java.io

Class FileInputStream

java.lang.Object
  java.io.InputStream
    java.io.FileInputStream

All Implemented Interfaces:

Closeable, AutoCloseable

```
public class FileInputStream
extends InputStream
```

A `FileInputStream` obtains input bytes from a file in a file system. What files are available depends on the host environment.

`FileInputStream` is meant for reading streams of raw bytes such as image data. For reading streams of characters, consider using `FileReader`.

Since:

JDK1.0

See Also:

`File`, `FileDescriptor`, `FileOutputStream`, `Files.newInputStream(java.nio.file.Path, java.nio.file.OpenOption...)`

Constructor Summary

Constructors

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

Method Summary

Methods

Modifier and Type	Method and Description
int	**available**() Returns an estimate of the number of remaining bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
void	**close**() Closes this file input stream and releases any system resources associated with the stream.
protected void	**finalize**() Ensures that the close method of this file input stream is called when there are no more references to it.
FileChannel	**getChannel**() Returns the unique **FileChannel** object associated with this file input stream.
FileDescriptor	**getFD**() Returns the FileDescriptor object that represents the connection to the actual file in the file system being used by this FileInputStream.
int	**read**() Reads a byte of data from this input stream.
int	**read**(byte[] b) Reads up to b.length bytes of data from this input stream into an array of bytes.
int	**read**(byte[] b, int off, int len) Reads up to len bytes of data from this input stream into an array of bytes.
long	**skip**(long n) Skips over and discards n bytes of data from the input stream.

Methods inherited from class java.io.InputStream

mark, markSupported, reset

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FileInputStream

```
public FileInputStream(String name)
                throws FileNotFoundException
```

Creates a FileInputStream by opening a connection to an actual file, the file named by the path name name in the file system. A new FileDescriptor object is created to represent

this file connection.

First, if there is a security manager, its `checkRead` method is called with the `name` argument as its argument.

If the named file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading then a `FileNotFoundException` is thrown.

Parameters:

`name` - the system-dependent file name.

Throws:

`FileNotFoundException` - if the file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.

`SecurityException` - if a security manager exists and its `checkRead` method denies read access to the file.

See Also:

`SecurityManager.checkRead(java.lang.String)`

FileInputStream

```
public FileInputStream(File file)
                throws FileNotFoundException
```

Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system. A new `FileDescriptor` object is created to represent this file connection.

First, if there is a security manager, its `checkRead` method is called with the path represented by the `file` argument as its argument.

If the named file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading then a `FileNotFoundException` is thrown.

Parameters:

`file` - the file to be opened for reading.

Throws:

`FileNotFoundException` - if the file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.

`SecurityException` - if a security manager exists and its `checkRead` method denies read access to the file.

See Also:

`File.getPath()`, `SecurityManager.checkRead(java.lang.String)`

FileInputStream

```
public FileInputStream(FileDescriptor fdObj)
```

Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.

If there is a security manager, its `checkRead` method is called with the file descriptor `fdObj` as its argument to see if it's ok to read the file descriptor. If read access is denied to the file descriptor a `SecurityException` is thrown.

If `fdObj` is null then a `NullPointerException` is thrown.

This constructor does not throw an exception if `fdObj` is `invalid`. However, if the methods are invoked on the resulting stream to attempt I/O on the stream, an `IOException` is thrown.

Parameters:

`fdObj` - the file descriptor to be opened for reading.

Throws:

`SecurityException` - if a security manager exists and its `checkRead` method denies read access to the file descriptor.

See Also:

`SecurityManager.checkRead(java.io.FileDescriptor)`

Method Detail

read

```
public int read()
         throws IOException
```

Reads a byte of data from this input stream. This method blocks if no input is yet available.

Specified by:

`read` in class `InputStream`

Returns:

the next byte of data, or `-1` if the end of the file is reached.

Throws:

`IOException` - if an I/O error occurs.

read

```
public int read(byte[] b)
         throws IOException
```

Reads up to `b.length` bytes of data from this input stream into an array of bytes. This method blocks until some input is available.

Overrides:

read in class InputStream

Parameters:

b - the buffer into which the data is read.

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the file has been reached.

Throws:

IOException - if an I/O error occurs.

See Also:

InputStream.read(byte[], int, int)

read

```
public int read(byte[] b,
       int off,
       int len)
         throws IOException
```

Reads up to `len` bytes of data from this input stream into an array of bytes. If `len` is not zero, the method blocks until some input is available; otherwise, no bytes are read and `0` is returned.

Overrides:

read in class InputStream

Parameters:

b - the buffer into which the data is read.

`off` - the start offset in the destination array b

`len` - the maximum number of bytes read.

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the file has been reached.

Throws:

NullPointerException - If b is `null`.

IndexOutOfBoundsException - If `off` is negative, `len` is negative, or `len` is greater than `b.length - off`

IOException - if an I/O error occurs.

See Also:

InputStream.read()

skip

```
public long skip(long n)
          throws IOException
```

Skips over and discards n bytes of data from the input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly `0`. If n is negative, an `IOException` is thrown, even though the `skip` method of the `InputStream` superclass does nothing in this case. The actual number of bytes skipped is returned.

This method may skip more bytes than are remaining in the backing file. This produces no exception and the number of bytes skipped may include some number of bytes that were beyond the EOF of the backing file. Attempting to read from the stream after skipping past the end will result in -1 indicating the end of the file.

Overrides:

`skip` in class `InputStream`

Parameters:

n - the number of bytes to be skipped.

Returns:

the actual number of bytes skipped.

Throws:

`IOException` - if n is negative, if the stream does not support seek, or if an I/O error occurs.

available

```
public int available()
             throws IOException
```

Returns an estimate of the number of remaining bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream. The next invocation might be the same thread or another thread. A single read or skip of this many bytes will not block, but may read or skip fewer bytes.

In some cases, a non-blocking read (or skip) may appear to be blocked when it is merely slow, for example when reading large files over slow networks.

Overrides:

`available` in class `InputStream`

Returns:

an estimate of the number of remaining bytes that can be read (or skipped over) from this input stream without blocking.

Throws:

`IOException` - if this file input stream has been closed by calling `close` or an I/O error occurs.

close

```
public void close()
           throws IOException
```

Closes this file input stream and releases any system resources associated with the stream.

If this stream has an associated channel then the channel is closed as well.

Specified by:

`close` in interface `Closeable`

Specified by:

`close` in interface `AutoCloseable`

Overrides:

`close` in class `InputStream`

Throws:

`IOException` - if an I/O error occurs.

getFD

```
public final FileDescriptor getFD()
                           throws IOException
```

Returns the `FileDescriptor` object that represents the connection to the actual file in the file system being used by this `FileInputStream`.

Returns:

the file descriptor object associated with this stream.

Throws:

`IOException` - if an I/O error occurs.

See Also:

`FileDescriptor`

getChannel

```
public FileChannel getChannel()
```

Returns the unique `FileChannel` object associated with this file input stream.

The initial position of the returned channel will be equal to the number of bytes read from the file so far. Reading bytes from this stream will increment the channel's position. Changing the channel's position, either explicitly or by reading, will change this stream's file position.

Returns:

the file channel associated with this file input stream

Since:

1.4

finalize

```
protected void finalize()
                 throws IOException
```

Ensures that the `close` method of this file input stream is called when there are no more references to it.

Overrides:

`finalize` in class `Object`

Throws:

`IOException` - if an I/O error occurs.

See Also:

`close()`