JAVA I/O CLASSES

SUBJECT:JAVA PROGRAMMING

Saviya Varghese
Dept of BCA

2020-21

Java i/o classes

- The java i/o package (java.io)provides an extensive set of classes for handling input and output to and from many different devices.
- **Java I/O** (Input and Output) is used *to process the input* and *produce the output*.
- The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java..
- Java uses the concept of a stream to make I/O operation fast.
- The java.io package contains all the classes required for input and output operations.
- We can perform **file handling in Java** by Java I/O API.

1.Java I/O Streams

- Java programs perform I/O through streams.
- A stream is a logical entity that can either collect from an i/p device or hand over to an o/p device any data or information as a flow of bytes or characters.
- There are two kinds of Streams −
- **InPutStream** − The InputStream is used to read data from a source.
- **OutPutStream** − The OutputStream is used for writing data to a destination.


Source
Program
Destination

1.1 Byte and Character Streams

- 2 types of java streams-byte stream and character stream.
- Byte streams are used for reading or writing binary data.
- Character streams are used for handling character type data.
- Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, **FileInputStream** and **FileOutputStream**.

Byte streams

- Byte Stream Classes are in divided in two groups - InputStream Classes - These classes are subclasses of an abstract class, InputStream and they are used to read bytes from a source(file, memory or console).
- OutputStream Classes - These classes are subclasses of an abstract class, OutputStream and they are used to write bytes to a destination(file, memory or console).

Character streams

- One of the limitations of byte stream classes is that it can handle only 8-bit bytes and cannot work directly with Unicode characters.
- To overcome this limitation, character stream classes have been introduced in java.io package to match the byte stream classes.
- The character stream classes support 16-bit Unicode characters, performing operations on characters, character arrays, or strings, reading or writing buffer at a time.
- Character stream classes are divided into two stream classes namely, Reader class and Writer class.
- Character stream I/O automatically translates this internal format to and from the local character set

Reader Classes

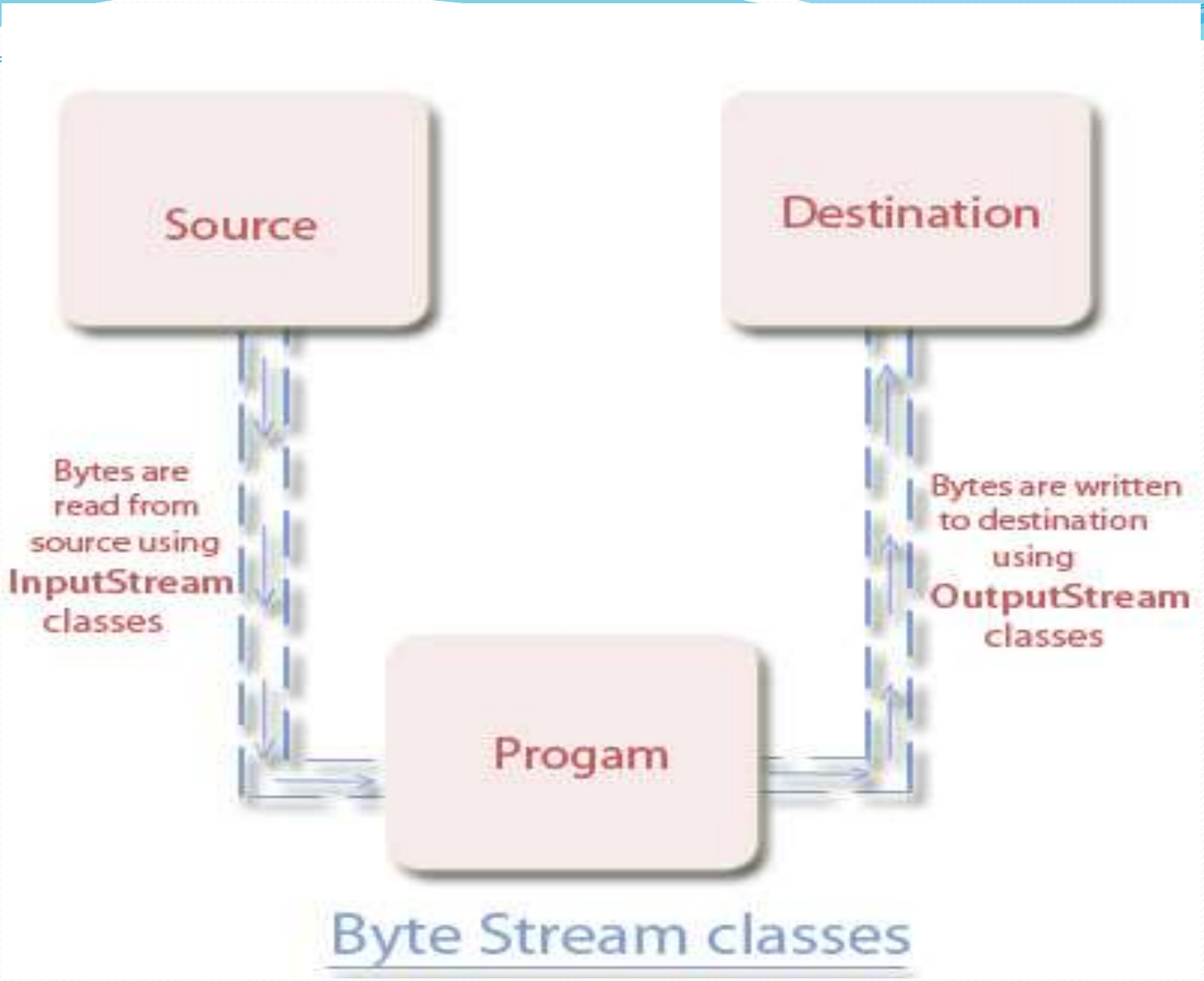
- Reader classes are used to read 16-bit unicode characters from the input stream.
- The Reader class is the superclass for all character-oriented input stream classes.
- All the methods of this class throw an IOException.
- Being an abstract class, the Reader class cannot be instantiated hence its subclasses are used

Table Reader Classes

Class	Description
BufferedReader	contains methods to read characters from the buffer
CharArrayReader	contains methods to read characters from a character array
FileReader	contains methods to read from a file
FilterReader	contains methods to read from underlying character-input stream
InputStreamReader	contains methods to convert bytes to characters
PipedReader	contains methods to read from the connected piped output stream
StringReader	contains methods to read from a string

Table Reader Class Methods

Method	Description
int read()	returns the integral representation of the next available character of input. It returns -1 when end of file is encountered
int read (char buffer [])	attempts to read buffer. length characters into the buffer and returns the total number of characters successfully read. It returns -I when end of file is encountered
int read (char buffer [], int loc, int nChars)	attempts to read 'nChars' characters into the buffer starting at buffer [loc] and returns the total number of characters successfully read. It returns -1 when end of file is encountered
void mark(int nChars)	marks the current position in the input stream until 'nChars' characters are read
void reset ()	resets the input pointer to the previously set mark
long skip (long nChars)	skips 'nChars' characters of the input stream and returns the number of actually skipped characters
boolean ready ()	returns true if the next request of the input will not have to wait, else it returns false
void close ()	closes the input source. If an attempt is made to read even after closing the stream then it generates IOException


Source
Destination
Bytes are read from source using InputStream classes
Bytes are written to destination using OutputStream classes
Progam
Byte Stream classes

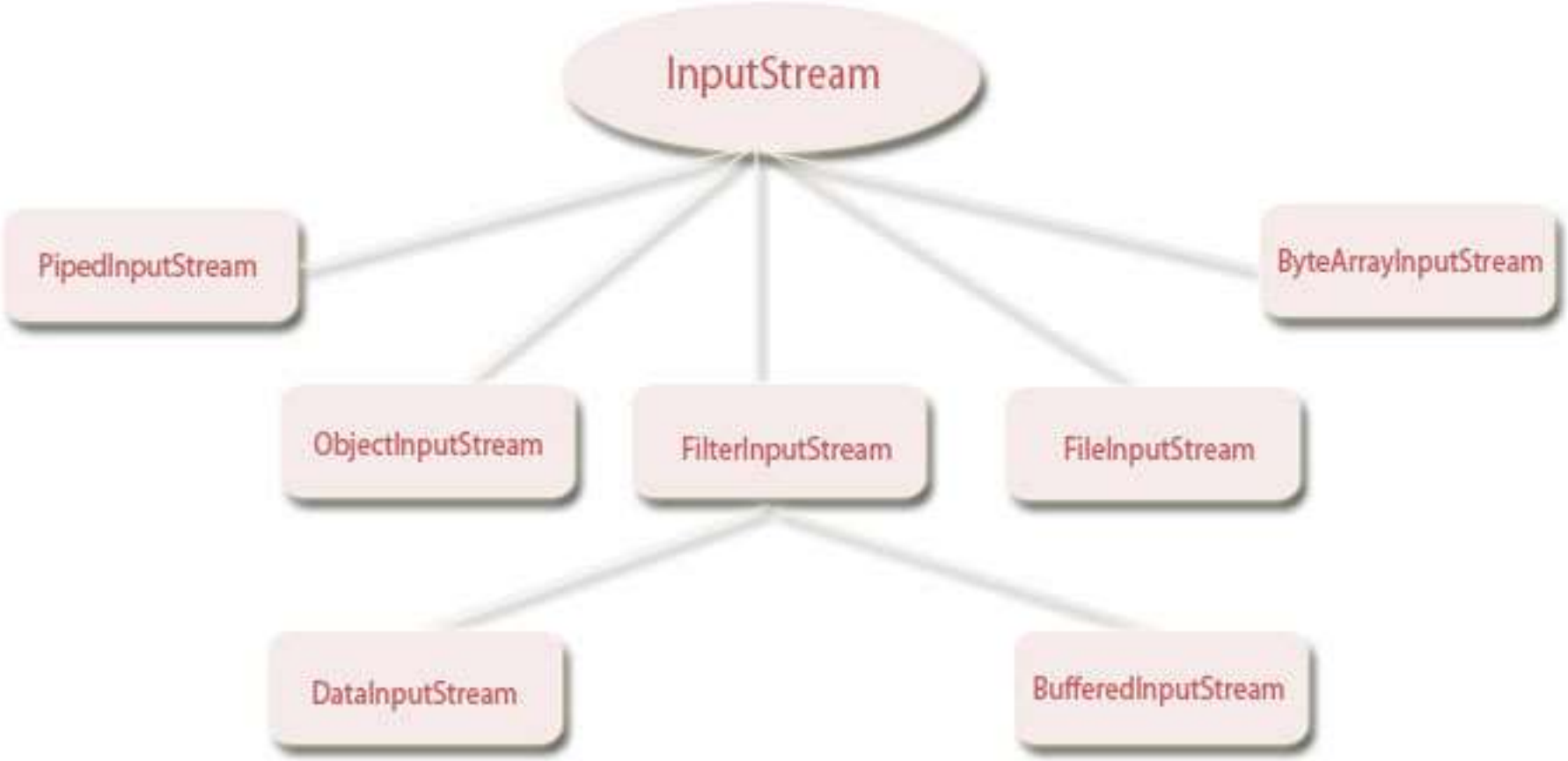
InputStream

InputStream class is a base class of all the classes that are used to read bytes from a file, memory or console.

InputStream is an abstract class and hence we can't create its object but we can use its subclasses for reading bytes from the input stream.

OutputStream

- OutputStream class is a base class of all the classes that are used to write bytes to a file, memory or console.
- OutputStream is an abstract class and hence we can't create its object but we can use its subclasses for writing bytes to the output stream.
- In the diagram below we have shown the hierarchy of OutputStream class and some of its important subclasses that are used to write bytes.



Hierarchy of InputStream classes

Table Input Stream Classes

Class	Description
BufferedInputStream	contains methods to read bytes from the buffer (memory area)
ByteArrayInputStream	contains methods to read bytes from a byte array
DataInputStream	contains methods to read Java primitive data types
FileInputStream	contains methods to read bytes from a file
FilterInputStream	contains methods to read bytes from other input streams which it uses as its basic source of data
ObjectInputStream	contains methods to read objects
PipedInputStream	contains methods to read from a piped output stream. A piped input stream must be connected to a piped output stream
SequenceInputStream	contains methods to concatenate multiple input streams and then read from the combined stream

Method	Description
int read()	returns the integral representation of the next available byte of input. It returns -1 when end of file is encountered
int read (byte buffer [])	attempts to read buffer. length bytes into the buffer and returns the total number of bytes successfully read. It returns -1 when end of file is encountered
int read (byte buffer [], int loc, int nBytes)	attempts to read 'nBytes' bytes into the buffer starting at buffer [loc] and returns the total number of bytes successfully read. It returns -1 when end of file is encountered
int available ()	returns the number of bytes of the input available for reading
Void mark(int nBytes)	marks the current position in the input stream until 'nBytes' bytes are read
void reset ()	Resets the input pointer to the previously set mark
long skip (long nBytes)	skips 'nBytes' bytes of the input stream and returns the number of actually skippedbyte
void close ()	closes the input source. If an attempt is made to read even after closing the stream then it generates IOException