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.

FileOutputStream class methods

Method	Description
protected void finalize()	It is used to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write **ary.length** bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write **len** bytes from the byte array starting at offset **off** to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
FileChannel getChannel()	It is used to return the file channel object associated with the file output stream.
FileDescriptor getFD()	It is used to return the file descriptor associated with the stream.
void close()	It is used to closes the file output stream.

Example

```
import java.io.FileOutputStream;

public class fileoutput {

   public static void main(String[] args) {
     try {
      //FileOutputStream fout= new FileOutputStream("fileout.txt");  // This for Update Mode
      FileOutputStream fout= new FileOutputStream("fileout.txt",true);  // This for Append Mode
      //String s="My name is krishan kumawat ";
      // ORRR
      String s=javax.swing.JOptionPane.showInputDialog(null,"Enter the Text..");
      byte b[]=s.getBytes();
      /*for(int i=0; i<b.length; i++){
          fout.write(b[i]);
      }*/
      //  ORRRR
      fout.write(b);
      fout.close();
     } catch (Exception e) {
        e.printStackTrace();
     }
      }
```

}

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.

Java FileInputStream class methods

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to **b.length** bytes of data from the input stream.
int read(byte[] b, int off, int len)	It is used to read up to **len** bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
FileChannel getChannel()	It is used to return the unique FileChannel object associated with the file input stream.
FileDescriptor getFD()	It is used to return the FileDescriptor object.
protected void finalize()	It is used to ensure that the close method is call when there is no more reference to the file input stream.
void close()	It is used to closes the stream.

Example

```
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class In_OR_OUT {

public static void main(String[] args) {
try {
FileOutputStream fout=new FileOutputStream("fout.txt",true);
String s="My name is krishan "+" now i am in BCA 3 year";
byte b[]=s.getBytes();
fout.write(b);

FileInputStream fin=new FileInputStream("fout.txt");
int i=0;
while(((i=fin.read())!=-1)){
```

```
System.out.print((char)i);
}
// Orr
/*while(true){
int i=fin.read();
if(i==-1)
break;
System.out.print((char)i);
}*/

} catch (Exception e) {
e.printStackTrace();
}

}
}
```

Java ByteArrayOutputStream Class

Java ByteArrayOutputStream class is used to write common data into multiple files. In this stream, the data is written into a byte array which can be written to multiple streams later.

The ByteArrayOutputStream holds a copy of data and forwards it to multiple streams.

The buffer of ByteArrayOutputStream automatically grows according to data.

Java ByteArrayOutputStream class declaration

Documentation of ByteArrayOutputStream.java

```
public class java.io.ByteArrayOutputStream extends java.io.OutputStream {
  protected byte[] buf;
  protected int count;
  public java.io.ByteArrayOutputStream();
  public java.io.ByteArrayOutputStream(int);
  public synchronized void write(int);
  public synchronized void write(byte[], int, int);
  public void writeBytes(byte[]);
  public synchronized void writeTo(java.io.OutputStream) throws java.io.IOException;
  public synchronized void reset();
  public synchronized byte[] toByteArray();
  public synchronized int size();
  public synchronized java.lang.String toString();
  public synchronized java.lang.String toString(java.lang.String) throws
java.io.UnsupportedEncodingException;
  public synchronized java.lang.String toString(java.nio.charset.Charset);
  public synchronized java.lang.String toString(int);
  public void close() throws java.io.IOException;
```

```
}
```

Java ByteArrayOutputStream class constructors

Constructor	Description
ByteArrayOutputStream()	Creates a new byte array output stream with the initial capacity of 32 bytes, though its size increases if necessary.
ByteArrayOutputStream(int size)	Creates a new byte array output stream, with a buffer capacity of the specified size, in bytes.

Java ByteArrayOutputStream class methods

Method	Description
int size()	It is used to returns the current size of a buffer.
byte[] toByteArray()	It is used to create a newly allocated byte array.
String toString()	It is used for converting the content into a string decoding bytes using a platform default character set.
String toString(String charsetName)	It is used for converting the content into a string decoding bytes using a specified charsetName.
void write(int b)	It is used for writing the byte specified to the byte array output stream.
void write(byte[] b, int off, int len	It is used for writing **len** bytes from specified byte array starting from the offset **off** to the byte array output stream.
void writeTo(OutputStream out)	It is used for writing the complete content of a byte array output stream to the specified output stream.
void reset()	It is used to reset the count field of a byte array output stream to zero value.
void close()	It is used to close the ByteArrayOutputStream.

Example

```
public static void main(String[] args) throws IOException
{
ByteArrayOutputStream bout=new ByteArrayOutputStream();
FileOutputStream fout = new FileOutputStream("bout.txt");
String s="My name is krishan"+" now";
byte b[]=s.getBytes();
bout.write(b);
bout.writeTo(fout);
}
```

Java ByteArrayInputStream class declaration

Documentation of ByteArrayInputStream.java

```
public class java.io.ByteArrayInputStream extends java.io.InputStream {
  protected byte[] buf;
  protected int pos;
```

```
  protected int mark;
  protected int count;
  public java.io.ByteArrayInputStream(byte[]);
  public java.io.ByteArrayInputStream(byte[], int, int);
  public synchronized int read();
  public synchronized int read(byte[], int, int);
  public synchronized byte[] readAllBytes();
  public int readNBytes(byte[], int, int);
  public synchronized long transferTo(java.io.OutputStream) throws java.io.IOException;
  public synchronized long skip(long);
  public synchronized int available();
  public boolean markSupported();
  public void mark(int);
  public synchronized void reset();
  public void close() throws java.io.IOException;
}
```

Java ByteArrayInputStream class constructors

Constructor	Description
ByteArrayInputStream(byte[] ary)	Creates a new byte array input stream which uses **ary** as its buffer array.
ByteArrayInputStream(byte[] ary, int offset, int len)	Creates a new byte array input stream which uses **ary** as its buffer array that can read up to specified **len** bytes of data from an array.

Java ByteArrayInputStream class methods

Methods	Description
int available()	It is used to return the number of remaining bytes that can be read from the input stream.
int read()	It is used to read the next byte of data from the input stream.
int read(byte[] ary, int off, int len)	It is used to read up to len bytes of data from an array of bytes in the input stream.
boolean markSupported()	It is used to test the input stream for mark and reset method.
long skip(long x)	It is used to skip the x bytes of input from the input stream.
void mark(int readAheadLimit)	It is used to set the current marked position in the stream.
void reset()	It is used to reset the buffer of a byte array.
void close()	It is used for closing a ByteArrayInputStream.

Java BufferedOutputStream Class

Java BufferedOutputStream class is used for buffering an output stream. It internally uses buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

Java BufferedOutputStream class declaration

```
public class java.io.BufferedOutputStream extends java.io.FilterOutputStream {
  protected byte[] buf;
  protected int count;
  public java.io.BufferedOutputStream(java.io.OutputStream);
  public java.io.BufferedOutputStream(java.io.OutputStream, int);
  public synchronized void write(int) throws java.io.IOException;
  public synchronized void write(byte[], int, int) throws java.io.IOException;
  public synchronized void flush() throws java.io.IOException;
}
```

Java BufferedOutputStream class constructors

Constructor	Description
BufferedOutputStream(OutputStream os)	It creates the new buffered output stream which is used for writing the data to the specified output stream.
BufferedOutputStream(OutputStream os, int size)	It creates the new buffered output stream which is used for writing the data to the specified output stream with a specified buffer size.

Java BufferedOutputStream class methods

Method	Description
void write(int b)	It writes the specified byte to the buffered output stream.
void write(byte[] b, int off, int len)	It write the bytes from the specified byte-input stream into a specified byte array, starting with the given offset
void flush()	It flushes the buffered output stream.

Example

```
public static void main(String[] args) throws IOException{
FileOutputStream fout=new FileOutputStream("buffered.txt");
BufferedOutputStream buff=new BufferedOutputStream(fout);
String s="Welcome in BufferedOutputStream";
byte b[]=s.getBytes();
```

```
buff.write(b);
buff.flush(); // It is mandatory to flush the data in file

}
```

Java BufferedInputStream Class

Java BufferedInputStream class is used to read information from stream. It internally uses buffer mechanism to make the performance fast.

The important points about BufferedInputStream are:

- When the bytes from the stream are skipped or read, the internal buffer automatically refilled from the contained input stream, many bytes at a time.
- When a BufferedInputStream is created, an internal buffer array is created.

Java BufferedInputStream class declaration

public class java.io.BufferedInputStream extends java.io.FilterInputStream {

protected volatile byte[] buf;

protected int count;

protected int pos;

protected int markpos;

protected int marklimit;

public java.io.BufferedInputStream(java.io.InputStream);

public java.io.BufferedInputStream(java.io.InputStream, int);

public synchronized int read() throws java.io.IOException;

public synchronized int read(byte[], int, int) throws java.io.IOException;

public synchronized long skip(long) throws java.io.IOException;

public synchronized int available() throws java.io.IOException;

public synchronized void mark(int);

public synchronized void reset() throws java.io.IOException;

public boolean markSupported();

public void close() throws java.io.IOException;

static {};

}

Java BufferedInputStream class constructors

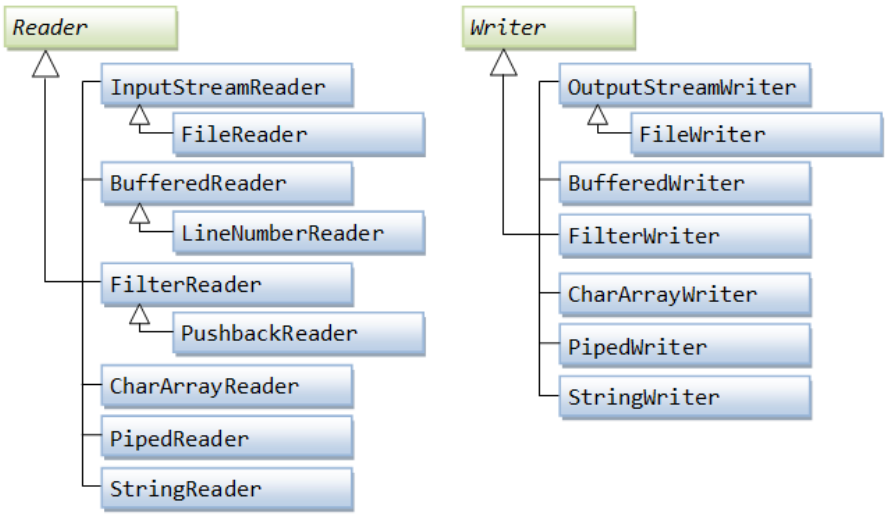
Constructor	Description
BufferedInputStream(InputStream IS)	It creates the BufferedInputStream and saves it argument, the input stream IS, for later use.
BufferedInputStream(InputStream IS, int size)	It creates the BufferedInputStream with a specified buffer size and saves it argument, the input stream IS, for later use.

Java BufferedInputStream class methods

Method	Description
int available()	It returns an estimate number of bytes that can be read from the input stream without blocking by the next invocation method for the input stream.
int read()	It read the next byte of data from the input stream.
int read(byte[] b, int off, int ln)	It read the bytes from the specified byte-input stream into a specified byte array, starting with the given offset.
void close()	It closes the input stream and releases any of the system resources associated with the stream.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readlimit)	It sees the general contract of the mark method for the input stream.
long skip(long x)	It skips over and discards x bytes of data from the input stream.
boolean markSupported()	It tests for the input stream to support the mark and reset methods.

Character Oriented Stream

hierarchy of Character Oriented Stream



Java Writer

It is an abstract class for writing to character streams. The methods that a subclass must implement are write(char[], int, int), flush(), and close(). Most subclasses will override some of the methods defined here to provide higher efficiency, functionality or both.

Implementation of Writer class

```
public abstract class java.io.Writer implements
java.lang.Appendable,java.io.Closeable,java.io.Flushable {

  protected java.lang.Object lock;

  public static java.io.Writer nullWriter();

  protected java.io.Writer();

  protected java.io.Writer(java.lang.Object);

  public void write(int) throws java.io.IOException;

  public void write(char[]) throws java.io.IOException;

  public abstract void write(char[], int, int) throws java.io.IOException;

  public void write(java.lang.String) throws java.io.IOException;

  public void write(java.lang.String, int, int) throws java.io.IOException;

  public java.io.Writer append(java.lang.CharSequence) throws java.io.IOException;

  public java.io.Writer append(java.lang.CharSequence, int, int) throws java.io.IOException;

  public java.io.Writer append(char) throws java.io.IOException;

  public abstract void flush() throws java.io.IOException;

  public abstract void close() throws java.io.IOException;

  public java.lang.Appendable append(char) throws java.io.IOException;

  public java.lang.Appendable append(java.lang.CharSequence, int, int) throws
java.io.IOException;

  public java.lang.Appendable append(java.lang.CharSequence) throws java.io.IOException;

}
```

Fields

Modifier and Type	Field	Description
protected Object	lock	The object used to synchronize operations on this stream.

Constructor

Modifier	Constructor	Description
protected	Writer()	It creates a new character-stream writer whose critical sections will

		synchronize on the writer itself.
protected	Writer(Object lock)	It creates a new character-stream writer whose critical sections will synchronize on the given object.

Methods

Modifier and Type	Method	Description
Writer	append(char c)	It appends the specified character to this writer.
Writer	append(CharSequence csq)	It appends the specified character sequence to this writer
Writer	append(CharSequence csq, int start, int end)	It appends a subsequence of the specified character sequence to this writer.
abstract void	close()	It closes the stream, flushing it first.
abstract void	flush()	It flushes the stream.
void	write(char[] cbuf)	It writes an array of characters.
abstract void	write(char[] cbuf, int off, int len)	It writes a portion of an array of characters.
void	write(int c)	It writes a single character.
void	write(String str)	It writes a string.
void	write(String str, int off, int len)	It writes a portion of a string.

Example

```
public static void main(String[] args) throws IOException {
Writer w = new FileWriter("Writer.txt");
String s="This is Writer Class";
w.write(s);
w.close();
System.out.println("Success");
}
```

Java BufferedWriter Class

Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits Writer class. The buffering characters are used for providing the efficient writing of single arrays, characters, and strings.

Class Implementation

```
public class java.io.BufferedWriter extends java.io.Writer {
  public java.io.BufferedWriter(java.io.Writer);
  public java.io.BufferedWriter(java.io.Writer, int);
  void flushBuffer() throws java.io.IOException;
  public void write(int) throws java.io.IOException;
  public void write(char[], int, int) throws java.io.IOException;
  public void write(java.lang.String, int, int) throws java.io.IOException;
  public void newLine() throws java.io.IOException;
```

```
  public void flush() throws java.io.IOException;
  public void close() throws java.io.IOException;
  static {};
}
```

Class constructors

Constructor	Description
BufferedWriter(Writer wrt)	It is used to create a buffered character output stream that uses the default size for an output buffer.
BufferedWriter(Writer wrt, int size)	It is used to create a buffered character output stream that uses the specified size for an output buffer.

Class methods

Method	Description
void newLine()	It is used to add a new line by writing a line separator.
void write(int c)	It is used to write a single character.
void write(char[] cbuf, int off, int len)	It is used to write a portion of an array of characters.
void write(String s, int off, int len)	It is used to write a portion of a string.
void flush()	It is used to flushes the input stream.
void close()	It is used to closes the input stream

```
public static void main(String[] args)throws IOException {
FileWriter fw=new FileWriter("./bufferwriter.txt");
BufferedWriter wr=new BufferedWriter(fw);
String s="jhkhhiuwhk";
wr.write(s);
wr.flush();
}
```

Java BufferedReader Class

Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast. It inherits Reader class.

Java BufferedReader class declaration

```
public class java.io.BufferedReader extends java.io.Reader {
  public java.io.BufferedReader(java.io.Reader, int);
  public java.io.BufferedReader(java.io.Reader);
  public int read() throws java.io.IOException;
  public int read(char[], int, int) throws java.io.IOException;
  java.lang.String readLine(boolean) throws java.io.IOException;
  public java.lang.String readLine() throws java.io.IOException;
```

```
  public long skip(long) throws java.io.IOException;
  public boolean ready() throws java.io.IOException;
  public boolean markSupported();
  public void mark(int) throws java.io.IOException;
  public void reset() throws java.io.IOException;
  public void close() throws java.io.IOException;
  public java.util.stream.Stream<java.lang.String> lines();
  static {};
}
```

Java BufferedReader class constructors

Constructor	Description
BufferedReader(Reader rd)	It is used to create a buffered character input stream that uses the default size for an input buffer.
BufferedReader(Reader rd, int size)	It is used to create a buffered character input stream that uses the specified size for an input buffer.

Java BufferedReader class methods

Method	Description
int read()	It is used for reading a single character.
int read(char[] cbuf, int off, int len)	It is used for reading characters into a portion of an array.
boolean markSupported()	It is used to test the input stream support for the mark and reset method.
String readLine()	It is used for reading a line of text.
boolean ready()	It is used to test whether the input stream is ready to be read.
long skip(long n)	It is used for skipping the characters.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readAheadLimit)	It is used for marking the present position in a stream.
void close()	It closes the input stream and releases any of the system resources associated with the stream.

Example

```
public static void main(String[] args)throws IOException {
FileReader fr=new FileReader("bufferwriter.txt");
BufferedReader br=new BufferedReader(fr);

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