**File IO – Part-03**

* **FileReader:**

We can use FileReader to read character data from the file.

* **Constructors:**

FileReader fr = new FileReader(String fileName);

FileReader fr = new FileReader(File f);

* **Methods:**

int read();

It attempts to read next character from the file and returns its 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 type casting.

Example:

FileReader fr = new FileReader(“abc.txt”);

int i = fr.read();

while(i != -1){

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

i = fr.read();

}

int read(char[] ch);

It attempts to read enough characters from the file into char[] and returns number of characters copied.

Example:

File f= new File(“abc.txt”);

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

FileReader fr = new FileReader(f);

fr.read(ch);

for(char c: ch){

System.out.print(c);

}

Input:

durga

Software

Solutions

void close();

Once the read or write operation is done, it is mandatory to call the close method.

Example:

import java.io.\*;

class FileReaderDemo{

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

File f = new File(“abc.txt”);

FileReader fr = new FileRead(f);

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

fr.read(ch);

for(char ch1: ch){

System.out.print(ch1);

}

System.out.println(“\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”);

FileReader fr1 = new FileReader(“abc.txt”);

int i = fr1.read();

while(i != -1){

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

i = fr1.read();

}

}

}

Note:

By using FileReader we can read data character by character which is not convenient to the programmer.

Usage of FileReader and FileReader is not recommended because.

1. While writing data by FileWriter we have to insert line separator (\n) manually. Which varies from system to system. It is difficult to the programmer.
2. By using FileReader, we can read data character by character, which is not convenient to the programmer.

To overcome these problems, we should for BufferedWriter and BuffereReader.

* **BufferedWriter:**

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

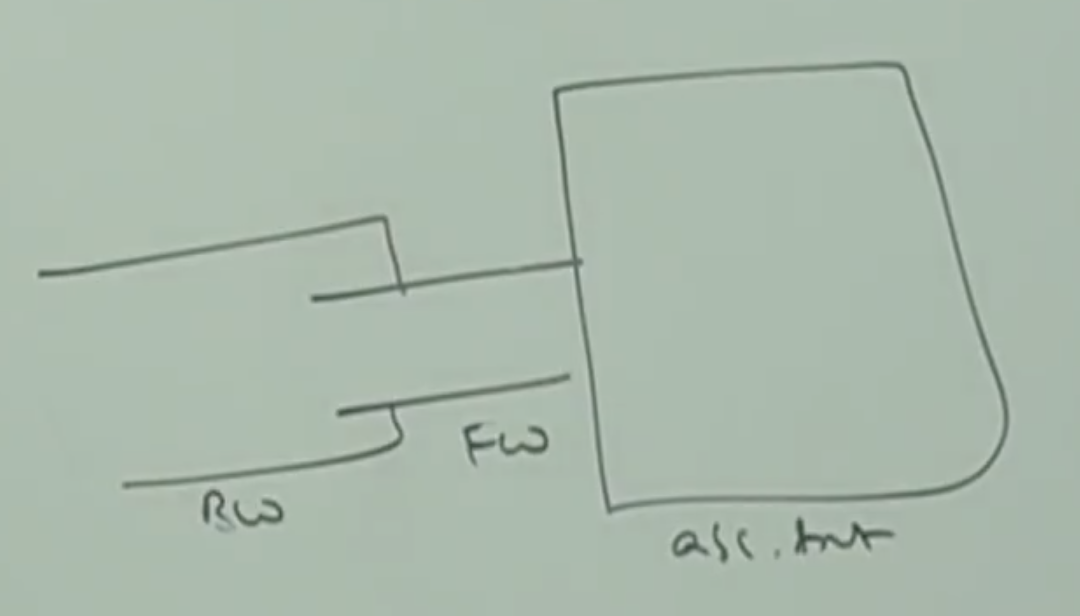
* **Constructors:**

BufferedWriter bw = new BufferedWriter(Writer w);

BufferedWriter bw = new BufferedWriter(Writer w, int bufferSize);

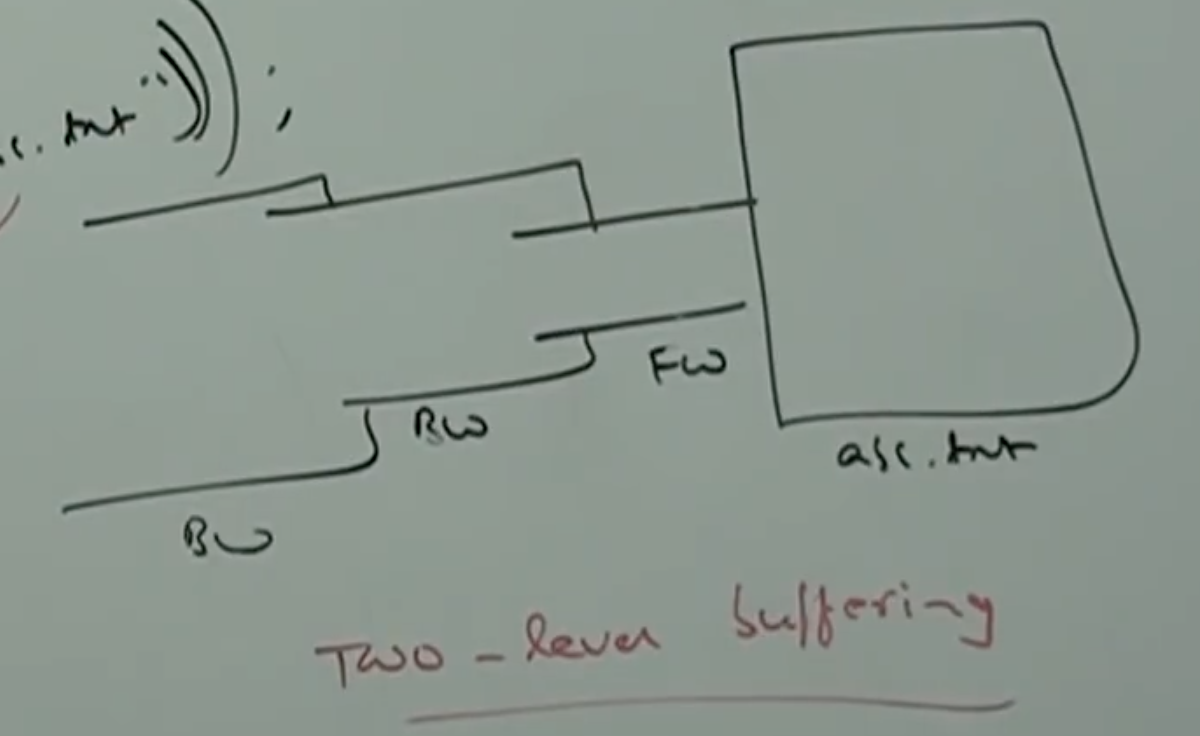
Note:

BufferedWriter can’t communicate directly with the file. It can communicate via some Writer object.



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 BufferedWriter(new FileWriter(“abc.txt”))); // valid

****

* **Methods:**

write(int ch);

write(char[] ch);

write(String s);

flush();

close();

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. // Answer.

Example:

class BufferedWriterDemo{

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

FileWriter fw = new FileWriter(“abc.txt”);

BufferedWriter bw = new BufferedWriter(fw);

bw.write(100);

bw.newLine();

char[] ch1 = {‘a’, ‘b’, ‘c’, ‘d’};

bw.write(ch1);

bw.newLine();

bw.write(“durga”);

bw.newLine();

bw.write(“Software solutions”);

bw.flush();

bw.close();

}

}

Output: d

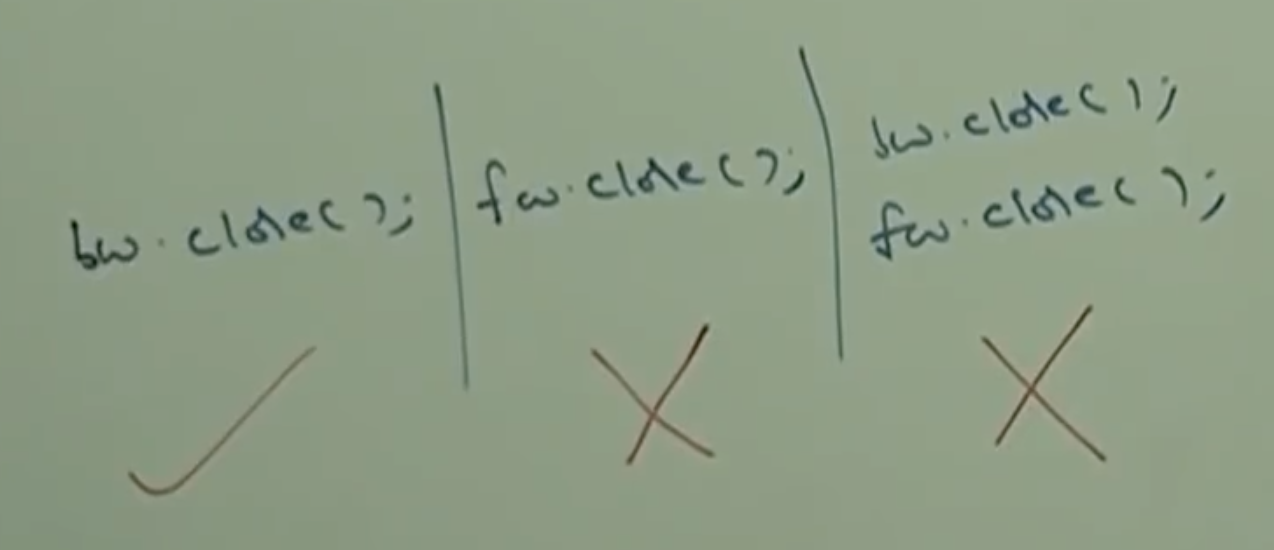
abcd

durga

Software solutions

Note:

Whenever we are closing BufferedWriter automatically internal FileWriter will be closed and we are not required to close explicitly.



* **BufferedReader:**

We can use BufferedReader to read character data from the file.

The advantage of BufferedRead when compared with FileReader is we can read data line by line in addition to character by character.

* **Constructors:**

BufferedReader br = new BufferedReader(Reader r);

BufferedReader br = new BufferedReader(Reader r, int bufferSize);

Note:

BufferedReader can’t communicate directly with the file and it can communicate via some Reader object.

* **Methods:**

int read();

int read(char[] ch);

void close();

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.

Example:

import java.io.\*;

class Test{

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

FileReader fr = new FileReader(“abc.txt”);

BufferedRead br = new BufferedReader(fr);

String line = br.readLine();

while(line != null){

System.out.println(line);

Line = br.readLine();

}

br.close();

}

}

**Note:**

Whenever we close BufferedReader it will automatically close the associated FileReader also. So, we are not required to close the FileReader explicitly.

Note\*\*\*

The most enhanced reader to read character data from the file is BufferedReader.