File Class Methods

```
import java.io.File;
import java.io.IOException;

public class f2
{
    public static void main(String[] args) throws IOException
    {
        //creating f1 directory
        File f1=new File("ankur19");
        System.out.println(f1.exists());

        f1.mkdir();
        System.out.println(f1.exists());

        //creating f2 file in f1 directory
        File f2=new File("ankur19","abc2.txt");
        f2.createNewFile();

        //creating a file in "e: drive \\ xyz folder"
        File f3=new File("E:\\Ankur Patel","vip12.txt");
        f3.createNewFile();
/*_____________________________________________________________________*/
        // 1 // public String getPath()
        //return The string form of this abstract pathname

        System.out.println("File path="+f2.getPath());
        //File path=ankur19\abc2.txt
/*_____________________________________________________________________*/

        // 2 // public String getAbsolutePath()
        //returns The absolute pathname string denoting the same file or
        //directory as this abstract pathname

        System.out.println("Absolute path="+f2.getAbsolutePath());
     //Absolute path=E:\Ankur Patel\SEM-2\JAVA- SYLLABUS\JAVA SUbject
Notes\FileIOProg\ankur19\abc2.txt

/*_____________________________________________________________________*/

        // 3 // public String getParent()
        //return  The pathname string of the parent directory named by this
        //abstract pathname, or null if this pathname does not name a parent
```

```
        System.out.println("Parent="+f2.getParent());
        //Parent=ankur19
/*_____________________________________________________________________________*/

        // 4 //  public File getParentFile()
        //return  The abstract pathname of the parent directory named by this
        //abstract pathname, or null if this pathname does not name a parent


        System.out.println("Parent File="+f2.getParentFile());
        //Parent File=ankur19
/*_____________________________________________________________________________*/

        // 5 // public String getName()
        //returns  The name of the file or directory denoted by this abstract
        //pathname, or the empty string if this pathname's name sequence is empty

        System.out.println("Name="+f2.getName());
        //Name=abc2.txt
/*_____________________________________________________________________________*/

        // 6 // public boolean isFile()
        //returns  true< if and only if the file denoted by this
        //abstract pathname exists andis a normal file; false otherwise

        System.out.println("is file ?"+f2.isFile());
        //is file ?true
/*_____________________________________________________________________________*/

        // 7 // public boolean isDirectory()
        //returns true if and only if the file denoted by this
        //abstract pathname exists and is a directory;false otherwise

        System.out.println("is Directory ?"+f2.isDirectory());
        //is Directory ?false
/*_____________________________________________________________________________*/

        // 8 // public boolean canRead()
        //returns true if and only if the file specified by this abstract pathname
        // exists and can be read by the application; false otherwise

        System.out.println(f2.canRead());
        //true
/*_____________________________________________________________________________*/
```

```
        // 9 // public boolean canWrite()
        //returns true if and only if the file system actually contains a file
        //denoted by this abstract pathname and the application is allowed to write
        //to the file; false otherwise.

        System.out.println(f2.canWrite());
        //true
/*_____________________________________________________________________________*/

        // 10 // public boolean canExecute()
        //returns true if and only if the abstract pathname exists and the
        //application is allowed to execute the file

        System.out.println(f2.canExecute());
        //true
/*_____________________________________________________________________________*/

        // 11 // public long length()
        //return The length, in bytes, of the file denoted by this abstract
        //pathname, or 0L if the file does not exist.

        System.out.println(f2.length());
        //0 (when the file is empty)
        //5 (written hello in the file)
/*_____________________________________________________________________________*/

        // 12 // public boolean equals(Object obj)
        //return true if and only if the objects are the same;false otherwise

        File f4=new File("ankur19","abc2.txt");
        System.out.println(f2.equals(f4));//false

        f4.createNewFile();
        System.out.println(f2.equals(f4));//true
/*_____________________________________________________________________________*/

  // 13 //  public File[] listFiles()
        //return  An array of abstract pathnames denoting the files and directories
        //in the directory denoted by this abstract pathname.The array will be
        //empty if the directory is empty.  Returns null if this abstract pathname
        //does not denote a directory
```

```
        File [] ff=f1.listFiles();

        for(File ff1: ff)
        {
            System.out.println("List of Files:"+ff1);
        }
       //List of Files:ankur19\abc2.txt
/*_____________________________________________________________________________*/

        System.out.println("F2 File path="+f2.getPath());
        //F2 File path=ankur19\abc2.txt

        System.out.println("F4 File path="+f4.getPath());
        //F4 File path=ankur19\abc2.txt
    }
}
```

Practice prog:1:-Demonstrate a program to count a number of files and directories in a given directory

```
import java.io.File;

import java.io.IOException;

public class F3

{

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

    {

        File F1= new File("AP1490");

        F1.mkdir();

        File F2= new File("AP1490","VP07");

        F2.mkdir();

        File F3= new File("AP1490","DBP08");

        F3.mkdir();

        File F4= new File("AP1490","AP1.txt");

        F4.createNewFile();

        File F5= new File("AP1490","AP2.txt");

        F5.createNewFile();
```

```
        File F6= new File("AP1490","AP3.txt");
        F6.createNewFile();
        String [] s= F1.list();
        int count=0;
        for(String s1:s)
        {
            count++;
            System.out.println("File or directory name:"+s1);
        }
        System.out.println("Total no of files and directories="+count);
    }
}
```

/*File or directory name:AP1.txt

File or directory name:AP2.txt

File or directory name:AP3.txt

File or directory name:DBP08

File or directory name:VP07

Total no of files and directories=5*/

Practice prog:2:-Demonstrate a program to count total no of files and total no of directories present in a particular directory

```
import java.io.File;
import java.io.IOException;
public class F3
{
    public static void main(String[] args) throws Exception
    {
        File F1= new File("AP1490");
        F1.mkdir();
        File F2= new File("AP1490","VP07");
```

```
        F2.mkdir();
        File F3= new File("AP1490","DBP08");
        F3.mkdir();
        File F4= new File("AP1490","AP1.txt");
        F4.createNewFile();
        File F5= new File("AP1490","AP2.txt");
        F5.createNewFile();
        File F6= new File("AP1490","AP3.txt");
        F6.createNewFile();
        String [] s= F1.list();
        int filecount=0;
        int directorycount=0;
        for(String s1:s)
        {
            File F=new File(F1,s1);
            if(F.isFile())
            {
                filecount++;
                System.out.println("File name:"+s1);
            }
            else
            {
                directorycount++;
                System.out.println("Directory name:"+s1);
            }
        }
        System.out.println("Total no of files ="+filecount);
        System.out.println("Total no of directories="+directorycount);
    }
}
```

/*File name:AP1.txt

File name:AP2.txt

File name:AP3.txt

Directory name:DBP08

Directory name:VP07

Total no of files =3

Total no of directories=2*/

Java FileOutputStream Class

- Java FileOutputStream is an output stream **used for writing data** to a file.
- If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

Java FileInputStream Class

- Java FileInputStream class obtains input bytes from a file. It is **used for reading byte-oriented data** (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

Sample Prog:

```
import java.io.*;

public class F6

{

    public static void main(String[] args) throws Exception
    {
        FileOutputStream fos=new FileOutputStream("hridu1.txt");
        fos.write(97);
        String s="nkur patel";
        byte [] b=s.getBytes();
        fos.write(b);
        fos.close();

        FileInputStream fis=new FileInputStream("hridu1.txt");
```

```
        int i=fis.read();
        while(i!=-1)
        {
            System.out.print((char)i);
            i=fis.read();
        }
    }
}
```