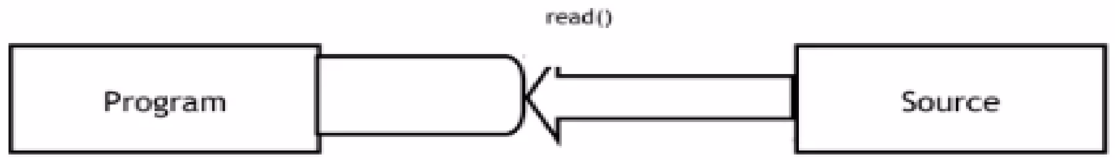
Introduction

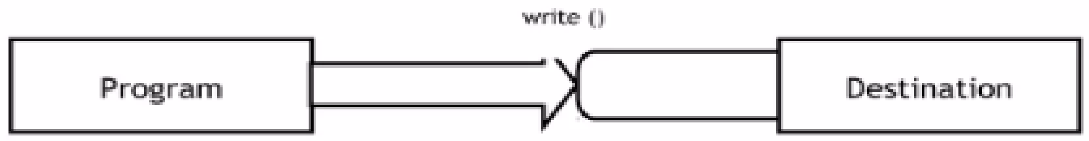
Streams are used to transfer the data between program and source/destination. They transfer the data in unique way irrespective of source/destination. Streams are defined in java.io package in java.

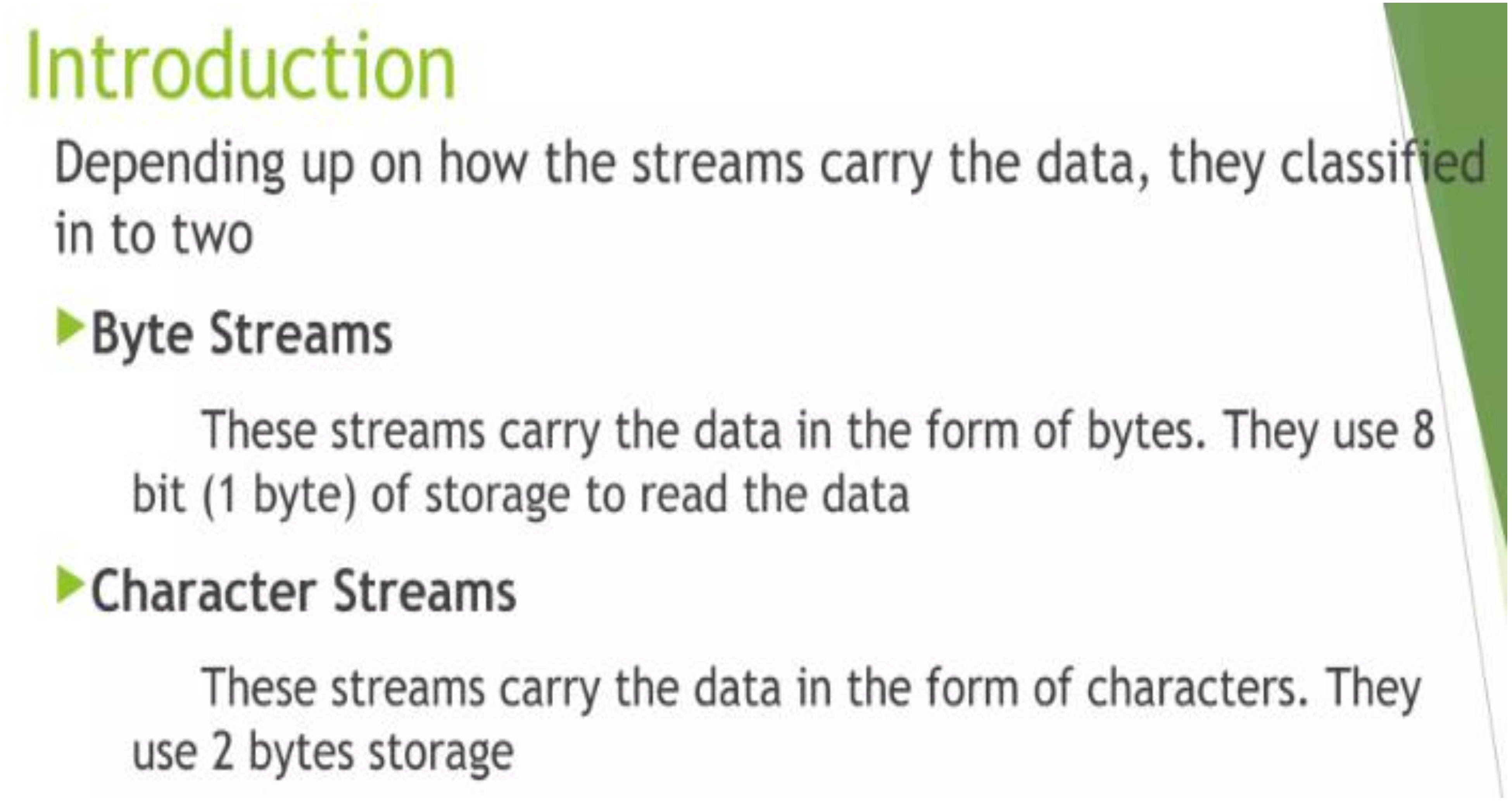
Depending up on the direction of the transfer the streams are classified in to two categories.

- **Input Stream:**



- **Output Stream**





Introduction

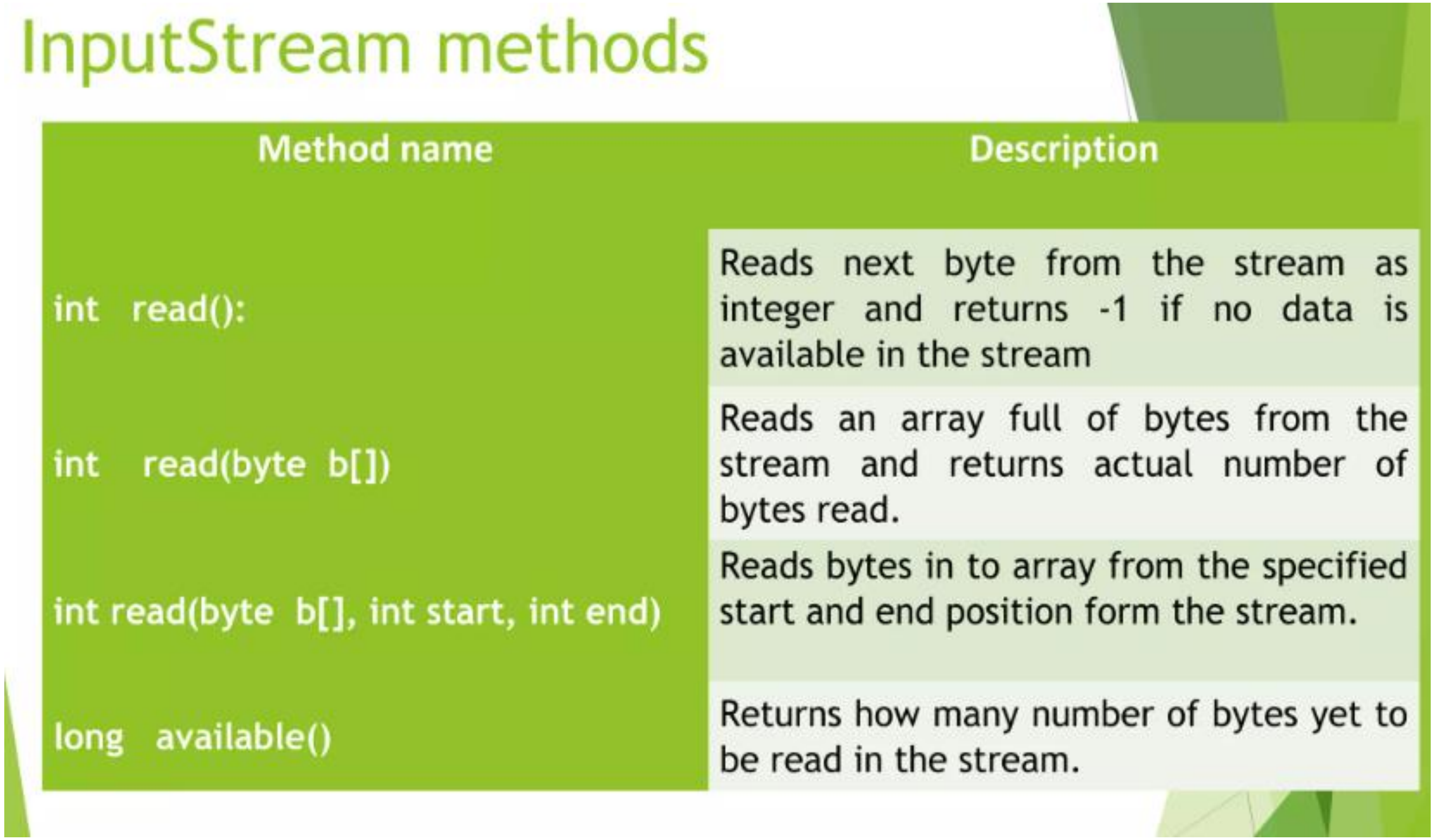
Depending up on how the streams carry the data, they classified in to two

- **Byte Streams**

  These streams carry the data in the form of bytes. They use 8 bit (1 byte) of storage to read the data

- **Character Streams**

  These streams carry the data in the form of characters. They use 2 bytes storage



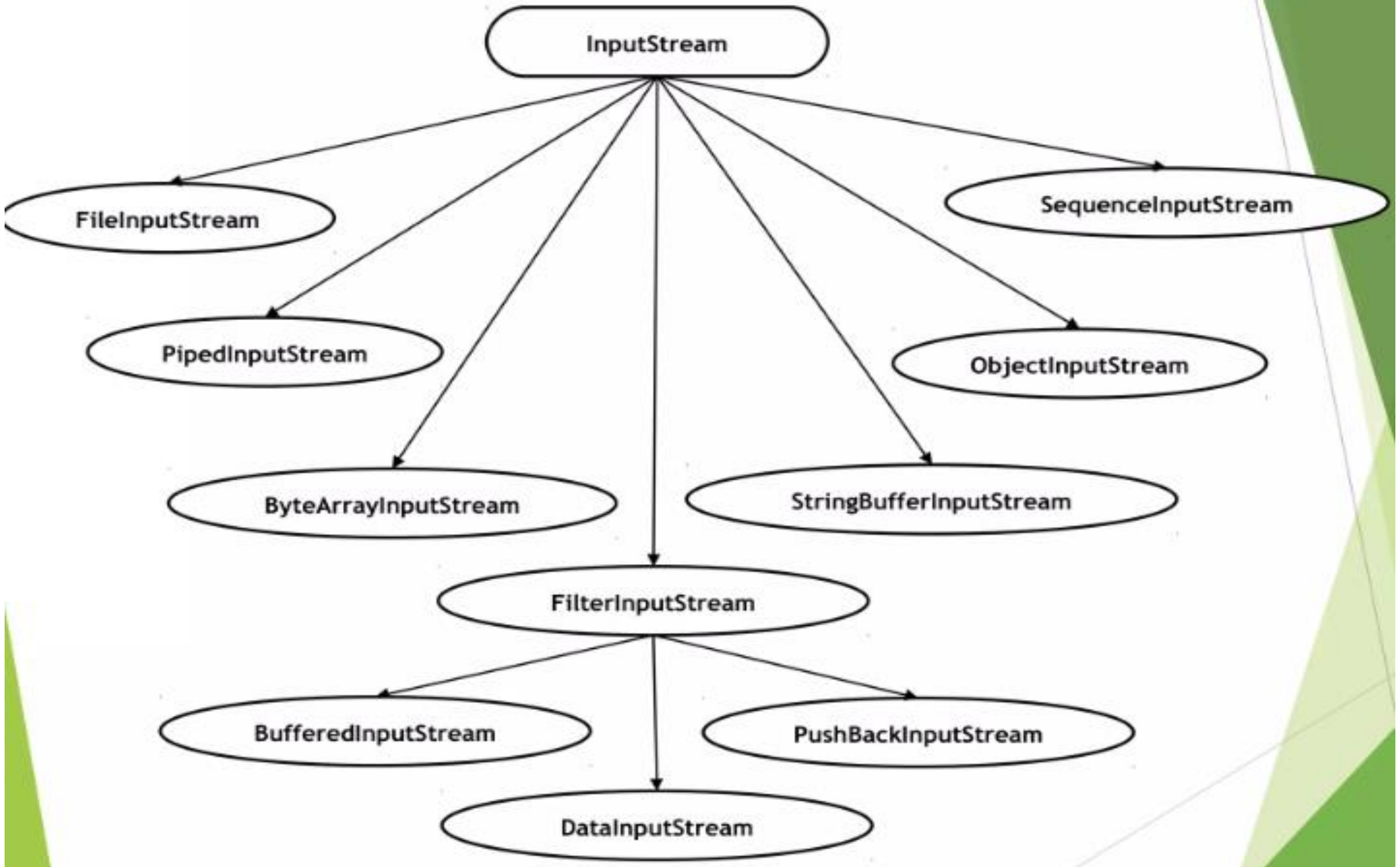
InputStream methods

Method name	Description
int read():	Reads next byte from the stream as integer and returns -1 if no data is available in the stream
int read(byte b[])	Reads an array full of bytes from the stream and returns actual number of bytes read.
int read(byte b[], int start, int end)	Reads bytes in to array from the specified start and end position form the stream.
long available()	Returns how many number of bytes yet to be read in the stream.

InputStream methods

Method name	Description
long skip(long n)	Skips specified number of bytes in the input stream and returns actual number of bytes skipped
void mark(int readLimit)	Marks the current position and it is valid till specified read limit.
void reset()	Moves to the recent marked position or beginning of the stream
void close()	Closes the stream

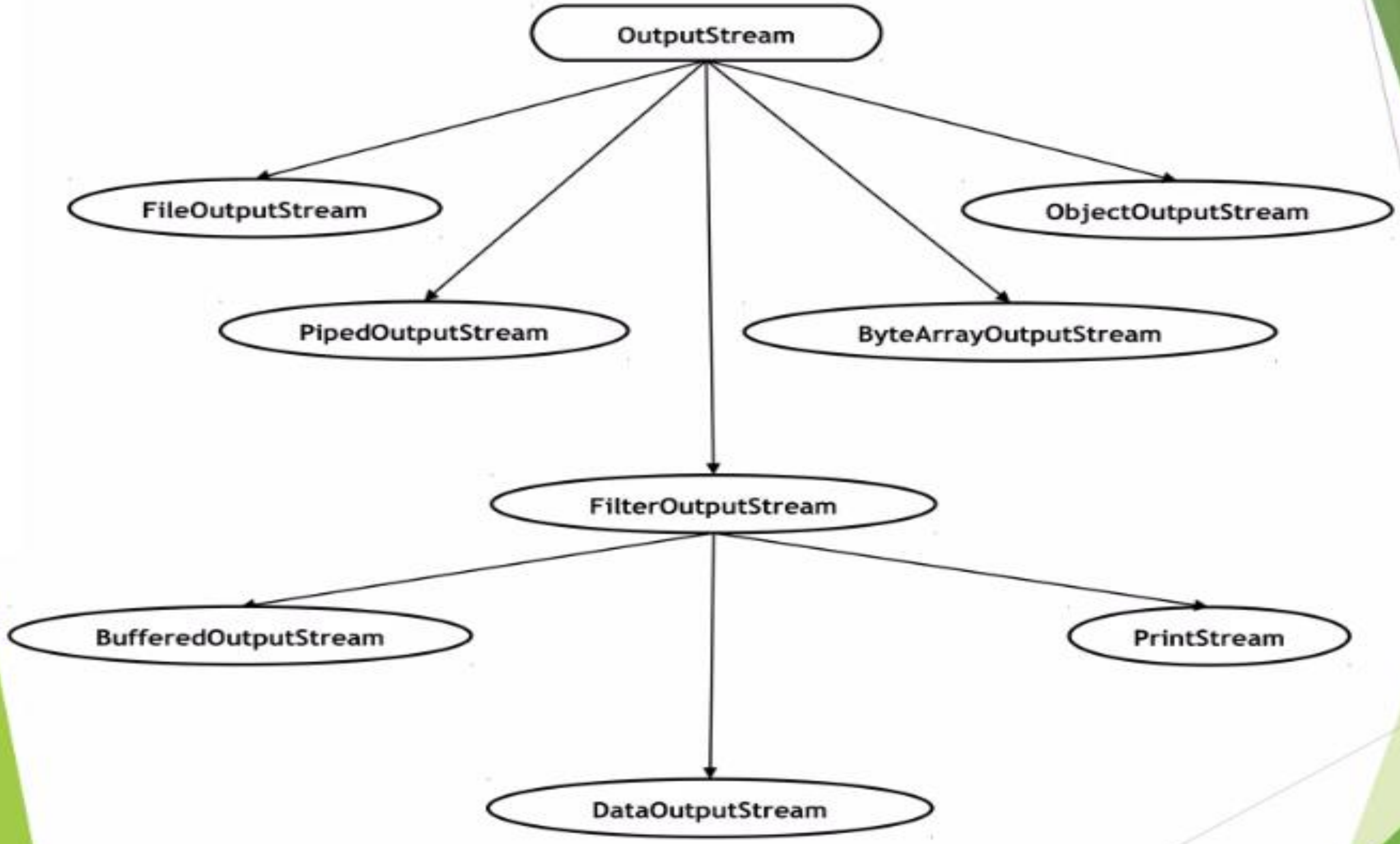
Byte Input Streams

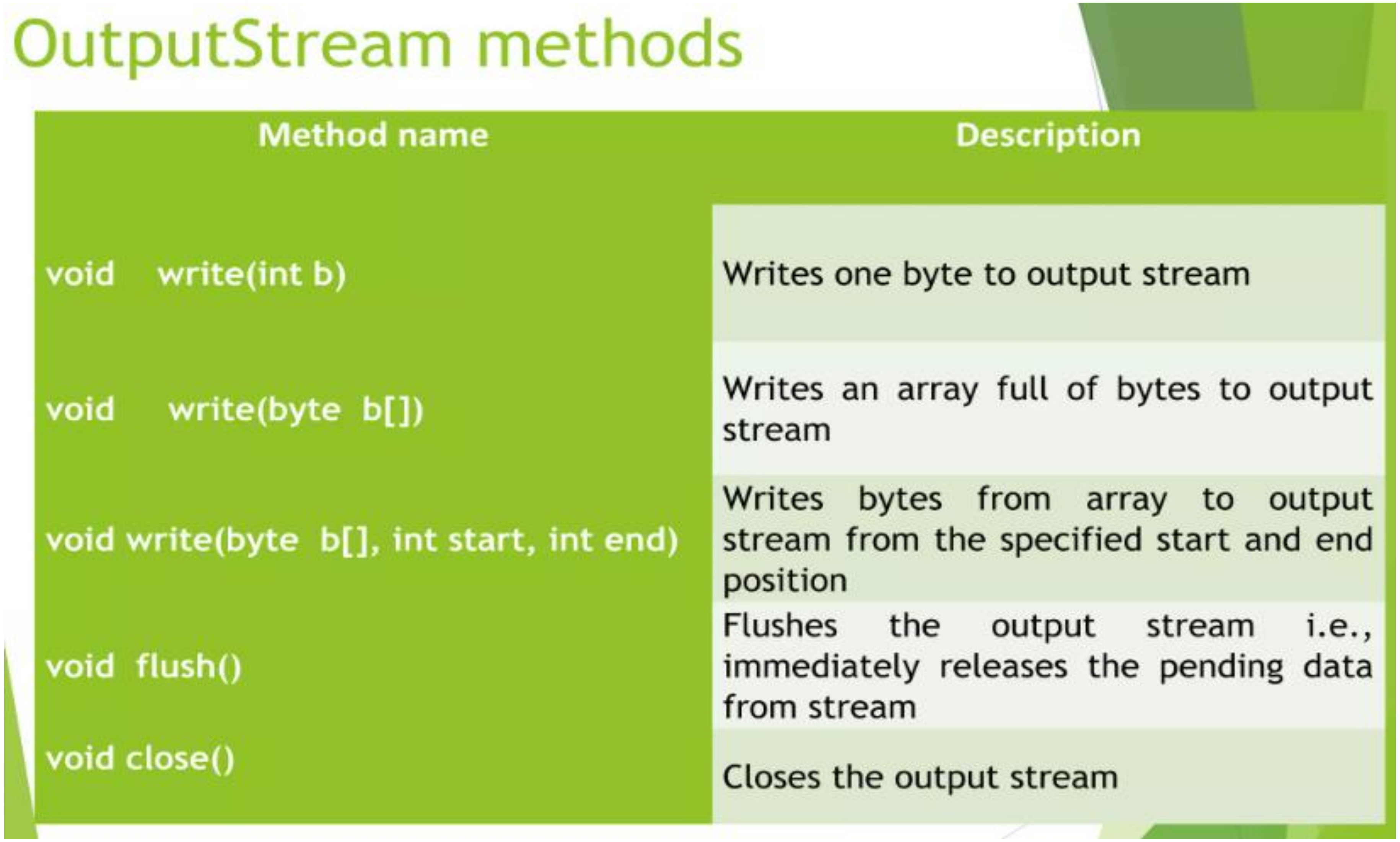


Various Byte Input Streams

Stream Class Name	Use
FileInputStream	used to read from files
PipedInputStream	used to read from pipes
ByteArrayInputStream	used to read from a byte array
StringBufferInputStream	used to read from a String buffer object
ObjectInputStream	used to read objects from an input stream
SequenceInputStream	used to combine two or more input streams
BufferedInputStream	provides buffer facility to the input stream
DataInputStream	used to read primitive data from the input stream
PushBackInputStream	provides un reading facility to the input stream

Byte Output Streams





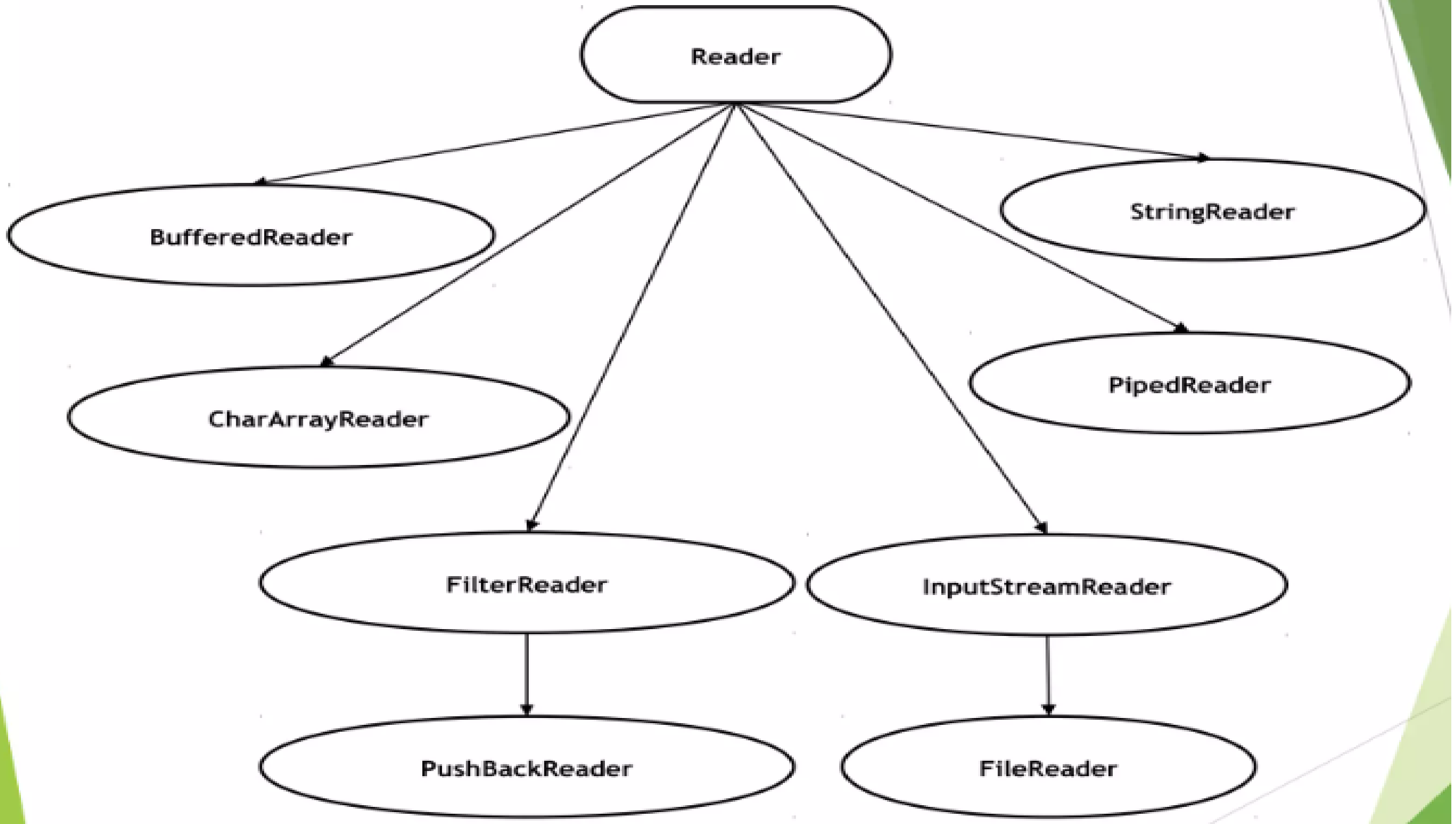
OutputStream methods

Method name	Description
void write(int b)	Writes one byte to output stream
void write(byte b[])	Writes an array full of bytes to output stream
void write(byte b[], int start, int end)	Writes bytes from array to output stream from the specified start and end position
void flush()	Flushes the output stream i.e., immediately releases the pending data from stream
void close()	Closes the output stream

Byte Output Streams

Stream Class Name	Use
FileOutputStream	used to write data into a file
PipedOutputStream	used to write data to a pipe
ByteArrayOutputStream	used to write data to a byte array
ObjectOutputStream	used to write objects to a output stream
BufferedOutputStream	provides buffer facility to the output stream
DataOutputStream	used to write primitive data to an input stream
PrintStream	Used to print any data on output stream

Character Input Streams

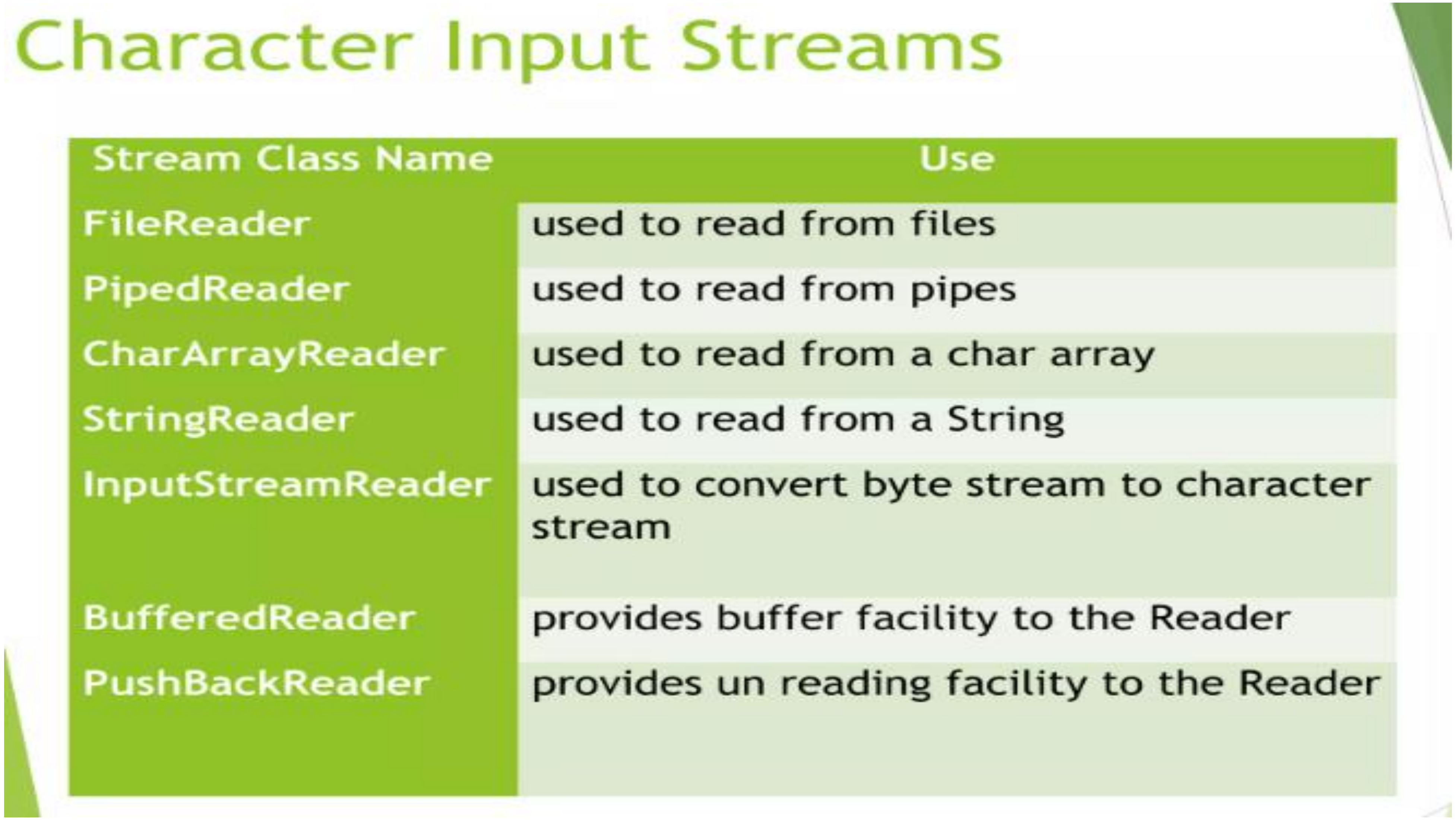


Reader methods

Method name	Description
int read():	Reads next character from the stream as integer and returns -1 if no data is available in the stream.
int read(char c[])	Reads an array full of characters from the stream and returns actual number of characters read
int read(char c[], int start, int end)	Reads characters in to array from the specified start and end position form the stream
long available()	Returns how many number of bytes yet to be read in the stream.

Reader methods

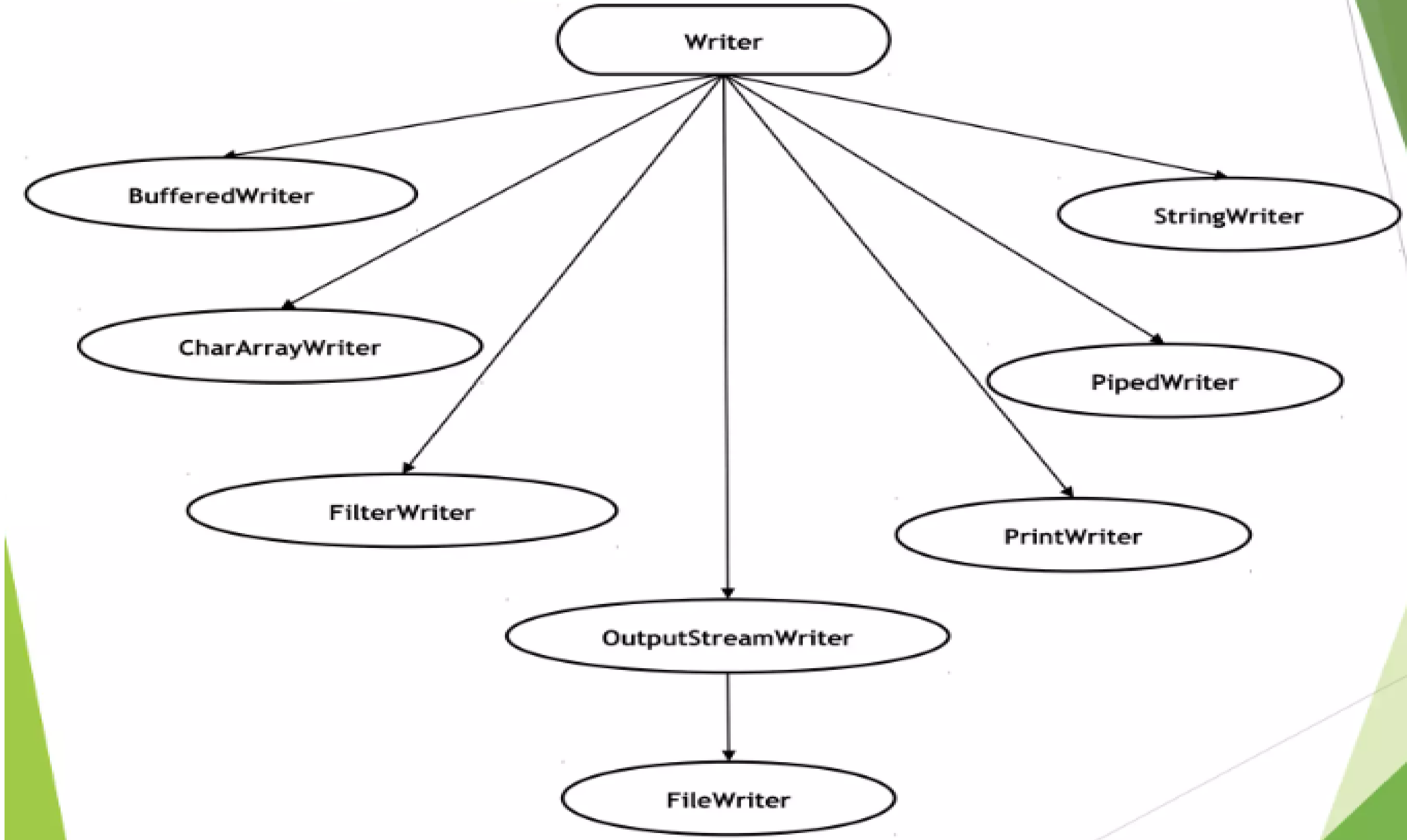
Method name	Description
long skip(long n)	Skips specified number of bytes in the input stream and returns actual number of bytes skipped
void mark(int readLimit)	Marks the current position and it is valid till specified read limit.
void reset()	Moves to the recent marked position or beginning of the stream
void close()	Closes the stream



Character Input Streams

Stream Class Name	Use
FileReader	used to read from files
PipedReader	used to read from pipes
CharArrayReader	used to read from a char array
StringReader	used to read from a String
InputStreamReader	used to convert byte stream to character stream
BufferedReader	provides buffer facility to the Reader
PushBackReader	provides un reading facility to the Reader

Character Output Streams



Writer methods

Method name	Description
void write(int c)	Writes one char to output stream
void write(char c[])	Writes an array full of chars to output stream
void write(char c[], int start, int end)	Writes chars from array to output stream from the specified start and end position
void flush()	Flushes the output stream i.e., immediately releases the pending data from stream
void close()	Closes the output stream

Character Output Streams

Stream Class Name	Use
FileWriter	used to write data into a file
PipedWriter	used to write data to a pipe
CharArrayWriter	used to write data to a byte array
StringWriter	used to write string to a Writer
PrintWriter	used to print any data on Writer
BufferedWriter	provides buffer facility to the Writer
OutputStreamWriter	used to convert character stream to byte stream

Exceptions

- FileNotFoundException

  Raises when an attempt is made to open a file which doesnot exist physically on the disk

- IOException
- Raises when

File class

File is a not a stream class but it is part of java.io package which is used to provide support for files and directories.

Constructors:

- File(String fileName):

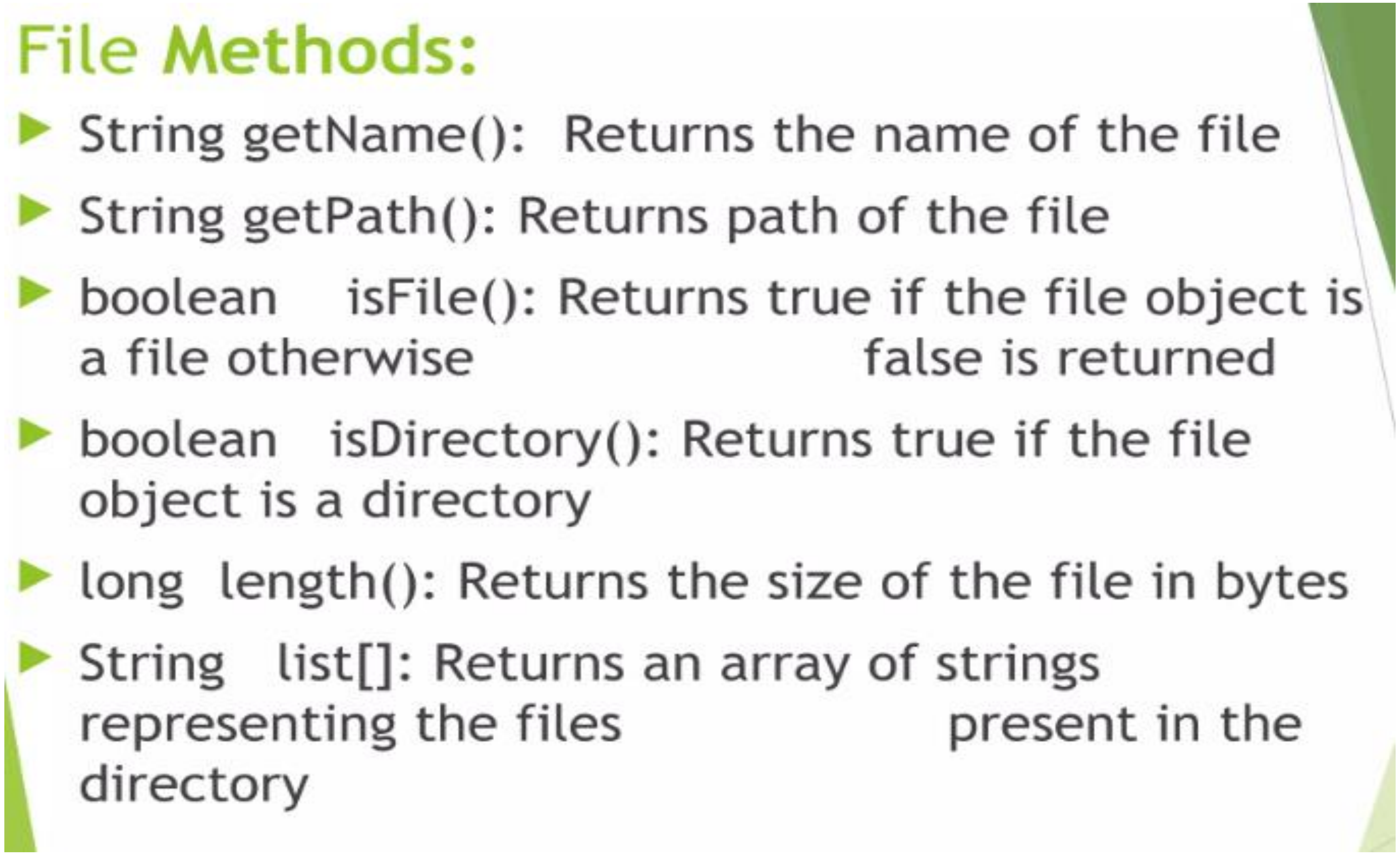
  Constructs a file object with full path of the file

  Eg: File f1=new File ("D:\Programs\Java\FileDemo.java");

- File(String parent, String fileName)

  Constructs a file object for the file at specified path

  Eg: File f2=new File ("D:\Program\Java\","FileDemo.java");



File Methods:

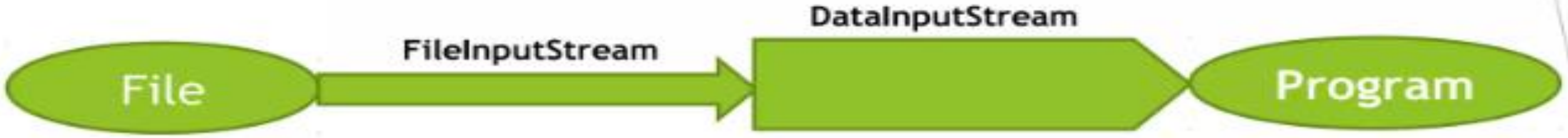
- String getName(): Returns the name of the file
- String getPath(): Returns path of the file
- boolean isFile(): Returns true if the file object is a file otherwise false is returned
- boolean isDirectory(): Returns true if the file object is a directory
- long length(): Returns the size of the file in bytes
- String list[]: Returns an array of strings representing the files present in the directory

Reading & writing files

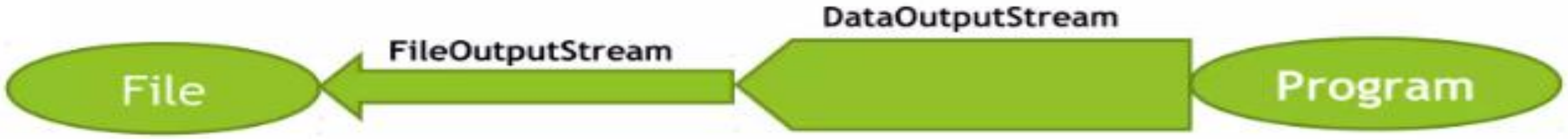
- Reading / Writing Bytes
  - FileInputStream
  - FileOutputStream
- Reading / Writing Characters
  - FileReader
  - FileWriter
- Reading / Writing Primitive data types
  - DataInputStream
  - DataOutputStream

Reading & writing files

Using DataInputStream and DataOutputStream



```
FileInputStream  fis=new  FileInputStream("Student.txt");
DataInputStream  dis=new DataInputStream(fis);
```



```
FileOutputStream  fos=new  FileOutputStream("Student.txt");
DataOutputStream  dos=new DataOutputStream(fos);
```

Output Streams

Java’s basic output class is java.io.OutputStream:

```
 public abstract class OutputStream
```

This class provides the fundamental methods needed to write data. These are:

```
public abstract void write(int b) throws IOException
public void write(byte[] data) throws IOException
public void write(byte[] data, int offset, int length)
      throws IOException
public void flush() throws IOException
public void close() throws IOException
```

Input Streams

Java’s basic input class is java.io.InputStream:

```
public abstract class InputStream
```

This class provides the fundamental methods needed to read data as raw bytes. These are:

```
public abstract int read() throws IOException
public int read(byte[] input) throws IOException
public int read(byte[] input, int offset, int length) throws IOException
public long skip(long n) throws IOException
public int available() throws IOException
public void close() throws IOException
```

Marking and Resetting

The InputStream class also has three less commonly used methods that allow programs to back up and reread data they’ve already read. These are:

```
public void mark(int readAheadLimit)
public void reset() throws IOException
public boolean markSupported()
```

Filter Streams

The filters come in two versions:

The filter streams

The readers and Writers

Buffered Streams

- The BufferedOutputStream class stores written data in a buffer(a protected byte array field named buf) until the buffer is full or the stream is flushed. Then it writes the dataonto the underlying output stream all at once.
- A single write of many bytes is almost always much faster than many small writes that add up to the same thing. This is espe-cially true of network connections because each TCP segment or UDP packet carries afinite amount of overhead, generally about 40 bytes' worth. This means that sending 1 kilobyte of data 1 byte at a time actually requires sending 40 kilobytes over the wire, whereas sending it all at once only requires sending a little more than 1K of data.
- Most network cards and TCP implementations provide some level of buffering themselves, so the real numbers aren't quite this dramatic. Nonetheless, buffering network output is generally a huge performance win.

- The BufferedInputStream class also has a protected byte array named buf that serves as a buffer. When one of the stream’s read() methods is called, it first tries to get the requested data from the buffer. Only when the buffer runs out of data does the stream read from the underlying source.
- At this point, it reads as much data as it can from the source into the buffer, whether it needs all the data immediately or not. Data that isn’t used immediately will be available for later invocations of read(). When reading files from a local disk, it’s almost as fast to read several hundred bytes of data from the underlying stream as it is to read one byte of data. Therefore, buffering can substantially improve performance. The gain is less obvious on network connections where the bot- tleneck is often the speed at which the network can deliver data rather than the speed at which the network interface delivers data to the program or the speed at which the program runs. Nonetheless, buffering input rarely hurts and will become more impor- tant over time as network speeds increase.
  - BufferedInputStream has two constructors, as does BufferedOutputStream:
  - public BufferedInputStream(InputStream in)
  - public BufferedInputStream(InputStream in, int bufferSize)
  - public BufferedOutputStream(OutputStream out)
  - public BufferedOutputStream(OutputStream out, int bufferSize)

PrintWriter

The PrintWriter class is a replacement for Java 1.0’s PrintStream class that properly handles multibyte character sets and international text.
Sun originally planned to dep-recate PrintStream in favor of PrintWriter but backed off when it realized this step would invalidate too much existing code, especially code that depended on System.out. Nonetheless, new code should use PrintWriter instead of PrintStream.
Aside from the constructors, the PrintWriter class has an almost identical collection of methods to PrintStream. These include:

```
public PrintWriter(Writer out)
public PrintWriter(Writer out, boolean autoFlush)
public PrintWriter(OutputStream out)
public PrintWriter(OutputStream out, boolean autoFlush)
public void flush()
public void close()
public boolean checkError()
public void write(int c)
public void write(char[] text, int offset, int length)
public void write(char[] text)
public void write(String s, int offset, int length)
public void write(String s)
public void print(boolean b)
public void print(char c)
public void print(int i)
public void print(long l)
public void print(float f)
public void print(double d)
public void print(char[] text)
public void print(String s)
public void print(Object o)
public void println()
public void println(boolean b)
public void println(char c)
public void println(int i)
public void println(long l)
public void println(float f)
public void println(double d)
public void println(char[] text)
public void println(String s)
public void println(Object o)
```