**BufferedReader :**

We can use BufferedReader to read character data from the file. The main advantage of BufferedReader when compared with FileReader is we can use data line by line in addition to character by character.

**Constructors:**

1. **BufferedReader br = new BufferedReader(Reader r);**
2. **BufferedReader br = new BufferedReader(Reader r, int buffersize);**

**Note:** BufferedReader can’t communicate directly to the file. It can communicate through some reader file.

**Methods:**

1. **int read()**
2. **int read(char[] ch)**
3. **void close()**
4. **String readLine()**

It returns the next line from the file. If the next is not available then this method returns null.

Ex:

FileReader fr = new FileReader(“abc.txt”);

BufferedReader br = new BufferedReader(fr);

String line = br.readLine();

while(line != null){

Sop(line);

line = br.readLine();

}

br.close();

O/P:

The content of abc.txt line by line

Note: Whenever we are closing BufferedReader automatically underlaying FileReader will be closed and we are not required to close explicitly.

The most enhanced to read character data from the file is Buffered Reader.

**PrintWriter:**

It is the most enhanced writer to write character data to the file The main advantage of PrintWriter over FileWriter and BufferedWriter is we can write any type of primitive data directly to the file.

**Constructors:**

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 to the file and can communicate via some writer object also.

**Methods:**

1. **write(int ch)**
2. **write(char[] ch)**
3. **write(String s)**
4. **flush()**
5. **close()**
6. **print(char ch)**
7. **print(int i)**
8. **print(double d)**
9. **print(boolean b)**
10. **print(String s)**
11. **println(char ch)**
12. **println(int i)**
13. **println(double d)**
14. **println(boolean b)**
15. **println(String s)**

Ex:

FileWriter fw = new Filewriter(“abc.txt”);

PrintWriter out = new PrintWriter(fw);

out.write(100);

out.println(100);

out.println(true);

out.println(‘c’);

out.println(“durga”);

out.flush();

out.close();

O/P:

d100

true

c

durga

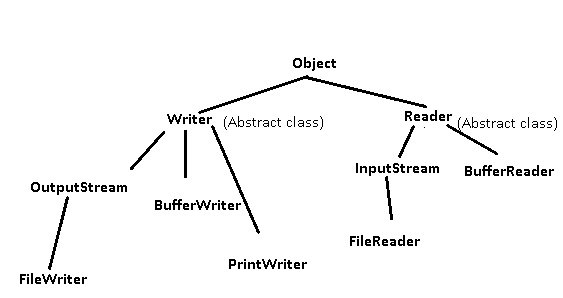
What is the difference between write(100) and print(100)?

In the case of write(100) the corresponding character d will be added to the file but in the case of print(100) the int value 100 will added to the file directly.

The most enhanced writer to write character data to the file is PrintWriter. Where as the most enhanced reader to read character data from the file is BufferedReader.

In general we can use readers and writers to handle character data(text data) where as we can use Streams to handle binary data like images, video file, audio file etc.

We can OutputStream to write binary data to the file InputStream to reads binary data from the file.



Q. WAP to merge data from two files into a third file.

import java.io.\*;  
  
public class MergeFilers {  
 public static void main(String[] args) throws IOException {  
 PrintWriter pw=new PrintWriter("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\file1.txt");  
 BufferedReader br = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\chiku.txt"));  
 String line = br.readLine();  
 while(line != null){  
 pw.println(line);  
 line = br.readLine();  
 }  
 br = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\sonu.txt"));  
 line = br.readLine();  
 while(line != null){  
 pw.println(line);  
 line = br.readLine();  
 }  
 pw.flush();  
 pw.close();  
 br = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\file1.txt"));  
 line = br.readLine();  
 while(line!=null){  
 System.*out*.println(line);  
 line = br.readLine();  
 }  
 }  
}

WAP to perform file extraction operation!

import java.io.\*;  
  
public class EtractionOfFile {  
 public static void main(String[] args) throws IOException {  
 PrintWriter pw = new PrintWriter("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\output.txt");  
 BufferedReader br = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\input.txt"));  
 String line = br.readLine();  
 while(line != null){  
 boolean avil= false;  
 BufferedReader br2 = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\delete.txt"));  
 String target = br2.readLine();  
 while(target != null){  
 if(line.equals(target)){  
 avil=true;  
 break;  
 }  
 target= br2.readLine();  
 }  
 if (avil==false){  
 pw.println(line);  
 }  
 line= br.readLine();  
 }  
 pw.flush();  
 pw.close();  
  
  
 }  
}

WAP to remove duplicates from the given input file.

import java.io.\*;

public class DeleteDuplicate {

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

PrintWriter pw = new PrintWriter("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\output.txt");

BufferedReader br1 = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\input.txt"));

String line = br1.readLine();

while(line != null){

boolean available = false;

BufferedReader br2 = new BufferedReader(new FileReader("C:\\Users\\Lenovo\\Desktop\\My Notes\\Me\\output.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();

}

}

}