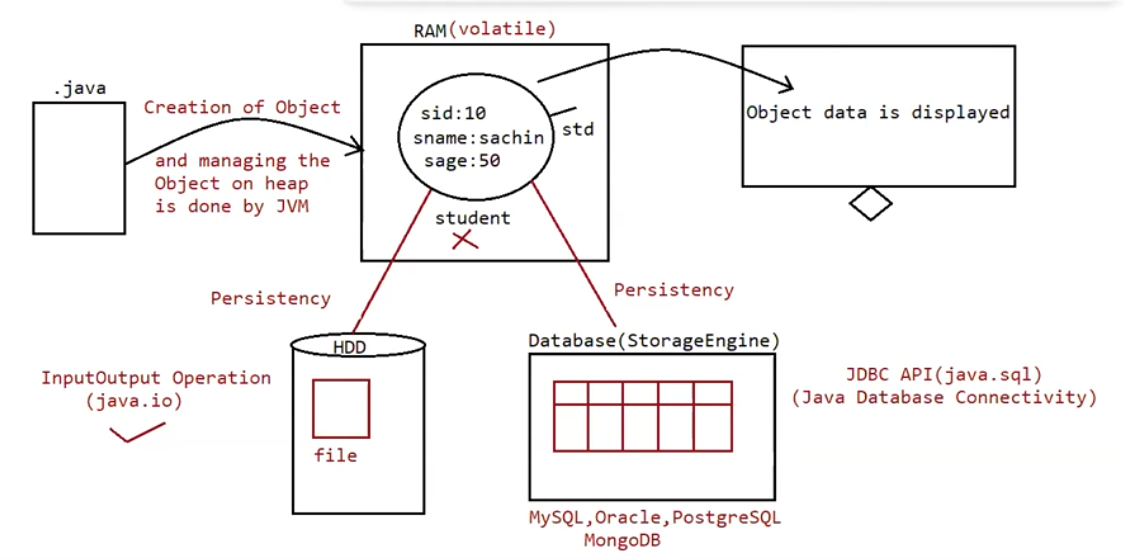
Eg: Io\_Introduction



What is persistency?

It is mechanism of storing the data permanently into a file.

In java persistency can be achieved through an API‘s available inside the package called “java.io”.

Eg: Io\_File\_Creation

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.

Eg: Io\_Directory\_Creation

Note:

In unix everything is file , java “file IO” is based on unix operating system

Hence in java we can represent both file and directory 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 in current working directory.

eg#1. File f=new File("abc.txt");

2. 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.

eg#1. File f1=new File("abc");

f1.mkdir();

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

Eg: Io\_File\_Inside\_Directory

Eg: Io\_File\_Inside\_Directory\_Eg1

Eg: Io\_Operations\_On\_File

Eg: Io\_Operations\_On\_File\_Eg1

Eg: Io\_FileWriter

FileWriter:

By using FileWriter object 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 the data to the file.

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.

2. write(char[] ch);

To write an array of characters to the file.

3. write(String s);

To write a String to the file.

4. flush();

To give the guarantee the total data include last character also written to the file.

5. close();

To close the stream.

Note:

=> The main problem with FileWriter is we have to insert line separator manually, which is difficult to the programmer. ('\n')

=> And even line separator varing from system to system.

Eg: Io\_FileWriter\_Eg2

FileReader:

=> By using FileReader object 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.

2. int i = fr.read();

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

As this method returns unicode value, while printing we have to perform type

casting.

4. int read(char[] ch);

It attempts to read enough characters from the file into char[] array and returns

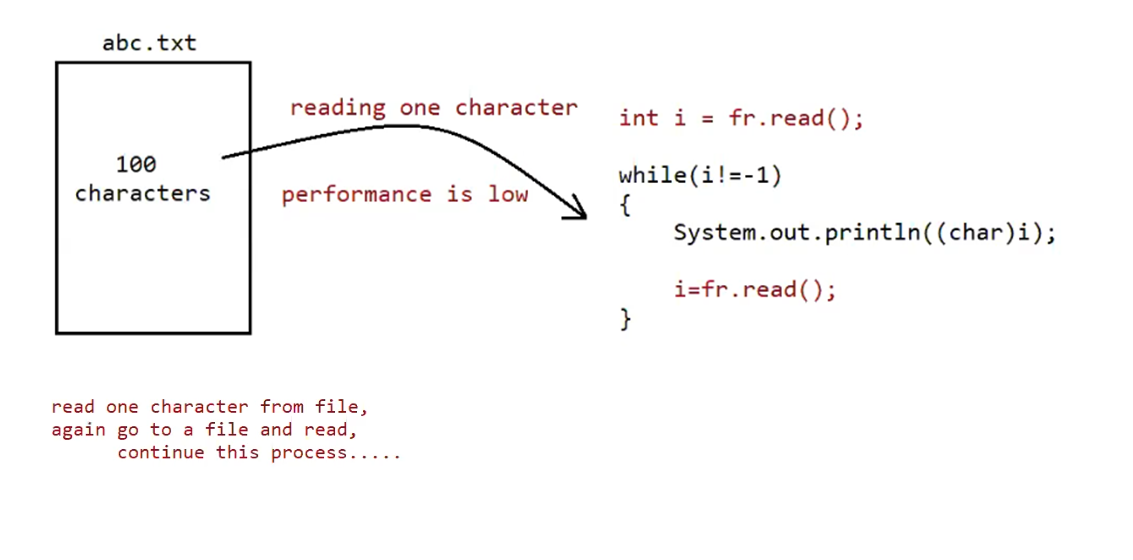
the no of characters copied from the file into char[] array.

5. File f=new File("abc.txt");

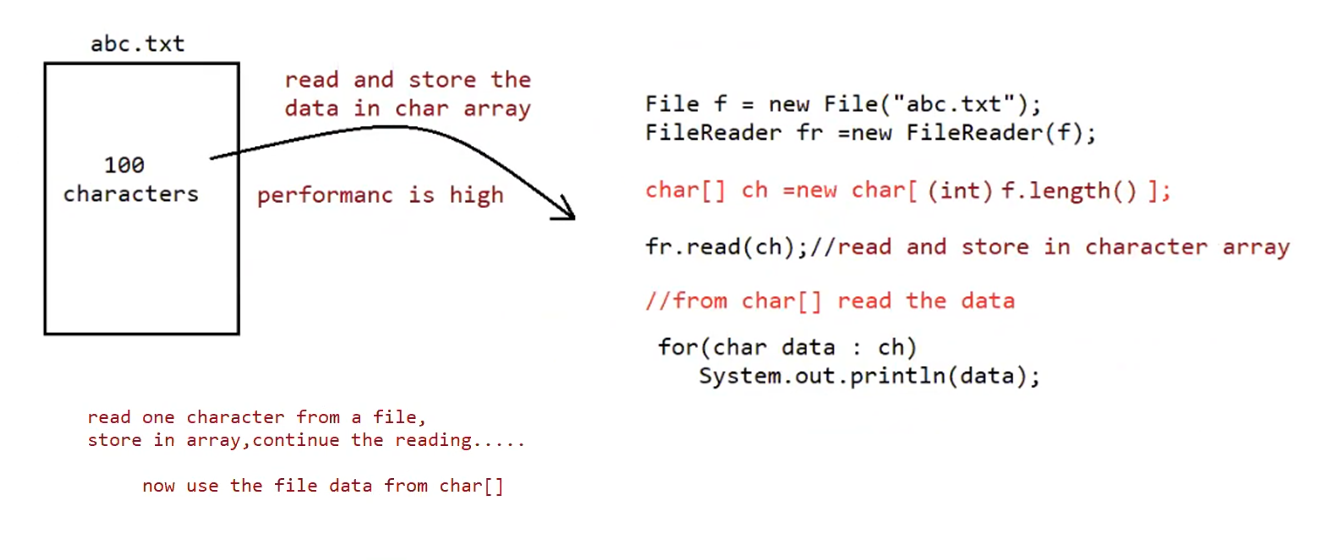
6. Char[] ch=new Char[(int)f.length()];

7. void close();

Eg: File\_Reader



Eg: File\_Reader\_Eg1



Usage of FileReader and FileWriter is not recommended because of following reason

1. While writing data by FileWriter compulsorily we should insert line separator (\n) which is a big headache to the programmer
2. While reading the data by FileReader we have to read character by character instead of line by line which is not convenient to the programmer

Assume we need to search for a 10 digit mobile number present in a file called “mobile.txt”

Since we can read only character by character just to search one mobile number 10 searching operations needs to be done , and to search 10000 mobile no we need to read 1cr times , so performance is very low.

1. So overcome these limitations we should use BufferedReader and BufferedWriter concpets.

BufferedWriter:

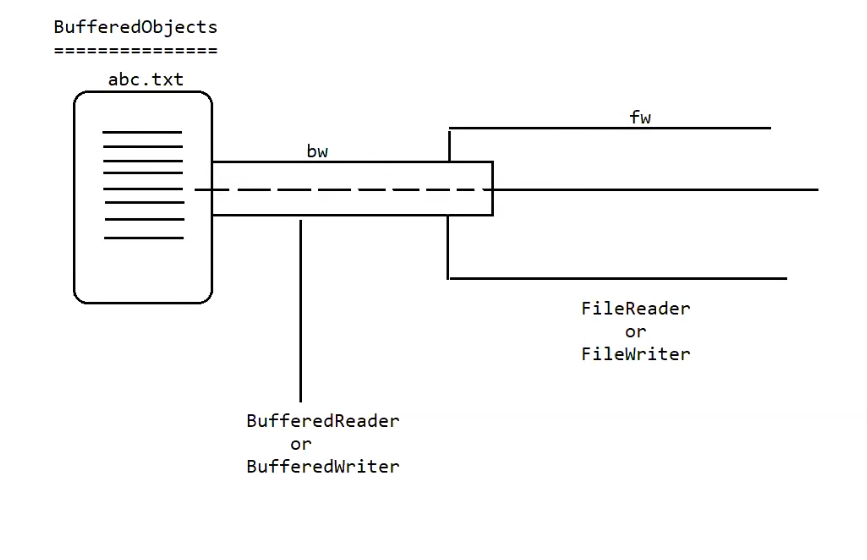
By using BufferedWriter object we can write character data to the file.

Constructor

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)

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

Eg: Buffered\_Writer

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.

Note:

When compared with FileWriter which of the following capability(facility) is available as method in BufferedWriter.

1. Writing data to the file.

2. Closing the writer.

3. Flush the writer.

4. Inserting newline character.

Ans. 4

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();

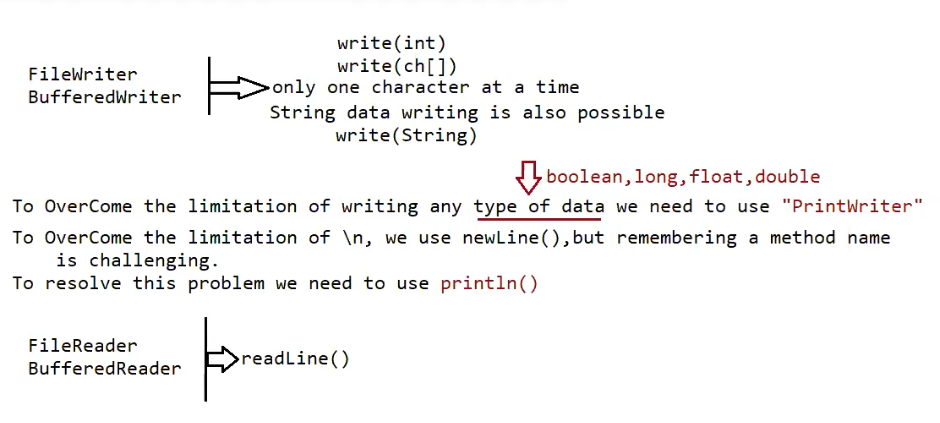
It attempts to read next line and return it, from the File. if the next line is not available then this method returns null.

4. void close();

Eg: Buffer\_Reader

Eg: BufferReader

//go through the code



Eg: Print\_Writer

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:

1. write(int ch);

2. write (char[] ch);

3. write(String s);

4. flush();

5. close();

6. print(char ch);

7. print (int i);

8. print (double d);

9. print (boolean b);

10.print (String s);

11.println(char ch);

12.println (int i);

13.println(double d);

14.println(boolean b);

15.println(String s);

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.

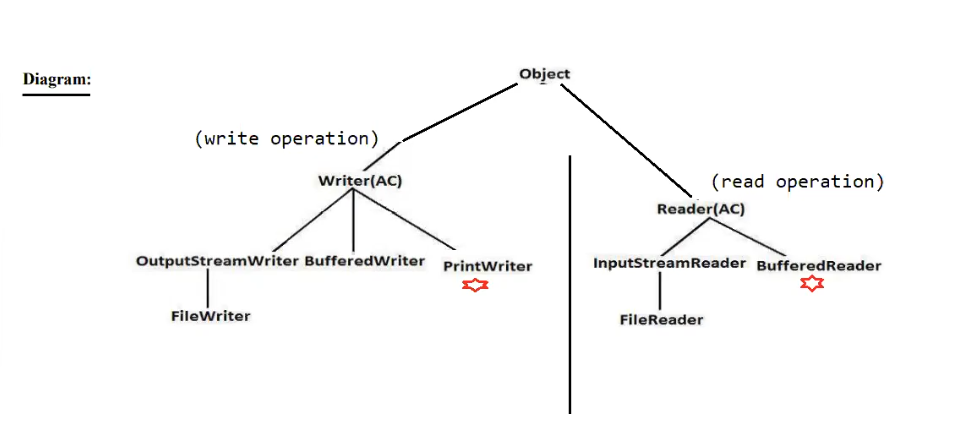
Note 2:

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 , video files etc.)
2. We can output stream to write the binary data to the file, and we can use input stream to read binary data from the file

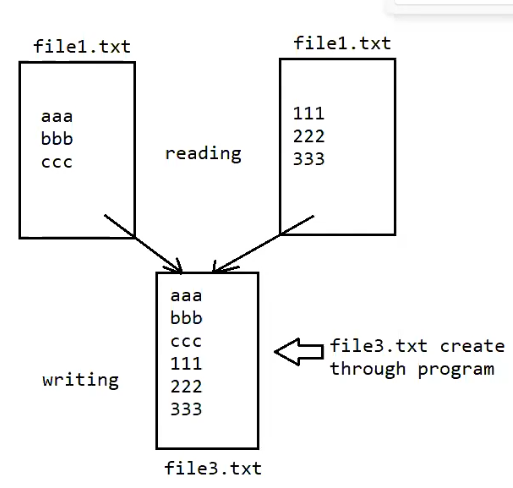
Character Data => Reader and Writer

Binary Data => InputStream and OutputStream

Hirearchy

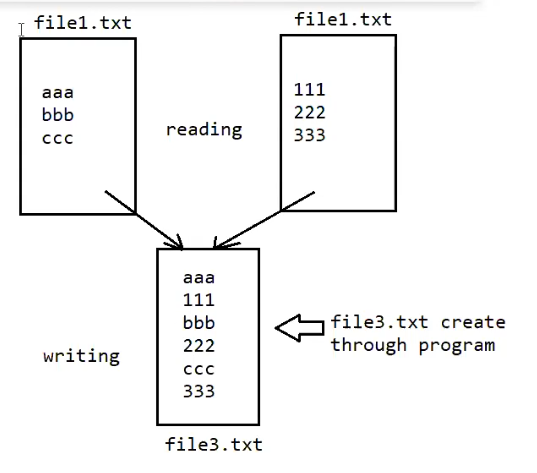


Eg: Copy\_File1Txt\_File2Txt\_To\_File3Txt



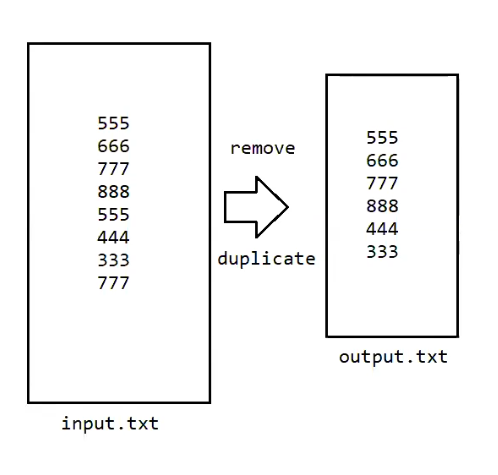
// go through the code

Eg: Copy\_File1Txt\_File2Txt\_LineByLine\_Alternatively\_To\_File3Txt



// go through the code

Eg: Extract\_Numbers\_Without\_Dupilcates\_From\_File1Txt



// go through the code