DAY-64 file IO

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

FileReader

----------

To read the data present in the file we will use FileReader class.

To activate fileReader class we must create fileReader class object and to do that we will use the fallowing constructor

Constructor:

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

1. FileReader fr = new FileReader(String fname);

2. FileReader fr = new FileReader(File f);

Methods present in the FileReader class:

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

1. int read()

--> retruns the character[unicode] from the file.If the character is not available it will return -1.

Note: Because this method returns unicode value in the form int during displaying the data we must perform type casting.

2. int read(char[] ch)

--> returns the character as per the size of the array.

3. void close()

--> to close the FileReader

Example-1

----------

// FileReader example-1

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

import java.io.\*;

class FileReaderDemo1

{

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

{

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

FileReader fr = new FileReader(f);

char[] ch = new char[(int)f.length()];

fr.read(ch);

System.out.print(ch);

}

}

/\*

output:

-------

Once a King is always a king..!

\*/

example-2

----------

// FileReader example-2

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

import java.io.\*;

class FileReaderDemo1

{

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

{

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

FileReader fr = new FileReader(f);

char[] ch = new char[(int)f.length()];

fr.read(ch);

System.out.println(ch);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

FileReader fr1 = new FileReader("abc.txt");

int i = fr1.read();

while(i!=-1)

{

System.out.print((char)i);

i=fr1.read();

}

}

}

/\*

output:

-------

Once a King is always a king..!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Once a King is always a king..!

\*/

Note: The disadvantage of FileReader is it will read the data in the form of characters.

To overcome the disadvantage of FileWriter and FileReader we will use :

1. BufferWriter

2. BufferedReader

BufferedWriter:

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

\* We can use BufferedWriter to write character data to the file.

Constructors in BufferedWriter:

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

1. BufferedWriter bw = new BufferedWriter(String name);

Note: BufferedWriter cant communicate directly with the file it can communicate via writer object.

2. BufferWriter bw = new BufferWriter(Write w);

3. BufferWriter bw = new BufferWriter(Write w,int buffersize)

Methods present in the BufferWriter:

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

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

6. newLine()

--> To insert a line separator

example-1

----------

// BufferedWriter example-1

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

import java.io.\*;

class BufferedWriterDemo1

{

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

{

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

BufferedWriter bw = new BufferedWriter(fw);

bw.write(100);

bw.newLine();

char[] ch = {'a','b','c','d'};

bw.write(ch);

bw.newLine();

bw.write("sagar");

bw.newLine();

bw.flush();

bw.close();

System.out.println("data is successfully written on to the file...!");

}

}

Note: It is better to close BufferWriter close method instead of FileWriter close method because if the BufferWriter is closed automatically

filewriter will be closed.

BufferedReader

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

We can use BufferedReader to read character data from the file. The advantage of BufferedReader over FileReader is we can read data line by line in

addition to character by character.

Constructors in the BufferReader:

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

1. Bufferreader br = new BufferReader(Reader r);

2. Bufferreader br = new BufferReader(Reader r,int bufferSize);

Note: BufferReader cant communicate directly with the file it can communicate via reader object.

Methods present in the BufferReader class:

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

1. int read()

--> retruns the character[unicode] from the file.If the character is not available it will return -1.

Note: Because this method returns unicode value in the form int during displaying the data we must perform type casting.

2. int read(char[] ch)

--> returns the character as per the size of the array.

3. void close()

--> to close the FileReader

Note: The disadvantage of FileReader is it will read the data in the form of characters.

4. String readLine()

--> It attempts to read next line from the file and returns it. Incase if the next line is not available then this method returns 'null'.

example-1

----------

// BufferedReader example-1

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

import java.io.\*;

class BufferedReaderDemo1

{

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

{

FileReader fr = new FileReader("abc.txt");

BufferedReader br = new BufferedReader(fr);

String line = br.readLine();

while(line!=null)

{

System.out.println(line);

line=br.readLine();

}

br.close();

System.out.println("read operation is successfully completed...!");

}

}

/\*

output:

------

once a king always a kingd

abcd

sagar

d

abcd

sagar

read operation is successfully completed...!

\*/

Note: It is better to close BufferReader close method instead of FileReader close method because if the BufferReader is closed automatically

FileReader will be closed.

PrintWriter

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

\* We can use PrintWriter to write data to the file.

Everytime calling the newLine method w.r.t BufferWriter is the disadvantage of BufferWriter to overcome this we can use PrintWriter.

Note: Using FileWriter and BufferWriter we can write only in the form of character where we cant use another primitive datatypes to write

the data for example int,boolean etc...

Constructors in the printWriter class:

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

1. PrintWriter pw = new PrintWriter(String fname);

2. PrintWriter pw = new PrintWriter(File f);

3. PrintWriter pw = new PrintWriter(Writer w);

Note: PrintWriter can communicate directly with the file and also it can communicate via some writer object too.

Methods present in the PrintWriter:

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

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

along with the above methods we also have:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print(char ch);

Print(int i);

Print(double d);

Print(boolean b);

Print(String s);

# for new line println is used :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Println(char ch);

Println(int i);

Println(double d);

Println(boolean b);

Println(String s);

example-1

-----------

// PrintWriter example-1

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

import java.io.\*;

class PrintWriterDemo1

{

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

{

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

PrintWriter pw = new PrintWriter(fw);

pw.println(99);

char[] ch = {'a','b','c'};

pw.println(ch);

pw.println("sagar");

pw.flush();

pw.close();

System.out.println("write operation is performed successfully...!");

}

}

Note: The most used classes to read the data is "BufferedReader" and to write the data is "PrintWriter".

In general we can use reader and writers to handle character data[text data] where as to handle binary data like image audio and video file

we will use streams

To read binary data we will use InputStream where to write binary data on to the file we will use OutputStream.

example:

---------

wap to take data from two different file and write it on a new file.

solution:

---------

// example merging of two file:

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

import java.io.\*;

class Task1

{

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

{

PrintWriter pw = new PrintWriter("pqr.txt");

FileReader fr1= new FileReader("abc.txt");

BufferedReader br1 = new BufferedReader(fr1);

FileReader fr2= new FileReader("xyz.txt");

BufferedReader br2 = new BufferedReader(fr2);

String line1 = br1.readLine();

String line2 = br2.readLine();

while((line1!=null)||(line2!=null))

{

if(line1!=null)

{

pw.println(line1);

line1= br1.readLine();

}

if(line2!=null)

{

pw.println(line2);

line2= br2.readLine();

}

}

pw.flush();

br1.close();

br2.close();

pw.close();

}

}

refer dia :

refer dia1:

example for binary data file handling:

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

import java.io.\*;

class Image\_copy\_ex

{

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

{

// to read the binary file

FileInputStream fis = new FileInputStream("pic.jpg");

int size = fis.available();

byte[] b = new byte[size];

fis.read(b);

System.out.println("reading binary file is done..!");

// to write the binary file

FileOutputStream fos = new FileOutputStream("lion.jpg");

fos.write(b);

System.out.println("writing binary file is done..!");

fis.close();

fos.close();

}

}

output:

-------

reading binary file is done..!

writing binary file is done..!