05/04/11

Java.io package

File I/O :-

1. File
2. FileWriter
3. FileReader
4. BufferedReader
5. BufferedWriter
6. PrintWriter.

(1) File :-

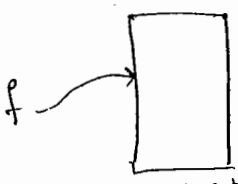
* File f = new File("abc.txt");

→ this line won't create any physical file, first it will check, is there any file named with abc.txt is available or not.

→ If it is available then f simply pointing to that file.

→ If it is not available then f represents just name of the file without creating any physical file.

```
File f = new File("abc.txt");
S.o.pln(f.exists()); // false
f.createNewFile();
S.o.pln(f.exists()); // true
```



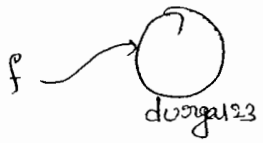
→ A Java file object can represent a directory also.

Ex:- File f = new File("durgai23");

S.o.p(f.exists()); false

f.mkdir();

S.o.p(f.exists()); true



Constructors :-

① File f = new File(String name);

→ Create a java file object to represent name of a file or directory.

② File f = new File(String subdir, String name);

→ To create a file or directory present in some other sub-directory.

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

Ex:- ① Write code to create a file named with abc.txt in current working directory.

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

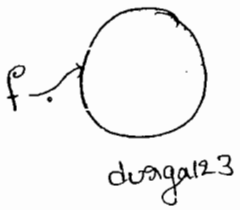
② w.c. to create a directory named with xyz in current working directory.

```
File f = new File("xyz");
f.mkdir();
```

③ w.c. to Create a directory named with durga123 in Current Working directory. in that directory Create a file named with abc.txt.

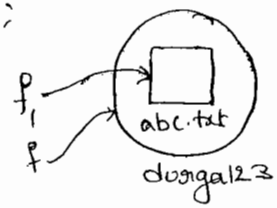
A)

```
File f = new File("durga123");
f.mkdir();
```




```
File f1 = new File("durga123", "abc.txt");
f1.CreateNewFile();
```

@a)




```
File f1 = new File(f, "abc.txt");
f1.CreateNewFile();
```

Important methods of File class :-

① boolean exists();

→ returns true if the physical file or directory present

② boolean CreateNewFile();

→ first this method will check whether the specified file is already available or not. if it is already available then this method written returns false without Creating newfile. if it is not already available then this method returns true after creating new file.

③ boolean mkdir();

④ boolean isFile();

(5) boolean isDirectory();

(6) String[] list();

→ it returns the names of all files & Sub-directories present in the specified directory.

(7) boolean delete();

→ To delete a file or directory

(8) long length();

→ returns the no. of characters present in the Specified-file

Ex:- W.a.p to print the names of all files & Sub-directories present in "D:\durga_classes".

```
import java.io.*;
class Test
{
  p.s.v.m(String[] args) throws Exception
  {
    File f = new File("D:\\durga_classes");
    String[] s = f.list();
    for(String s1 : s)
    {
      S.o.println(s1);
    }
  }
}
```

(2) FileWriter :-

→ We can use FileWriter object to write character data to the file.

Constructors :-

① FileWriter fw = new FileWriter(String name);

② FileWriter fw = new FileWriter(File f);

→ The above 2 constructors ment for overridding. If we want to perform append instead of overridding then we have to use the following constructors.

③ FileWriter fw = new FileWriter(String name, boolean append);

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

→ If the specified file is not already available then the above constructors will Create that file.

Methods of FileWriter :-

① write(int ch);

To write a single character to the file.

② write(char[] ch);

To write an array of characters to the file.

③ write(String s);

To write a String to the file.

(4) flush() :-

→ to give the guarantee that Last character of the data also reflect to the file.

(5) close() :-

Ex!- demo program for the FileWriter.

```
import java.io.*;
Class FileWriterDemo2
{
 P.S.V.m(String[] args)
  {
   FileWriter fw = new FileWriter("wc.txt", true);   → appending
   fw.write(100); // adding a single character
   fw.write("Durgaln SoftwareSolutions");
   char[] ch1 = {'a','b','c'};
   fw.write('\n');
   fw.write(ch1);
   fw.write('\n');
   fw.flush();
   fw.close();
  }
}
```

o/p!- durga
SoftwareSolutions
abc

wo → d

(3) FileReader :-

→ We can use FileReader to Read character data from the file

Constructors :-

1. FileReader fr = new FileReader(String name);
2. FileReader fr = new FileReader(File f);

Methods of FileReader :-

(i) int read() :-

* It attempts to read next character from the file and returns its Unicode value.
* If the next character is not available, then this method returns "-1".

(ii) int read(char[] ch) :-

* It attempts to read enough characters from the file into the char array & returns the no. of characters which are copied from file to the char[].

(iii) close();

Ex:- on FileReader

```
import java.io.*;
Class FileReadDemo
{
  P.S.V.M(String[] args) throws IOException
```

```
File f = new File("wc.txt");
FileReader fr = new FileReader(f);
S.o.pln(fr.read()); //Unicode of first character
Char[] ch2 = new Char[(int)(f.length())];
fr.read(ch2); // file data Copied to array
for(Char ch : ch2)
{
  System.out.print(ch);
}
S.o.pln("***----------");
FileReader fr1 = new FileReader(f);
int i = fr1.read();
While (i != -1)
{
  S.o.pln((Char) i);
  i = fr1.read;
}
}
}
```

* Usage of FileWriter & FileReader is not recommended because :-

1) While writing data by FileWriter we have to insert line Seperators manually which is a bigger headache to the programmer.

2) By using FileReader we can read data character by character which is not convineiant to the programmer.

3) To resolve these problem SUN people introduced BufferedWriter & BufferedReader classes.

(ii) BufferedWriter :-

→ We can use BufferedWriter to write character data to the file.

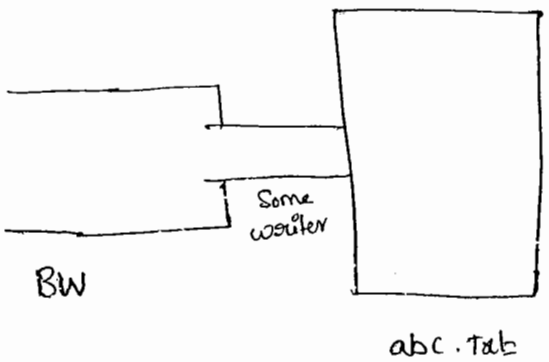
Constructors :-

1) BufferedWriter bw = new BufferedWriter(Writer w);

2) BufferedWriter bw = new BufferedWriter(Writer w, int buffersize);

Note:-

* BufferedWriter never communicates directly with the file compulsory it should communicate via some Writer object only.



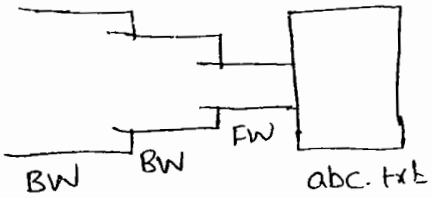
Q) Which of the following are valid.

✗ ① BufferedWriter bw = new BufferedWriter("abc.txt");

✗ ② BufferedWriter bw = new BW(new file("abc.txt"));

✓ ③ BufferedWriter bw = new BufferedWriter(new FileWriter("abc.txt"));

✓ ④ BW bw = new BW(new BW(new FW(new file("abc.txt"))));



Important methods of BufferedWriter :-

① write(int ch)

② write(char[] ch)

③ write(String s)

④ flush()

⑤ close()

⑥ newLine(); :- to insert a newline character

Q:- When compared with FileWriter which of the following capability is available as a seperate method in BufferedWriter.

A) ④

① writing data to the file. ✓ ④ inserting a line seperator.

② flushing the stream.

③ closing the stream.

Ex:-

```
import java.io.*;
class BufferedWriterDemo
{
  P.S.V.m(String[] args) throws IOException
  {
    File f = new File("wc.txt");
    FileWriter fw = new FileWriter(f);
    BufferedWriter bw = new BufferedWrite(fw);
    bw.write(100);
    bw.newLine();
    char[] ch1 = {'a','b','c','d'};
    bw.write(ch1);
    bw.newLine();
    bw.write("duvga");
    bw.newLine();
    bw.write("Software Solutions");
    bw.flush();
    bw.close();
  }
}
```

o/p:-

```
d
abcd
duvga
SoftwareSolutions
```

wc.txt

Note:- When ever we are closing BufferedWriter automatically underlaying writers will be closed

BW.close();	fw.close();	fw.close(); bw.close();
✓	X	X

(iv) BufferedReader() :-

* The main Advantage of BufferedReader over FileReader is we can read the data Line by Line instead of Reading character by character. This approach improves performance of the System by reducing the no. of read operations.

Constructors :-

(i) BufferedReader br = new BufferedReader(Reader r);

(ii) " (Reader r, int buffer size);

Note :-

* BufferedReader can't communicate directly with the file compulsary it should communicate via some reader object.

Important Methods :-

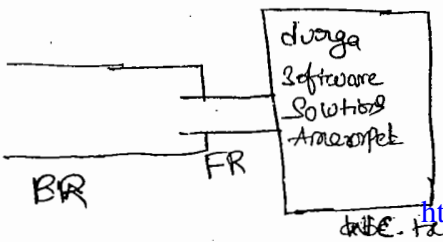
① int read();

② int read(char[] ch);

③ close();

④ String readLine();

* It attempts to find the nextLine & if the nextLine is available then it returns it, otherwise it returns null.



Ex:-
```
import java.io.*;
class Buffered
{
  p.s.v.m(String[] args) throws Exception
  {
    FileReader fr = new FileReader("wc.txt");
    BufferedReader br = new BufferedReader(fr);
    String line = br.readLine();
    while(line != null)
    {
      S.o.println(line);
      line = br.readLine();
    }
    br.close();
  }
}
```

o/p:-
```
durga
Software
Solutions
Ameerpet
```

Note:-

* when ever your closing BufferedReader underlying Readers will be closed.

⑪ PrintWriter() :-

* this is the most enhanced writer to write a character data to file. By using FileWriter & BufferedWriter we can write only character data but by using PrintWriter we can write any primitive data types to the file.

Constructors:-

① PrintWriter pw = new PrintWriter(String name);

② PrintWriter pw = new PrintWriter(File f);

③ PrintWriter pw = new PrintWriter(Writer w);

Methods:-

① write(int ch)
② write(char[] ch)
③ write(String s)
④ flush()
⑤ close()

print(char ch)
print(int i)
print(long l)
print(double d)
print(String s)
print(char[] ch)
⋮

println(char ch)
println(int i)
println(long l)
println(double d)
println(String s)
println(char[] ch)
⋮

ex:-

```
import java.io.*;
class PrintWriterDemo1
{
  p.s.v.m(String[] args) throws IOException
  {
```

```
FileWriter fw = new FileWriter("coc.txt");
PrintWriter pw = new PrintWriter(fw);
pw.write(100); // d
pw.println(100); // 100
pw.println(true); // true
pw.println('c'); // c
pw.println("durga"); // durga
pw.flush();
pw.close();
}
}
```

o/p:-

```
d100
true
c
durga
```

coc.txt

Q:- what is the diff. b/w the following

(a) pw.write(100);

(b) pw.print(100);

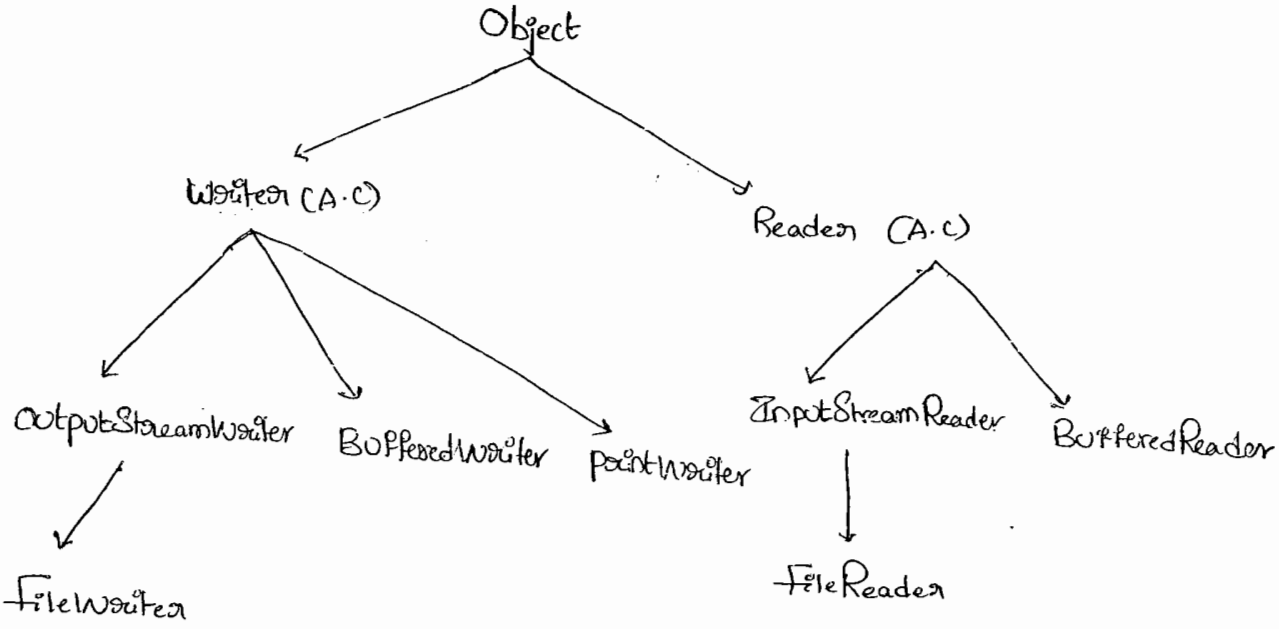
Ans:-

pw.write(100);

→ In this case the corresponding Character d will be added to the file

pw.print(100)

→ In this case the corresponding int value 100 directly will be added to the file

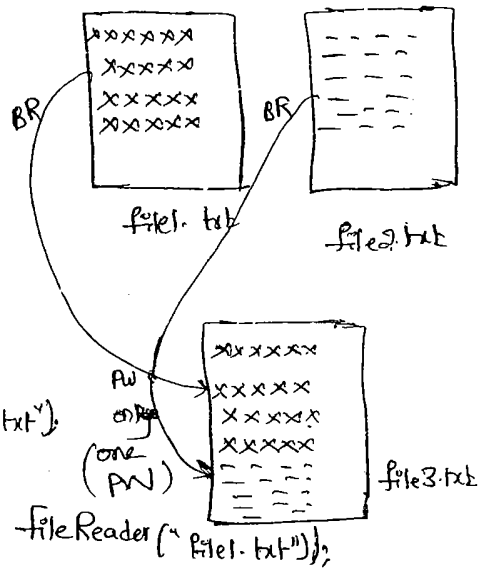


Note:-

* Readers & writers meant for handling character data (any primitive data type) +
* To handle Binary data (like images, movie files, jar files ....) we should go for Streams.
* We can use InputStream to Read Binary data & OutputStream to write a Binary data.
* We can use ObjectInputStream & ObjectOutputStream to read & write Objects to a file respectively (Serialization).
* The most Enhanced Writer to write character data is PrintWriter where as the most Enhanced Reader to read character data is BufferedReader.

*) W.a.p to Merge data from Two files into a 3rd file. ಪ್ರಶ್ನೆ

file3.txt = file1.txt + file2.txt




```
import java.io.*;
class FileMerger
{
 p.s.v.m (String[] args) throws Exception
 {
  PrintWriter pw = new PrintWriter("output.txt");
  BufferedReader br = new BufferedReader(new FileReader("file1.txt"));
  String line = br.readLine();
  while (line != null)
  {
    pw.println(line);
    line = br.readLine();
  }
  br = new BufferedReader(new FileReader("file2.txt"));
  line = br.readline();
  while (line != null)
  {
    pw.println(line);
    line = br.readLine();
  }
 pw.flush();
 br.close();
 pw.close();
}
```

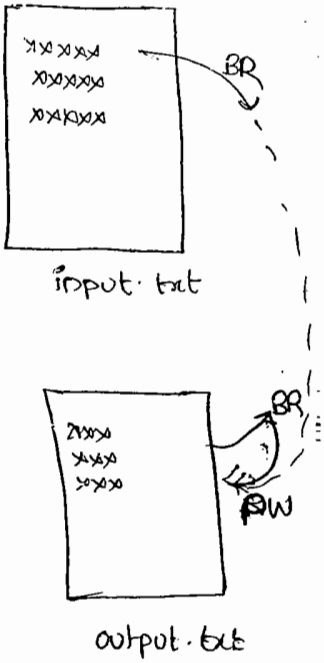
Ex2:-

w.a.p to merge data from 2 files into a 3rd file but merging should be done line by line alternatively.

```
import java.io.*;
Class FileMerger2
{
  P.S.V.m(String[] args) throws IOException
  {
    PrintWriter pw = new PrintWriter("output.txt");
    BufferedReader br1 = new BufferedReader(new FileReader("file1.txt"));
    BufferedReader br2 = new BufferedReader(new fileReader("file2.txt"));
    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();                    pw.close();
br1.close();
br2.close();
}
```

3)- W.a.p to merge data from two files into a 3rd file but merging should be done para by para. assume that there is a Blank Line B/w Every 2 paras?

4)- W.a.p to delete duplicates from the given i/p file ?




```
import java.io.*;
class DuplicateEliminator
{
 p.s.v.m(String[] args)throws Exception
 {
 BufferedReader br1 = new BufferedReader(new FileReader("i/p.txt"));
 PrintWriter pw = new PrintWriter("o/p.txt");
 String line = br1.readLine();
 while(line != null)
 {
   boolean available = false;
   BR br2 = new BR(new FR("o/p.txt"));
   String target = br2.readLine();
   while(target != null)
    {
     if(line.equals(target))
     {
      available = true;
       break;
     }
    target =br2.readLine();
    }
```

```
if(available == false)
 {
   pw.println(line);
   pw.flush();
 }
  line = br1.readLine();
}
 pw.flush();
 br1.close();
 br2.close();
 pw.close();
 }
}
```

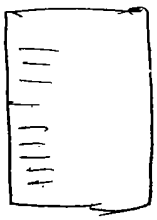
(25) W.a.p to perform file Extraction (result.txt = total.txt - delete.txt)

```
import java.io.*;
Class FileExtraction
{
  p.s.v.m (String[] args) throws Exceptions
   {
     BufferedReader br1 = new BufferedReader(new FileReader
                                  ("mobile.txt"));
     PrintWriter pw = new printWriter("output.txt");
     String line = br1.readLine();
     while(line != null)
     {
       boolean available = false;
       BR br2 = new BR(new FR("delete.txt"));
```

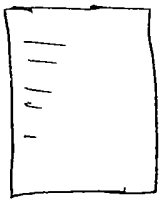
1) operators
2) Exception handling

```
String target = br2.readLine();
while(target != null)
{
 if(line.equals(target))
 {
   available = true;
    break;
 }
  target = br2.readLine();
}
 if(available == false)
 {
  pw.println(line);
  }
  line = br1.readline();
 }
  pw.flush();
  br1.close();
  br2.close();
  pw.close();
 }
}
```

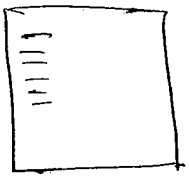
durgajobs.com



total.txt



choice.txt



result.txt