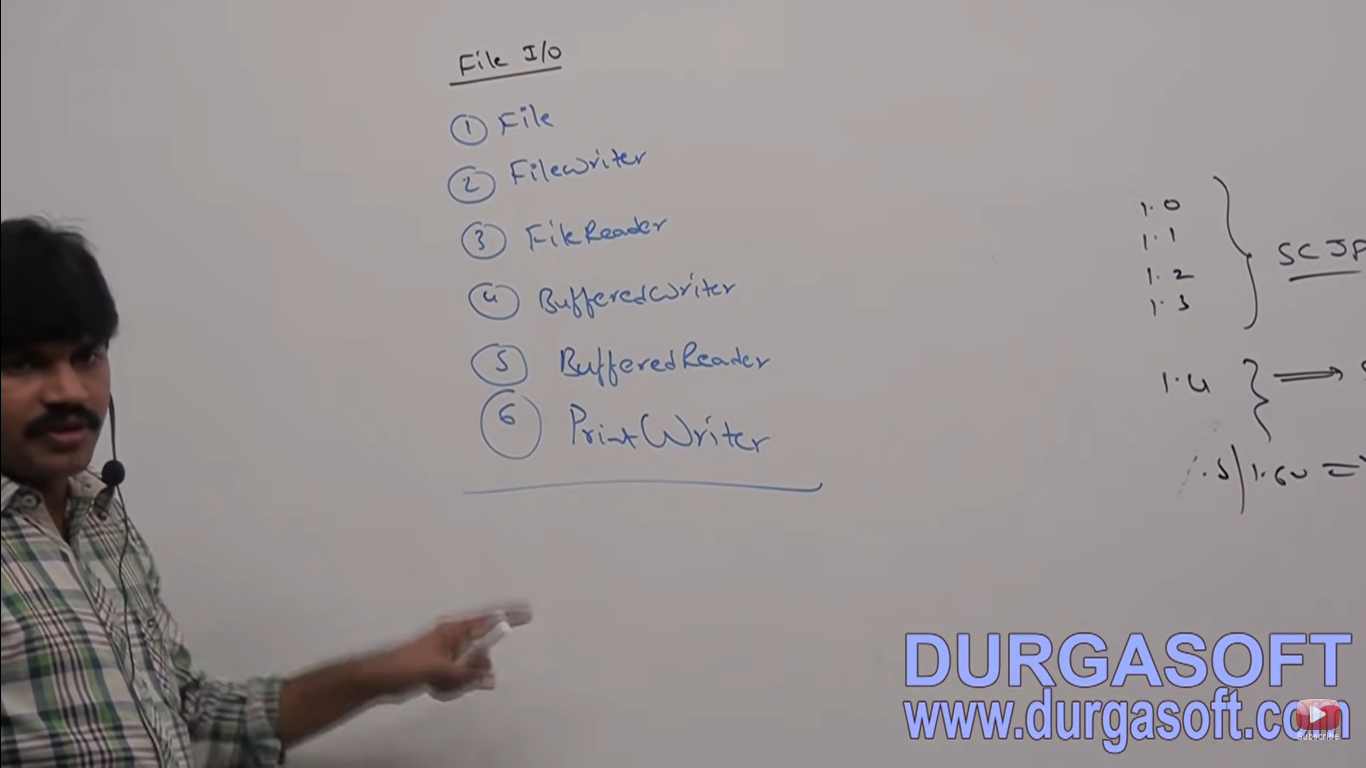
**File I/O**

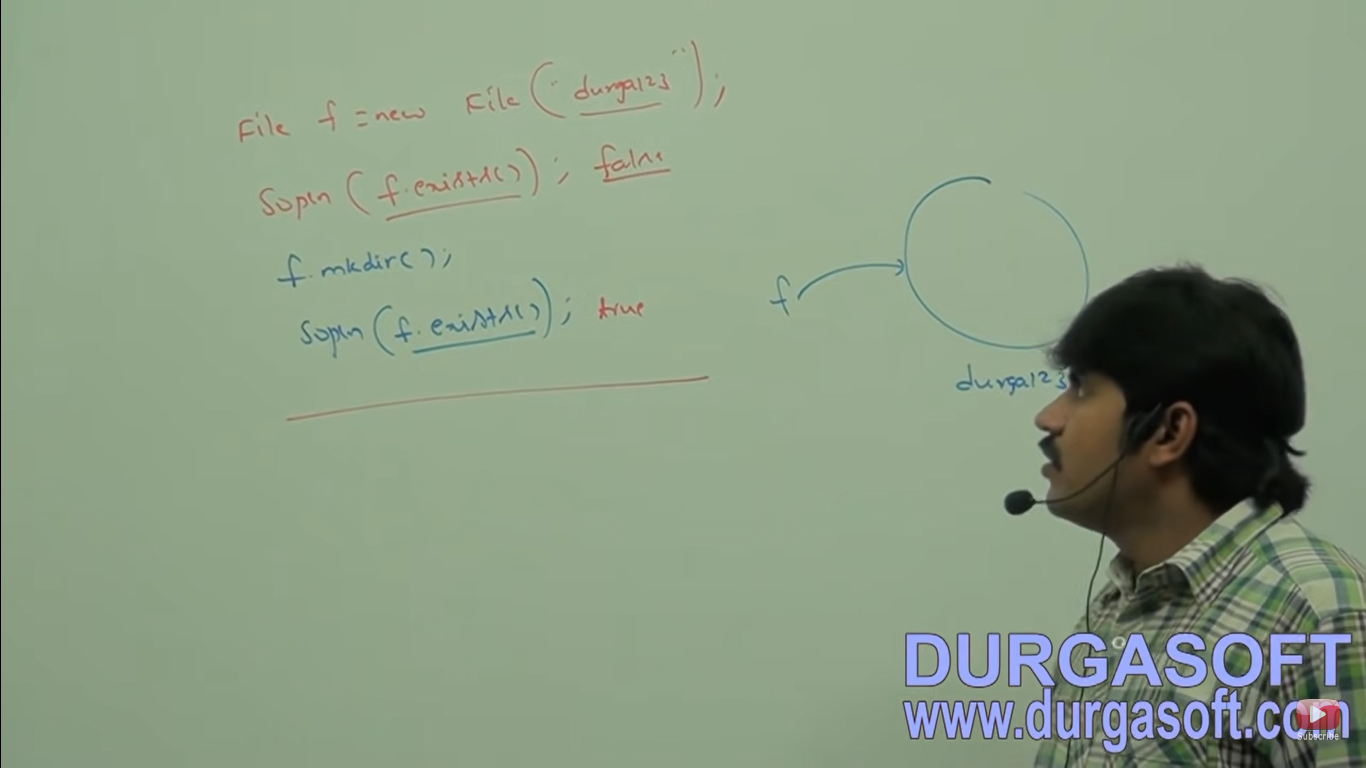




**File f = new File(“abc.txt”);**

This line won’t create any physical file, first it will check is there any physical file named with abc.txt is available or not. If it is available then f simply refers that file. If it is not available then we are just creating java file object to represent the name abc.txt

We can use Java file object to represent directory also



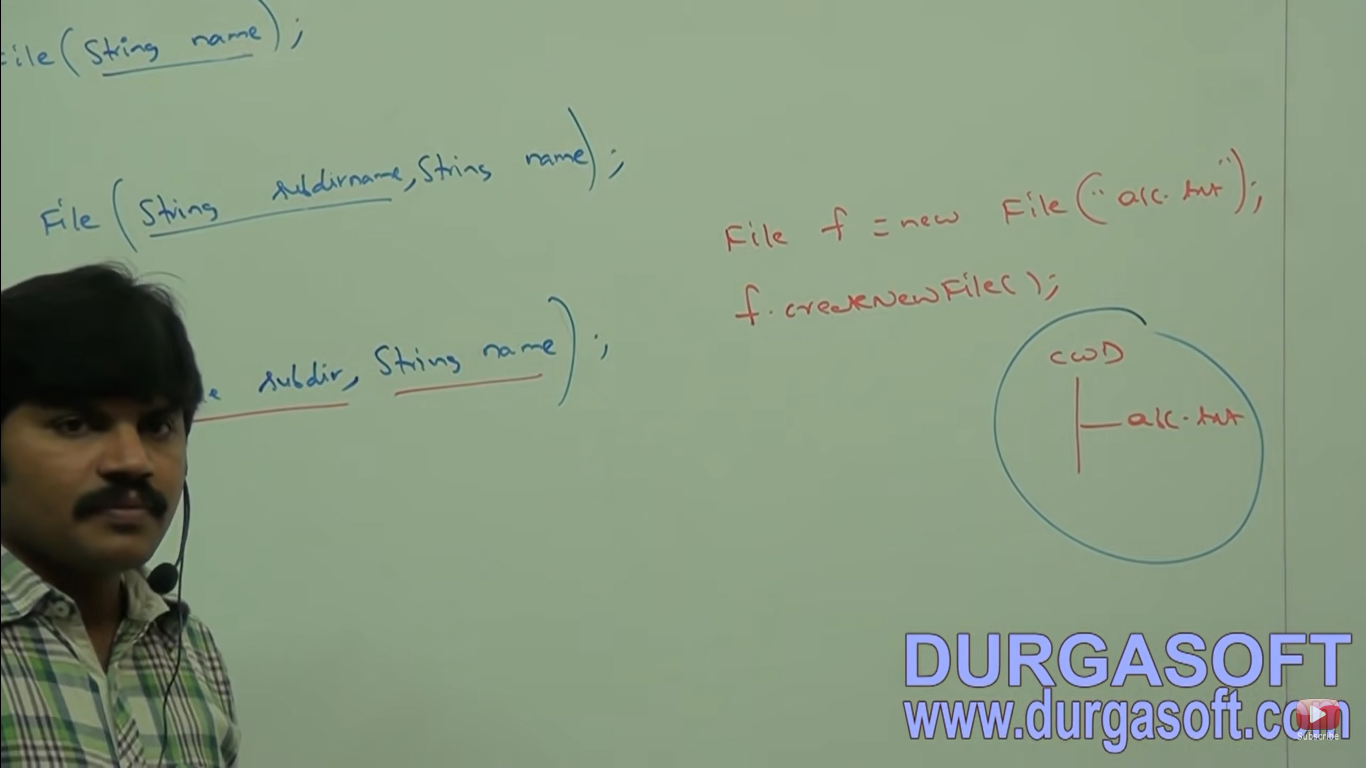
**Note – In Unix everything is treated as a file. Java File I/O concept is implemented based on Unix operating system. Hence Java file object can be used to represent both files and directories.**

**File Class Constructors**

1. File f = new File(String name);  **//parameters may be file name or directory name present in the current working directory**
2. File f = new File(String subdirName, String name)**; //First parameters represent the subdirectory name**
3. File f = new File(File subdir, String name); **//If first parameters already point by some file name**
4. Creates a java file object to represent name of the file or directory in current working directory.
5. Creates a java file object to represent name of the file or directory present in specified subdirectory.

**Write code to create a file named with abc.txt in current working directory**

File f = new File(“abc.txt”);



**Write code to create directory named with durga123 in current working directory and create a file named with demo.txt in that directory.**

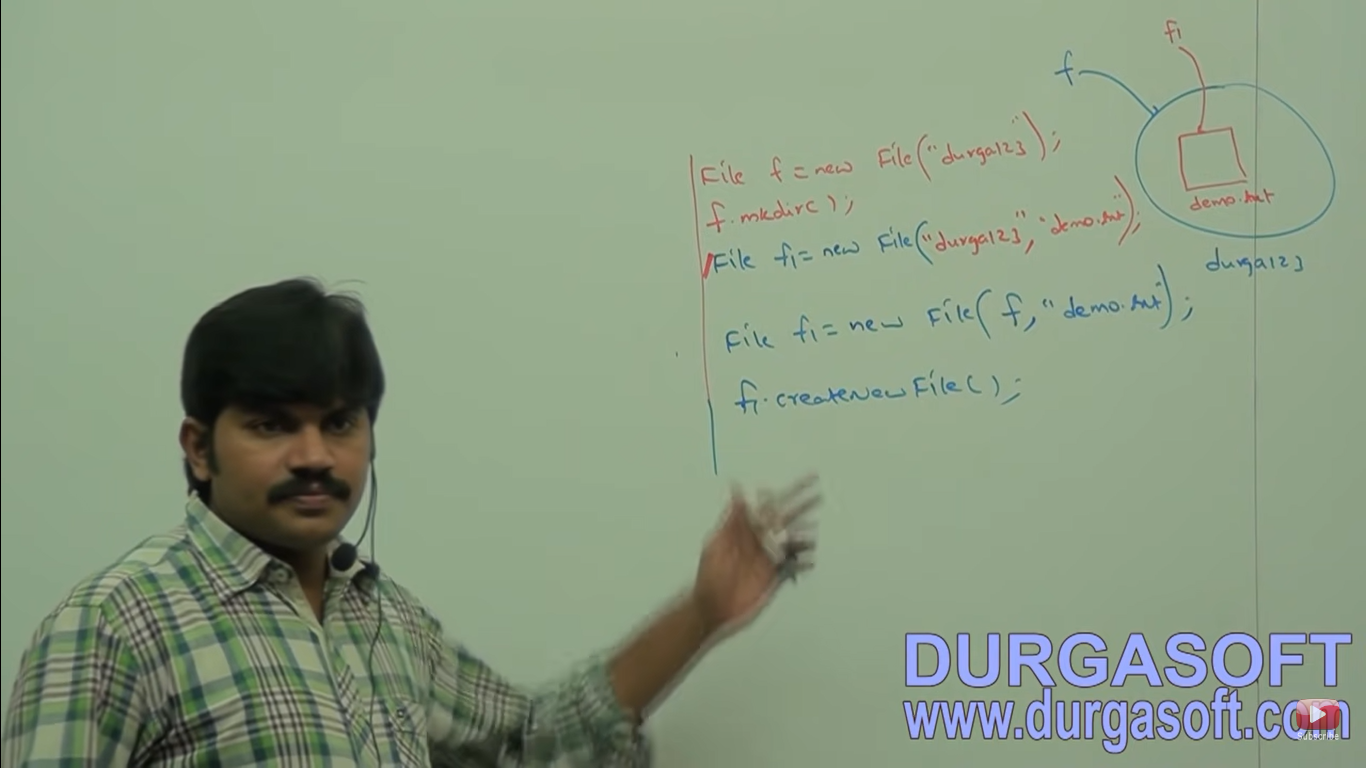
File f = new File(“durga123”);

f.mkdir();

File f1 = new File(“durga123”,”demo.txt”);

File f1 = new File(f,”demo.txt”);

F1.createNewFile();

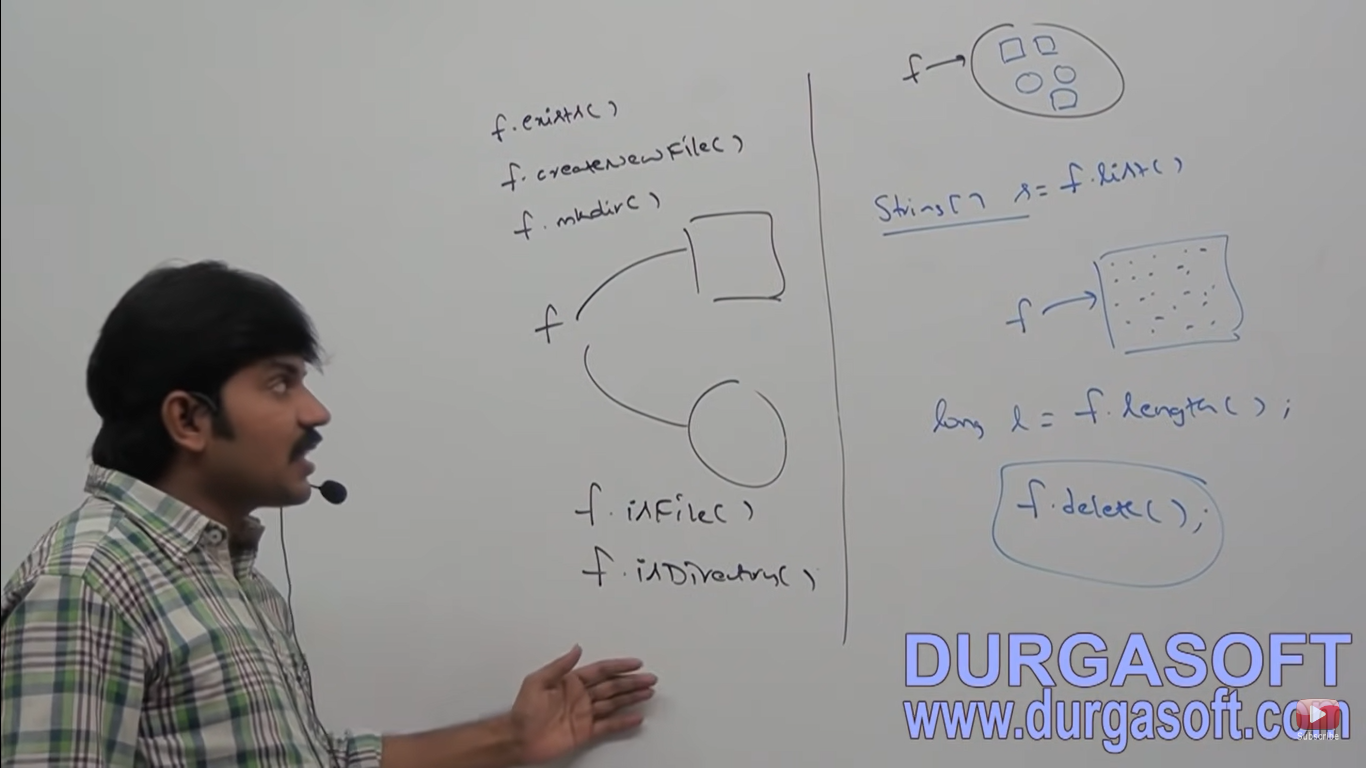


**Write code to create a file named with abc.txt in E:\xyz folder**

File f = new File(“E:\xyz”,”abc.txt”);

f.createNewFile();





**Important methods present in file class**

1. **boolean exists();**

returns true if the specified file or directory available

1. **boolean createNewFile();**

first this method will check whether the specified file is already available or not, if it is already available then this method returns false without creating any physical file. If the file is not already available then this method will create a new file and returns true.

1. **boolean mkdir();**

**Same as 2**

1. **boolean isFile();**

returns true if the specified file object pointing to physical file.

1. **boolean isDirectory();**

Return true if f pointing to directory

1. **String[] list()**

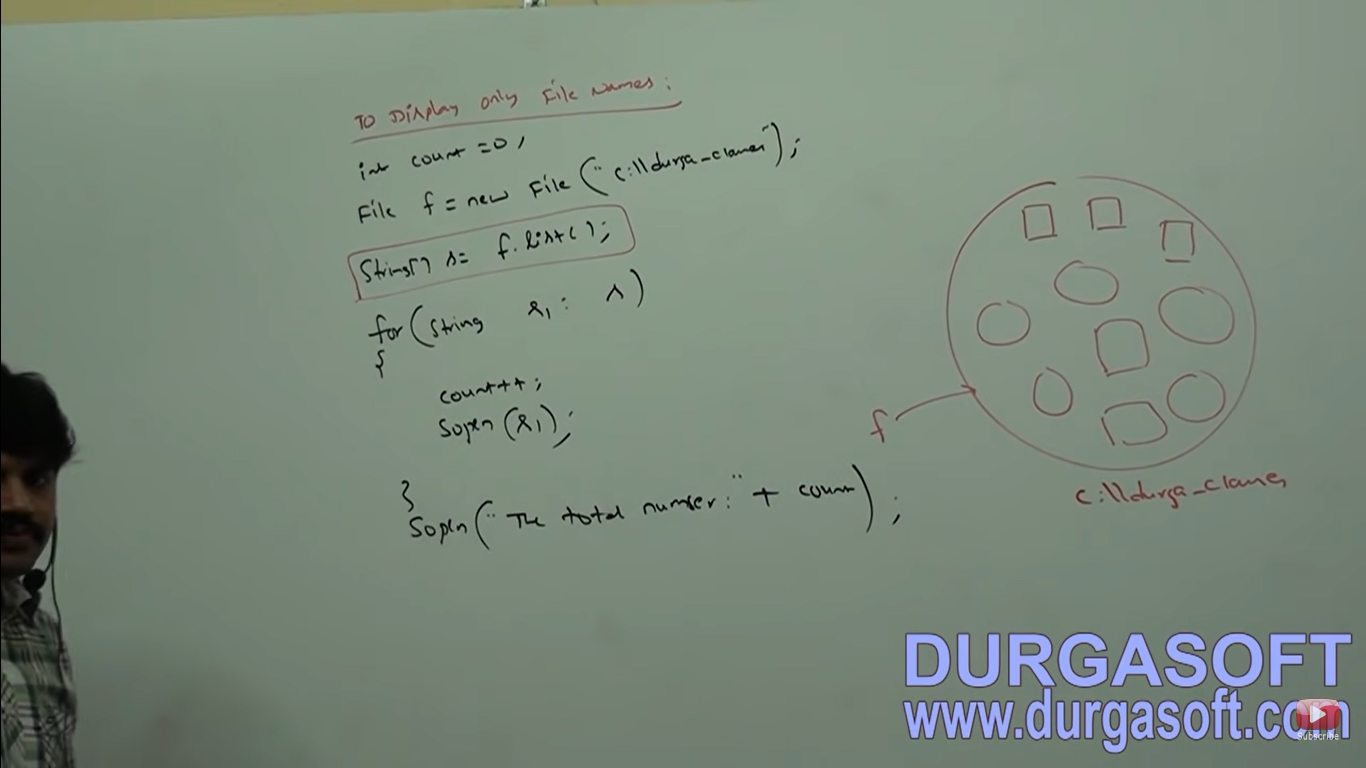
This method return the name of all file names and sub directories present in the specified directory

1. **long length()**

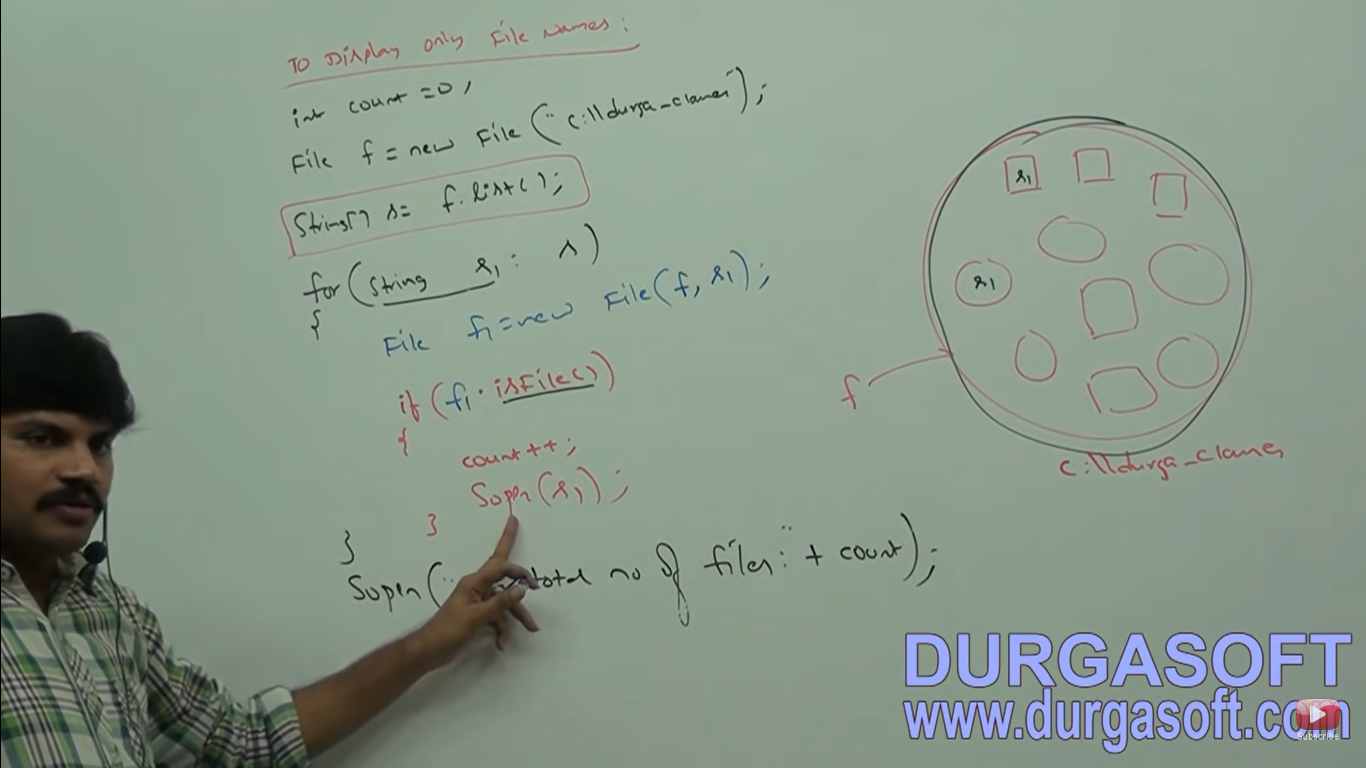
Returns the number of characters present in the specified file.

1. **boolean delete()**

**Write a program to display the names of all files and directories present in C:\DurgaClasses**



**Write a program to display only file names?**



**FileWriter**

We can use file writer to write character data to the file.

**Constructors**

1. **FileWriter fw = new FileWriter(String fileName); //It will override the file**
2. **FileWriter fw = new FileWriter(File f); //It will override the file**
3. **FileWriter fw = new FileWriter(String filename, boolean append); //This will append the data.**
4. **FileWriter fw = new FileWritter(File f, boolean append); //This will append the data in the file.**

1 ,2 – The above file writers meant for overriding of existing data instead of overriding if we want append operation then we have to create fileWriter by using 3,4.

**Note – If the specified file is not already available then all the above constructors will create that file**

**Methods**

1. **write(int ch) method 🡪** To write a single character . Eg fw.write(97) and fw.write(‘a’) is same
2. **write(char[] ch) 🡪** To write an array of characters to the file.
3. **write(String s) 🡪** To write string to the file.

**We have FW, BW, PW (File writer, Print writer, Buffered writer)**

1 – To write a single character

2 – To write an array of characters to the file.

3 – To write string to the file

1. **flush()** 🡪 To give the guarantee that total data including last character will be written to the file
2. **close() 🡪** To close the writer.
3. package JavaConceptsFileIO;
4. import java.io.FileWriter;
5. import java.io.IOException;
6. public class FileWriterEx {
7. public static void main(String[] args) throws IOException {
8. FileWriter fw = new FileWriter("abc11113131.txt",true);
9. fw.write(100);
10. fw.write("urga\nSoftwareSolutions");
11. fw.write("\n");
12. char[] ch1 = {'a','b','c'};
13. fw.write(ch1);
14. fw.flush();
15. fw.close();
16. }
17. }

In the above program file writer can perform overriding of existing data. Instead of overriding if we want append operation then we have to create FileWriter object as follows:

FileWriter fw = new FileWriter("abc11113131.txt",true);

**Note – The main problem with FW is we have to insert lines separator (\n) manually which is varied from system to system. It is difficulty to the programmer. We can solve this problem by using BW or PW.**

**FileReader**

We can use FileReader to read character data from the file

**Constructors:**

1. FileReader fr = new FileReader(String fName);
2. FileReader fr = new FileReader(File f);

**Methods:**

1. Int read() 🡪 returns the character (Unicode value)

int i = fr.read();

sop(i);//100

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

File f = new File(“abc.text”);

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

FileReader fr = new FileReader(f);

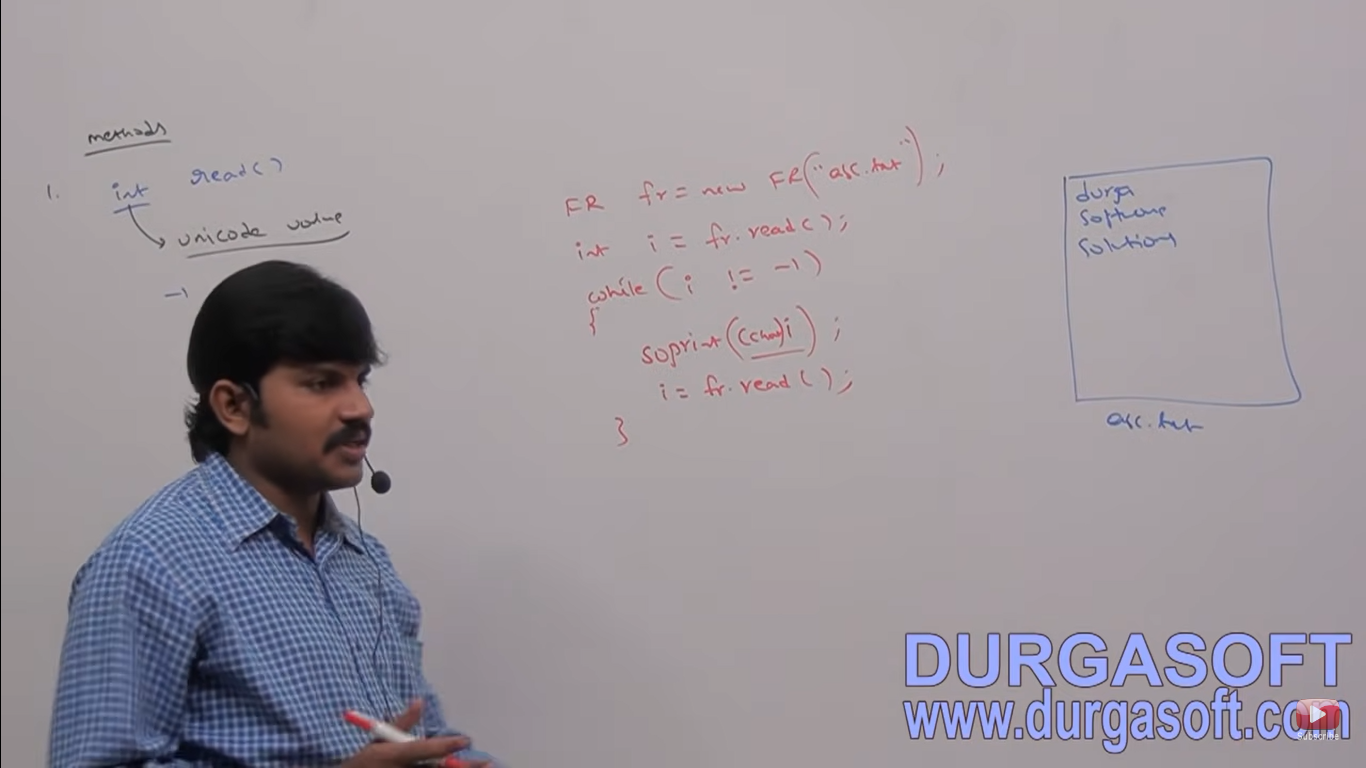
Fr.read(ch);

for(char ch1: ch){

System.out.println(ch1);

}

1. void close();



It attempts to read the next character from the file and returns it’s Unicode value. If the next character not available then this method returns -1. As this method returns Unicode value (int value), at the time of printing we have to perform typecasting.

**Note – By using File Reader we can read the data character by character which is not convenient to the programmer.**

**Usage of File Write and File Reader is not recommended because**

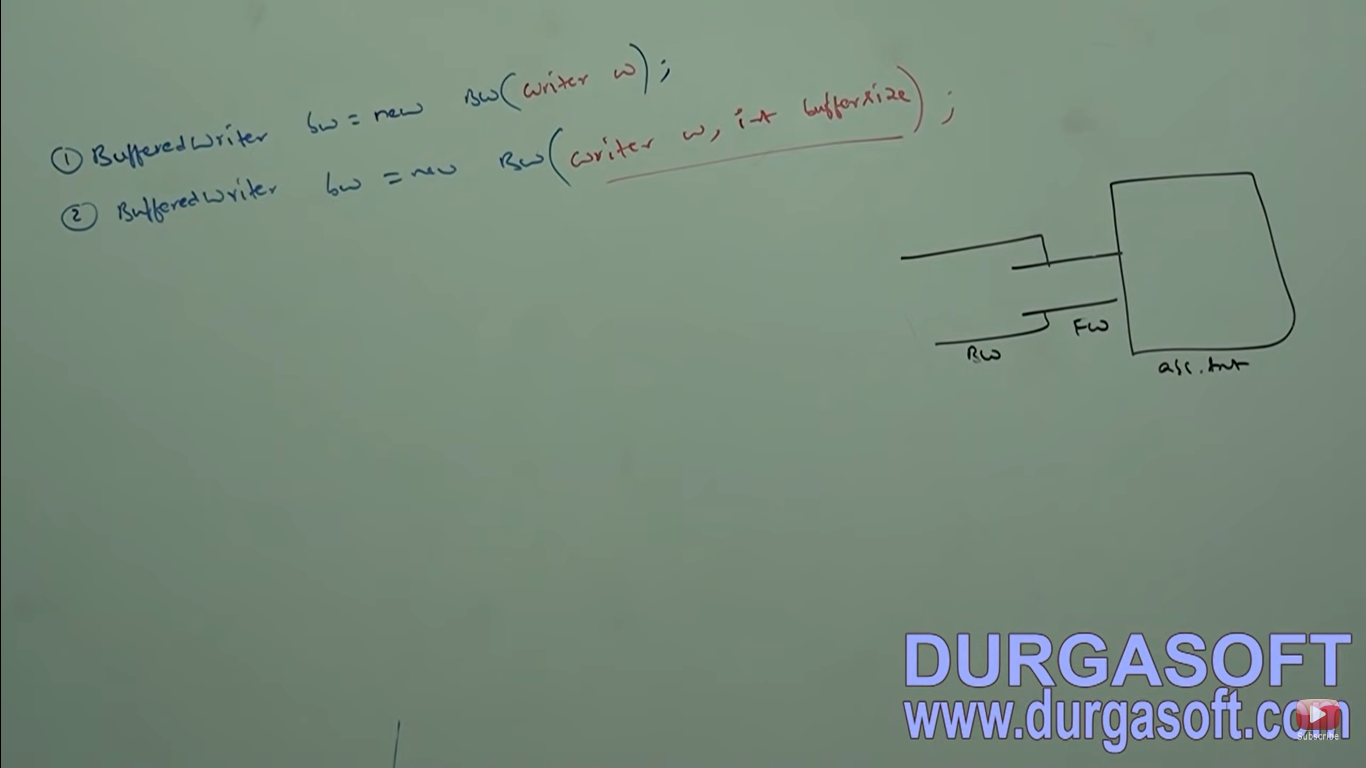
1. While writing data by FW we have to insert line separator(\n) manually which is varied from system to system it is difficult to the programmer.
2. By using File Reader we can read data character by character, which his not convenient to the programmer. To overcome these problems we should go for bufferedWriter and bufferedReader.

**Buffered Writer**

We can use buffered writer to write character data to the file.

**Constructors**

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



Note – BufferedWriter can’t communicate directly with the file. It can communicate via some writer object.

Q. Which of the following are valid?

1. BufferedWriter bw = new BufferedWriter(“abc.txt”);**//invalid**

**2.** BufferedWriter bw = new BufferedWriter(new File(“abc.txt”)); **//invalid**

3. BufferedWriter bw = new BufferedWriter(new FileWriter(“abc.txt”)); **//valid**

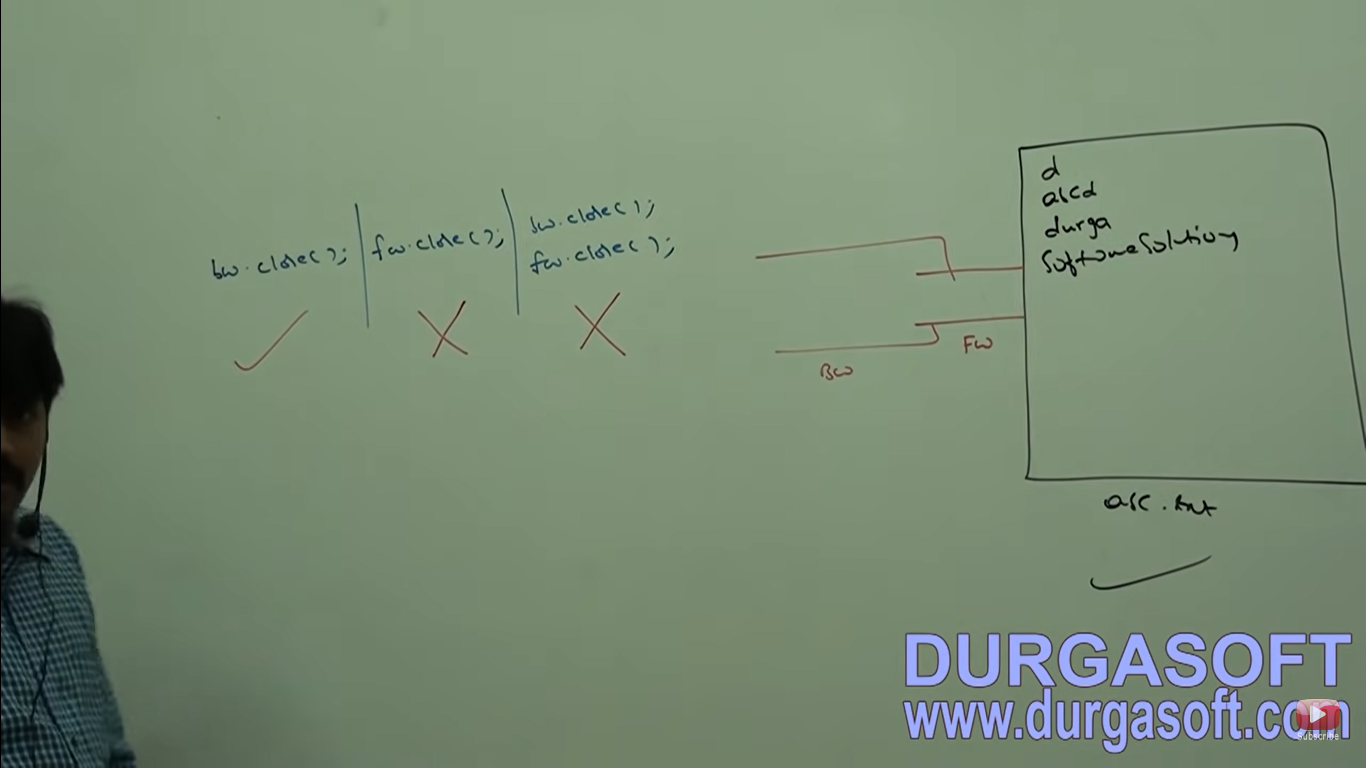
**4.** BufferedWriter bw = new BufferedWriter(new BW(new FW(“abc.txt”)); **//Two level buffering – valid**

**Methods:**

1. **write(int ch) -> To write single character to the file**
2. **write(char[] ch) -> To write an array of character**
3. **write(String s) -> To write string to the file**
4. **flush()**
5. **close()**
6. **newLine() - > To insert a line separator**

When compared with fileWriter which of the following capability available extra in method form in buffered writer?

1. Writing data to the file
2. Close the file
3. Flushing the file
4. **Inserting a new line character**



**Note – Whenever we are closing buffered writer automatically internal file writer will be closed and we are not required to close explicitly.**

**BufferedReader**

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

**Constructors**

1. **BufferedReader br = new BufferedReader(Reader r); //It can’t directly communicate with file**
2. **BufferedReader br = new BR(Reader r, int buffersize);**

**Note – Buffered Reader can’t communicate directly with the file and it can communicate via some Reader object.**

**Methods:**

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

**It attempts to read next line from the file and returns it. If the next line not available then this method returns null.**

**FR fr = new FR(“abc.txt”);**

**BR br = new BR(fr);**

**String line = br.readLine();**

**While(line!=null){**

**Sop(line);**

**Line = br.readLine();**

**}**

**Note – Whenever we are closing buffered reader automatically underlying file reader will be closed and we are not required to close explicitly. The most enhanced reader to read character data from the file is Buffered Reader.**