CORE JAVA

With

SCJP / OCJP

Study Material

Chapter 11 : File I/O

DURGA M.Tech

(Sun certified & Realtime Expert)

Ex. IBM Employee

Trained Lakhs of Students
for last 14 years across INDIA

File IO Package

Agenda:

1. **File**
2. **FileWriter**
3. **FileReader**
4. **BufferedWriter**
5. **BufferedReader**

File:

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

This line 1st checks whether abc.txt file is already available (or) not if it is already available then "f" simply refers that file.
If it is not already available then it won't create any physical file just creates a java File object represents name of the file.

```
Example:
import java.io.*;
class  FileDemo
{
      public static void main(String[] args)throws IOException
      {
            File f=new File("cricket.txt");
            System.out.println(f.exists());//false
            f.createNewFile();
            System.out.println(f.exists());//true
            }
}


output :
1st run :
false
true

2nd run :
true
true
```

A java File object can represent a directory also.

```
Example:
import java.io.*;
class  FileDemo
{
      public static void main(String[] args)throws IOException
      {
```

```
            File f=new File("cricket123");
            System.out.println(f.exists());//false
            f.mkdir();
            System.out.println(f.exists());//true
      }
}
```

Note: in UNIX everything is a file, java "file IO" is based on UNIX operating system hence in java also we can represent both files and directories by File object only.

File class constructors:

1. **File f=new File(String name);**
   Creates a java File object that represents name of the file or directory in current working directory.
2. **File f=new File(String subdirname,String name);**
   Creates a File object that represents name of the file or directory present in specified sub directory.
3. **File f=new File(File subdir,String name);**

Requirement: **Write code to create a file named with demo.txt in current working directory.**

Program:

```
import java.io.*;
class  FileDemo
{
      public static void main(String[] args)throws IOException
      {
            File f=new File("demo.txt");
            f.createNewFile();
      }
}
```

Requirement: **Write code to create a directory named with SaiCharan123 in current working directory and create a file named with abc.txt in that directory.**

Program:

```
import java.io.*;
class  FileDemo
{
      public static void main(String[] args)throws IOException
      {
            File f1=new File("SaiCharan123");
            f1.mkdir();
            File f2=new File("SaiCharan123","abc.txt");
            f2.createNewFile();
      }
}
```

Requirement: **Write code to create a file named with demo.txt present in c:\saicharan folder.**

Program:

```
import java.io.*;
```

```
class  FileDemo
{
      public static void main(String[] args)throws IOException
      {
            File f=new File("c:\\saiCharan","demo.txt");
            f.createNewFile();
      }
}
```

Import methods of file class:

1. **boolean exists();**
   Returns true if the physical file or directory available.
2. **booleancreateNewFile();**
   This method 1st checks whether the physical file is already available or not if it is already available then this method simply returns false without creating any physical file.
   If this file is not already available then it will create a new file and returns true
3. **booleanmkdir();**
   This method 1st checks whether the directory is already available or not if it is already available then this method simply returns false without creating any directory.
   If this directory is not already available then it will create a new directory and returns true
4. **booleanisFile();**
   Returns true if the File object represents a physical file.
5. **booleanisDirectory();**
   Returns true if the File object represents a directory.
6. **String[] list();**
   It returns the names of all files and subdirectories present in the specified directory.
7. **long length();**
   Returns the no of characters present in the file.
8. **boolean delete();**
   To delete a file or directory.

Requirement: **Write a program to display the names of all files and directories present in c:\\charan_classes.**

```
Program:
import java.io.*;
class  FileDemo
{
      public static void main(String[] args)throws IOException
      {     int count=0;
            File f=new File("c:\\charan_classes");
            String[] s=f.list();
            for(String s1=s) {
            count++;
            System.out.println(s1);
```

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

        }
}
```

Requirement: **Write a program to display only file names**

```
Program:
import java.io.*;
class  FileDemo
{
        public static void main(String[] args)throws IOException
        {       int count=0;
                File f=new File("c:\\charan_classes");
                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("total number : "+count);

        }
}
```

Requirement: **Write a program to display only directory names**

```
Program:
import java.io.*;
class  FileDemo
{
        public static void main(String[] args)throws IOException
        {       int count=0;
                File f=new File("c:\\charan_classes");
                String[] s=f.list();
                for(String s1=s) {
                 File f1=new file(f,s1);
                if(f1.isDirectory()) {
                count++;
                System.out.println(s1);
               }
              }
        System.out.println("total number : "+count);

        }
}
```

FileWriter:

By using FileWriter object we can write character data to the file.

Constructors:

FileWriterfw=new FileWriter(String name);
FileWriterfw=new FileWriter(File f);

The above 2 constructors meant for overriding.

Instead of overriding if we want append operation then we should go for the following 2 constructors.

FileWriterfw=new FileWriter(String name,boolean append);
FileWriterfw=new FileWriter(File f,boolean append);

If the specified physical file is not already available then these constructors will create that file.

Methods:

1. **write(intch);**
   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 the total data include last character also written to the file.
5. **close();**
   To close the stream.

```
Example:
import java.io.*;
classFileWriterDemo
{
      public static void main(String[] args)throws IOException
      {
            FileWriterfw=new FileWriter("cricket.txt",true);
            fw.write(99);//adding a single character
            fw.write("haran\nsoftware solutions");
            fw.write("\n");
            char[] ch={'a','b','c'};
            fw.write(ch);
            fw.write("\n");
            fw.flush();
            fw.close();
      }
```

```
}
Output:
charan
software solutions
abc
```

Note :

- **The main problem with FileWriter is we have to insert line separator manually , which is difficult to the programmer. ('\n')**
- **And even line separator varing from system to system.**

FileReader:

By using FileReader object we can read character data from the file.

Constructors:

FileReaderfr=new FileReader(String name);
FileReaderfr=new FileReader (File f);

Methods:

1. **int read();**
   It attempts to read next character from the file and return its Unicode value. If the next character is not available then we will get -1.

```
2. int i=fr.read();
3. System.out.println((char)i);
```

   As this method returns unicodevalue , while printing we have to perform type casting.

4. **int read(char[] ch);**
   It attempts to read enough characters from the file into char[] array and returns the no of characters copied from the file into char[] array.

```
5.  File f=new File("abc.txt");
6.  Char[] ch=new Char[(int)f.length()];
```

7. **void close();**

```
Approach 1:
import java.io.*;
classFileReaderDemo
{
      public static void main(String[] args)throws IOException
      {
```

```
            FileReaderfr=new FileReader("cricket.txt");
            int i=fr.read();     //more amount of data
            while(i!=-1)
            {
                  System.out.print((char)i);
                  i=fr.read();
            }
      }
}
Output:
Charan
Software solutions
ABC
Approach 2:
import java.io.*;
classFileReaderDemo
{
      public static void main(String[] args)throws IOException
      {
            File f=new File("cricket.txt");
            FileReaderfr=new FileReader(f);
            char[] ch=new char[(int)f.length()];  //small
amount of data
            fr.read(ch);
            for(char ch1:ch)
            {
                  System.out.print(ch1);
            }
      }
}
Output:
XYZ
Software solutions.
```

Usage of *FileWriter* and *FileReader* is not recommended because :

1. **While writing data by FileWriter compulsory we should insert line separator(\n) manually which is a bigger headache to the programmer.**
2. **While reading data by FileReader we have to read character by character instead of line by line which is not convenient to the programmer.**
3. **To overcome these limitations we should go for BufferedWriter and BufferedReader concepts.**

BufferedWriter:

By using BufferedWriter object we can write character data to the file.

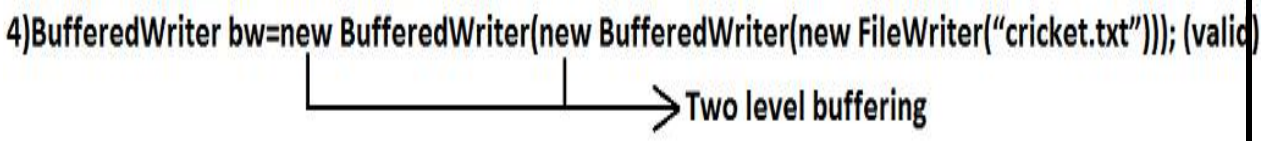
Constructors:

BufferedWriterbw=new BufferedWriter(writer w);
BufferedWriterbw=new BufferedWriter(writer w,intbuffersize);

Note:BufferedWriter never communicates directly with the file it should communicates via some writer object.

Which of the following declarations are valid?

1. **BufferedWriterbw=new BufferedWriter("cricket.txt"); (invalid)**
2. **BufferedWriterbw=new BufferedWriter (new File("cricket.txt")); (invalid)**
3. **BufferedWriterbw=new BufferedWriter (new FileWriter("cricket.txt")); (valid)**



Methods:

1. **write(intch);**
2. **write(char[] ch);**
3. **write(String s);**
4. **flush();**
5. **close();**
6. **newline();**
   Inserting a new line character to the file.

When compared with FileWriter which of the following capability(facility) is available as method in BufferedWriter.

1. **Writing data to the file.**
2. **Closing the writer.**
3. **Flush the writer.**
4. **Inserting newline character.**

Ans : 4

Example:

```
import java.io.*;
classBufferedWriterDemo
{
      public static void main(String[] args)throws IOException
      {
            FileWriterfw=new FileWriter("cricket.txt");
            BufferedWriterbw=new BufferedWriter(fw);
            bw.write(100);
            bw.newLine();
            char[] ch={'a','b','c','d'};
            bw.write(ch);
            bw.newLine();
            bw.write("SaiCharan");
            bw.newLine();
            bw.write("software solutions");
```

```
            bw.flush();
            bw.close();
            }
}
Output:
d
abcd
SaiCharan
software solutions
```

*Note :***When ever we are closing BufferedWriter automatically underlying writer will be closed and we are not close explicitly.**

BufferedReader:

This is the most enhanced(better) Reader to read character data from the file.

Constructors:

BufferedReaderbr=new BufferedReader(Reader r);
BufferedReaderbr=new BufferedReader(Reader r,intbuffersize);

Note:BufferedReadercan not communicate directly with the File it should communicate via some Reader object.
The main advantage of BufferedReader over FileReader is we can read data line by line instead of character by character.

Methods:

1. **int read();**
2. **int read(char[] ch);**
3. **String readLine();**
   It attempts to read next line and return it , from the File. if the next line is not available then this method returns null.
4. **void close();**

```
Example:
import java.io.*;
classBufferedReaderDemo
{
      public static void main(String[] args)throws IOException
      {
            FileReaderfr=new FileReader("cricket.txt");
            BufferedReaderbr=new BufferedReader(fr);
            String line=br.readLine();
            while(line!=null)
            {
                  System.out.println(line);
                  line=br.readLine();
            }
            br.close();
```

```
                }
}
```

Note:

- **Whenever we are closing BufferedReader automatically underlying FileReader will be closed it is not required to close explicitly.**
- **Even this rule is applicable for BufferedWriter also.**

PrintWriter:

- **This is the most enhanced Writer to write text data to the file.**
- **By using FileWriter and BufferedWriter we can write only character data to the File but by using PrintWriter we can write any type of data to the File.**

Constructors:

PrintWriter pw=new PrintWriter(String name);
PrintWriter pw=new PrintWriter(File f);
PrintWriter pw=new PrintWriter(Writer w);

PrintWriter can communicate either directly to the File or via some Writer object also.

Methods:

1. **write(intch);**
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);**

```
Example:
import java.io.*;
classPrintWriterDemo {
public static void main(String[] args)throws IOException
      {
            FileWriterfw=new FileWriter("cricket.txt");
            PrintWriter out=new PrintWriter(fw);
            out.write(100);
            out.println(100);
            out.println(true);
            out.println('c');
            out.println("SaiCharan");
            out.flush();
            out.close();
      }
}
Output:
d100
true
c
SaiCharan
```

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) "100" value will be added directly to the File.

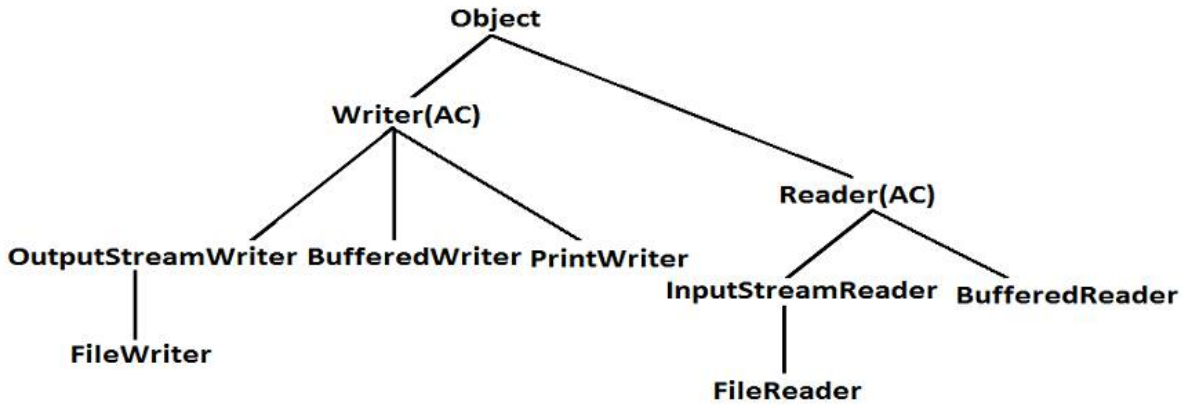
Note 1:

1. **The most enhanced Reader to read character data from the File is BufferedReader.**
2. **The most enhanced Writer to write character data to the File is PrintWriter.**

Note 2:

1. **In general we can use Readers and Writers to handle character data. Where as we can use InputStreams and OutputStreams to handle binary data(like images, audio files, video files etc).**
2. **We can use OutputStream to write binary data to the File and we can use InputStream to read binary data from the File.**

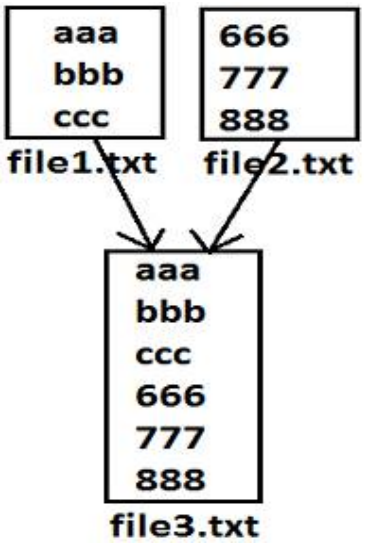
Diagram:



Requirement:

Write a program to perform File merge(combine) operation.

Diagram:

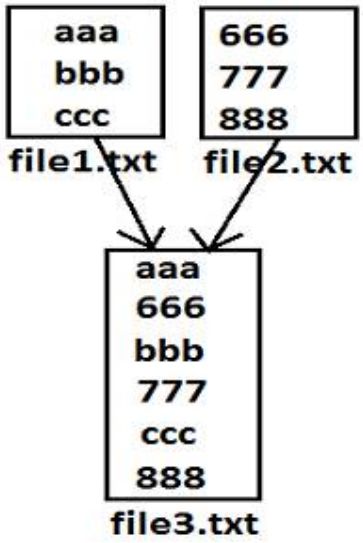



```
Program:
import java.io.*;
class FileWriterDemo1
{
public static void main(String[] args)throws IOException {
  PrintWriter pw=new PrintWriter("file3.txt");
BufferedReaderbr=new BufferedReader(new
FileReader("file1.txt"));
  String line=br.readLine();
while(line!=null)
```

```
  {
pw.println(line);
line=br.readLine();
   }
br=new BufferedReader(new FileReader("file2.txt"));    //reuse
line=br.readLine();
while(line!=null)
     {
pw.println(line);
line=br.readLine();
      }
      pw.flush();
      br.close();
      pw.close();
      }
}
```

***Requirement*: Write a program to perform file merge operation where merging should be performed line by line alternatively.**

Diagram:




```
Program:
import java.io.*;
class FileWriterDemo1 {
public static void main(String[] args)throws IOException {
PrintWriter pw=new PrintWriter("file3.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();
            br1.close();
            br2.close();
            pw.close();
  }
}
```

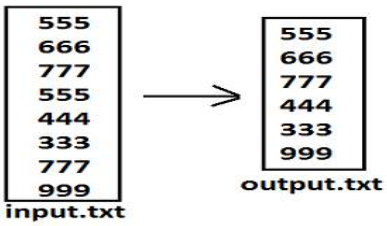
***Requirement*: Write a program to merge data from all files present in a folder into a new file**

Program:

```
import java.io.*;
classTotalFileMerge {
public static void main(String[] args)throws IOException {
PrintWriter pw=new PrintWriter("output.txt");
      File f=new File("E:\\xyz");
      String[] s=f.list();
      for(String s1=s) {
BufferedReader br1=new BufferedReader(new File(f,s1));
    String line=br.readLine();
while(line!=null)
      {
      pw.println(line);
      line=br.readLine();
      }
      }
      pw.flush();
  }
}
```

***Requirement:* Write a program to delete duplicate numbers from the file.**

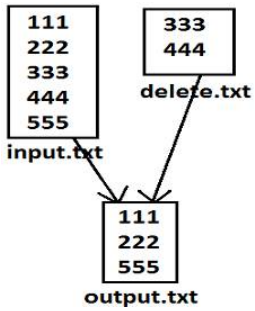
Diagram:




```
Program:
import java.io.*;
class FileWriterDemo1
{
public static void main(String[] args)throws IOException
  {
BufferedReader br1=new BufferedReader(new
FileReader("input.txt"));
PrintWriter out=new PrintWriter("output.txt");
   String target=br1.readLine();
while(target!=null)
   {
boolean available=false;
BufferedReader br2=new BufferedReader(new
FileReader("output.txt"));
       String line=br2.readLine();
      while(line!=null)
      {
      if(target.equals(line))
          {
            available=true;
            break;
          }
      line=br2.readLine();
      }
      if(available==false)
      {
      out.println(target);
      out.flush();
      }
      target=br1.readLine();
      }
   }
}
```

Requirement: **write a program to perform file extraction operation.**

Diagram:




```
Program:
import java.io.*;
class FileWriterDemo1
{
public static void main(String[] args)throws IOException
   {
BufferedReader br1=new BufferedReader(new
FileReader("input.txt"));
PrintWriter pw=new PrintWriter("output.txt");
   String line=br1.readLine();
while(line!=null)


 {
boolean available=false;
BufferedReader br2=new BufferedReader(new
FileReader("delete.txt"));
     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();
   }
}
```