**Java I/O Tutorial**

Java I/O (Input and Output) is used to process the input and produce the output.

Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform file handling in java by Java I/O API.

**Stream**

A stream is a sequence of data. In Java a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

In java, 3 streams are created for us automatically. All these streams are attached with console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

Java IO

**OutputStream Class**

Java application uses an output stream to write data to a destination, it may be a file, an array, peripheral device or socket.

OutputStream class is an abstract class. It is the super class of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

Java output stream hierarchy

**InputStream**

Java application uses an input stream to read data from a source, it may be a file, an array, peripheral device or socket.

InputStream class is an abstract class. It is the super class of all classes representing an input stream of bytes.

Java input stream hierarchy

**Writer Class Hierarchy**



**Reader Class Hierarchy**



**Creating, Writing, Appending and Reading a text file**

**import** java.io.\*;

**public** **class** Creating\_Reading\_Writing\_Text\_Files {

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

// Creating, writing and appending a file

CharSequence csq = "ddh a a";

FileWriter f = **new** FileWriter("D:\\temp.txt");

f.write("hello world");

f.append(csq);

f.flush();

f.close();

// reading using FileReader

/\*

\* FileReader r = new FileReader("D:\\temp.txt"); int ch; while(true) { ch =

\* r.read(); if(ch == -1 ) { break; }else { System.out.print((char) ch); }

\*

\* }

\*/

// reading using bufferedreader

FileReader fr = **new** FileReader("D:\\temp.txt");

BufferedReader br = **new** BufferedReader(fr);

String S = **null**;

**while** ((S = br.readLine()) != **null**) {

System.***out***.println(S);

}

br.close();

fr.close();

}

}

**Properties class in Java**

The properties object contains key and value pair both as a string. The java.util.Properties class is the subclass of Hashtable.

It can be used to get property value based on the property key. The Properties class provides methods to get data from properties file and store data into properties file. Moreover, it can be used to get properties of system.

**Advantage of properties file**

Recompilation is not required, if information is changed from properties file: If any information is changed from the properties file, you don't need to recompile the java class. It is used to store information which is to be changed frequently.

Methods of Properties class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void load(Reader r) | loads data from the Reader object. |
| public void load(InputStream is) | loads data from the InputStream object |
| public String getProperty(String key) | returns value based on the key. |
| public void setProperty(String key,String value) | sets the property in the properties object. |
| public void store(Writer w, String comment) | writers the properties in the writer object. |
| public void store(OutputStream os, String comment) | writes the properties in the OutputStream object. |
| storeToXML(OutputStream os, String comment) | writers the properties in the writer object for generating xml document. |
| public void storeToXML(Writer w, String comment, String encoding) | writers the properties in the writer object for generating xml document with specified encoding. |

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.util.Iterator;

**import** java.util.Properties;

**import** java.util.Set;

**public** **class** read\_properties\_file {

**public** **static** **void** main(String[] args) {

**try** {

// System.out.println(System.getProperty("user.dir"));

String path=System.*getProperty*("user.dir")+"\\test.properties";

// System.out.println(path);

Properties prop = **new** Properties();

FileInputStream fs = **new** FileInputStream(path);

prop.load(fs);

System.***out***.println("Checks whether the key is present or not = " + prop.containsKey("city"));

Set<String> s1 = prop.stringPropertyNames();

Iterator<String> i1 = s1.iterator();

String a = **null**;

**while**(i1.hasNext()) {

a = i1.next().toString();

System.***out***.println(a + " = " + prop.getProperty(a));

}

// System.out.println("State = " + prop.getProperty("state"));

// System.out.println("Pincode = " + prop.getProperty("pincode"));

// System.out.println("City = " + prop.getProperty("city"));

} **catch** (Exception e) {

e.printStackTrace();

}

}

}