Java

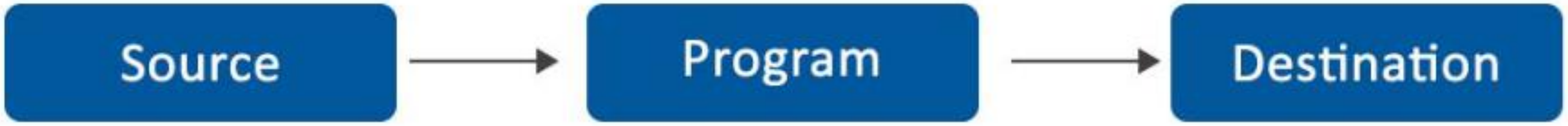
I/O

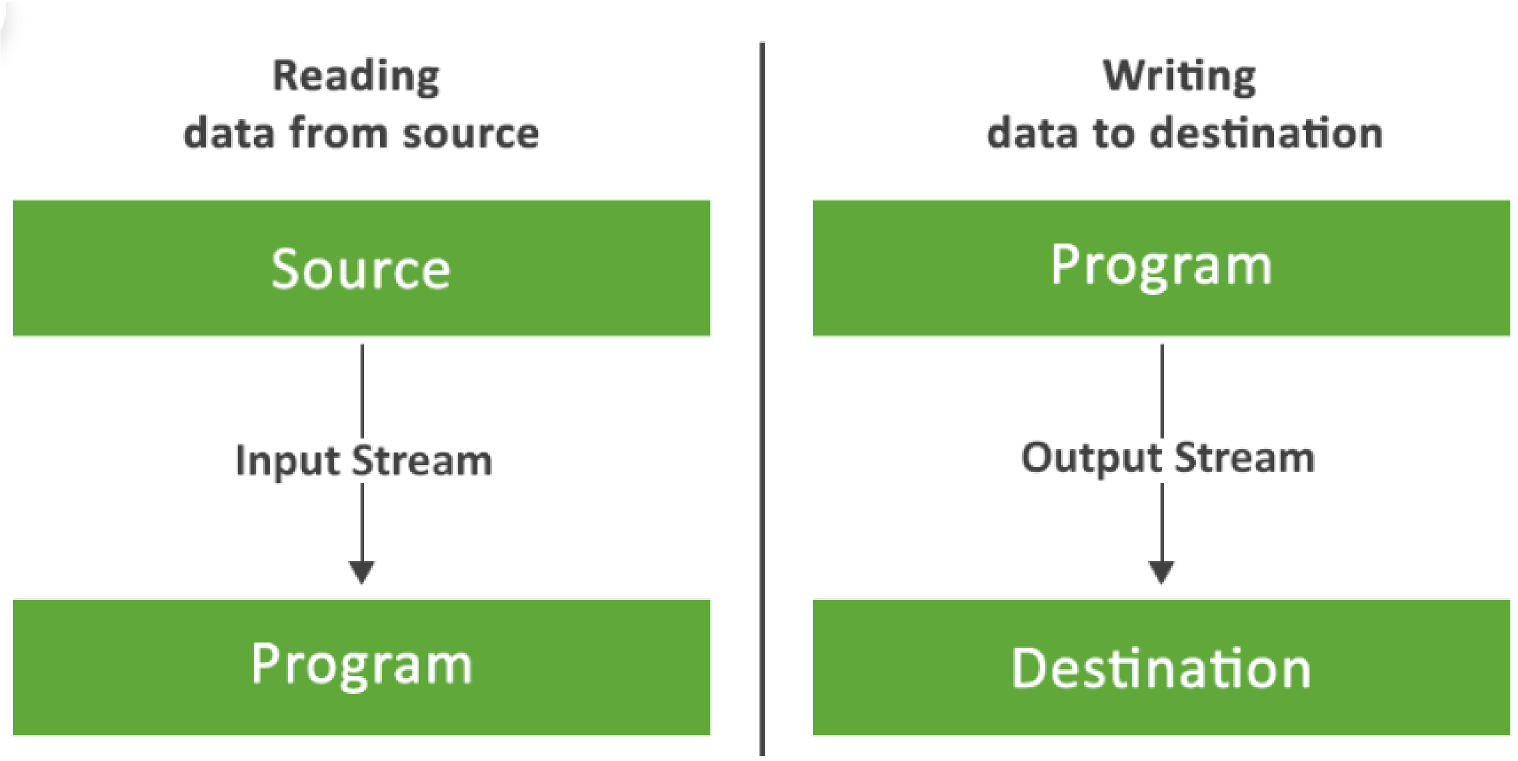
Department of Data Science
& Engineering, DSCA, MIT

Input / Output

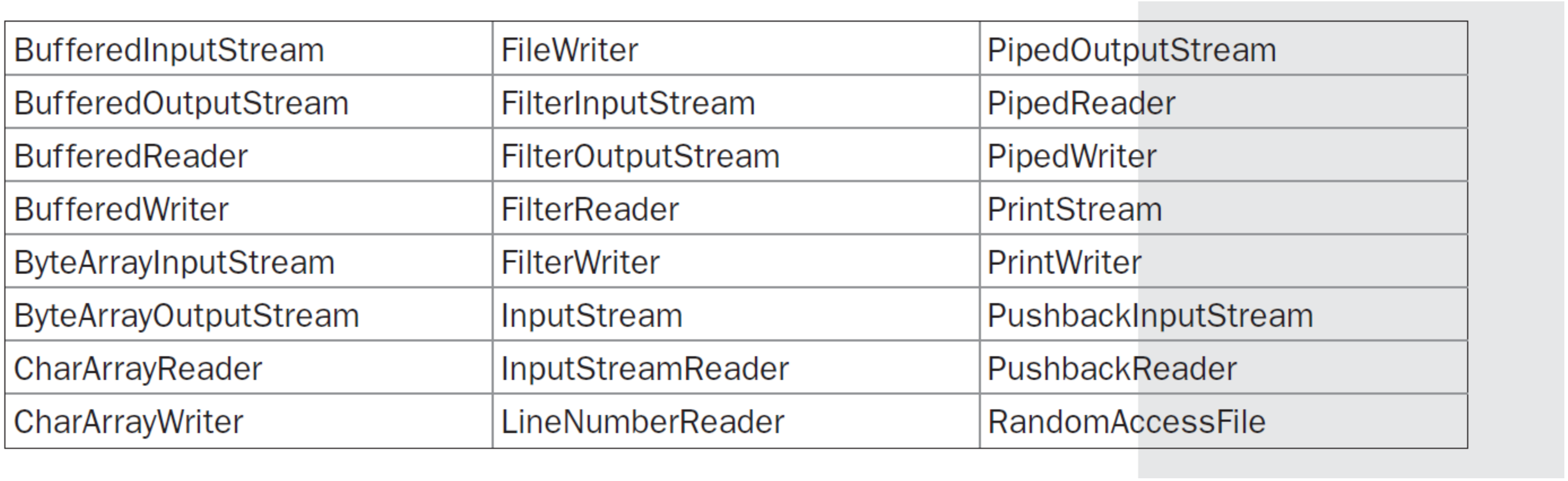
- Java IO API helps in working with files, console and network
- Package---- "**java.io.....**"
- The Java IO package focuses on input and output to files, network streams, etc.
- But the Java IO package does not include classes to open network sockets which are essential for network communication.

Input / Output

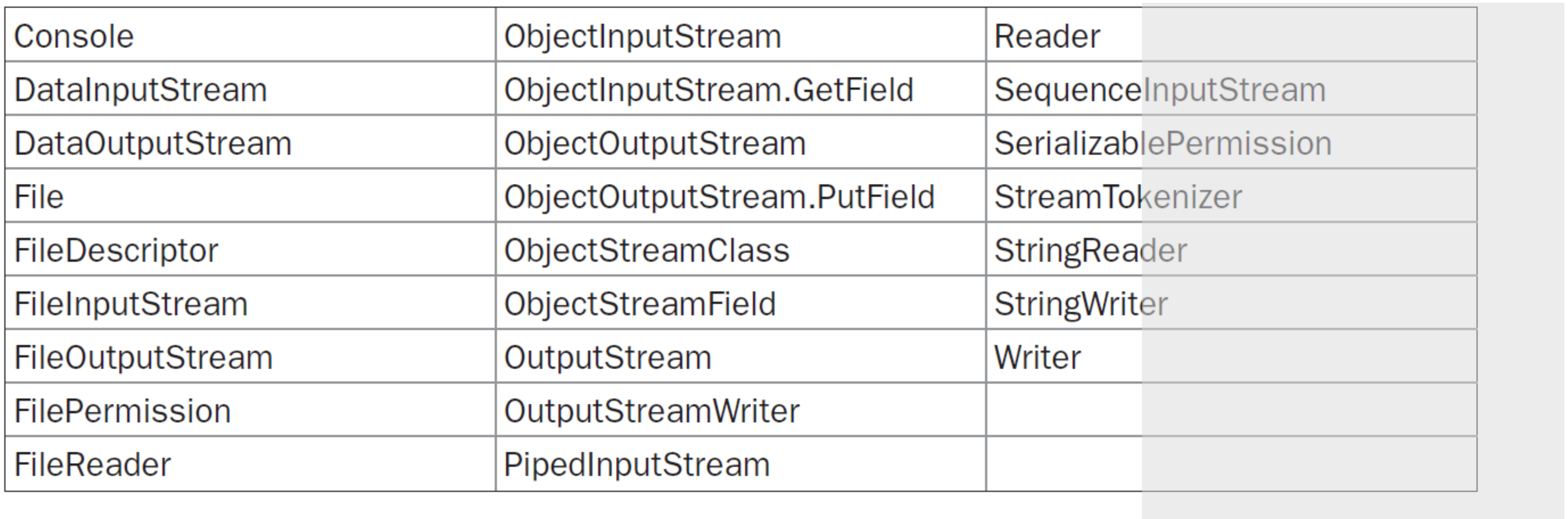




IO classes



BufferedInputStream	FileWriter	PipedOutputStream
BufferedOutputStream	FilterInputStream	PipedReader
BufferedReader	FilterOutputStream	PipedWriter
BufferedWriter	FilterReader	PrintStream
ByteArrayInputStream	FilterWriter	PrintWriter
ByteArrayOutputStream	InputStream	PushbackInputStream
CharArrayReader	InputStreamReader	PushbackReader
CharArrayWriter	LineNumberReader	RandomAccessFile



Console	ObjectInputStream	Reader
DataInputStream	ObjectInputStream.GetField	SequenceInputStream
DataOutputStream	ObjectOutputStream	SerializablePermission
File	ObjectOutputStream.PutField	StreamTokenizer
FileDescriptor	ObjectStreamClass	StringReader
FileInputStream	ObjectStreamField	StringWriter
FileOutputStream	OutputStream	Writer
FilePermission	OutputStreamWriter	
FileReader	PipedInputStream	

File

- Long-term storage of large amounts of data
- Persistent data exists after termination of program
- Files stored on secondary storage devices
  - Magnetic disks
  - Optical disks
  - Magnetic tapes
- Sequential and random access files

File Class

- Provides useful information about a file or directory
- Does not open files or process files
- To obtain or manipulate path, time, date, permissions etc. it has predefined methods.
- Constructor:
  - File(String directoryPath)
  - File(String directoryPath, String fileName)
  - File(File dirObj, String fileName)
  - File(URI uriObj)
- ***Example***: *FileDemo.java*

File Class

Method	**Description**
void deleteOnExit( )	Removes the file associated with the invoking object when the Java Virtual Machine terminates.
long getFreeSpace( )	Returns the number of free bytes of storage available on the partition associated with the invoking object. (Added by Java SE 6.)
long getTotalSpace( )	Returns the storage capacity of the partition associated with the invoking object. (Added by Java SE 6.)
long getUsableSpace( )	Returns the number of usable free bytes of storage available on the partition associated with the invoking object. (Added by Java SE 6.)
boolean isHidden( )	Returns **true** if the invoking file is hidden. Returns **false** otherwise.
boolean setLastModified(long *millisec*)	Sets the time stamp on the invoking file to that specified by *millisec*, which is the number of milliseconds from January 1, 1970, Coordinated Universal Time (UTC).
boolean setReadOnly( )	Sets the invoking file to read-only.

Directory Class

- Directories are also files
- Contains list of files and directories
- For Directory *isDirectory()* returns true

  String[] and list()

  – returns an array of strings that gives the files and directories contained

  File[] and listFiles()

  – Returns array of File objects
- ***Example**: DirectoryDemo.java*

Directory Class

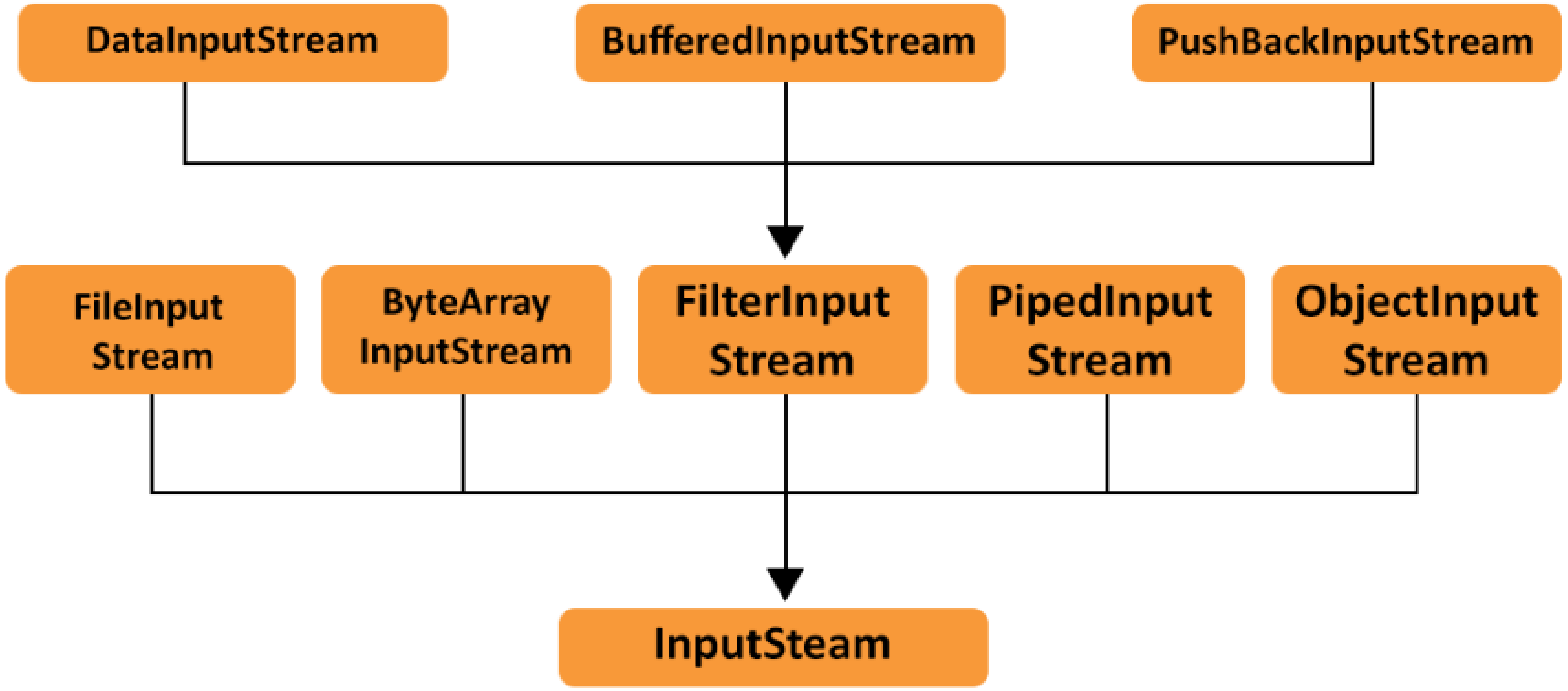
```
import java.io.*;

public class OnlyExt implements FilenameFilter {
  String ext;

  public OnlyExt(String ext) {
    this.ext = "." + ext;
  }

  public boolean accept(File dir, String name) {
    return name.endsWith(ext);
  }
}
```

```
import java.io.*;

class DirListOnly {
  public static void main(String args[]) {
    String dirname = "/java";
    File f1 = new File(dirname);
    FilenameFilter only = new OnlyExt("html");
    String s[] = f1.list(only);

    for (int i=0; i < s.length; i++) {
      System.out.println(s[i]);
    }
  }
}
```

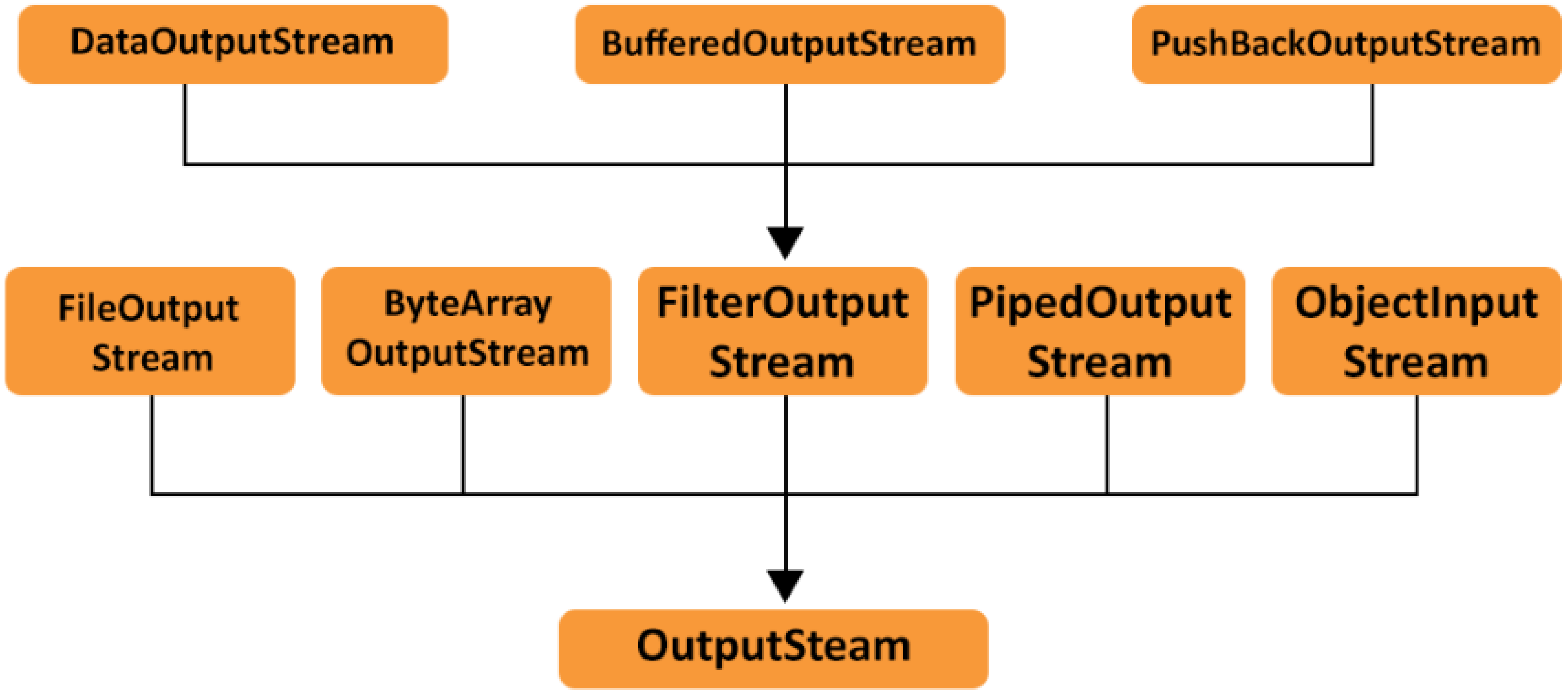
Stream Classes

- Java views a File as a stream of bytes.
  - File ends with end-of-file marker or a specific byte number
  - File as a stream of bytes associated with an object.
  - Java also associates streams with devices
    - System.in, System.out, and System.err
  - Streams can be redirected
- Stream is an abstraction that either produces or consumes information

Stream Classes



Stream Classes



Stream Classes

- Java’s stream-based I/O is built upon four abstract classes.
  - InputStream, OutputStream (for byte streams)
  - Reader, Writer (for character streams)
- They form separate hierarchies
- Use the character stream classes when working with characters or strings
- Use the byte stream classes when working with bytes or other binary objects

Byte Stream Classes

- Topped by ***InputStream*** and ***OutputStream*** classes
- ***InputStream*** is an abstract class that defines Java's model of streaming byte input.

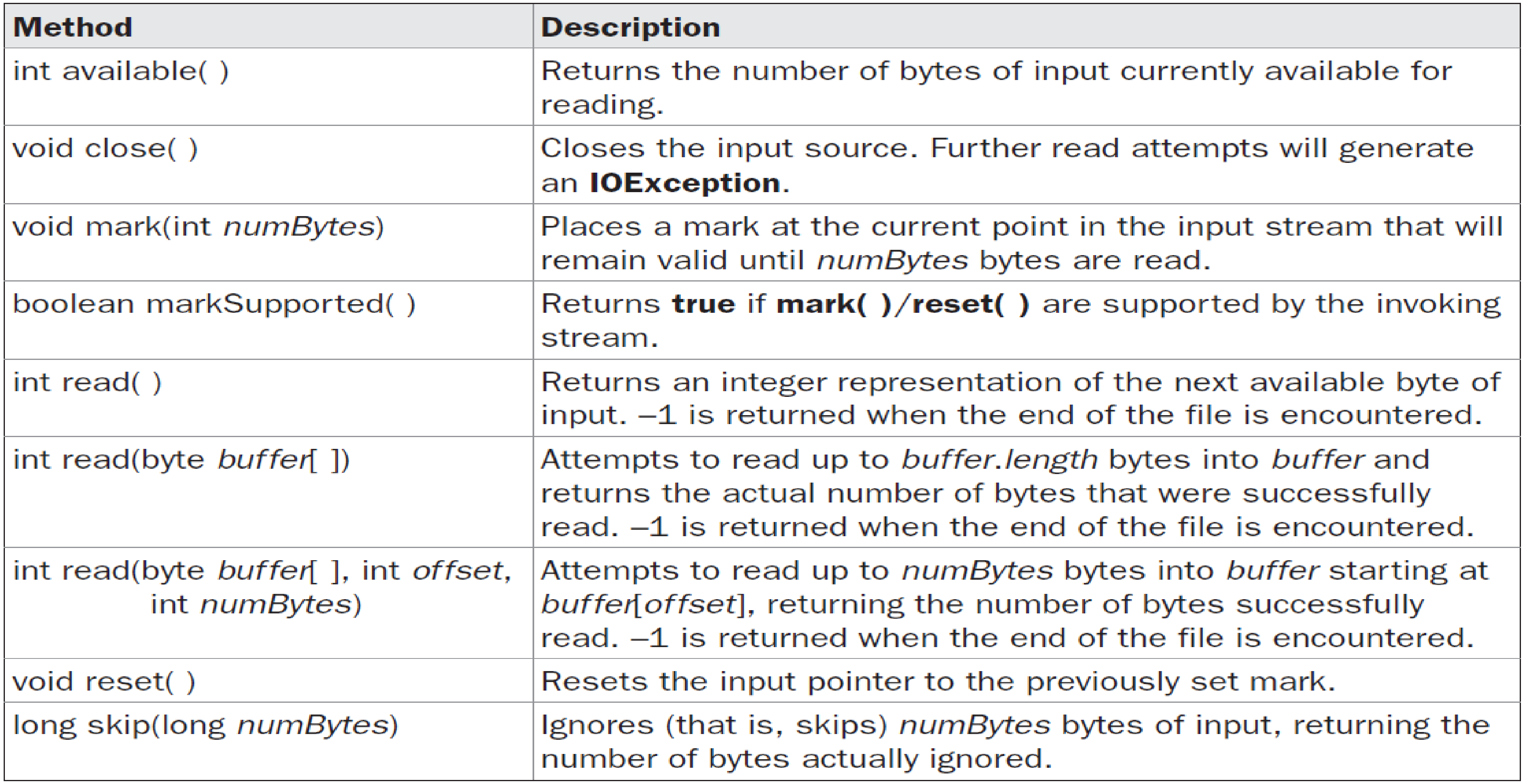
  int available() *void close()* *int read()*

  int read(byte buff[]) *int read(byte buff[], int off, int num)*
- ***OutputStream*** is an abstract class that defines Java's model of streaming byte output.

  void flush() *void close()* *void write(int b)*

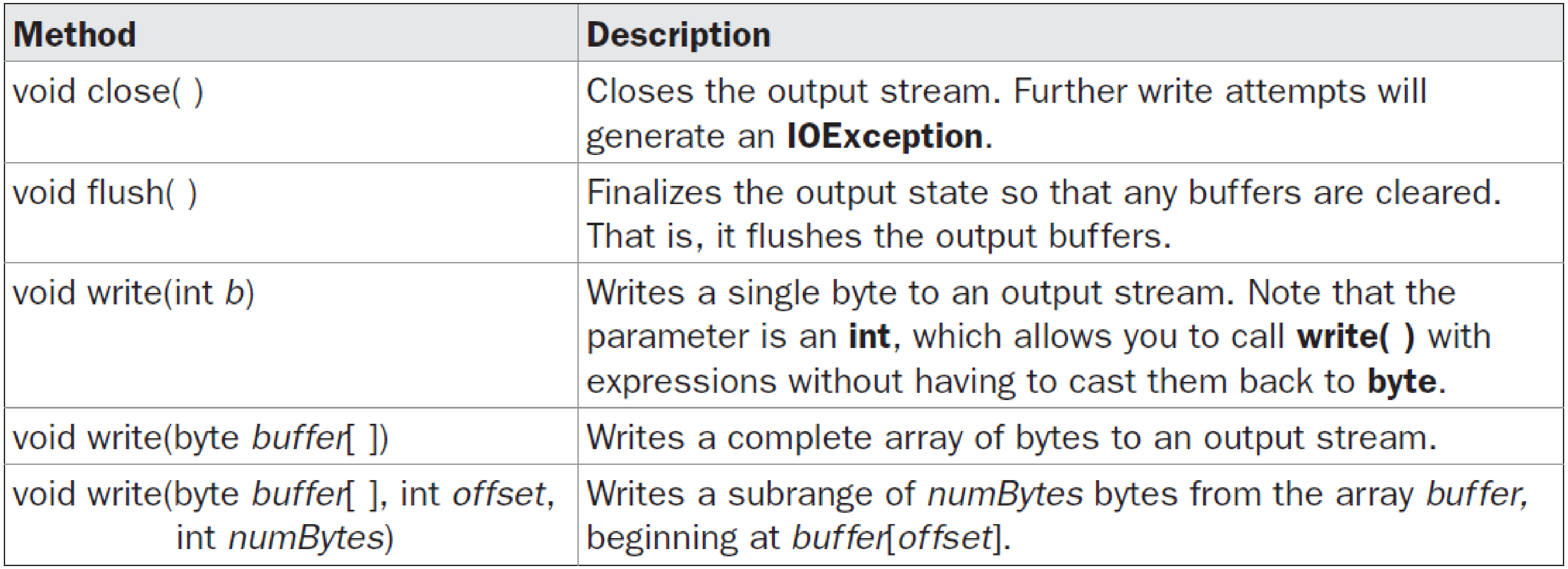
  void write(byte buff[]) *void write(byte buff[], int off, int num)*

InputStream Methods



Method	**Description**
int available( )	Returns the number of bytes of input currently available for reading.
void close( )	Closes the input source. Further read attempts will generate an **IOException**.
void mark(int *numBytes*)	Places a mark at the current point in the input stream that will remain valid until *numBytes* bytes are read.
boolean markSupported( )	Returns **true** if **mark( )**/**reset( )** are supported by the invoking stream.
int read( )	Returns an integer representation of the next available byte of input. –1 is returned when the end of the file is encountered.
int read(byte *buffer*[ ])	Attempts to read up to *buffer.length* bytes into *buffer* and returns the actual number of bytes that were successfully read. –1 is returned when the end of the file is encountered.
int read(byte *buffer*[ ], int *offset*, int *numBytes*)	Attempts to read up to *numBytes* bytes into *buffer* starting at *buffer*[*offset*], returning the number of bytes successfully read. –1 is returned when the end of the file is encountered.
void reset( )	Resets the input pointer to the previously set mark.
long skip(long *numBytes*)	Ignores (that is, skips) *numBytes* bytes of input, returning the number of bytes actually ignored.

OutputStream Methods



Method	Description
void close( )	Closes the output stream. Further write attempts will generate an **IOException**.
void flush( )	Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.
void write(int *b*)	Writes a single byte to an output stream. Note that the parameter is an **int**, which allows you to call **write( )** with expressions without having to cast them back to **byte**.
void write(byte *buffer*[ ])	Writes a complete array of bytes to an output stream.
void write(byte *buffer*[ ], int *offset*, int *numBytes*)	Writes a subrange of *numBytes* bytes from the array *buffer*, beginning at *buffer*[*offset*].

FileInputStream

- ***FileInputStream*** class creates an ***InputStream*** that you can use to read bytes from a file
- Constructors
  - FileInputStream(String filePath)
  - FileInputStream(File fileObj)
- ***Example***: *FileInputStreamDemo.java*

FileOutputStream

- ***FileOutputStream*** class creates an ***OutputStream*** that you can use to write bytes to a file
- Constructors
  - FileOutputStream(String filePath)
  - FileOutputStream(File fileObj)
  - FileOutputStream(String path, boolean append)
  - FileOutputStream(File obj, boolean append)
- ***Example****: FileOutputStreamDemo.java,*

  FileCopyDemo.java

Character Stream Classes

- Topped by ***Reader*** and ***Writer*** classes
- ***Reader*** is an abstract class that defines Java's model of streaming character input

  void close() *int read()* *int read(char buff[])*

  int read(char buff[], int off, int num)

- ***Writer*** is an abstract class that defines Java's model of streaming character output

  void flush() void close() void write(int ch)

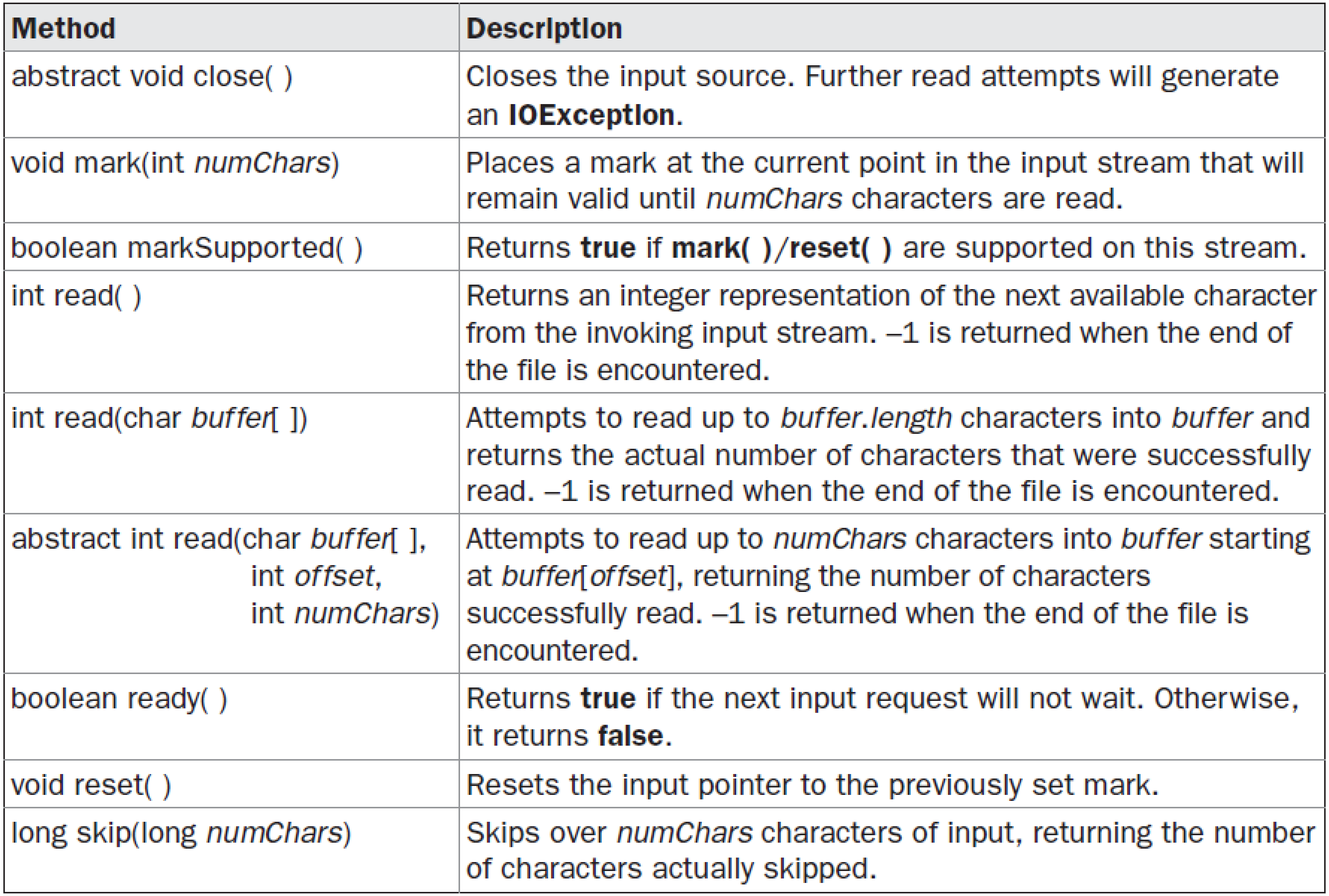
  void write(char buff[]) void write(char buff[], int off, int num)

  void write(String s) *void write(String s, int off, int num)*

FileReader

- ***FileReader*** class creates a ***Reader*** that you can use to read the contents of a file
- Constructors
  - FileReader(String filePath)
  - FileReader(File fileObj)
- ***Example***: *FileReaderDemo.java*

FileReader Methods

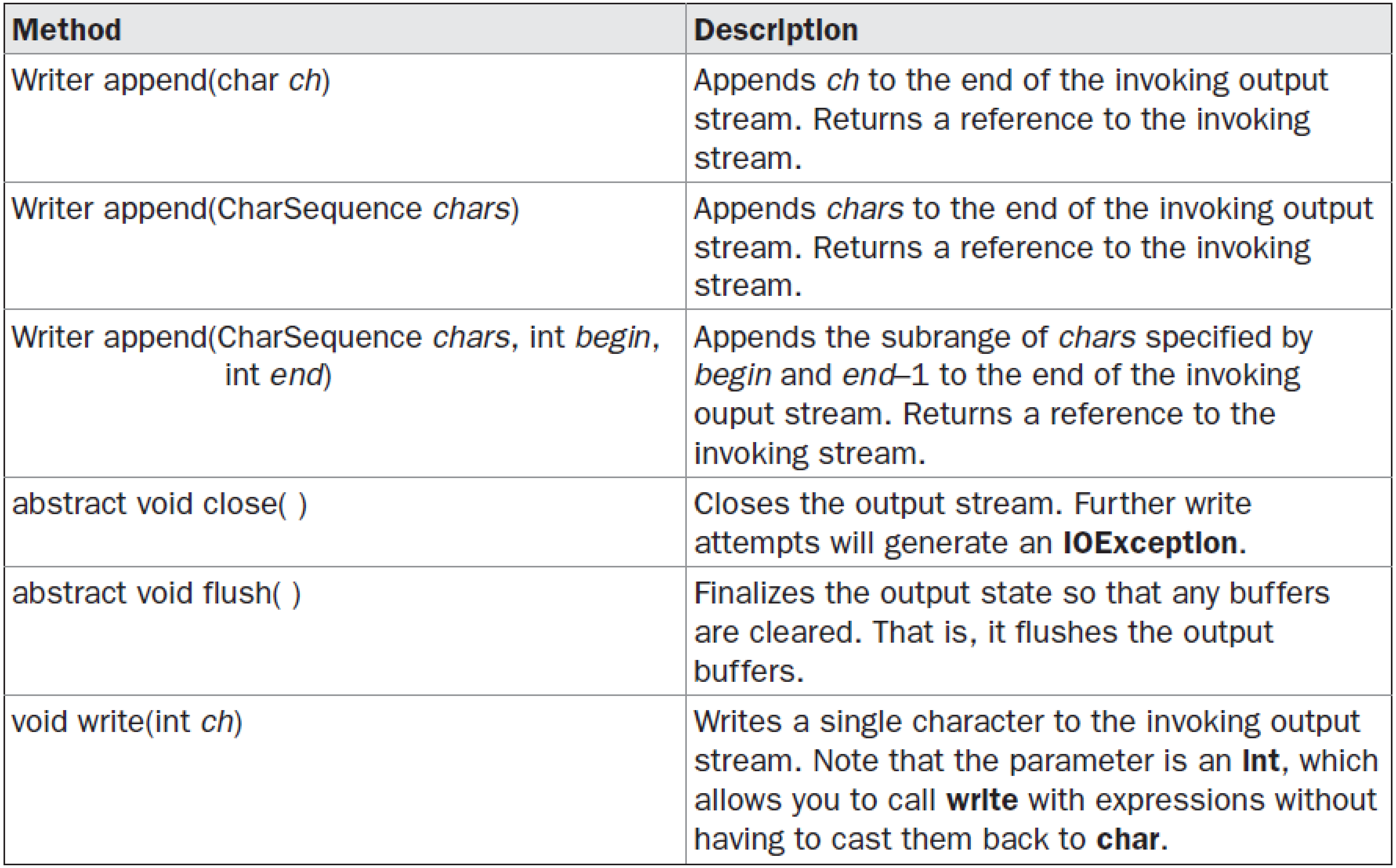


Method	Description
abstract void close( )	Closes the input source. Further read attempts will generate an **IOException**.
void mark(int *numChars*)	Places a mark at the current point in the input stream that will remain valid until *numChars* characters are read.
boolean markSupported( )	Returns **true** if **mark( )**/**reset( )** are supported on this stream.
int read( )	Returns an integer representation of the next available character from the invoking input stream. –1 is returned when the end of the file is encountered.
int read(char *buffer*[ ])	Attempts to read up to *buffer.length* characters into *buffer* and returns the actual number of characters that were successfully read. –1 is returned when the end of the file is encountered.
abstract int read(char *buffer*[ ], int *offset*, int *numChars*)	Attempts to read up to *numChars* characters into *buffer* starting at *buffer*[*offset*], returning the number of characters successfully read. –1 is returned when the end of the file is encountered.
boolean ready( )	Returns **true** if the next input request will not wait. Otherwise, it returns **false**.
void reset( )	Resets the input pointer to the previously set mark.
long skip(long *numChars*)	Skips over *numChars* characters of input, returning the number of characters actually skipped.

FileWriter

- ***FileWriter*** class creates a ***Writer*** that you can use to write to a file
- Constructors
  - FileWriter(String filePath)
  - FileWriter(File fileObj)
  - FileWriter(String path, boolean append)
  - FileWriter(File obj, boolean append)
- ***Example***: *FileWriterDemo.java*

FileWriter Methods



Method	Description
Writer append(char *ch*)	Appends *ch* to the end of the invoking output stream. Returns a reference to the invoking stream.
Writer append(CharSequence *chars*)	Appends *chars* to the end of the invoking output stream. Returns a reference to the invoking stream.
Writer append(CharSequence *chars*, int *begin*, int *end*)	Appends the subrange of *chars* specified by *begin* and *end*–1 to the end of the invoking ouput stream. Returns a reference to the invoking stream.
abstract void close( )	Closes the output stream. Further write attempts will generate an **IOException**.
abstract void flush( )	Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.
void write(int *ch*)	Writes a single character to the invoking output stream. Note that the parameter is an **int**, which allows you to call **write** with expressions without having to cast them back to **char**.

BufferedReader

- ***BufferedReader*** is a ***Reader*** that buffers input
- It improves performance by reducing the number of times data is actually physically read from the input stream
- Constructors
  - BufferedReader(Reader reader)
  - BufferedReader(Reader reader, int buffSize)
- ***Example***: *BufferedReaderDemo.java*

  BufferedReaderDemoWithExecption.java

BufferedWriter

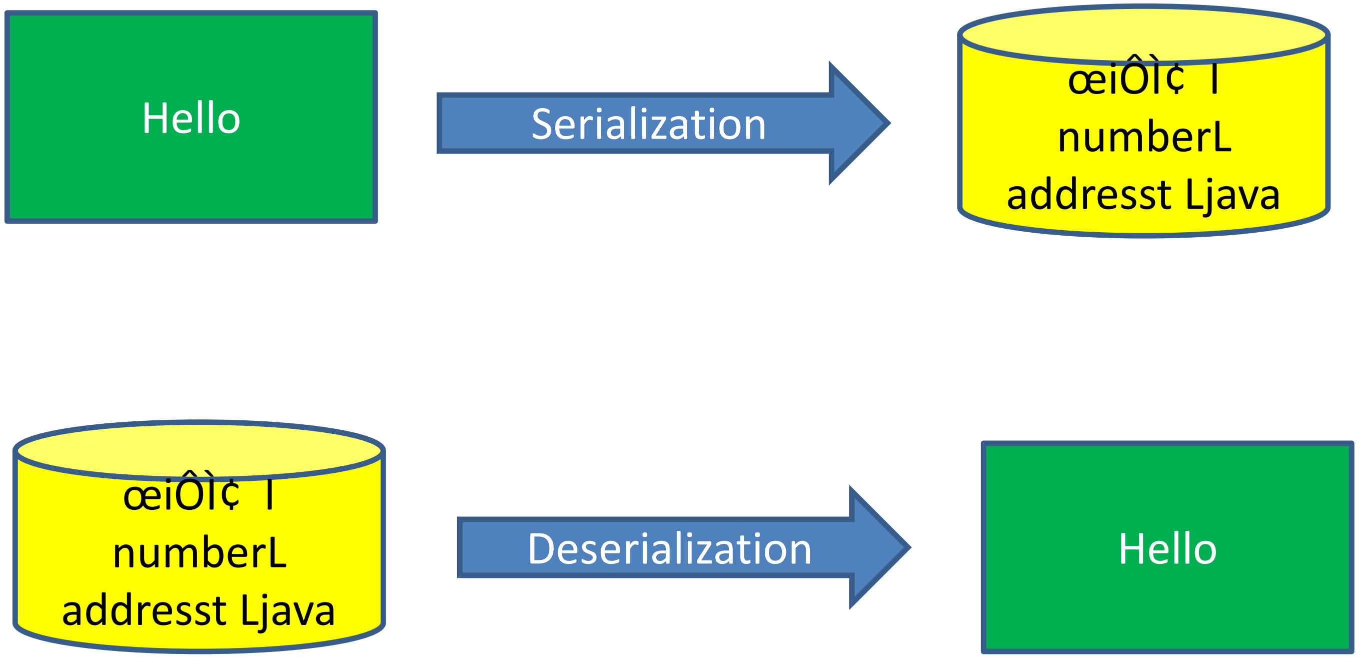
- ***BufferedWriter*** is a ***Writer*** that buffers output
- It improves performance by reducing the number of times data actually physically written to the output stream
- Constructors
  – BufferedWriter(Writer writer)
  – BufferedWriter(Writer writer, int buffSize)
- ***Example***: *BufferedWriterDemo.java*

Serialization

- *Serialization* is the process of writing the state of an object to a byte stream
  - When you create a class and its object, the *object is destroyed* when the programs terminated.
  - What if we want class *without recreating the object*?
- Then we can use Serialization and Deserialization.
- *Serialization*- converting object to ByteStream
- *Deserialization* – converting ByteStream to object
- Serialization can be achieved by implementing ***Serializable*** interface

Serialization

Serialization



Example: *StudentDemo.java*
SerializeDemo.java
DeserializeDemo.java

Object(Input/Output)Stream

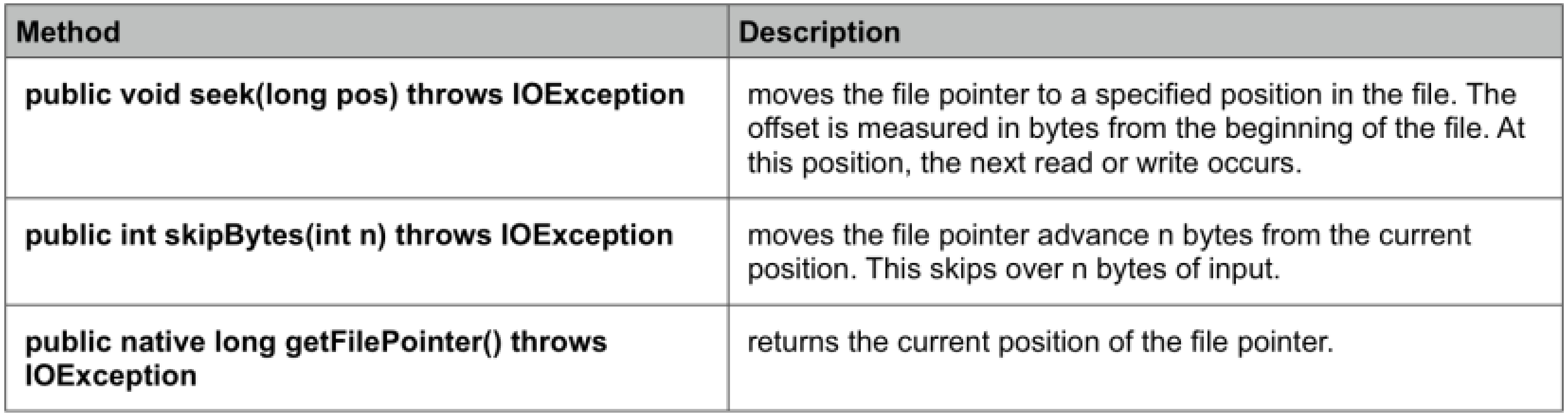
- ***ObjectInputStream*** class extends the ***InputStream*** class
    - It is responsible for reading objects from a stream
- ***ObjectOutputStream*** class extends the ***OutputStream*** class
    - It is responsible for writing objects to a stream

Data(Input/Output)Stream

- ***DataInputStream*** & ***DataOutputStream*** enable to write or read primitive data to or from a stream
- They implement the ***DataOutput*** & ***DataInput*** interfaces respectively
- Constructors
  - DataOutputStream(OutputStream os)
  - DataInputStream(InputStream is)

RandomAccessFile

- This class support both reading and writing to a random access file
- A random access file behaves like a large array of bytes stored in the file system
- The file pointer can be read by the **getFilePointer** method and set by the **seek** method
- ***Example***: *RandomAccessFileDemo.java*



Method	Description
public void seek(long pos) throws IOException	moves the file pointer to a specified position in the file. The offset is measured in bytes from the beginning of the file. At this position, the next read or write occurs.
public int skipBytes(int n) throws IOException	moves the file pointer advance n bytes from the current position. This skips over n bytes of input.
public native long getFilePointer() throws IOException	returns the current position of the file pointer.