DAY-63 file IO

---------------

In any programming language,in any application,providing input to the Applications and getting output from the Applications is essential.

In case of C and C++ applications,we are able to perform input and output operations by using some predefined library

in the form of printf(),scanf(),cin>>,cout<<,......

Similarly in Java Applications,to perform input and output operations we have to use "streams".

Java has represented all the streams in the form of predefined classes in "java.io" package.

refer dia:1

Stream:

--------

Stream is medium or channel,it will allow the data in continuous flow from input devices to java

program and from Java program to output devices

refer dia:2

In Java IOStreams are divided into following ways:

--------------------------------------------------

1.Byte oriented Streams.

2.Character-Oriented Streams

1.Byte-Oriented Streams:

-----------------------

These are Streams, which will allow the data in the form of bytes from input devices to Java

program and from java program to output devices.

There are two types of Byte-Oriented Streams

1.InputStream

2.OutputStream

1.InputStream:

---------------

It is a byte-oriented Stream,it will allow data in the form of bytes from input devices(keyborad) to Java

Applications.[writing operation]

2.OutputStream:

---------------

It is a byte-oriented Stream,it will allow the data in the form of bytes from Java applications to

output devices[monitor screen].

NOTE: The length of data items in Byte Oriented Streams is 1 byte.

NOTE: The length of the data in characters-oriented stream is 2 bytes.

2.Character-Oriented Streams:

------------------------------

These are the Streams,which will allow the data in the form of characters from input devices to java program and form java program to output devices.

There are two bytes of character-oriented streams

1.Reader

2.Writer

Data stored permanently in the hard disk is called as "FILE"

Managing the files is reffered as "file handling"

'Small amount of data is handled in the form of file where as large amount data is handled in the form of database'.

File handling concept involves :

--------------------------------

1. creation of the file

2. writing the data on to the file

3. reading the data present in the file

4. closing the file

writing and reading operation on the file is considered as 'File I/O concept'

File handling concept consist of fallowing concepts:

----------------------------------------------------

1. File

2. FileWriter

3. FileReader

4. BufferedWriter

5. BufferedReader

6. PrintWriter

FILE:

---------------

import java.io.\*;

class Demo

{

public static void main(String[] args)

{

File f = new File("abc.txt");

System.out.println(f.exists()); // false

}

}

--> This line wont create any physical file. First it will check is there any physical file named with abc.txt is present or not

If it is present then f will refers that file.

If the file is not present then we are just creating java file object to represent the name abc.txt

example program for creating the new file:

------------------------------------------

import java.io.\*;

class Demo

{

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

{

File f = new File("abc.txt");

System.out.println(f.exists()); // false

f.createNewFile();

System.out.println(f.exists()); // true

}

}

Creation of the file:

---------------------

File f = new File("abc.txt"); # creates java file object

s.o.p(f.exists()); # returns boolean result which indicates file is present or not

f.createNewFile(); # creation of new file in the current working directory

s.o.p(f.exists())

Creation of the Directory:

---------------------

File f = new File("abc.txt"); # creates java file object

s.o.p(f.exists()); # returns boolean result which indicates file is present or not

f.mkdir(); # creation of new folder[directory] in the current working directory

s.o.p(f.exists());

import java.io.\*;

class Demo

{

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

{

File f = new File("abcd.txt");

System.out.println(f.exists()); // false

f.mkdir();

System.out.println(f.exists()); // true

}

}

File class constructors:

------------------------

1. File f = new File(String name);

--> pointing the file present in current working directory.

2. File f = new File(String subdirname,String name);

--> pointing the file present in specified sub-directory using sub-directory name.

3. File f = new File(File subdir,String name);

--> pointing the file present in specified sub-directory using the java file object refernce.

// file class constructor example-1

------------------------------------

import java.io.\*;

class Demo

{

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

{

//File f = new File(String name);

File f = new File("xyz.txt");

System.out.println(f.exists()); // false

f.createNewFile();

System.out.println(f.exists()); // true

}

}

// file class constructor example-2

------------------------------------

import java.io.\*;

class Demo

{

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

{

//File f = new File(String name);

File f = new File("myFolder");

f.mkdir();

// File f = new File(String subdirname,String name);

File f1 = new File("myFolder","myFile.txt");

f1.createNewFile();

System.out.println(f.exists()); // true

}

}

// file class constructor example-3

-------------------------------------

import java.io.\*;

class Demo

{

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

{

//File f = new File(String name);

File f = new File("myFolder");

f.mkdir();

// File f = new File(File subdir,String name);

File f1 = new File(f,"myFile1.txt");

f1.createNewFile();

System.out.println(f.exists()); // true

}

}

// creation of folder and file in a specified directory

-------------------------------------------------------

import java.io.\*;

class Demo

{

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

{

File f = new File("D:\\lmn");

f.mkdir();

File f1 = new File("D:\\lmn","opq.txt");

f1.createNewFile();

}

}

Methods present in the file class:

-----------------------------------

1. boolean exists()

--> This method returns true if the file is present else returns false

2. boolean createNewFile()

--> This method creates a new physical file in the current working directory.

3. boolean mkdir()

--> This method creates a new physical folder in the current working directory.

4. boolean isFile()

--> returns true if the java file object refernce is refering to the physical file else returns false.

5. boolean isDirectory()

--> returns true if the java file object refernce is refering to the physical folder else returns false.

6. Strig[] list()

--> returns all the sub-directory and file names present in the specified directory.

7. long length()

--> returns the number of file and sub-directory present in the specified directory.

8. boolean delete()

--> To delete specified file or directory.

// methods in file class example for Strig[] list()

----------------------------------------------------

import java.io.\*;

class Demo

{

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

{

int count=0;

File f = new File("C:\\Program Files");

String s[]=f.list();

for (String s1:s )

{

count++;

System.out.println(s1);

}

System.out.println("The total number : " +count);

}

}

OUTPUT:

-------

Common Files

CUAssistant

desktop.ini

EditPlus 3

Intel

Internet Explorer

Java

McAfee Security Scan

Microsoft Office 15

Microsoft Update Health Tools

ModifiableWindowsApps

Mozilla Firefox

MySQL

NVIDIA Corporation

OpenShot Video Editor

Realtek

rempl

Sublime Text 3

Synaptics

Uninstall Information

UNP

VideoLAN

Windows Defender

Windows Mail

Windows Media Player

Windows Multimedia Platform

Windows NT

Windows Photo Viewer

Windows Portable Devices

Windows Security

Windows Sidebar

WindowsApps

WindowsPowerShell

The total number : 33

// methods in file class example for isFile() and isDirectory

--------------------------------------------------------------

import java.io.\*;

class Demo

{

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

{

int count=0;

File f = new File("C:\\Windows");

String s[]=f.list();

for (String s1:s )

{

File f1=new File(f,s1);

if (f1.isFile())

{

count++;

System.out.println(s1);

}

}

System.out.println("The total number : " +count);

}

}

ouput:--> gives the number files from the folder by neglating the sub-folders.

FileWriter

-----------

FileWriter is required to write data on to the file

FileWriter class Constuctors are:

----------------------------------

1. FileWriter fw = new FileWriter(string fname);

--> by providing specified file name as argument we can write the data on to the file.

Note: If the file not present then the fileWriter creates a brand new empty file with the given name and then writes the data on to the file.

2. FileWriter fw= new FileWriter(File f);

--> instead of file name we can also pass file object refence as argument.

Note: By default FileWriter will perform overwriting of the data to avoid that we must enable append opeartion while writing

3. FileWriter fw = new FileWriter(string fname,boolean append);

4. FileWriter fw= new FileWriter(File f,boolean append);

// fileWriter example program-1

--------------------------------

import java.io.\*;

class Demo

{

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

{

FileWriter fw = new FileWriter("abc.txt");

fw.write(100); // d --> ASCII CODE

fw.write('\n');

fw.write("100"); // 100 STRING

fw.write('\n');

String s ="study online";

fw.write(s);

fw.write('\n');

fw.close();

}

}

Methods present in the FileWriter class:

-----------------------------------------

1. Write(int ch)

--> To write the single character to the file.

2. Write(char[] ch)

--> To write the array of character to the file.

3. Write(String s)

--> To write the String to the file.

4. flush()

--> To ensure that all the data is wriiten on to the file

5. close()

--> To close the FileWriter

// fileWriter example program-2

------------------------------------

import java.io.\*;

class Demo

{

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

{

FileWriter fw = new FileWriter("abc.txt",true); // append mode enabled

fw.write(100); // d --> ASCII CODE

fw.write('\n');

fw.write("100"); // 100 STRING

fw.write('\n');

String s ="study online";

fw.write(s);

fw.write('\n');

fw.flush();

fw.close();

}

}

Note: \n is platform dependent to avoid this we will use BufferedWriter and PrintWriter