File I/O

1.. 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.

```java
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.

```java
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.

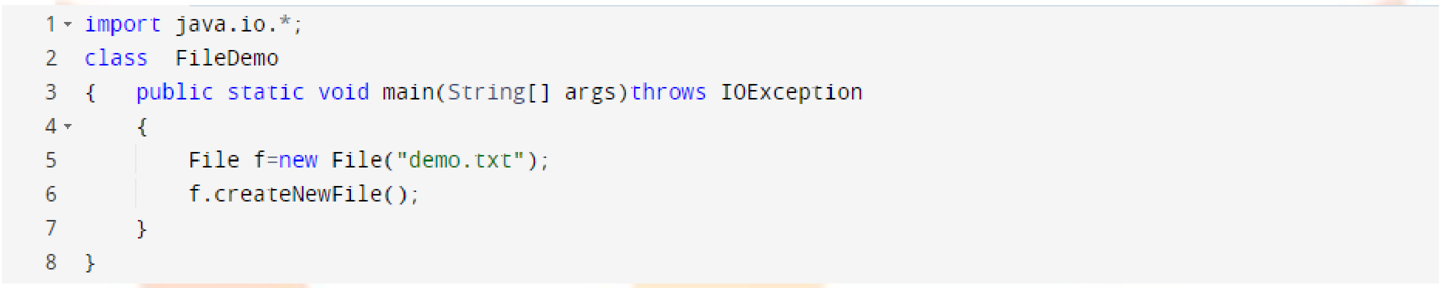
1.1 File Class Constructors

- **File f=new File(String name);**
  Creates a java File object that represents name of the file or directory in current working directory.

- **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.

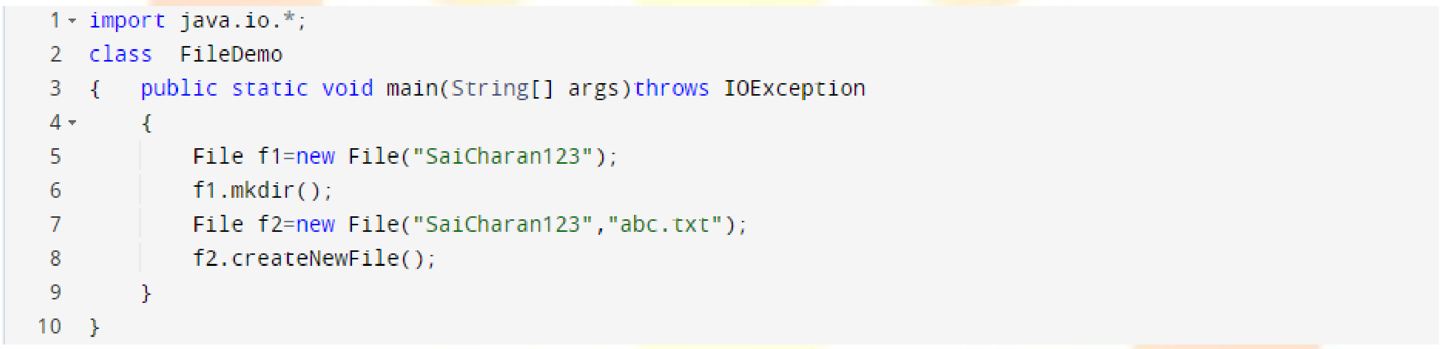
- **File f=new File(File subdir, String name);**

1.2 Write code to create a file named with demo.txt in current working directory.



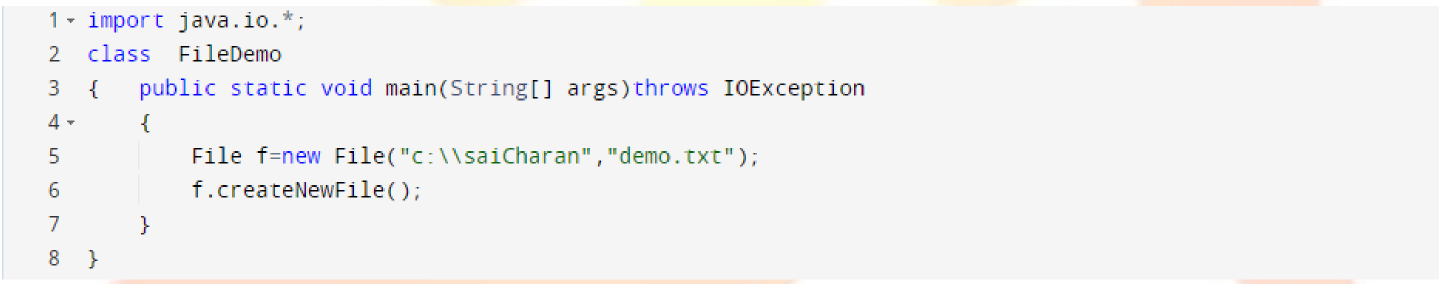
```java
import java.io.*;
class  FileDemo
{   public static void main(String[] args)throws IOException
    {
        File f=new File("demo.txt");
        f.createNewFile();
    }
}
```

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



```java
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();
    }
}
```

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



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

1.5 Methods present in File Class

- **boolean exists();**
  Returns true if the physical file or directory available.

- **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 without creating any physical file. If this file is not already available then it will create a new file and returns true

- **boolean mkdir();**
  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

- **boolean isFile();**
  Returns true if the File object represents a physical file.

- **boolean isDirectory();**
  Returns true if the File object represents a directory.

- **String[] list();**
  It returns the names of all files and subdirectories present in the specified directory.

- **long length();**
  Returns the number of characters present in the file.

- **boolean delete();**
  To delete a file or directory.

Example:

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

```java
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);
    }
}
```

Write a program to display only file names

```java
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);
    }
}
```

Write a program to display only directory names

```java
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);
    }
}
```

2.. FileWriter

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

2.1 Constructors of FileWriter Class

- FileWriter fw = new FileWriter(String name);
- FileWriter fw = new FileWriter(File f);

  The above two constructors meant for overriding. Instead of overriding if we want append operation then we should go for the following two 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.

2.2 Methods of FileWriter Class

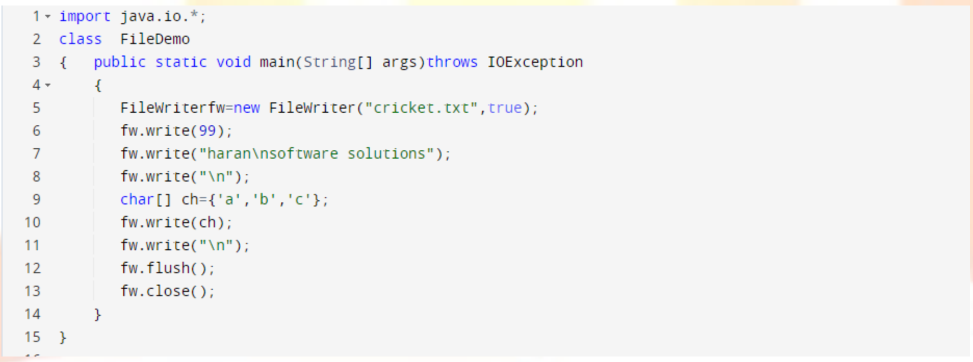
- **write(int ch);**
  To write a single character to the file.

- **write(char[] ch);**
  To write an array of characters to the file.

- **write(String s);**
  To write a String to the file.

- **flush();**
  To give the guarantee the total data include last character also written to the file.

- **close();**
  To close the stream.



```java
import java.io.*;
class  FileDemo
{   public static void main(String[] args)throws IOException
    {
        FileWriterfw=new FileWriter("cricket.txt",true);
        fw.write(99);
        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 ('\n') manually, which is difficult to the programmer And even line separator varying from system to system.

3.. FileReader

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

3.1 Constructors of FileReader

- FileReader fr=new FileReader(String name);
- FileReader fr=new FileReader (File f);

3.2 Methods of FileReader

- **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.

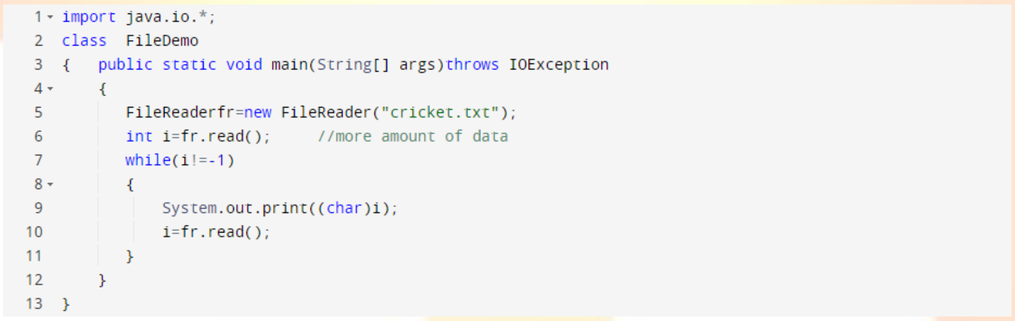
  int i = fr.read();
  System.out.println((char)i);

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

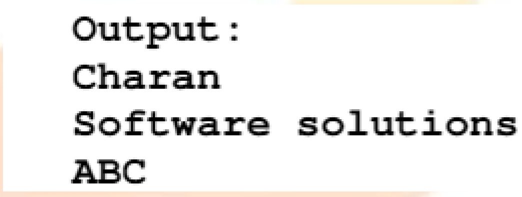
- **int read(char[] ch);**
  It attempts to read enough characters from the file into char[] array and returns the number of characters copied from the file into char[] array.

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

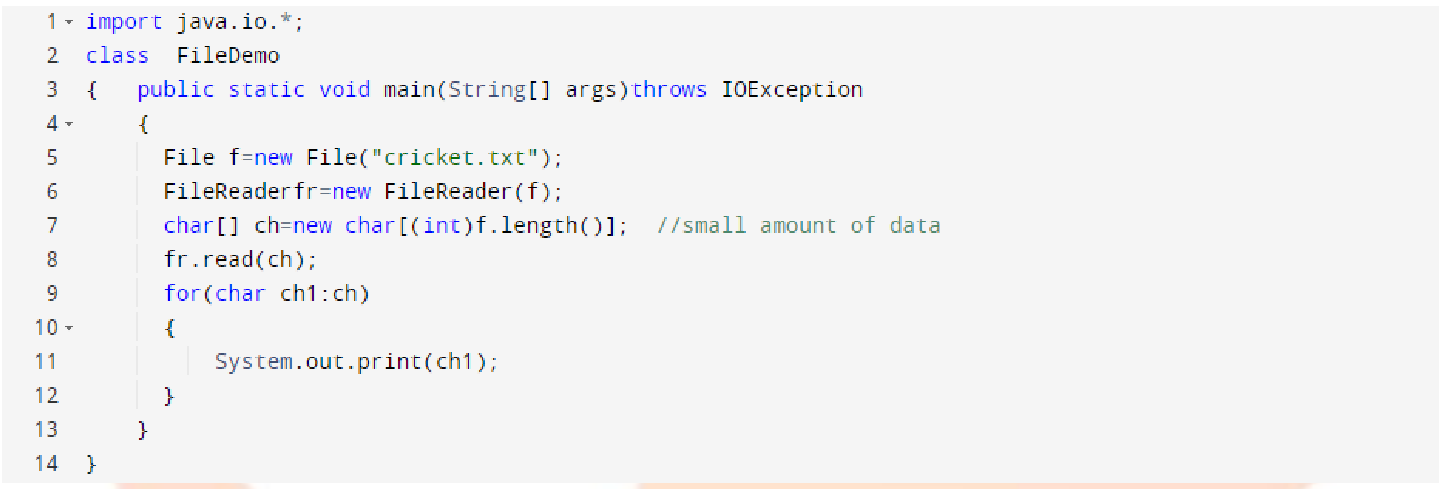
- **void close();**



```
import java.io.*;
class  FileDemo
{   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
```



```java
import java.io.*;
class  FileDemo
{   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);
      }
    }
}
```

Usage of FileWriter and FileReader is not recommended because while writing data by **FileWriter** compulsory we should insert line separator(\n) manually which is a bigger headache to the programmer and while reading data by **FileReader** we have to read character by character instead of line by line which is not convenient to the programmer. So, to overcome these limitations we should go for **BufferedWriter** and **BufferedReader** concepts.

4.. BufferedWriter and BufferedReader

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

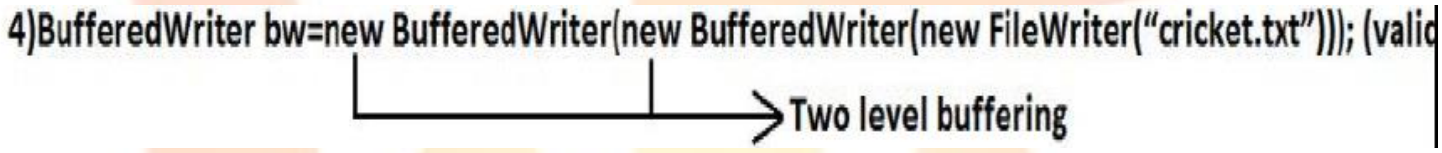
4.1 Constructors of BufferedWriter Class

- 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?

- BufferedWriter bw=new BufferedWriter("cricket.txt"); (invalid)
- BufferedWriter bw=new BufferedWriter (new File("cricket.txt")); (invalid)
- BufferedWriter bw=new BufferedWriter (new FileWriter("cricket.txt")); (valid)



4)BufferedWriter bw=new BufferedWriter(new BufferedWriter(new FileWriter("cricket.txt"))); (vali

4.2 Methods of BufferedWriter Class

- write(int ch);
- write(char[] ch);
- write(String s);
- flush();
- close();
- newline(); Inserting a new line character to the file.

```java
import java.io.*;
class  FileDemo
{   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: Whenever we are closing BufferedWriter automatically underlying writer will be closed and we are not close explicitly.

4.3 BufferedReader (Constructors and Methods)

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 cannot 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:

- int read();
- int read(char[] ch);
- 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.

- void close();

```java
import java.io.*;
class  FileDemo
{   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.

5.. PrintWriter Class

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.

5.1 Constructors of PrintWriter Class

- 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.

5.2 Methods of PrintWriter Class

- 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);

```
import java.io;
class FileDemo {
    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("SaiCharan");
        out.flush();
        out.close();
    }
}
```

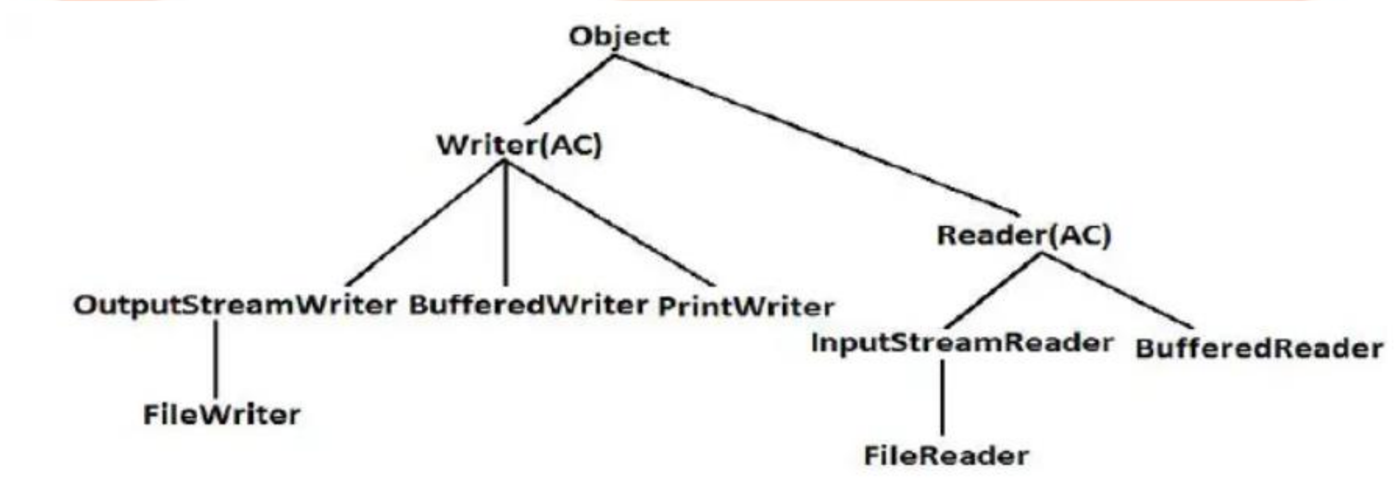
```
Output:
d100
true
c
SaiCharan
```

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: The most enhanced Reader to read character data from the File is BufferedReader and The most enhanced Writer to write character data to the File is PrintWriter.

Note: Generally, we can use Readers and Writers to handle character data. Whereas we can use **InputStreams** and **OutputStreams** to handle binary data (like images, audio files, video files etc.). We can use **OutputStream** to write binary data to the File and we can use **InputStream** to read binary data from the **File**.

Diagram


Object
Writer(AC)
Reader(AC)
OutputStreamWriter
BufferedWriter
PrintWriter
FileWriter
InputStreamReader
BufferedReader
FileReader