Đọc và ghi tập tin với Java

ThS. *Nguyễn Trường Sơn*

Nội dung trình bày

- Các hàm xử lý tập tin / thư mục
  - java.io.File
  - java.nio.Files
- Ghi và đọc dữ liệu từ tập tin bằng Stream
  - Characters Stream
  - Bytes Stream

Các hàm xử lý tập tin / thư mục

- import **java.io.File**;

- Một số hằng số:
  - File.separator
  - File.separatorChar
  - File.pathSeparator
  - File.pathSeparatorChar

- Một số hàm thông dụng chung:
  - File.createTempFile (String prefix, String suffix)
  - File.createTempFile(String prefix, String suffix, File dir)
  - File.listRoots()

Lưu ý: Ký tự phân tách đường dẫn (File.separator)trong

- HĐH Windows: \
- HĐH Linux, OS X: /

Các hàm quản lý tập tin / thư mục

- **File**
  - import **java.io.File**;
- Hàm khởi tạo:
  - File f = new File("/Users/ntson/Misc/abc.txt");
- Một số hàm kiểm tra
  - isFile ()
  - isDirectory ()
  - isHidden ()
  - canRead ()
  - canWrite ()
  - canExecute ()
  - exists ()

Các hàm quản lý tập tin / thư mục

- Một số hàm kiểm tra
  - getPath()
  - getAbsolutePath()
  - getAbsoluteFile()
  - getName()
  - getParent()
  - getParentFile()
  - listFiles()
    - → Lấy danh sách tất cả các tập tin / thư mục trong một thư mục
  - listFiles(FilenameFilter)
    - → Lấy danh sách tất cả các tập tin / thư mục trong một thư mục theo một filter (điều kiện lọc)

Ví du: Lấy thông tin của môt tập

```
1  package demojavaio;
2
3  import java.io.File;
4  public class Main {
5     public static void main(String[] args) {
6        File f = new File ("files/path.txt");
7        if (f.isDirectory())
8           System.out.println("Is folder !!!");
9        if (f.isFile())
10          System.out.println("Is file !!!");
11       if (f.exists())
12          System.out.println("Exists !!!");
13       else
14           System.out.println("Not exists !!!");
15       System.out.println("Absolute path: " + f.getAbsolutePath());
16       System.out.println("Get path: " + f.getPath());
17       System.out.println("Get name: " + f.getName());
18       System.out.println("getParent: " + f.getParent());
19    }
20 }
```

```
File f = new File ("files"+ File.separator +"path.txt");
```

Đoạn chương trình này có thể bị lỗi nếu chạy trên HĐT Windows

Ví dụ: Hiển thị danh sách tập tin của môt thư muc

```
1  package demojavaio;
2  import java.io.File;
3
4  public class DemoFolder {
5
6      public static void main(String[] args) {
7          File f = new File("path_of_folder");
8
9          if (f.isDirectory()) {
10             File[] list = f.listFiles();
11             for (File item : list) {
12                 System.out.println(item.getPath());
13             }
14         }
15     }
16 }
17
```

Ví du: Hiển thi danh sách tập tin

```
1  package demojavaio;
2  import java.io.File;
3  import java.io.FilenameFilter;
4  public class ListFolder {
5     public static void main(String[] args) {
6        File f = new File("path_of_folder");
7        FilenameFilter textFilter = new FilenameFilter() {
8           public boolean accept(File dir, String name) {
9              String lowercaseName = name.toLowerCase();
10             if (lowercaseName.endsWith(".txt")) {
11                return true;
12             } else {
13                return false;
14             }
15          }
16       };
17       if (f.isDirectory()) {
18          File[] list = f.listFiles(textFilter );
19          for (File item : list) {
20             System.out.println(item.getPath());
21          }
22       }
23    }
24 }
```

Khai báo đối tượng filter

Các hàm quản lý tập tin / thư mục

- Một số hàm khác:
  - Lấy thư mục hiện hành:
    - **System.getProperty("user.dir")**;

```
1  package demojavaio;
2
3  public class DemoFolder {
4
5     public static void main(String[] args) {
6        String curDir = System.getProperty("user.dir");
7              System.out.println(curDir);
8        }
9  }
16
17
```

Các hàm quản lý tập tin / thư mục

- **Files**
  - import **java.nio.File**; (NIO: New IO Java SE 7)
- Các hàm thông dụng:
  - static List<String> **readAllLines**(Path p, Charset cs)
    - Đọc tất cả các dòng của một tập tin → mảng các dòng (List)
  - static byte[] **readAllBytes**(Path path)
    - Đọc toàn bộ nội dung của tập tin → mảng các byte

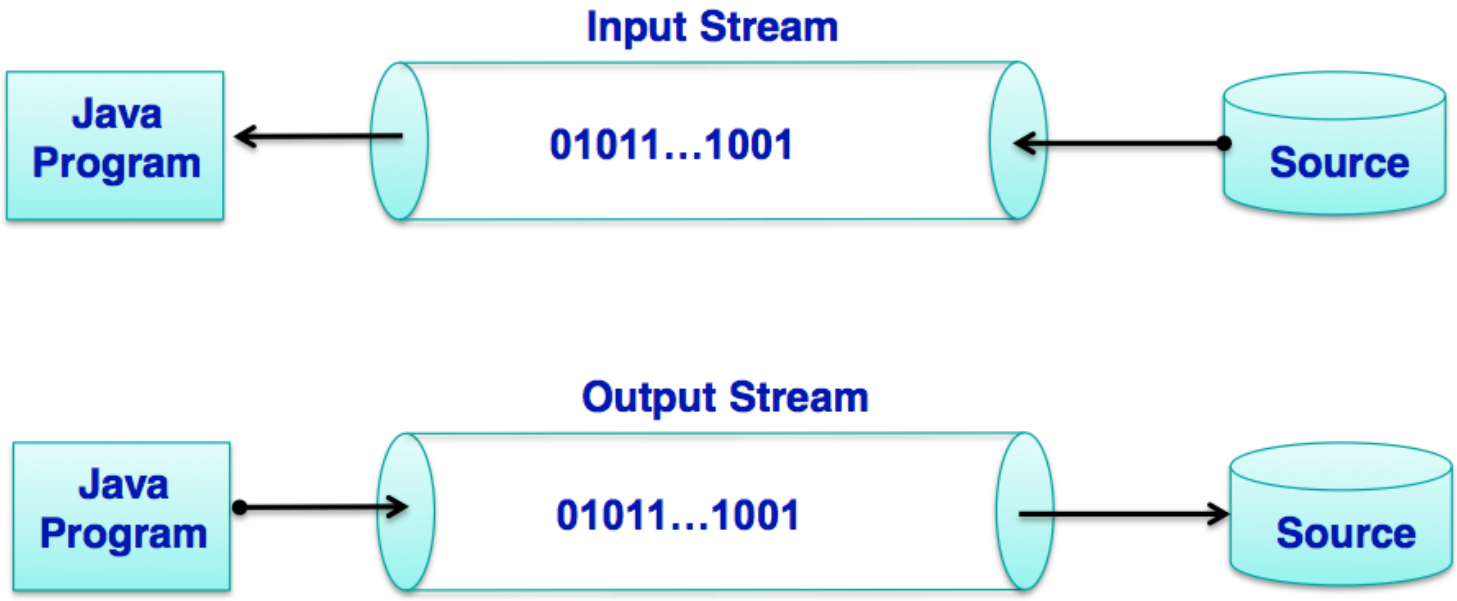
Đọc nội dung toàn bộ tập tin

```
1  package demojavaio;
2
3  import java.io.IOException;
4
5  import java.nio.charset.StandardCharsets;
6  import java.nio.file.Files;
7  import java.nio.file.Paths;
8
9  public class DemoDocNoiDung {
10    public static void main(String[] args) throws IOException {
11       byte[] encoded = Files.readAllBytes(Paths.get("path.txt"));
12
13       String s = new String(encoded, StandardCharsets.UTF_8);
14
15       System.out.println(s);
     }
}
```

Đọc nội dung toàn bộ tập tin

```
1  package demojavaio;
2
3  import java.io.IOException;
4
5  import java.nio.charset.StandardCharsets;
6  import java.nio.file.*;
7  import java.util.List;
8  public class DemoJavaNIO2 {
9
10    public static void main(String[] args) throws IOException {
11       List<String> list =
   Files.readAllLines(Paths.get("path.txt"),StandardCharsets.UTF_8);
12
13       for(String s: list)
14       {
15          System.out.println(s);
16       }
17    }
18 }
```

Stream



Java sử dụng Stream để Read và Write data

Stream

- Stream là một dãy tuần tự các byte có chiều dài không xác định.

- **Input Stream** là các stream thực hiện việc đưa dãy các byte vào trong chương trình Java từ một nguồn bên ngoài.

- **Output Stream** đưa dãy các byte từ chương trình Java đến các nơi bên ngoài

FileInputStream & FileOutputStream

- FileOutputStream: Sử dụng để ghi dữ liệu ra tập tin
  - **Hàm khởi tạo**:
    - **FileOutputStream**(File file)
      - Tạo một đối tượng để ghi vào một tập tin (đại diện bởi đối tượng file: File)
    - **FileOutputStream**(File file, boolean append)
      - Tạo một đối tượng để ghi vào một tập tin, append = true thì nội dung mới sẽ được ghi thêm vào cuối tập tin
    - **FileOutputStream**(String name)

```
1   FileOutputStream fos = new FileOutputStream("path.txt");
```

- **FileOutputStream**(String name, boolean append)

```
1   FileOutputStream fos = new FileOutputStream("path.txt", true);
```

```
1   File f = new File("path.txt");
2   FileOutputStream fos = new FileOutputStream(f);
```

FileInputStream & FileOutputStream

- FileInputStream: Sử dụng để truy xuất dữ liệu từ tập tin
  - **Hàm khởi tạo**:
    - **FileInputStream**(File file)
      - Tạo một đối tượng để đọc dữ liệu của một tập tin (đại diện bởi đối tượng file: File)
    - **FileInputStream**(String name)
      - Tạo một đối tượng từ đường dẫn tập tin

```
1    FileInputStream fis = new FileInputStream("path.txt");
```

```
1    File f = new File("path.txt");
2    FileInputStream fis = new FileInputStream(f);
```

Các bước đọc / ghi tập tin

- Ghi tập tin
  - Tạo một đối tượng **FileOutputStream** từ một đường dẫn tập tin hoặc từ một đối tượng **File**
  - Tạo các đối tượng **Writer/XXXOutputStream** kết nối với đối tượng **FileOutputStream**
  - Sử dụng các đối tượng **Writer/XXXOutputStream** để ghi dữ liệu

- Đọc tập tin
  - Tạo một đối tượng **FileInputStream** từ một đường dẫn tập tin hoặc đối tượng **File**
  - Tạo các đối tượng **Reader /InputStream** kết nối với FileInputStream
  - Sử dụng các đối tượng **Reader/InputStream** để đọc dữ liệu

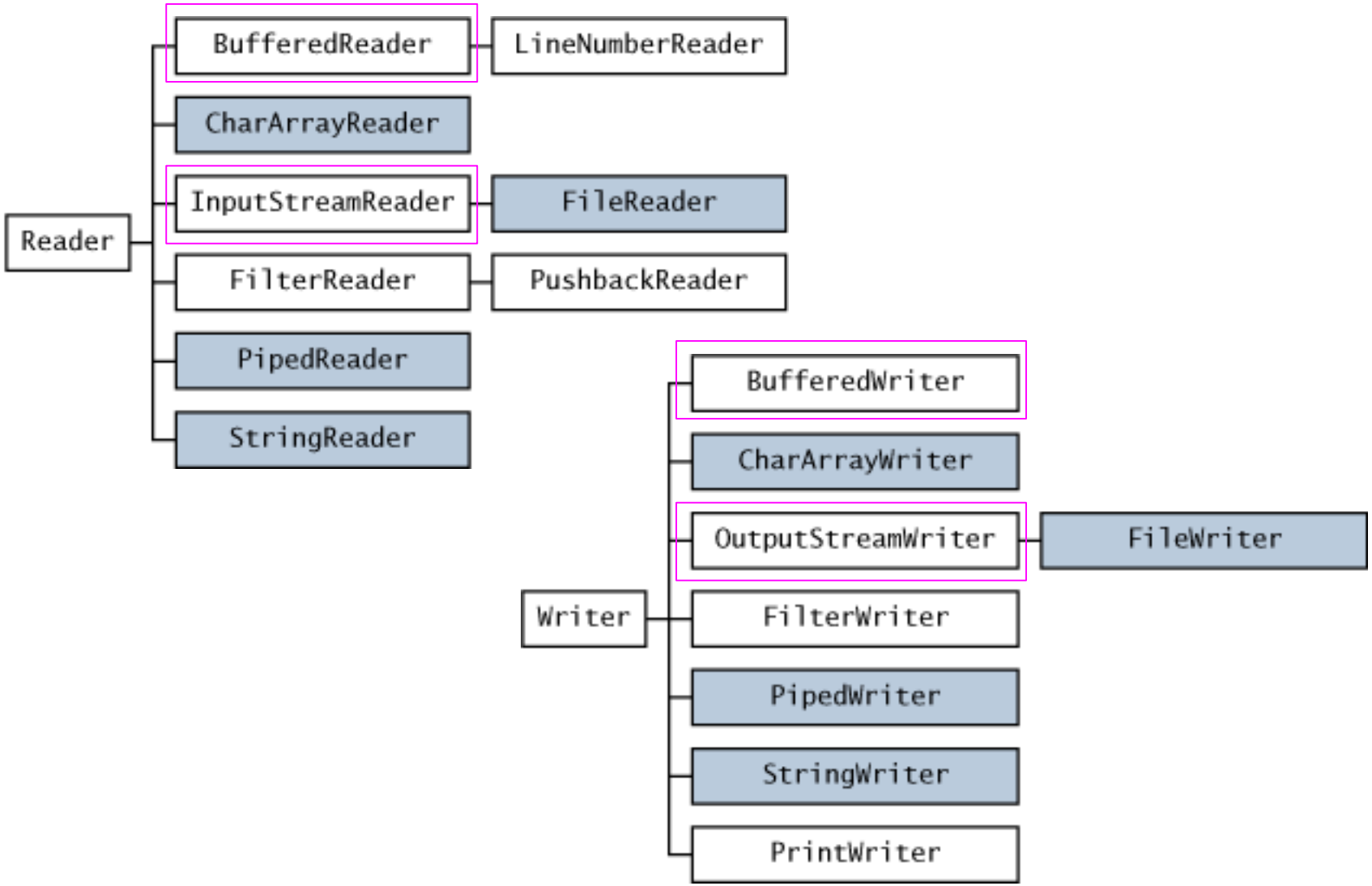
Hai cách đọc/ghi dữ liệu từ Stream

- Package **java.io** bao gồm các Stream Class:

- **Character Streams**:
  - Được sử dụng cho 16-bit characters
  - Sử dụng Read & Write classes
  - Thuận tiện khi đọc/ghi các tập tin văn bản

- **Byte Streams**:
  - Được sử dụng cho 8-bit bytes
  - Sử dụng InputStream & OutputStream classes
  - Thuận tiện để đọc/ghi các tập tin nhị phân

Character Stream



Các lớp có tên ___Reader hoặc ___Writer

Character Stream – Đọc / Ghi tập tin

- Ghi tập tin
  - Tạo một đối tượng **FileOutputStream** từ một đường dẫn tập tin hoặc từ một đối tượng **File**
  - Tạo các đối tượng **Writer** kết nối với đối tượng **FileOutputStream**
  - Sử dụng các đối tượng **Writer** để ghi dữ liệu

- Đọc tập tin
  - Tạo một đối tượng **FileInputStream** từ một đường dẫn tập tin hoặc đối tượng **File**
  - Tạo các đối tượng **Reader** kết nối với FileInputStream
  - Sử dụng các đối tượng **Reader** để đọc dữ liệu

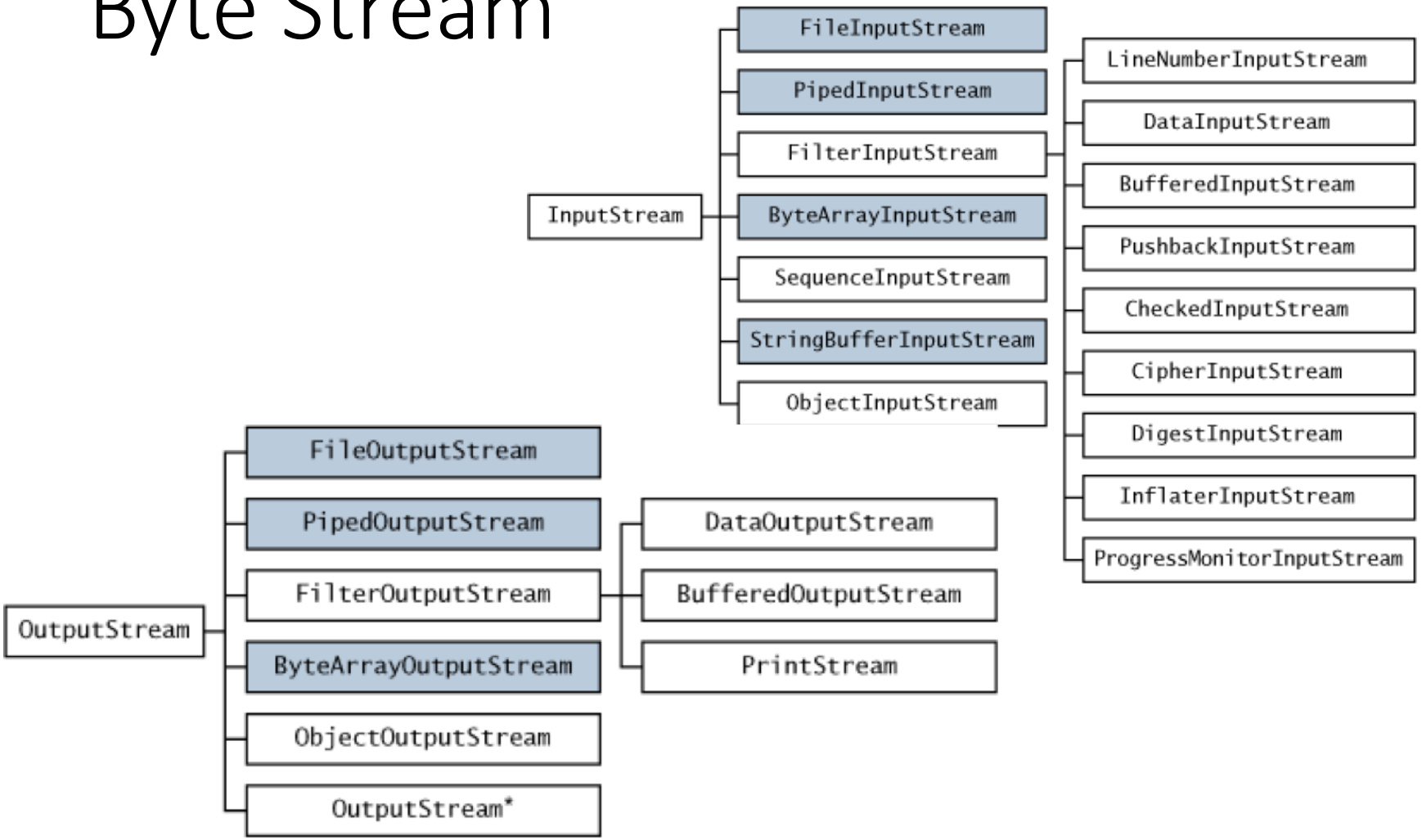
Character Stream - Ghi tập tin

```
1  package demojavaio;
2  import java.io.*;
3  public class Main {
4     public static void main(String[] args) throws IOException {
5        FileOutputStream fos = new FileOutputStream(”filename.txt");
6        OutputStreamWriter writer = new OutputStreamWriter(fo.s, "UTF-
7  8");
8        BufferedWriter bw = new BufferedWriter(writer);
9        String[] strs = new String[] {
10          ”TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN",
11          ”KHOA CÔNG NGHỆ THÔNG TIN”
12       };
13       for(String s: strs) {
14          bw.write(s);
15          bw.write("\n");
16       }
17       bw.close();
   }
}
```

Character Stream – Đọc tập tin

```
1  package demojavaio;
2  import java.io.*;
3  public class DemoDocTapTin {
4     public static void main(String[] args) throws IOException {
5        FileInputStream fis = new FileInputStream("path.txt");
6        InputStreamReader reader = new InputStreamReader(fis, "UTF-8");
7        BufferedReader br = new BufferedReader(reader);
8        String s;
9        do
10       {
11          s = null;
12          s = br.readLine();
13          System.out.println(s);
14       }
15       while (s != null);
16       br.close();
17    }
   }
```

Byte Stream



Các lớp có tên ___InputStream hoặc ___OutputStream

Byte Stream – Đọc/Ghi nội dung

```
1  package demojavaio;
2  import java.io.Serializable;
3  public class PhanSo implements Serializable {
4      private int tuso;
5      private int mauso;
6      public PhanSo (int t, int m)
7      {
8          this.tuso = t;
9          this.mauso = m;
10     }
11     public PhanSo ()
12     {
13         this.tuso = 0;
14         this.mauso = 1;
15     }
16     public void xuat()
17     {
18         System.out.print(this.tuso + "/" + this.mauso);
19     }
}
```

Các lớp phải implement lại interface Serializable thì mới có thể đọc/ghi đối tượng của lớp

Byte Stream – Ghi nội dung tập tin

```
1  package demojavaio;
2  import java.io.FileOutputStream;
3  import java.io.IOException;
4  import java.io.ObjectOutputStream;
5  public class GhiNhiPhan {
6     public static void main(String[] args) throws IOException {
7
8        FileOutputStream fos = new FileOutputStream("path.obj");
9        ObjectOutputStream oos = new ObjectOutputStream(fos);
10       oos.writeChar(100);
11       oos.writeObject(new PhanSo(1, 3));
12       oos.writeObject(new PhanSo(2, 5));
13       oos.writeObject(new PhanSo(3, 4));
14       oos.writeObject(new PhanSo(4, 5));
15       oos.close();
16    }
17 }
```

Byte Stream – Đoc nôi dung tập

```
1  package demojavaio;
2  import java.io.FileInputStream;
3  import java.io.IOException;
4  import java.io.ObjectInputStream;
5  public class DocNhiPhan {
6     public static void main(String[] args) throws IOException,
7  ClassNotFoundException {
8        FileInputStream fos = new FileInputStream("path.obj");
9        ObjectInputStream ois = new ObjectInputStream(fos);
10       int t = ois.readChar();
11       System.out.print(t);
12       PhanSo p = (PhanSo)ois.readObject();
13       p.xuat();
14       p = (PhanSo)ois.readObject();
15       p.xuat();
16       p = (PhanSo)ois.readObject();
17       p.xuat();
18       p = (PhanSo)ois.readObject();
19       p.xuat();
20       ois.close();
     }
}
```

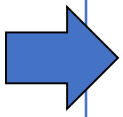
Ví dụ:

- Cho nội dung tập tin **template.txt** như sau:

```
MANV: {MANV}
Họ tên: {HOTEN}
Địa chỉ: {DIACHI}
Ngày sinh: {NGAYSINH}
```

- Hãy xuất thông tin của một nhân viên (ra màn hình ...ra vào:

```
NhanVien nv = new NhanVien();
nv.setHOTEN("Nguyễn Văn A”
nv.setDIACHI("Hồ Chí Minh");
nv.setMANV(12345);
nv.setNGAYSINH(new Date());
```



Kết quả:

```
MANV: 12345
Họ tên: Nguyễn Văn A
Địa chỉ: Hồ Chí Minh
Ngày sinh: Wed Apr 16 23:41:39
ICT 2014
```

Đọc và ghi dữ liệu XML

ThS. *Nguyễn Trường Sơn*

Nội dung trình bày

- Cấu trúc tài liệu XML
- Đọc tài liệu XML
  - DOM Parser
  - XPath Expression
  - StAX Parser
- Ghi tài liệu XML

XML (eXtensible Markup Language)

- XML: Ngôn ngữ đánh dấu văn bản
- Làm dễ dàng cho việc chia sẽ / trao đổi thông tin qua internet, giữa các hệ thống.
- Có khả năng tự mô tả

Ví dụ về một tài liệu XML

```
<?xml version="1.0" encoding="UTF-8"?>
<group>
  <student id="0112123" >
    <name>Nguyễn Văn A</name>
  </student>
  <student id="0112124" >
    <name>Trần Thị B</name>
  </student>
</group>
```

Thẻ con

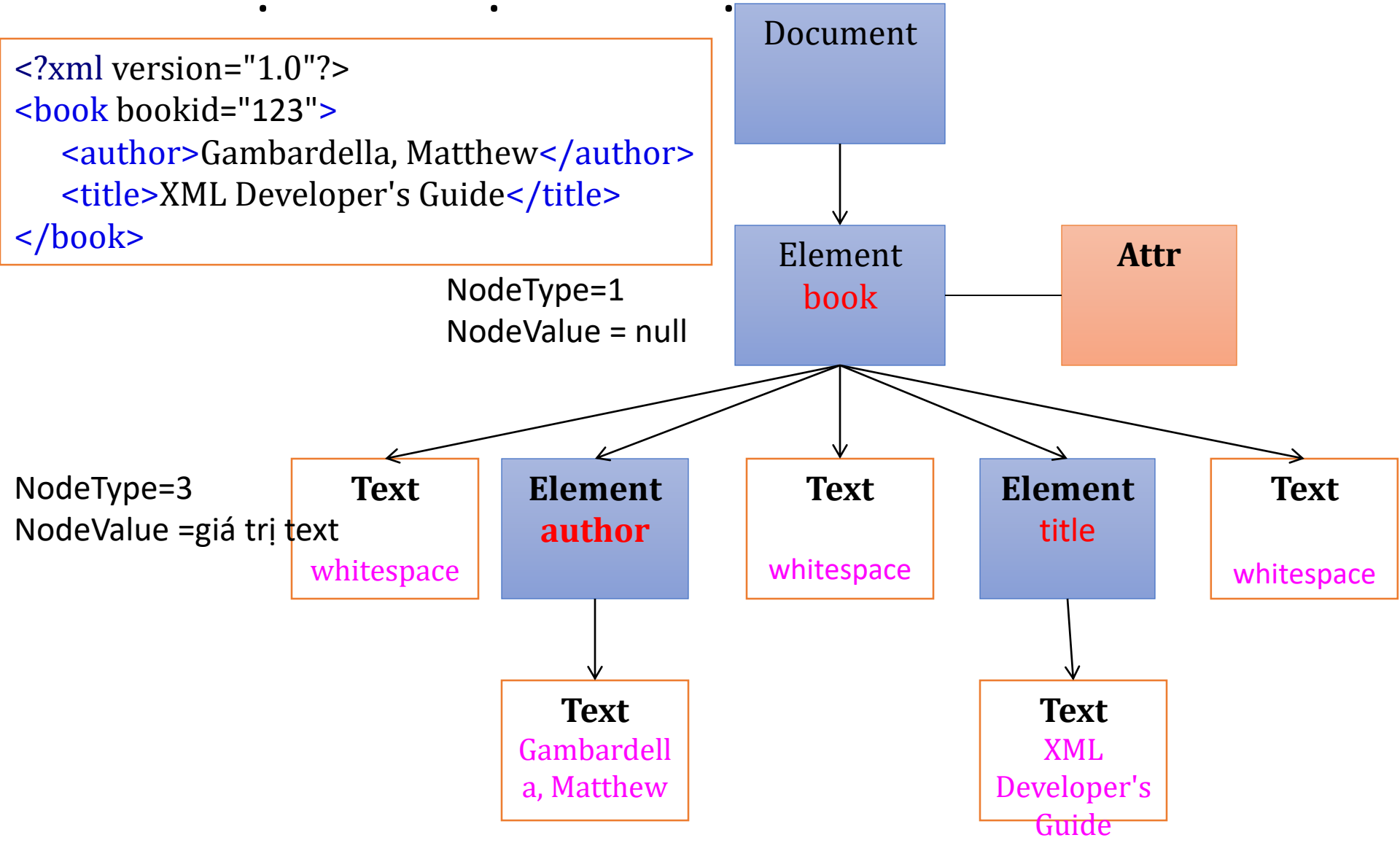
Thẻ con

Cặp thuộc tính = giá trị

Thẻ gốc: Bao toàn bộ tài liệu XML

Ví dụ về một tài liệu XML

```
<?xml version="1.0"?>
<book bookid="123">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
</book>
```



XML Parser

- Java hỗ trợ 2 loại xml parser:
  - **Tree Parser - DOM Parser**: phân tích nội dung tài liệu XML theo mô hình cây phân cấp
  - **Streaming Parser - SAX Parser** (Simple API for XML), **StAX Parser**: phát sinh các sự kiện trong quá trình duyệt tài liệu Xml

Đọc và ghi tài liệu XML

- Đọc tài liệu XML
  - **DOM Parser**
  - XPath Expression
  - StAX Parser
- Ghi tài liệu XML

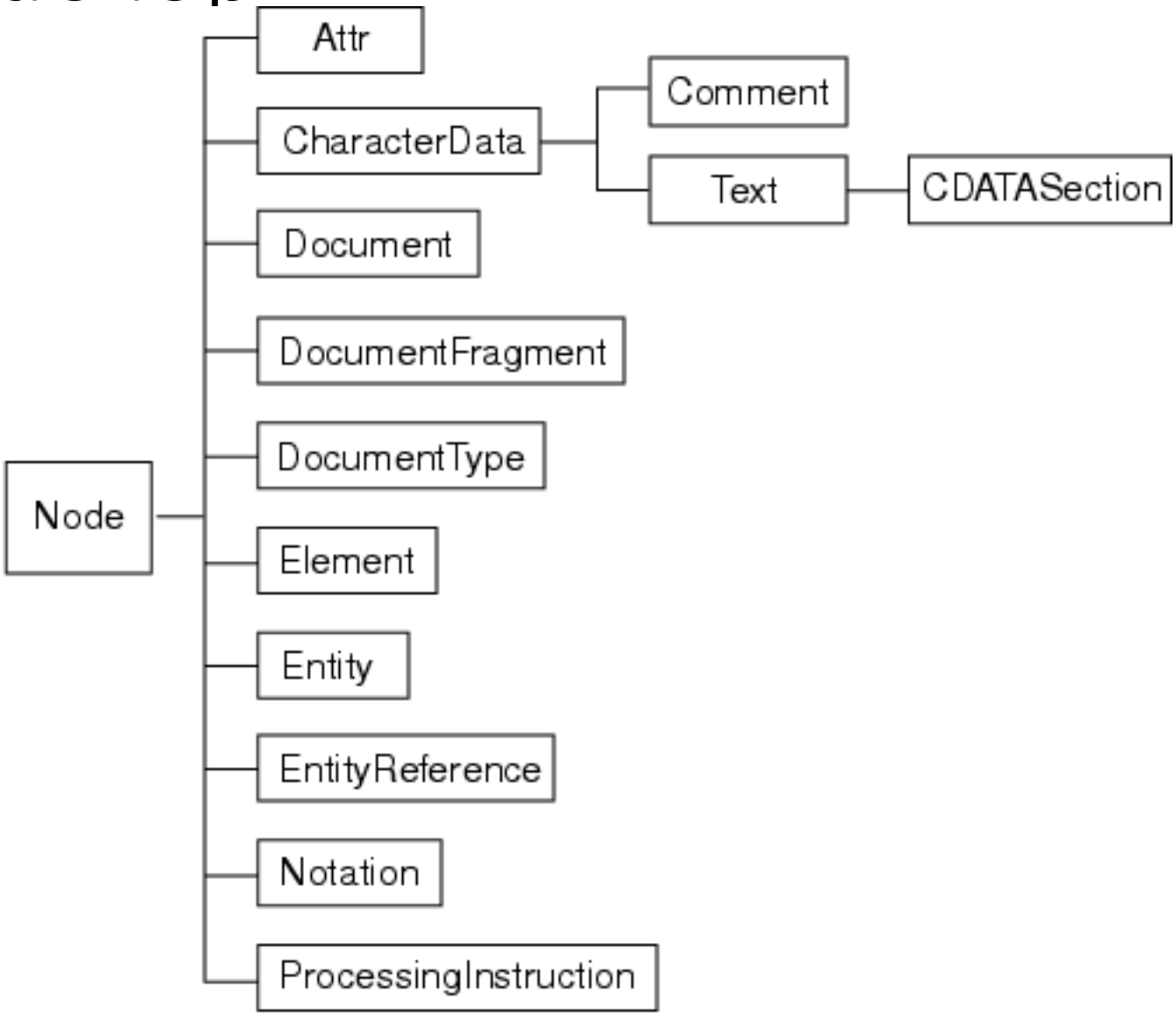
Đọc tài liệu XML – DOM Parser

- Các thư viện cần khai báo:

```
1  import javax.xml.parsers.*;
2  import org.w3c.dom.*;
```

- **Các lớp thường dùng**:
  - DocumentBuilderFactory
  - DocumentBuilder
  - Document
  - Element
  - NodeList
  - Node
  - …

Sơ đồ lớp


Node
Attr
CharacterData
Comment
Text
CDATASection
Document
DocumentFragment
DocumentType
Element
Entity
EntityReference
Notation
ProcessingInstruction

Đọc tài liệu XML – DOM Parser

- **Bước 1**: Tạo đối tượng **Document** chứa toàn bộ tài

```
1  DocumentBuilderFactory factory =
2  DocumentBuilderFactory.newInstance();
3
4  DocumentBuilder builder = factory.newDocumentBuilder();
5  File f = new File("files/book.xml");
6  Document doc = builder.parse(f);
```

```
<?xml version="1.0"?>
<book bookid="123">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
</book>
```

Đọc tài liệu XML – DOM Parser

- Bước 2: Truy xuất các thành phần con
  - Truy xuất **node** gốc:
    - **Element root** = doc.**getDocumentElement**();
  - Lấy về danh sách node con của một node
    - **NodeList list** = **anode**.**getChildNodes**();
    - //anode có thể là một node thuộc loại bất kỳ: Document, Element, Text, Comment, DocumentType, …
  - VD: Lấy danh sách các node con của node gốc (các node

Element
book

```
1  NodeList list = root.getChildNodes();
2  for(int i=0;i<list.getLength();++i)
3  {
4        // xử lý từng node
5        Node node = list.item(i);
6  …
7  }
```

Text	Element author	Text	Element title	Text
whitespace		whitespace		whitespace

Đọc tài liệu XML – DOM Parser

- Kiểm tra một node thuộc loại nào:
  - node ***instanceof*** **Element**
  - node ***instanceof*** **Comment**
  - node ***instanceof*** **Document**

```
1  NodeList list = root.getChildNodes();
2  for(int i=0;i<list.getLength();++i)
3  {
4        // xử lý từng node
5        Node node = list.item(i);
6        if(node instanceof Element)
7        {
               Element element = (Element) node;
         ...
         }
...
}
```

Ép kiểu sang lớp đối tượng tương ứng

Element **author**

Element title

Đọc tài liệu XML – DOM Pa

Hằng số	
DOCUMENT_NODE	9
COMMENT_NODE	8
ELEMENT_NODE	1
ATTRIBUTE_NODE	2
TEXT_NODE	3
…	

- Kiểm tra một node thuộc loại nào:
  - node. **getNodeType**() == Node.*ELEMENT_N*
  - node. **getNodeType**() == Node.*DOCUMENT_N*
  - node. **getNodeType**() == Node.*COMMENT_NO*

```
1  NodeList list = root.getChildNodes();
2  for(int i=0;i<list.getLength();++i)
3  {
4        // xử lý từng node
5        Node node = list.item(i);
6        if(node.getNodeType() == Node.ELEMENT_NODE)
7        {
               Element element = (Element) node;
         …
         }
…
}
```

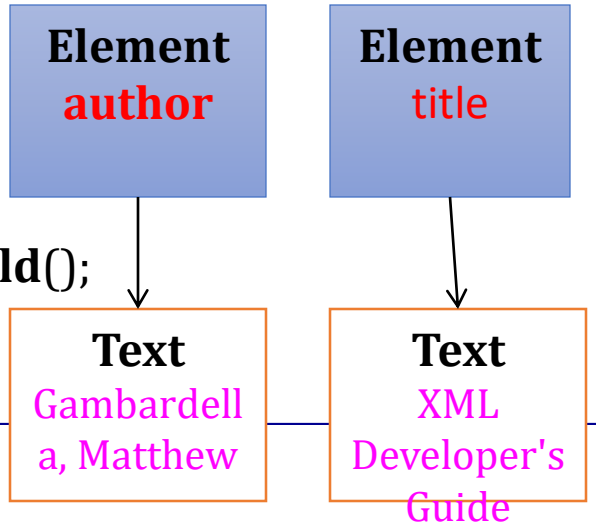
Element
author

Element
title

Đọc tài liệu XML – DOM Parser

- Lấy node con đầu tiên:
  - Node node = element.**getFirstChild**();
  - Node node = element.**getChildNodes**().**item**(0);

```
1  NodeList list = root.getChildNodes();
2      for(int i=0;i<list.getLength();++i)
3      {
4         Node node = list.item(i);
5         if (node instanceof Element)
6         {
7            Element element = (Element)node;
             Text text = (Text)element.getFirstChild();
             System.out.println(text.getData());
          }
```



Đọc tài liệu XML – DOM Parser

- Một số hàm thông dụng của một node:
  - node.**getNodeValue**();
  - node.**getNodeType**();
  - node.**getNodeName**();

  - String textnode.**getData**()

node	getNodeValue()	getNodeType()	getNodeName()
Document	null	9	#document
Element	null	1	Tên của thẻ XML
Text	Giá trị của chuỗi	3	#text
Attr	Giá trị của thuộc tính	2	Tên của thuộc tính

Đọc tài liệu XML – DOM Parser

- Truy xuất thuộc tính
  - **Attr** node.**getAttributeNode**(String tenThuocTinh);
  - **String** node.**getAttribute**(String tenThuocTinh);
  - NamedNodeMap node.**getAttributes**();

```
1   Attr a = root.getAttributeNode("id");
2   System.out.println("VALUE: " + a.getNodeValue());
3   System.out.println("NAME: " + a.getNodeName());
4   System.out.println("TYPE: " + a.getNodeType());
```

```
<?xml version="1.0"?>
<book bookid="123">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
</book>
```

Đọc tài liệu XML – DOM Parser

- Truy xuất danh sách thuộc tính
  - **NamedNodeMap** node.**getAttributes**();

```
1  NamedNodeMap attributes = root.getAttributes();
2      for(int i=0;i<attributes.getLength();++i)
3      {
4         Node attributeNode = attributes.item(i);
5         String name = attributeNode.getNodeName();
6         String value = attributeNode.getNodeValue();
7         System.out.println(name);
8         System.out.println(value);
9      }
```

Đọc và ghi tài liệu XML

- Đọc tài liệu XML
  - DOM Parser
  - **XPath Expression**
  - StAX Parser
- Ghi tài liệu XML

Xpath

- Là các *biểu thức đường dẫn* cho phép truy cập đến các node trong cây tài liệu dễ dàng mà không cần phải duyệt và tìm kiếm trên toàn bộ cây tài liệu.
- Khởi tạo đối tượng XPath:
  - **import javax.xml.xpath.*;**
  - XPathFactory xpFactory = XPathFactory.newInstance();
  - XPath path = xpFactory.newXPath();

Tập tin XML ví dụ

```
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
  </book>
</catalog>
```

Xpath

- Lấy giá trị của một thẻ hoặc một thuộc tính trong tài liệu

- Lấy giá trị của thẻ title trong cuốn sách đầu tiên
  - Đường dẫn Xpath: "**/catalog/book[1]/title**"

- Lấy giá trị của thẻ price trong cuốn sách đầu tiên
  - Đường dẫn Xpath: "**/catalog/book**[1]/**price**"

- Lấy giá trị của thuộc tính id trong cuốn sách đầu tiên
  - Đường dẫn Xpath: "/**catalog/book**[1]/**@id**"

Đọc tài liệu XML – Xpat

```
import java.io.File;
import java.io.IOException;
import javax.xml.parsers.*;
import javax.xml.xpath.*;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
```

```
1  package demojavaxml;
2  ....
3  public class DemoXPath {
4      public static void main(String[] args) throws IOException,
5  ParserConfigurationException, XPathExpressionException, SAXException
6  {
7          DocumentBuilderFactory factory =
8  DocumentBuilderFactory.newInstance();
9          DocumentBuilder builder = factory.newDocumentBuilder();
           Document doc = builder.parse(new File("files/book.xml"));
           XPathFactory xpFactory = XPathFactory.newInstance();
           XPath path = xpFactory.newXPath();
           String s = path.evaluate("/catalog/book[1]/title",doc);
           double price =
   Double.parseDouble(path.evaluate("/catalog/book[1]/price",doc));
           String id = path.evaluate("/catalog/book[1]/@id",doc);
           System.out.println(s);
           System.out.println(price);
           System.out.println(id);
       }
   }
```

Xpath

- Lấy về danh sách các Node
  - NodeList list = (NodeList)path.evaluate("/**catalog**/**book**", doc, XPathContants.NODESET);
- Lấy về 1 node
  - Node node = (Node)path.evaluate("/**catalog**/**book**[1]", doc, XPathContants.NODE);
- Lấy về kết quả là giá trị của một hàm
  - int count = ((Number)path.evaluate("**count**(/**catalog**/**book**)", doc,XPathConstants.NUMBER)).intValue();

Đọc và ghi tài liệu XML

- Đọc tài liệu XML
  - DOM Parser
  - XPath Expression
  - StAX Parser

- Ghi tài liệu XML

Ghi tài liệu XML

- Tạo nội dung tài liệu XML:
  - createElement(“nodeName”);
  - createTextNode(“textContent”);
  - appendChild(nodeName);
  - setAttribute(attributeName , value);
  - …

Tạo tài liệu XML

- Ví dụ:

```
DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder =
docFactory.newDocumentBuilder();
Document doc = docBuilder.newDocument()

Element root =  doc.createElement("root");
Element child1 = doc.createElement("child");
child1.setAttribute("at1","value1");
Text textNode1 = doc.createTextNode("text content");
doc.appendChild(root);
root.appendChild(child);
child.appendChild(textNode1);
```

Ghi tài liệu XML

- XML DOM API hiện thời không hỗ trợ việc kết xuất nội dung cây tài liệu lên bộ nhớ phụ. Ta có thể sử dụng XSLT để thực hiện:

```
//write the content into xml file
TransformerFactory transformerFactory =
TransformerFactory.newInstance();
Transformer transformer =
transformerFactory.newTransformer();
DOMSource source = new DOMSource(doc);

StreamResult result =  new StreamResult(new
File("testing.xml"));
transformer.transform(source, result);
System.out.println("Done");
```

Tạo tài liệu XML

- Kết quả:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root>
   <child at1="value1">text content</child>
</root>
```

Bài tập:

- Hãy viết chức năng:
  - Lưu trữ danh sách nhân viên xuống tập tin XML
  - Tạo tập tin cấu hình (config.xml) để lưu các tham số chung của hệ thống như: cấu hình CSDL, …