FILE & XML HANDLING

### File Handling:

* It is used to store particular program's output data in a file and these files may used to perform operations.
* java.io package is used to access the File class.

### File class Methods:

* canRead() - File Readable or not.
* canWrite() - File writable or not.
* createNewFile() - Create new file.
* delete() - Delete a file.
* exists() - File exist or not.
* length() - size of file in bytes.
* getName() - Name of the file.
* list() - Array of list.
* mkdir() - Creates new directory.
* getAbsolutePath() - Absolute path of file.

### File Operations:

# Create File: createNewFile()

Ex:

import java.io.\*;

public class Test

{

public static void main(String[] args) throws Exception

{

File newfile=new File("newFile.txt");

System.out.println(newfile.createNewFile());

}

}

# Read File: read()

Ex:

# (Method 1):

import java.io.\*;

import java.util.\*;

public class Test

{

public static void main(String[] args) throws Exception

{

File newfile=new File("newFile.txt");

Scanner readFile=new Scanner(newfile);

while(readFile.hasNextLine())

{

System.out.println(readFile.nextLine());

}

}

}

# (Method 2):

import java.io.\*;

import java.util.\*;

public class Test

{

public static void main(String[] args) throws Exception

{

char[] file=new char[100];

FileReader readfile=new FileReader("newFile.txt");

readfile.read(file);

System.out.println(file);

}

}

# Write File: write()

Ex:

import java.io.\*;

public class Test

{

public static void main(String[] args) throws Exception

{

String newData="File Writer";

FileWriter writefile=new FileWriter("newFile.txt");

writefile.write(newData);

writefile.close();

char[] file=new char[100];

FileReader readfile=new FileReader("newFile.txt");

readfile.read(file);

System.out.println(file);

readfile.close();

}

}

# Delete File: delete()

Ex:

import java.io.\*;

public class Test

{

public static void main(String[] args) throws Exception

{

File file=new File("newFile.txt");

System.out.println(file.delete());

}

}

### Streams:

* Sequence of input.
* Perform I/O operations.
* Two types of streams:

1)Input Stream:

* Used to read data.
* Is an Abstract Class.
* Ex:Keyboard
* To create InputStream

InputStream obj = new FileInputStream();

2)Output Stream:

* Used to write data.
* Is an Abstract Class.
* Subclasses are used to write data.
* Ex:file,monitor.
* To create OutputStream

OutputStream obj = new FileOutputStream();

# Based on the data type, there are two types of streams :

# 1)Byte Stream:

# Byte Input Stream: To read byte data from different devices.

# Byte Output Stream: To write byte data to different devices.

# 2)Character Stream:

# Character Input Stream: To read character data from different devices.

# Character Output Stream: To write character data to different devices.

### XML Handling:

* XML stands for Extensible Markup Language.
* XML tags are not predefined like HTML tags.
* XML was designed to carry data.
* HTML was designed to display data.
* XML stores data in plain text format.

Ex:

<?xml version="1.0" encoding="UTF-8" ?>

<Students>

<Student id="1">

<name>name1</name>

<age>age1</age>

</Student>

<Student id="2">

<name>name2</name>

<age>age2</age>

</Student>

</Students>

# note:

* XML Attribute Values Must Always be Quoted.
* XML Stores New Line as LF.
* Text content : name1,age1,name2,age2
* Element : Student
* Attribute : id

### XML Tree Structure:

* Starts at root element and branches from the root to child elements.

Ex:

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

### XML predefined entity reference:

* &lt; < less than
* &gt; > greater than
* &amp; & ampersand
* &apos; ' apostrophe
* &quot; " quotation mark

### XML Naming Rules:

* Element names are case sensitive.
* Starts with letters,underscore,digits,hyphens, underscores, and periods.
* Not start with Xml,xML and white space.

### NameSpace:

* The namespace declaration has the following syntax. xmlns:*prefix*="*URI*".
* When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.
* Namespaces can also be declared in the XML root element.

### Xpath:

* XPath uses path expressions to select nodes or node-sets in an XML document.XPath is a major element in the XSLT standard.

1)nodename - Selects all nodes with the name "nodename"

2) / - Selects from the root node

3) // - Selects nodes in the document from the current node that match the selection no matter where they are.

4) . - Selects the current node

5) .. - Selects the parent of the current node

6) @ - Selects attributes

### XML DOM:(Document Object Model)

* The XML DOM defines a standard way for accessing and manipulating XML documents.
* Not able to edit the value of attribute.
* With DOM functions we can able to
* Create nodes
  + - Remove nodes
    - Change their contents
    - Traverse the node hierarchy.
* DOM is part of the Java API for XML processing (JAXP).
* It presents an XML document as a tree-structure.
* DOM parser traverses the XML file and creates the corresponding DOM objects. These DOM objects are linked together in a tree structure. The parser reads the whole XML structure into the memory.
* SAX is an alternative JAXP API to DOM. SAX parsers are event-based; they are faster and require less memory.
* Tasks,
  + - Sorting elements.
    - Rearranging elements or looking up elements.
* DocumentBuilderFactory enables applications to obtain a parser that produces DOM object trees from XML documents.
* DocumentBuilder defines the API to obtain DOM Document instances from an XML document or to create a new DOM Document.
* Java DOM uses a Transformer to generate the XML file. It is called transformer, because it can also transform the document with XSLT language.

# Node types:

* + - Attr – Represents attribute.
    - CDATASection – Block of text.
    - Comment – Content.
    - Document – Represents XML or HTML document.
    - DocumentFragment – Represent a portion of XML document.
    - Element – Basic branches of XML tree.
    - Node – Primary data of DOM.
    - NodeList – Collection of Nodes.
    - Text – Textual contents.

# Read using (DOM parser, NodeIterator, NodeFilter & TreeWalker)

# Write using DOM

# SAX:

## SAX parser works on following events:

1. startDocument  
 2. startElement  
 3. characters  
 4. comments  
 5. processing instructions  
 6. endElement  
 7. endDocument

### STAX Parser:

* The StAX stands for the “Streaming API for XML” which is used for reading and writing XML documents.
* StAX is a pull API where SAX is a push API.
* StAX can read and write XML documents where SAX can only read XML documents.

# STAX API:

# Iterator API

* + - XMLEventReader
    - XMLEventWriter

# Cursor API

* + XMLStreamReader
  + XMLStreamWriter

### XSLT:

* We can use the XSLT APIs defined in javax.xml.transform to write XML data to a file or convert the XML document into other forms, such as HTML or PDF.
* Java DOM uses a Transformer to generate the XML file. It is called transformer, because it can also transform the document with XSLT language.

### JaxB:(Marshaller)

(Marshal convert object to xml) - (read)

(UnMarshal convert xml to object) - (write)

* Create POJO or bind the schema and generate the classes .
* Create the JAXBContext object .
* Create the Marshaller objects .
* Create the content tree by using set methods .
* Call the marshal method .