Ficha de Trabalho nº 6
JDOM: Criar e Manipular XML

1. Bibliografia

http://www.jdom.org/docs/apidocs/index.html

2. Introdução

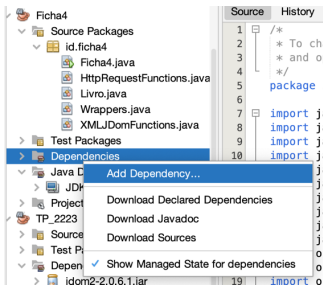
Nesta ficha de trabalho pretende-se que os alunos explorem a API JDOM para manipulação de ficheiros XML. Os ficheiros disponibilizados no Moodle para a realização desta ficha de trabalho são:

- **`XMLJDomFunctions.java`**
  - demonstração de algumas funções JDOM.
  - Grave este ficheiro para a pasta dos ficheiros *src* do projeto da aula anterior e corrija o nome da *package* (linha 5)
- **`JDOM2`**
  - para adicionar às *Libraries* do projeto Netbeans como indicado de seguida
  -

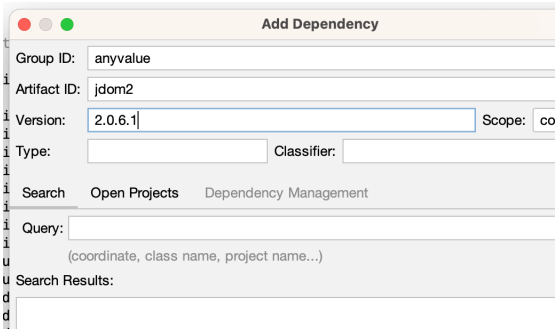
3. Adicionar JDOM ao projeto

Para acrescentar o JDOM ao Projeto da última aula siga os seguintes passos:

- Descarregue do Moodle o ficheiro ZIP e descompacte-o para uma pasta à sua escolha.
- No projeto da aula anterior aceda à pasta *Dependencies* (veja no Project Explorer como se indica na figura):



- Preencha os campos da janela como indicado abaixo e finalize clicando em ADD:



4. Utilização da API – funções do JDOM

As funções disponibilizadas no ficheiro **`XMLJDomFunctions.java`** permitem executar as seguintes tarefas:

4.1 Ler um ficheiro XML

Ler um ficheiro XML para que possa ser pesquisado/transformado/alterado.
Função: **`public static Document lerDocumentoXML(String caminhoFicheiro)`**

4.2 Gravar um documento XML para disco

Criar em disco um ficheiro XML usando o conteúdo de um documento XML em memória.
Função:

```
public static void escreverDocumentoParaFicheiro(Document doc, String caminhoFicheiro)
```

4.3 Ler um documento XML e criar uma String com o seu conteúdo

Coloca o conteúdo de um documento numa String.
Função:

```
public static String escreverDocumentoString(Document doc) {
```

4.4 Algumas funções do API JDOM

a) CRIAR UM ELEMENTO: `Element pai = new Element("pessoa");`

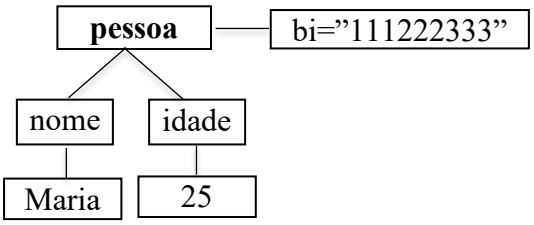
b) CRIAR UM ATRIBUTO E ASSOCIAR A UM ELEMENTO:

```
Attribute a = new Attribute ("bi","111222333");
pai.setAttribute(a);
```

c) ADICIONAR UM ELEMENTO FILHO AO ELEMENTO PAI:

```
Element filho = new Element("nome").addContent("Maria");
pai.addContent(filho);
filho = new Element("idade").addContent("25");
pai.addContent(filho);
```

as instruções acima criam a estrutura XML



d) REMOVER UM ELEMENTO: `pai.removeContent();` **//remove <u>todos</u> os filhos de pai**

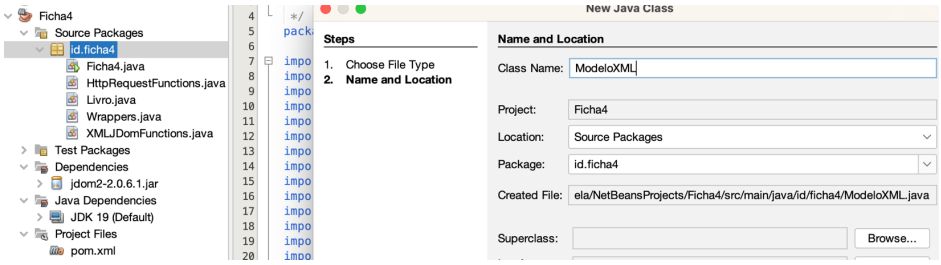
e) CRIAR DOCUMENTO E GRAVAR EM DISCO:

```
Document doc = new Document(pai); //usar o elemento raiz no construtor Document
escreverDocumentoParaFicheiro(doc,"pessoa.xml");
```

5. Exercícios

Usando o Projeto da aula anterior crie um novo ficheiro Java de nome **ModeloXML:**

- Com o botão direito em cima da package ***id.ficha4*** escolha **New-Java Class:**



5.1 No primeiro exercício pretende-se criar o seguinte ficheiro XML usando o API JDOM:

```
<catalogo>
	<livro isbn='1111' paginas='23'>
		<titulo>...</titulo>
		<autor>...</autor>
		<capa>...</capa>
		<editora>...</editora>
		<preco store = 'bertrand'>...</preco>
		<preco store = 'wook'>...</preco>
	</livro>
	<livro isbn='2222' paginas='68'>
		<titulo>...</titulo>
		<autor>...</autor>
		<capa>...</capa>
		<editora>...</editora>
		<preco store = 'bertrand'>...</preco>
		<preco store = 'wook'>...</preco>
	</livro>
	...
<catalogo>
```

a) No ficheiro ModeloXML implemente a função:

```
public static Document adicionaLivro (Livro liv, Document doc)
```

A função recebe uma instância da classe **Livro**, já com os campos preenchidos e o Document XML previamente inicializado. A função deve:

- Verificar se o ficheiro XML existe ou não.
- Se não existir, deve ser criado um novo **Document** com um elemento raiz <catalogo>.
- Se existir, deve ser obtido o elemento raiz usando o método **getRootElement**:

```
Element raiz;
if (doc == null) {
    raiz = new Element("catalogo"); //cria <catalogo>...</catalogo>
    doc = new Document(raiz);
} else {
    raiz = doc.getRootElement();
}
```

Com os métodos do API JDOM mostrados anteriormente no exemplo da pessoa, implemente o restante código que faça as seguintes tarefas:

- Criar um elemento **<livro>**
- Criar um atributo isbn (o valor do isbn é obtido com o getter *getIsnb()*)
- Criar um atributo paginas (o valor das paginas é obtido com o getter *getPaginas()*)
- Para o atributo **paginas** use a função **String x = Integer.toString(int x)** para converter um inte para String.
- Associe os atributos anteriores ao elemento <livro>
- Criar o elemento **<titulo>** e atribuir-lhe o conteúdo da variável liv (usar *addContent(liv.getTitulo()*)
- Repetir o passo anterior para os restantes campos
- Para os elementos **<preco>** use a função **String x = String.valueOf(double a)** para converter um double para String.
- Adicionar os filhos criados ao elemento <livro> (usar o método *addContent*)
- Adicionar o elemento livro à raiz (usar o método *addContent*)

b) Teste a função anterior no *main*

```
//Cria Livro
Livro liv = new Livro("1111", "Os Maias", "Eça de queiroz", "http://imagem", "Editora AAA", 110, 15.90, 13.50);
//Inicializa Doc XML
      Document doc = XMLJDomFunctions.lerDocumentoXML("livro.xml");
      //Chama a função para adicionar o livro ao XML
      doc = ModeloXML.adicionaLivro(liv, doc);
      //grava o ficheiro XML em disco
      XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "livro.xml");
```

Verifique se na pasta do projeto foi criado o ficheiro **livro.xml** de forma correta

c) Altere no *main.* Use as funções implementadas nas aulas anteriores:

- Wrappers: obter informação do site Bertrand:

```
String obtem_titulo(String isbn)
String obtem_autor(String isbn)
String obtem_capa(String isbn)
String obtem_editora(String isbn)
int obtem_paginas(String isbn)
double obtem_preco(String isbn)
double obtem_preco2(String isbn)
```

- Criar o objeto Livro:

```
Livro criaLivro(String isbn)
```

- Adicionar o livro ao XML:

```
Document adicionaLivro (Livro liv, Document doc)
```

Teste com os seguintes ISBN: 9789897224607,9789722129220,9789722532877,9789892314044,9789722533492

Verifique se na pasta do projeto o ficheiro **livro.xml** possui todos os livros inseridos

d) No ficheiro ModeloXML implemente a função

```
public static Document removeLivroAutor (String procura, Document doc)
```

esta função remove todos os elementos **<livro>** de um autor que contém a String dada como argumento (use o método *contains* da classe String). O Document XML previamente inicializado é também um argumento da função. A função devolve o Document actualizado.

- Verifique se o ficheiro XML existe ou não. Se não existir, deve ser enviado um aviso ao utilizador e sair da função. Se existir, deve ser obtido o elemento raiz usando o método getRootElement (variável raiz)

- Depois crie uma lista com todos os filhos <**livro**> do elemento **<catalogo>**:
  `List todosLivros = raiz.getChildren("livro");`

- Percorra a lista e <u>remova os livros</u> usando o método *removeContent*:

```
boolean found = false;
for(int i=0; i<todosLivros.size();i++){
      Element livro = (Element)todosLivros.get(i); //obtem livro i da Lista
      if (livro.getChild("autor").getText().contains(procura)){
          livro.getParent().removeContent(livro);
          System.out.println("Livro removido com sucesso!");
          found = true;
      }
  }
  if(!found){
      System.out.println("Autor " + procura + " não foi encontrado");
      return null;
    }
```

- Devolva o Document XML: `return doc;`

Teste a função no main:

```
public static void main(String[] args){
      ...
      //Inicializa Doc XML
      Document doc = XMLJDomFunctions.lerDocumentoXML("livro.xml");
      //Chama a função para remover  livros ao XML
      doc=ModeloXML.removeLivroAutor("Auster", doc);
      //grava o ficheiro XML em disco
      if(doc!=null)
          XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "livro.xml");
}
```

verifique o conteúdo do ficheiro **livro.xml** e verifique se livro do *Paul Auster* foi removido.
Tente remover um livro com um autor inexistente.

e) No ficheiro ModeloXML implemente a função

```
public static Document removeLivroISBN(String isbn, Document doc)
```

É semelhante à função anterior, mas remove um livro que tenha um ISBN igual ao enviado por argumento. O ISBN é um atributo do elemento Livro.

```
if (livro.getAttributeValue("isbn").equals(isbn)) {
   livro.getParent().removeContent(livro);
  System.out.println("Livro removido com sucesso!");
   found = true;
 }
```

f) No ficheiro ModeloXML implemente a função

```
public static Document alteraPrecoLivro (String isbn, double novoPreco, String loja,  Document doc)
```

esta função altera o valor do preço (da loja indicada – bertrand ou wook) de um livro cujo **isbn** é dado como argumento da função. O Document XML previamente inicializado é também um argumento da função. A função devolve o Document actualizado.
Implemente o seguinte código na função:

- Verifique se o ficheiro XML existe ou não.
  - Se não existir, deve ser enviado um aviso ao utilizador e sair da função.
  - Se existir, deve ser obtido o elemento raiz usando o método getRootElement (variável raiz)
- Crie uma lista com todos os filhos <livro> do elemento <catalogo>:

```
List todosLivros = raiz.getChildren("livro");
```

- Percorra a lista com um ciclo. Se encontrar o isbn dado como argumento:
  - Mostre o titulo do livro e o preço atual da loja indicada (use os métodos *getChild* e *getText* para ter acesso ao elemento <titulo> e <preco>)
  - Mostre o titulo do livro e o preço atual (se os métodos *getChild* e *getText* para ter acesso ao elemento <titulo> e <preco>)
  - Altere o preço do livro para o valor dado no argumento (use *getChild* e *setText* para alterar o valor do elemento <preco>)
- Se o isbn não foi encontrado escreve uma mensagem na consola e devolva **null**
- Caso contrario devolva o Document

Teste a função no main:

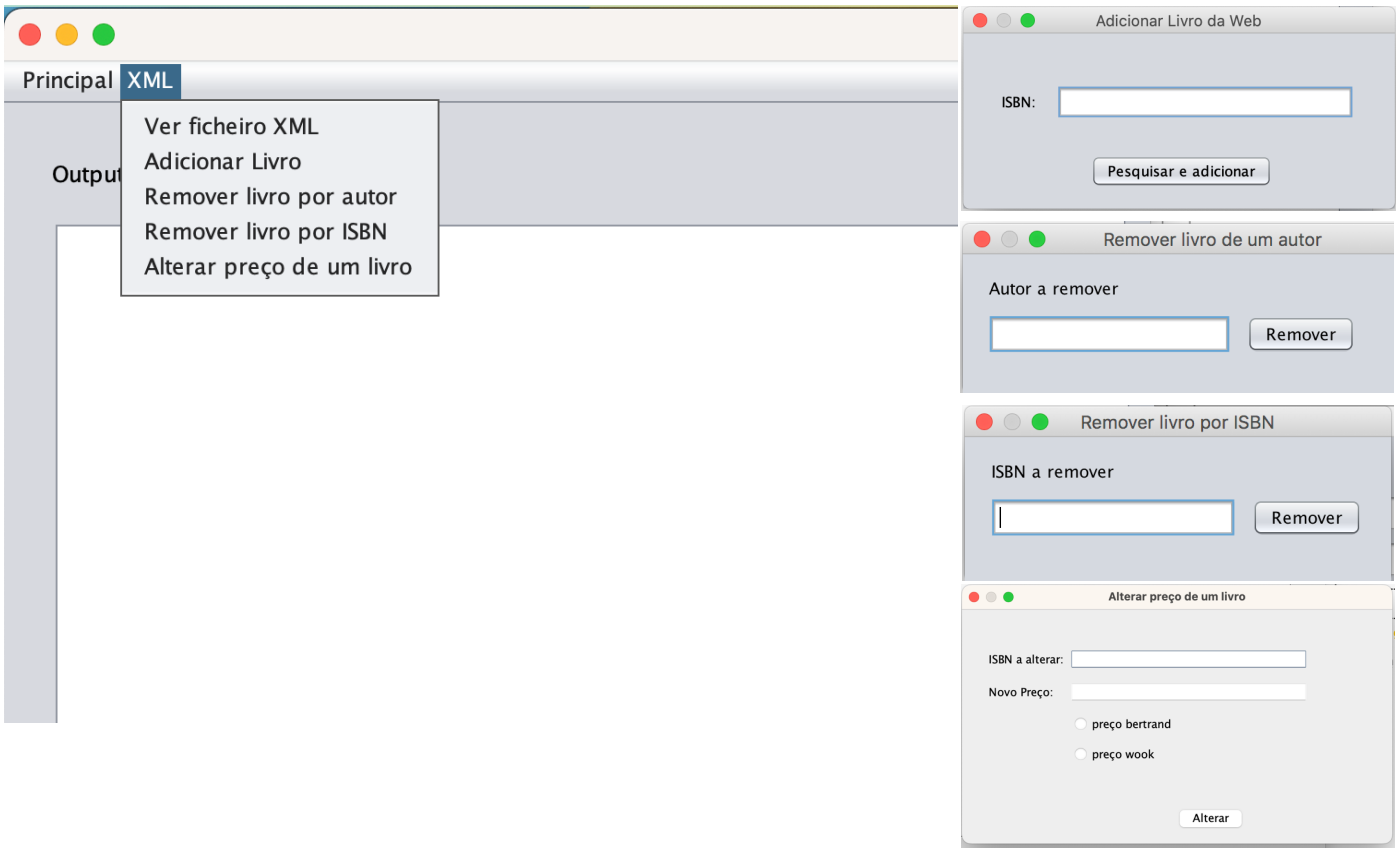
```
public static void main(String[] args){
        ...
        //Inicializa Doc XML
        Document doc = XMLJDomFunctions.lerDocumentoXML("livro.xml");
        //Chama a função para alterar o preço de um livro para 25 euros
        doc= ModeloXML.alteraPrecoLivro("9789722533492", 25.00, "wook", doc);
        //grava o ficheiro XML alterado em disco
        if(doc!=null)
            XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "livro.xml");
}
```

verifique o conteúdo do ficheiro **livro.xml** e veja se o preço do livro indicado foi alterado.

g) Crie um interface GUI simples para aceder às funções anteriores.

- Crie um **JFrame** contendo:
  - **MenuBar** com as opções :
    - Principal – Sair
    - XML com as e várias opções que vê na figura abaixo
  - **TextArea** para visualizar resultados

- Crie quatro **jDialog** de acordo com a figura abaixo. Os **jDialog** servirão para pedir os dados ao utilizador nas opções remover e adicionar livro, alterar preço.



Na opção **Ver ficheiro livro.xml** e sempre que queira visualizar na **textArea** o conteúdo de um ficheiro XML use o seguinte código:

```
Document doc = XMLJDomFunctions.lerDocumentoXML("livro.xml");
String texto = XMLJDomFunctions.escreverDocumentoString(doc);
jTextArea1.setText(texto);
```

Programe o código dos menus e botões usando as funções implementadas anteriormente.

Após cada operação, envie uma janela de informação (Ver Ficha 2). Por exemplo:

```
JOptionPane.showMessageDialog(this,
                              "Livro removido com sucesso",
                              "Informação",
                              JOptionPane.INFORMATION_MESSAGE);
```