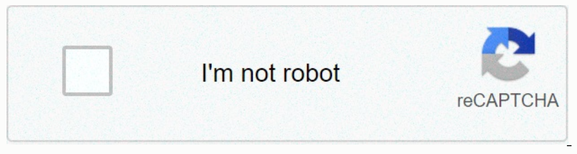

Continue

Bufferedinputstream android studio

bufferedInputStream public class extends FilterInputStream A BufferedInputStream adds functionality to other stream inputs i.e., the ability to buffer inputs and to support token and reset methods. When BufferedInputStream is created, an internal buffer array is created. Because bytes of the stream are read or skipped, the internal buffer is replenished as necessary from the contained input stream, many bytes at a time. The period remembering operation in the input stream and the reset operation cause all bytes read since the latest mark operation to be read back before the new byte is retrieved from the contained input stream. protected bytes[] buf The internal buffer array where the data is stored. Index protected int number one is greater than the last valid byte index in the buffer. Maximum protected marklimit int is read in front of the allowed after call to the sign method before the next call to the reset method fails. markpos int protected The value of the postal field at the time the last sign method was called. protected int post The current position is in the buffer. int available() Returns the estimated number of bytes that can be read (or skipped) from this input stream without blocking by calling the next method for this input stream. void close() Closes this input stream and releases any system resources associated with the stream. void mark(int readlimit) See common contracts of the InputStream mark method. boolean markSupported() Tests whether this input stream supports the mark and reset method. int read() See common contracts of the InputStream read method. int read(byte[] b, int off, int len) Reads bytes of this byte-input stream into the specified byte array, starting at the given offset. void reset() See common contracts of the InputStream reset method. long skip(long n) See common contracts of the InputStream skip method. From the java.io.FilterInputStream int read() Read the next byte of data from this input stream. int read(byte[] b, int off, int len) Read up to len byte data from this input stream into byte arrays. int read(byte[] b) Reads up to byte.length of data from this input stream into byte arrays. void reset() Repositions this stream to the position at the time the last sign method was called on this input stream. long pass (long n) Pass and discard n bytes of data from the input stream. From the java.io.InputStream int available() class Returns the estimated number of bytes that can be (or skipped) from this input stream without blocking by calling the next method for this input stream. void close() Closes this input stream and releases any system resources associated with the stream. void mark(int readlimit) Marks the current position in this input stream. boolean markSupported() Tests whether this input stream supports the mark and reset method. int abstract read() Read the next byte of data from the input stream. int read(byte[] b, int off, int len) Read up to len byte data from the input stream into the byte array. int read(byte[] b) Reads some bytes of the input stream and stores it into the buffer array b. void reset() Repositions this stream to the position at which the last sign method was called on this input stream. long pass (long n) Skip and discard n bytes of data from this input stream. From the java.lang.Object Clone() class Creates and returns a copy of this object. boolean equals (Object obj) Indicates whether some other object is equal to this one. void finalize() Is called by the garbage collector on the object when garbage collection specifies that there is no longer a reference to the object. Final Class<?> getClass() Returns the runtime class of the Object. int hashCode() Returns the hash code value for the object. String toString() Returns the representation of an object string. final void wait(long timeout, int nanos) Causes the thread to currently wait until another thread calls the notify() method or notifyAll() method for this object, or some other thread interrupts the current thread, or a certain amount of real time has passed. end void wait (timeout) Causes the current thread to wait until another thread calls the notify() method or notifyAll() method for this object, or a specified amount of time has elapsed. the final void wait() causes the current thread to wait until another thread calls the notify() method or the notifyAll() method for this object. From the java.io.Closeable abstract void close() interface closes this stream and releases any system resources associated with it. Byte-protected fields[] buf The internal buffer array where the data is stored. When necessary, it may be replaced by other arrays of different sizes. Index protected int number one is greater than the last valid byte index in the buffer. This value is always in the range of 0 to buf.length; buf element[0] via buf[count-1] contains buffer input data obtained from the underlying input stream. Maximum protected marklimit int is read in front of the allowed after call to the sign method before the next call to the reset method fails. Whenever the difference between a post and a markpos exceeds the marklimit, then the mark can be dropped by setting the markpos to -1. See also: markpos int protected postal field value at the time the last sign method was called. This value is always in the range -1 by post. If no position is marked in the input stream, this field is -1. If there is a marked position in the input stream, then buf[markpos] is the first byte provided as input after the reset operation. If markpos is not -1, then all bytes of the buf[markpos] position via buf[post-1] must remain in the buffer array (although it can be moved elsewhere in the buffer array, with adjustments corresponding to the count, postal, and markpos values); they may not be discarded unless and until the difference between post and markpos exceeds marklimit. protected int post The current position is in the buffer. This is the next character index to be read from the buf array. This value is always in the range of 0 to count. If it is less than a count, then buf[postal] is the next byte to be given as input; if it is the same as calculating, then the next read or pass operation will require more bytes to read than the contained input stream. The Public BufferedInputStream (InputStream in) Constructor Creates a BufferedInputStream and stores its argument, the incoming input stream, for later use. An internal buffer array is created and stored in buf. Parameters in InputStream: the underlying input stream. Public BufferedInputStream (InputStream in, int size) Creates a BufferedInputStream with the specified buffer size, and stores its arguments, the input flow in, for later use. An internal buffer array of length is created and stored in buf. Parameters in InputStream: the underlying input stream. int size: buffer size. Throw IllegalArgumentException if size < = 0. Publicly available method () Returns the estimated number of bytes that can be read (or skipped) from this input stream without blocking by calling the next method for this input stream. The next call may be the same thread or another thread. One read or pass byte this much won't block, but it can read or pass fewer bytes. This method returns the number of bytes left to read in the buffer (count - post) and the result of the in.available() call. Returns the estimated number of bytes that can be read (or skipped) from this input stream without blocking. Throw IOException if this input stream has been closed by calling the close() method, or an I/O error occurs. After the stream is closed, calling read(), available(), reset(), or skip() will further throw IOException. Closing previously closed streams has no effect. Throw IOException in case of a public void mark (int readlimit) error See common contracts of the InputStream mark method. Parameters int: the maximum limit of bytes that can be read before the position of the sign becomes invalid. public boolean () Tests whether this input stream supports the sign and reset method. The markSupported bufferedInputStream method returns true. Returns a boolean that indicates whether this type of stream supports the mark and reset method. See also: InputStream.mark(int)InputStream.reset() public int read () See general contract of InputStream reading method. Returns the next data byte int, or -1 if the end of the flow is reached. Throw IOException if this input stream has been closed by calling the close() method, or an I/O error occurs. See also: public int reading (byte[] b, int off, int len) Reading bytes of this byte-input flow into the specified byte array, starting at the given offset. This method implements the appropriate General InputStream#read(byte], int, int) method contract of the InputStream class. As an added convenience, he tries to read as many bytes as possible by repeatedly calling the underlying flow reading method. This recurring reading continues until one of the following conditions becomes true: The specified number of bytes has been read, the underlying flow reading method returns -1, indicates the end of the file, or the available method of the underlying stream returns zero, indicating that further input requests will block. If the first read on the underlying stream returns -1 to indicate the end of the file then this method returns -1. Otherwise, this method returns the number of bytes actually read. This class subclass is encouraged, but not necessary, to try to read as many bytes as possible in the same way. Parameter b byte: destination buffer. off int: offset where to start saving bytes. len int: the maximum number of bytes to read. Returns the number of bytes read, or -1 if the end of the stream has been reached. Throw IOException if this input stream has been closed by calling the close() method, or a public vacancy reset I/O error occurs () View the general contract of the InputStream reset method. If the markpos is -1 (no sign is set or the mark has been cancelled), IOException will be thrown. Otherwise, the post is set to the same as the markpos. Throw IOException if this stream has not been marked or, if the mark has been cancelled, or the stream has been closed by calling the close() method, or a public long n skip I/O error See the general contract of the InputStream skip method. Parameter n length: the number of bytes to pass. Returns the length of the actual number of bytes skipped. Throw IOException if the stream does not support searching, or if this input stream has been closed by calling the close() method, or an I/O error occurs. Happen.