Ficha de Trabalho nº 5
JDOM: Criar e Manipular XML

1. Bibliografia

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

2. Introdução

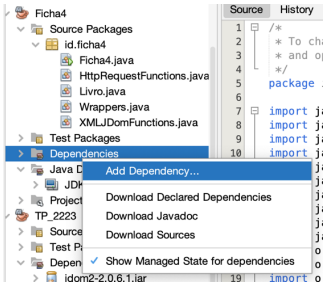
Nesta ficha de trabalho pretende-se que os alunos explorem a API JDOM para manipulação de ficheiros XML. Os ficheiros disponibilizados no Moodle para a realização desta ficha de trabalho são:

- **`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
  -

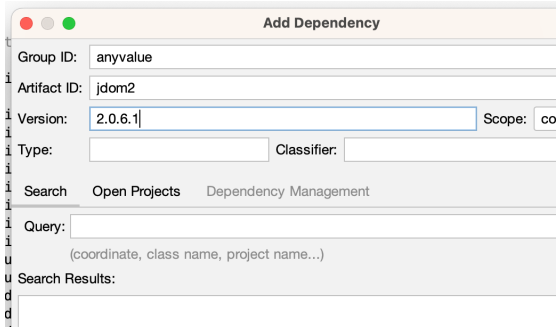
3. Adicionar JDOM ao projeto

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");
```

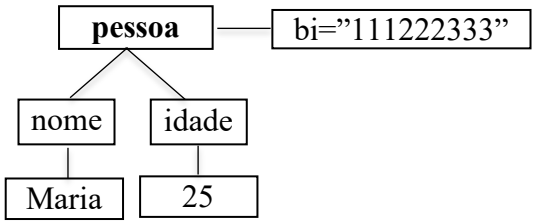
b) CRIAR UM ATRIBUTO E ASSOCIAR A UM ELEMENTO:

```
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**

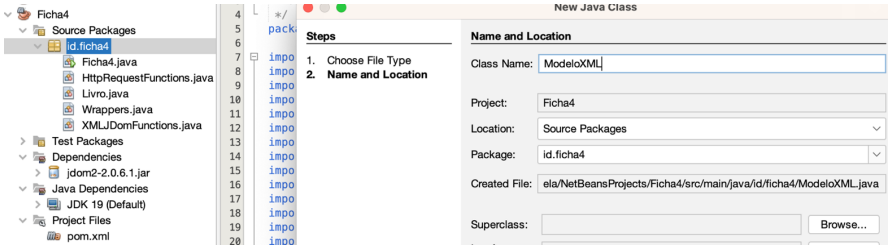
e) CRIAR DOCUMENTO E GRAVAR EM DISCO:

```
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:

```
<lista>
        <escritor nome="José Saramago">
          <nome_completo>José de Sousa Saramago</nome_completo>
          <nacionalidade>português</nacionalidade>
          <nascimento>16 de novembro de 1922</nascimento>
          <foto>https://pt.wikipedia.org/wiki/Ficheiro:JSJoseSaramago.jpg</foto>
          <ocupacoes>
               <ocupacao>serralheiro</ocupacao>
               <ocupacao>mecânico</ocupacao>
               <ocupacao>funcionário público</ocupacao>
               ...
          </ocupacoes>
      </escritor>
          ...
          ...
</lista>
```

a) No ficheiro ModeloXML implemente a função:

```
public static Document adicionaEscritor (Escritor escritor, Document doc)
```

A função recebe uma instância da classe **Escritor**, 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 <lista>.
- Se existir, deve ser obtido o elemento raiz usando o método **getRootElement**:

```
Element raiz;
if (doc == null) {
    raiz = new Element("lista"); //cria <lista>...</lista>
    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 **<escritor>**
- Criar um atributo nome (o valor do nome é obtido com o getter *getNome()*)
- Associe o atributo ao elemento <**escritor**>
- Criar o elemento **<nome_completo>** e atribuir-lhe o conteúdo da variável ***esc*** (usar *addContent(esc.getNome_completo()*)
- Repetir o passo anterior para os restantes campos
- Adicionar os filhos criados ao elemento <escritor> (usar o método *addContent*)
- Para o elemento **<ocupacoes>** iterar o ArrayList e acrescentar os vários elementos **<ocupacao>**
- Adicionar o elemento escritor à raiz (usar o método *addContent*)

b) Teste a função anterior no *main*. Use as funções implementadas nas aulas anteriores:

- Criar o objeto Escritor:

```
Escritor criaEscritor(String nome)
```

- Adicionar o escritor ao XML:

```
Document adicionaEscritor (Escritor esc, Document doc)
```

```
//Cria Escritor
Escritor esc = Wrappers.criaEscritor("Fernando Pessoa");
//Inicializa Doc XML
Document doc = XMLJDomFunctions.lerDocumentoXML("escritor.xml");
//Chama a função para adicionar o escritor ao XML
doc = ModeloXML.adicionaEscritor(esc, doc);
//grava o ficheiro XML em disco
XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "escritor.xml");
```

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

Adicione mais alguns escritores: José Saramago, Mário Cesariny, Paulo Coelho, António Lobo Antunes.

Verifique se na pasta do projeto o ficheiro **escritor.xml** possui todos os escritores inseridos

c) No ficheiro ModeloXML implemente a função

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

esta função remove o elemento **<escritor>** cujo atributo nome 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 <**escritor** > do elemento **<lista>**:

```
List todos = raiz.getChildren("escritor");
```

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

```
boolean found = false;
for(int i=0; i<todos.size();i++){
      Element esc = (Element)todos.get(i); //obtem escritor i da Lista
      if (esc.getAttributeValue("nome").contains(procura)) {
            esc.getParent().removeContent(esc);
           System.out.println("Escritor removido com sucesso!");
           found = true;
      }
 }
 if(!found){
      System.out.println("Escritor " + 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("escritor.xml");
//Chama a função para remover  escritor ao XML
doc=ModeloXML.removeEscritor("Saramago", doc);
//grava o ficheiro XML em disco
if(doc!=null)
    XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "escritor.xml");
}
```

verifique o conteúdo do ficheiro **escritor.xml** e verifique se o escritor José Saramago foi removido. Tente remover um escritor inexistente.

d) No ficheiro ModeloXML implemente a função

```
public static Document removeOcupacao(String escritor, String ocupacao, Document doc)
```

É semelhante à função anterior, mas ao encontrar o escritor, avança na lista das ocupações e remove a ocupação indicada caso ela exista

```
for (int i = 0; i < todos.size(); i++) {
    Element esc = (Element) todos.get(i); //obtem escritor i da Lista
    if (esc.getAttributeValue("nome").contains(escritor)) {
        Element ocup = (Element) esc.getChild("ocupacoes");
        List lista_oc = ocup.getChildren("ocupacao");
        for (int j = 0; j < lista_oc.size(); j++) {
            ..... // COMPLETAR
```

Teste a função no main:

```
public static void main(String[] args){
       ...
       //Inicializa Doc XML
       Document doc = XMLJDomFunctions.lerDocumentoXML("escritor.xml");
       //Chama a função para remover uma ocupação
       doc= ModeloXML.removeOcupacao("José Saramago", "contista", doc);
       //grava o ficheiro XML alterado em disco
       if(doc!=null)
           XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "escritor.xml");
}
```

verifique o conteúdo do ficheiro **escritor.xml** e veja se a ocupação do escritor indicado foi removida.

e) No ficheiro ModeloXML implemente a função

```
public static Document alteraNacionalidade (String nome, double novaNac,  Document doc)
```

esta função altera o valor da nacionalidade do escritor de @nome=**nome** (dado como argumento da função). O Document XML previamente inicializado é também um argumento da função. A função devolve o Document atualizado.
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 <escritor> do elemento <lista>:

```
List todos = raiz.getChildren("escritor");
```

- Percorra a lista com um ciclo. Se encontrar o nome dado como argumento:
  - Mostre o nome do escritor e a nacionalidade atual
  - Altere o a nacionalidade o valor dado no argumento (use *getChild* e *setText* para alterar o valor do elemento <nacionalidade>)

- Se o nome não foi encontrado escreve uma mensagem na consola e devolva **null**
- Caso contrário devolva o Document

Teste a função no main:

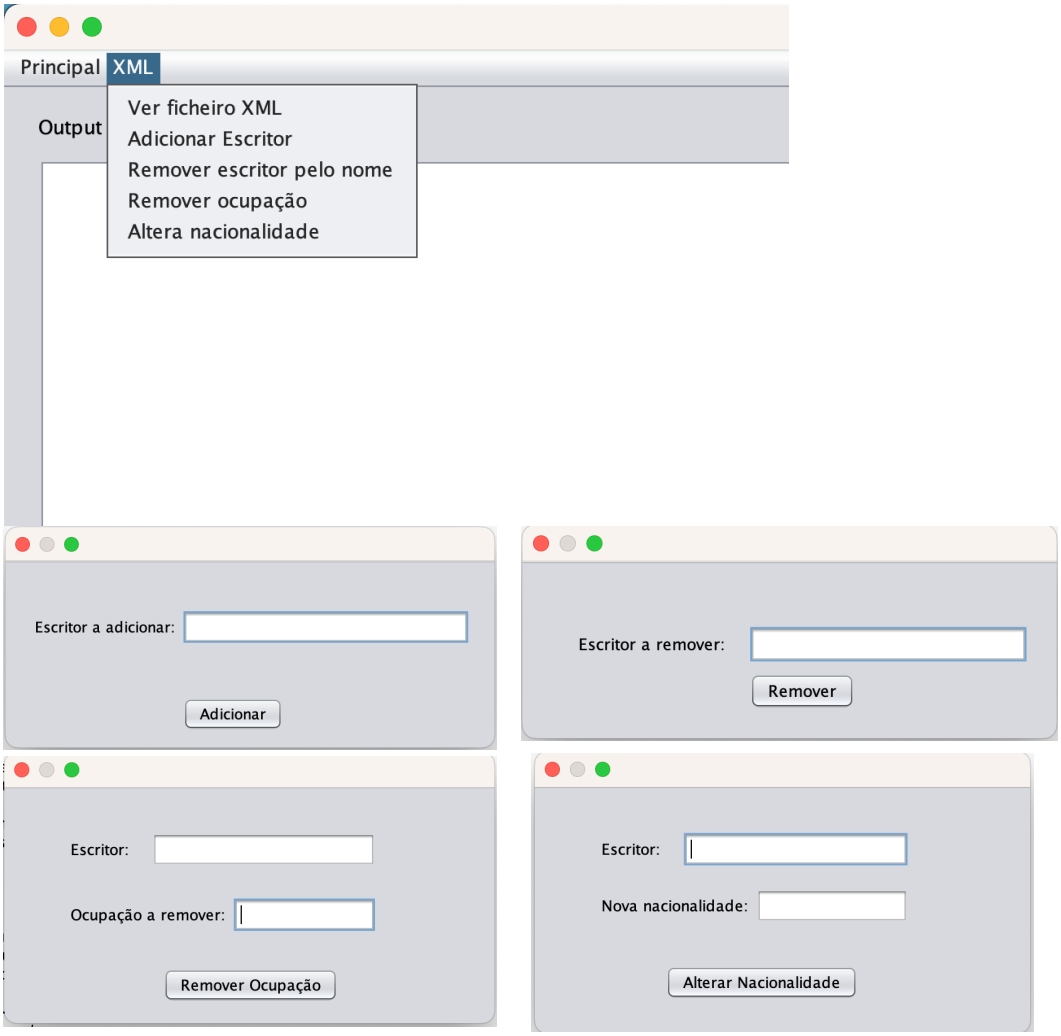
```
public static void main(String[] args){
        ...
        //Inicializa Doc XML
        Document doc = XMLJDomFunctions.lerDocumentoXML("escritor.xml");
        //Chama a função para alterar a nacionalidade para PORTUGUESA
        doc= ModeloXML.alterNacionalidade("José Saramago", "PORTUGUESA", doc);
        //grava o ficheiro XML alterado em disco
        if(doc!=null)
            XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "escritor.xml");
}
```

verifique o conteúdo do ficheiro **escritor.xml** e veja se a nacionalidade do escritor indicado foi alterada. Experimente com escritores que não existam no ficheiro, ou ocupações que não estejam presentes.

f) 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 escritor, remover ocupação e alterar nacionalidade.



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

```
Document doc = XMLJDomFunctions.lerDocumentoXML("escritor.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,
                              "Escritor removido com sucesso",
                              "Informação",
                              JOptionPane.INFORMATION_MESSAGE);
```