**Java.IO Package**

**Agenda:**

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

**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

}

}

* 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.

1. **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.

1. **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 bhaskar123 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("bhaskar123");

f1.mkdir();

File f2=new File("bhaskar123","abc.txt");

f2.createNewFile();

}

}

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

**Program:**

import java.io.\*;

class FileDemo

{

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

{

File f=new File("c:\\bhaskar","demo.txt");

f.createNewFile();

}

}

**Import methods of file class:**

1. **boolean exists();**

* Returns true if the physical file or directory available.

1. **boolean createNewFile();**

* This method 1st checks whether the physical file is already available or not if it is already available then this method simply returns false. If this file is not already available then it will create a new file and returns true

1. **boolean mkdir();**
2. **boolean isFile();**

* Returns true if the File object represents a physical file.

1. **boolean isDirectory();**
2. **String[] list();**

* It returns the names of all files and subdirectories present in the specified directory.

1. **long length();**

* Returns the no of characters present in the file.

1. **boolean delete();**

* To delete a file or directory.

**FileWriter:**

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

**Constructors:**

**FileWriter fw=new FileWriter(String name);**

**FileWriter fw=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.

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

**FileWriter fw=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(int ch);**

* To write a single character to the file.

1. **write(char[] ch);**

* To write an array of characters to the file.

1. **write(String s);**

* To write a String to the file.

1. **flush();**

* To give the guarantee the last character of the data also added to the file.

1. **close();**

* To close the stream.

**Example:**

import java.io.\*;

class FileWriterDemo

{

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

{

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

fw.write(98);//adding a single character

fw.write("haskar\nsoftware solutions");

fw.write("\n");

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

fw.write(ch);

fw.write("\n");

fw.flush();

fw.close();

}

}

**Output:**

Bhaskar

Software solutions

ABC

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

**Constructors:**

**FileReader fr=new FileReader(String name);**

**FileReader fr=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.

1. **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.

1. **void close();**

**Approach 1:**

import java.io.\*;

class FileReaderDemo

{

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

{

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

int i=fr.read();

while(i!=-1)

{

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

i=fr.read();

}

}

}

**Output:**

Bhaskar

Software solutions

ABC

**Approach 2:**

import java.io.\*;

class FileReaderDemo

{

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

{

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

FileReader fr=new FileReader(f);

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

fr.read(ch);

for(char ch1:ch)

{

System.out.print(ch1);

}

}

}

**Output:**

VBR

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 which is not convent 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:**

BufferedWriter bw=new BufferedWriter(writer w);

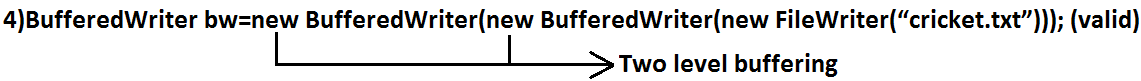
BufferedWriter bw=new BufferedWriter(writer w,int buffersize);

**Note:**

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

**Which of the following declarations are valid?**

1. **BufferedWriter bw=new BufferedWriter(“cricket.txt”); (invalid)**
2. **BufferedWriter bw=new BufferedWriter (new File(“cricket.txt”)); (invalid)**
3. **BufferedWriter bw=new BufferedWriter (new FileWriter(“cricket.txt”)); (valid)**



**Methods:**

1. **write(int ch);**
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.(x)
2. Closing the writer.(x)
3. Flush the writer.(x)
4. Inserting newline character. ()

**Example:**

import java.io.\*;

class BufferedWriterDemo

{

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

{

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

BufferedWriter bw=new BufferedWriter(fw);

bw.write(100);

bw.newLine();

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

bw.write(ch);

bw.newLine();

bw.write("bhaskar");

bw.newLine();

bw.write("software solutions");

bw.flush();

bw.close();

}

}

**Output:**

d

abcd

bhaskar

software solutions

**BufferedReader:**

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

**Constructors:**

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

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

**Note:**

* BufferedReader can 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 and return next line from the File. if the next line is not available then this method returns null.

1. **void close();**

**Example:**

import java.io.\*;

class BufferedReaderDemo

{

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

{

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

BufferedReader br=new BufferedReader(fr);

String line=br.readLine();

while(line!=null)

{

System.out.println(line);

line=br.readLine();

}

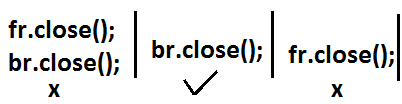
br.close();

}

}

**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:**

write(int ch);

write (char[] ch);

write(String s);

flush();

close();

print(char ch);

print (int i);

print (double d);

print (boolean b);

print (String s);

println(char ch);

println (int i);

println(double d);

println(boolean b);

println(String s);

**Example:**

import java.io.\*;

class PrintWriterDemo

{

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

{

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

PrintWriter out=new PrintWriter(fw);

out.write(100);

out.println(100);

out.println(true);

out.println('c');

out.println("bhaskar");

out.flush();

out.close();

}

}

**Output:**

d100

true

c

bhaskar

**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.

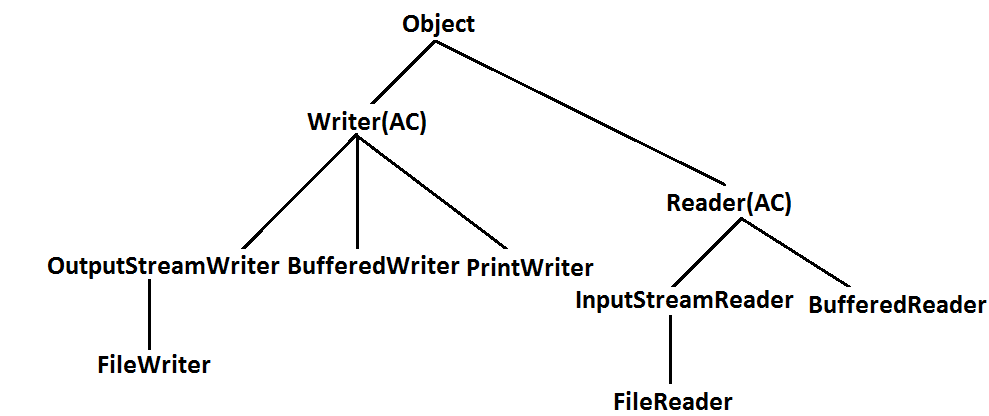
**Note 1:**

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:**

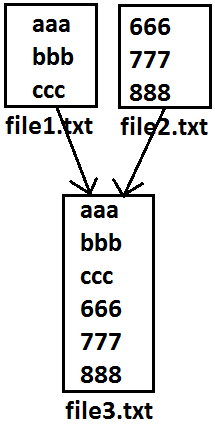
1. In general we can use Readers and Writers to handle character data. Where as we can use InputStreams and OutputStreams to handle binary data(like images, audio files, video files etc).
2. We can use OutputStream to write binary data to the File and we can use InputStream to read binary data from the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

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")); **reuse**

line=br.readLine();

while(line!=null)

{

pw.println(line);

line=br.readLine();

}

pw.flush();

br.close();

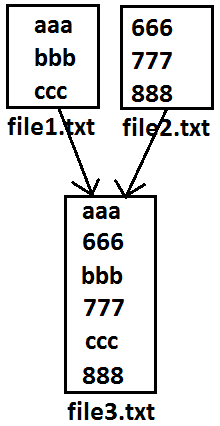
pw.close();

}

}

**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();

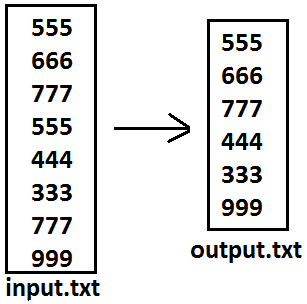
pw.close();

}

}

**Requirement:** Write a program to 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();

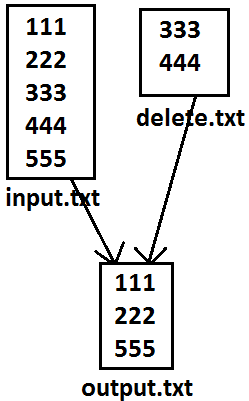
}

}

}

**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();

}

}