.out.print("\nEscriu l'id d'un curs: ");
                idCurs = sc.next();

                // Avaluem una expressió XPath que selecciona l'id de tots els cursos i obtenim una llista de nodes

                NodeList listCursos = (NodeList) xPath.compile("/cursos/curs/@id").evaluate(doc, XPathConstants.NODESET);

                // Comprovem si l'id introduït existeix dins del fitxer XML

                for (int i = 0; i < listCursos.getLength(); i++) {
                    if (idCurs.equals(listCursos.item(i).getNodeValue())) {
                        idValid = true;
                    }
                }

                if (!idValid) {
                    System.out.println("El curs amb id " + idCurs + " no existeix!");
                }
            }

            // Demanem el nom d'un nou alumne

            System.out.print("\nEscriu el nom del nou alumne: ");
            String nomAlumne = sc.next();

            // Avaluem una expressió XPath que selecciona els alumnes d'un curs

            XPathExpression expression = xPath.compile("/cursos/curs[@id='" + idCurs + "']/alumnes/alumne[text()='" + nomAlumne + "']");

            // Obtenim un node de la expressió

            Node nodeAlumne = (Node) expression.evaluate(doc, XPathConstants.NODE);

            // Eliminem l'alumne sel·leccionat

            if (nodeAlumne != null) {

                nodeAlumne.getParentNode().removeChild(nodeAlumne);

                // Crea un transformador XSLT

                Transformer transformer = TransformerFactory.newInstance().newTransformer();

                // Estableix la propietat OMIT_XML_DECLARATION a "no" per no ometre la declaració XML del document XML resultant

                transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");

                // Estableix la propietat INDENT a "yes" per indentar el document XML resultant

                transformer.setOutputProperty(OutputKeys.INDENT, "yes");

                // Crea una instància de DOMSource a partir del document XML

                DOMSource source = new DOMSource(doc);

                // Crea una instància de StreamResult a partir del camí del fitxer XML

                StreamResult result = new StreamResult(new FileOutputStream(new File(System.getProperty("user.dir") + "/data/cursos.xml")));

                // Transforma el document XML especificat per source i escriu el document XML resultant a l'objecte especificat per result

                transformer.transform(source, result);
            }

        } catch (XPathExpressionException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (TransformerException e) {
            e.printStackTrace();
        }
    }
}
```