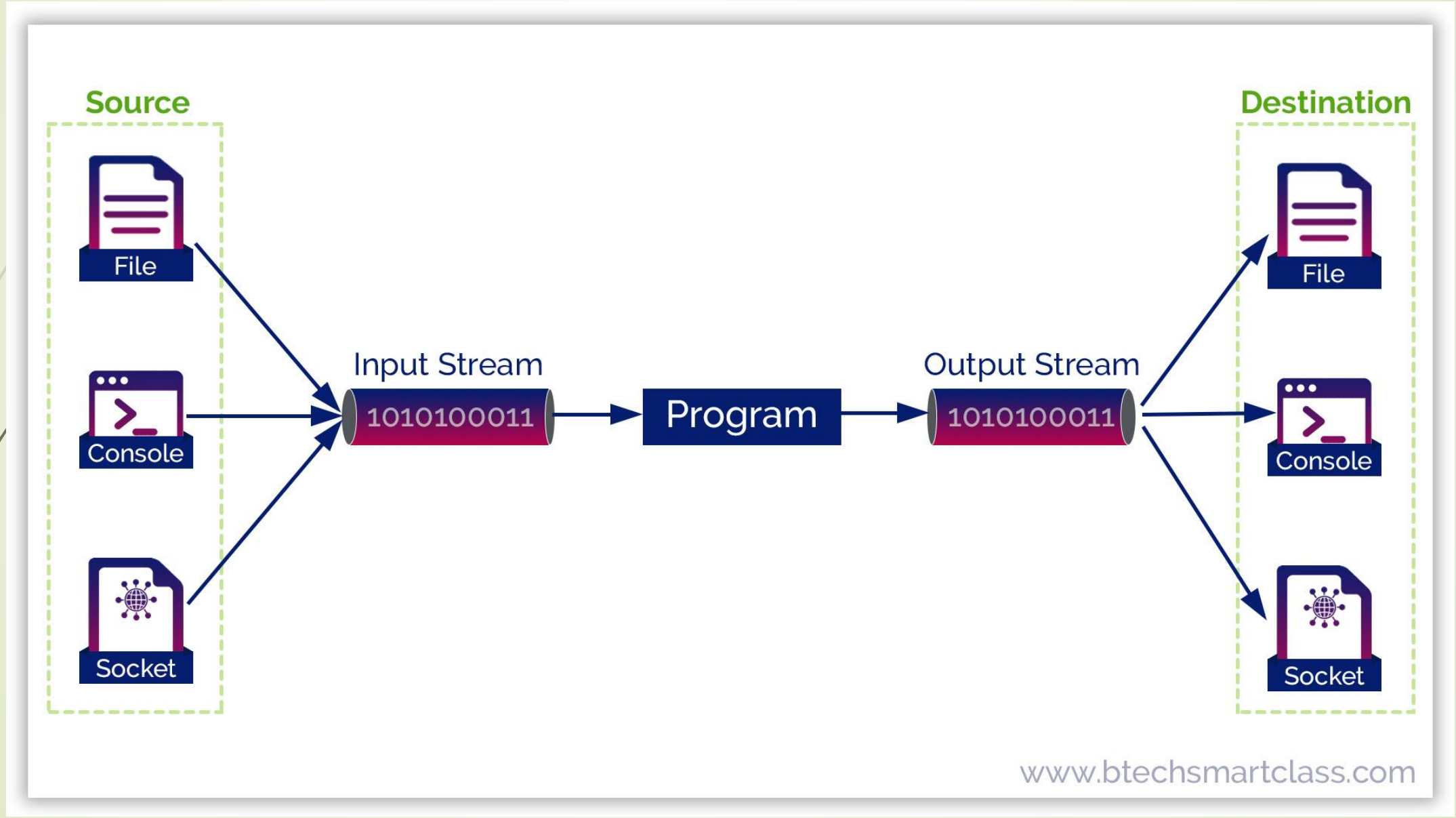
Java I/O

Java I/O

- The basic units of Input/Output in Java are IO Streams
- A stream can be thought of as a flow of data from a source to a destination i.e from InputStream to OutputStream

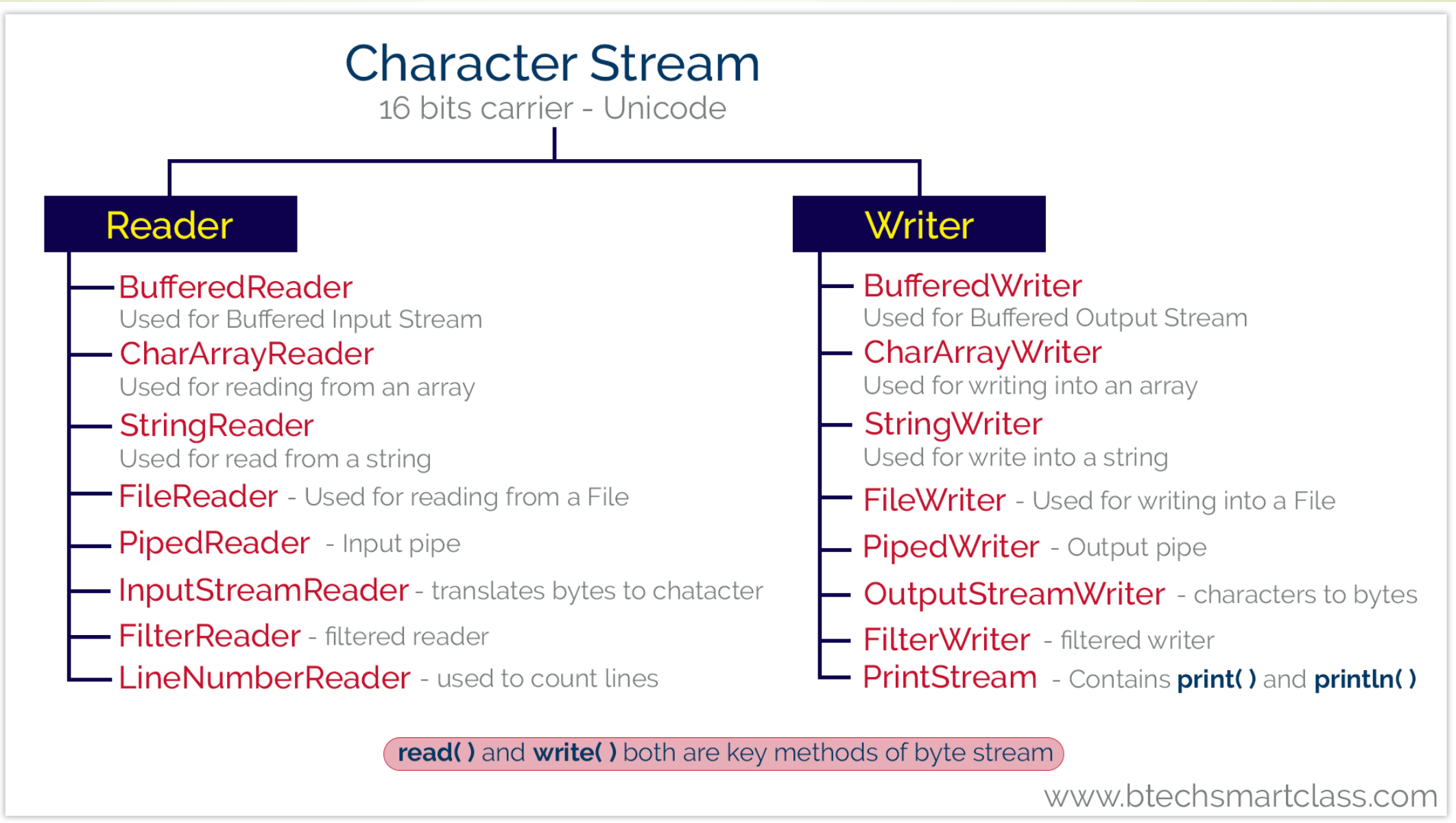
Java I/O


Source
File
Console
Socket
Input Stream
1010100011
Program
Output Stream
1010100011
Destination
File
Console
Socket
www.btechsmartclass.com

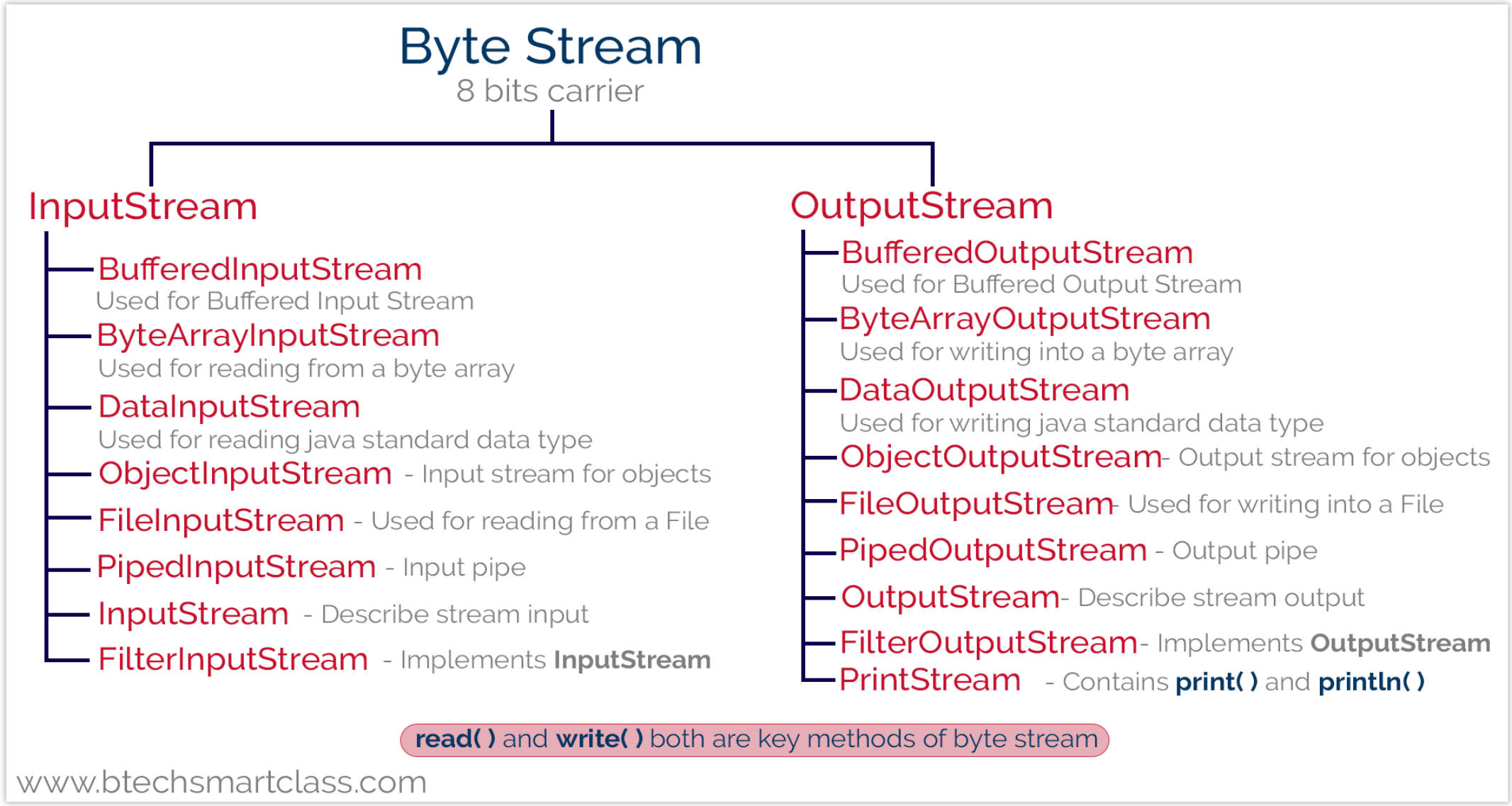
Java I/O

- Java supports two types of streams
  - Character streams (use Unicode)
  - Byte streams (use byte)
- Input and output of character data are handled by readers and writers
- Input and output of byte data are handled by input streams and output streams

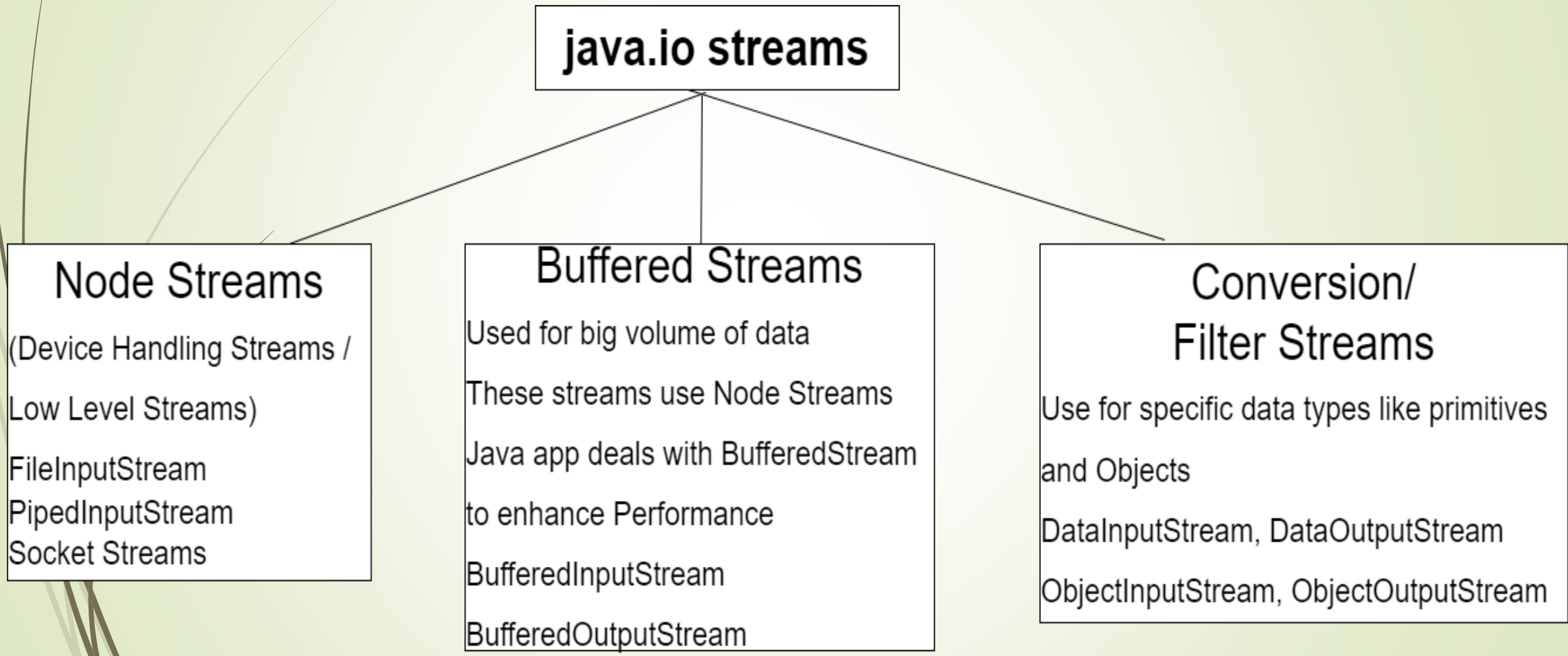
Character Stream



Byte Stream


Byte Stream
8 bits carrier
InputStream
BufferedInputStream
Used for Buffered Input Stream
ByteArrayInputStream
Used for reading from a byte array
DataInputStream
Used for reading java standard data type
ObjectInputStream - Input stream for objects
FileInputStream - Used for reading from a File
PipedInputStream - Input pipe
InputStream - Describe stream input
FilterInputStream - Implements InputStream
OutputStream
BufferedOutputStream
Used for Buffered Output Stream
ByteArrayOutputStream
Used for writing into a byte array
DataOutputStream
Used for writing java standard data type
ObjectOutputStream- Output stream for objects
FileOutputStream- Used for writing into a File
PipedOutputStream - Output pipe
OutputStream- Describe stream output
FilterOutputStream- Implements OutputStream
PrintStream - Contains print( ) and println( )
read( ) and write( ) both are key methods of byte stream
www.btechsmartclass.com

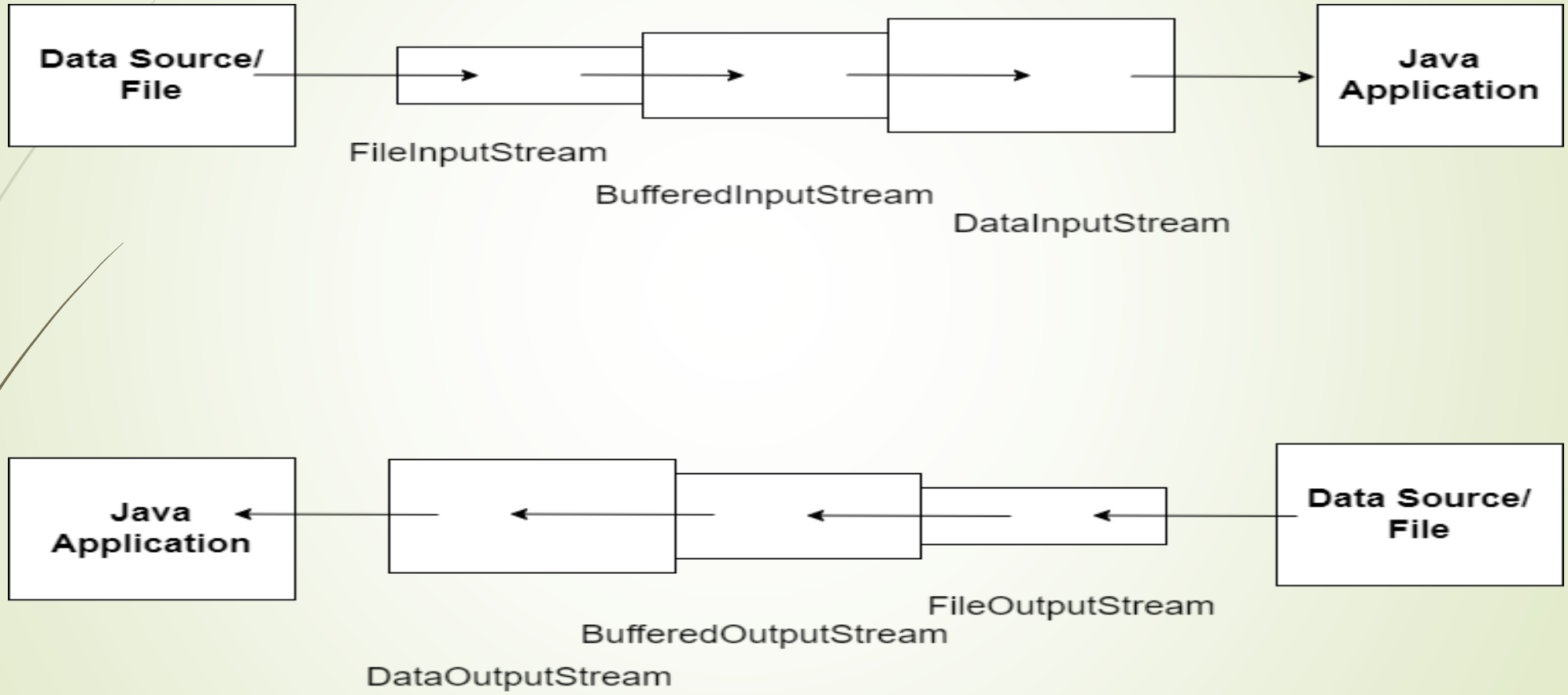
Stream Types



BufferedWriter vs PrintWriter

BufferedWriter	PrintWriter
BufferedWriter can specify size of buffer	PrintWriter can not specify size of buffer
BufferedWriter has write methods	PrintWriter has Write and Print methods
Can not wrap character oriented streams only	Can wrap character oriented and byte oriented streams only
Does not support flushing	Supports Auto flushing

Stream Chaining



Java Serialization

- Read or write a Java object to a stream
- Saving an object to persistent storage is called persistence
- Java provides a Serializable interface
- Serializable Interface has no methods and is a marker interface
- When an object is serialized, only data of the object are preserved

Java Serialization

- If a data variable is an object, data members of the object are also serialized if that object's class is serializable
- If a serializable object contains reference to non-serializable element, the entire serialization fails
- "**transient**" fields does not get serialized
- **ObjectInputStream** and **ObjectOutputStream** are used for serialization

Java Serialization

- **What is serialVersionUID?**
  - Each time an object is serialized, the object is **Stamped** with serialVersionUlD, and it's calculated based on information about the class and fields.
  - When object is being deserialized, if the class has changed since the object was serialized, the class could have a different serialVersionUID, and deserialization will fail.(java.lang.InvalidClassException).
- Serialization format overview
  1. Magic Number.
  2. Serialization format version number.
  3. Class description -> class name, serialVersionUID, description of data members(class signature)
  4. State of the object.(non static & non transient data members)

File class

- The File class has methods to perform below operation on a file
  - Create File objects
  - Check a File object can be read and written to.
  - Check if File object exists.
  - Get file (and path) names.
  - Get a list of file names inside a directory.
  - Create a Directory.
  - Rename a file.
  - Last modification date.
  - Delete the File object.