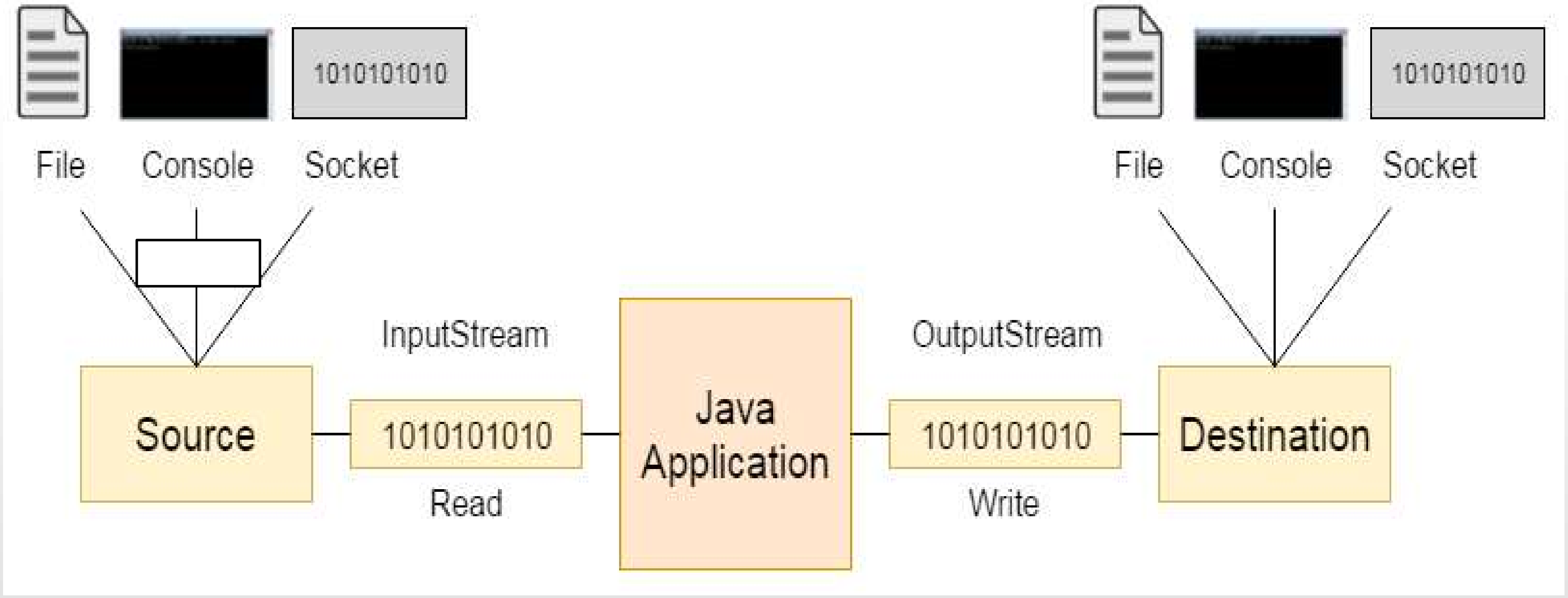
IO Streams

PresentationPoint



There are many classes and interfaces in the **java.io** package.

File

- A **File** object is used to obtain or manipulate the information(properties) associated with a disk file.

- Ex: permissions, time, date, and directory path etc.

- **File** class does not specify how information is retrieved from or stored in files;

File

- A directory in Java is treated simply as a **File** with one additional property:
    - contains a list of other files and directories
- List of filenames that can be examined by the **list( )** method.

Constructors used to create **File** objects:

File(String *directoryPath***)**

File(String *directoryPath*, String *filename***)**

File(File *dirObj*, String *filename***)**

Constructors used to create **File** objects:

Method-1 File(String *directoryPath*)

File f1 = new File(“D:/javaPrograms/Streams/prg1.txt");

Method-2 File(String *directoryPath*, String *filename*)

File f2 = new File(" D:/javaPrograms/Streams/", “prg1.txt”);

Method-3 *File(File dirObj, String filename)*

File f3 = new File("D:/javaPrograms/Streams/"); // dirObj

File f4 = new File(f3,”prg1.txt”);

Windows:

recognize forward slash(/).

If we use backaword slash, we need to use escape sequence(\\)

Unix :

recognize backword slash(\).

Methods :

String getName() – returns name of the file

String getParent() - returns the name of the parent directory

String getPath() –returns Relative path

String getAbsolutePath()- returns Absolute path

boolean isAbsolute()-Absolute path **(true)** or Relative path **(false)**

long length() - File size

Methods :

boolean exists() -**true** if the file exists, **false** if it does not

boolean canWrite() –**true** if writable, **false** if it is not

boolean canRead() - **true** if Readable, **false** if it is not

boolean canExecute () -**true** if Executable, **false** otherwise.

boolean isDirectory() - **true** if it is a directory, **false** otherwise.

boolean isFile() - **true** if file, **false** if not.

Methods :

boolean setReadOnly () - Sets the invoking file to read-only

boolean setWritable (boolean writable, boolean ownerOnly) - Sets/resets the invoking file to writable

long getFreeSpace () - Returns the number of bytes of storage available on the partition associated with the invoking object

long getTotalSpace () - Returns the storage capacity of the partition associated with the invoking object.

long getUsableSpace () - Returns the number of usable free bytes of storage available on the partition associated with the invoking object.

Methods :

long lastModified() -Last modified time.

boolean renameTo (File *newName*)

- *newName* becomes the new name of the invoking **File** object.

- It will return **true** upon success and **false** if the file cannot be renamed.

(if you either attempt to rename a file so that it uses an existing filename).

boolean delete()

To delete file and directory.

To delete directory, it should be empty

Creating Directories

boolean mkdir( ) -Creates a directory; returns true on success, false on failure.

Example:

```
File f = new File ("newdir");
boolean b = f.mkdir();
          OR
new File ("newdir").mkdir();
```

boolean mkdirs( )- Creates a directory and all the parents(even if do not exist) of the directory.

Example:

```
File f = new File ("tmp/one/two/three");
boolean b = f.mkdirs();
          OR
new File ("tmp/one/two/three").mkdirs();
```

mkdirs : To create a directory for which no path exists. It creates a directory and all the parents of the directory.

Creating Directories

```
File f = new File("D:/MIT");

if (f.mkdir())
{
    System.out.println("Directory is created");
}
else
 {
        System.out.println("Directory cannot be created");
 }
```

Retrieving Directory contents

String [ ]   list()

Call list( ) on that object to extract the list of other files and directories inside.

The listFiles( ) Alternative

- Is a Variation of **list( )** method.
  - Which returns File(object) **instead of string**

File[ ] listFiles( )

File[ ] listFiles(FilenameFilter *FFObj*)

Using FilenameFilter

- To include only those files that match a certain filename pattern, or ***filter.***

  String[ ] list(FilenameFilter *FFObj*)

- **FFObj** is an object of **a class that** implements the **FilenameFilter** interface.

- **FilenameFilter** defines a single method, **accept( )**, which is called once for each file in a list.

boolean **accept**(File *directory*, String filename)

- The **accept( )** method returns **true** for files in the directory specified by *directory* that should be included in the list (that is, those that match the *filename* argument), and returns **false** for those files that should be excluded.

- The **OnlyExt** class implements **FilenameFilter**.

FilenameFilter - Demo

```
import java.io.*;
class OnlyExt implements FilenameFilter
{
            String ext;
            public OnlyExt(String ext)
             {
                        this.ext = "." + ext;
             }
            public boolean accept(File dir, String name)
            {
                        return name.endsWith(ext);
            }
}
```

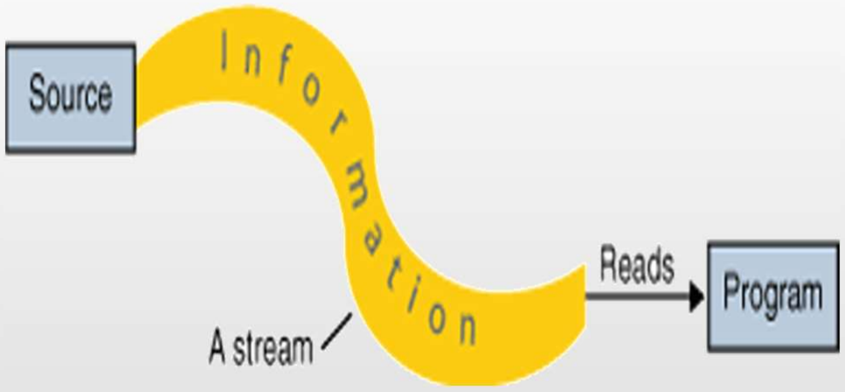
The **OnlyExt** class implements **FilenameFilter**.

FilenameFilter - Demo…

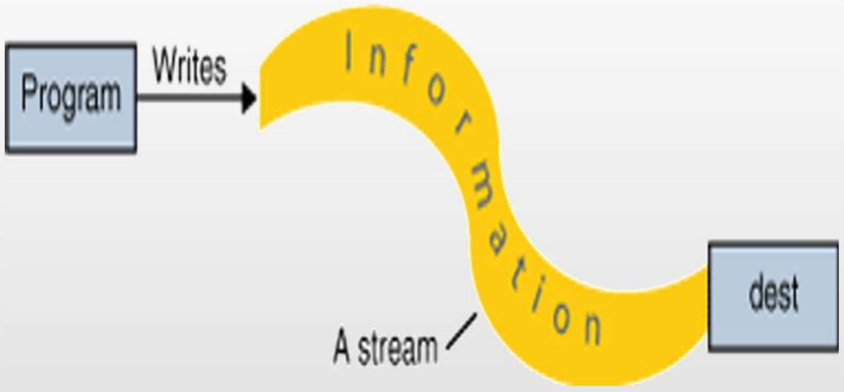
```
class DirListOnly1
 {           public static void main(String args[])
            {
                        String dirname = "D:/javaPrograms/Streams";
                        File f1 = new File(dirname);
                        FilenameFilter only = new OnlyExt(“java");
                        String s[] = f1.list(only);   // accept() returns a string of files in
                                                      // D:/javaPrograms/Streams with .java
                        for (int i=0; i < s.length; i++)
                        {     System.out.println(s[i]);
                        }
            } }
```

Streams

- **Stream**
  - A logical entity that either produces or consumes information
    - Linked to a physical device by the Java I/O system
    - All streams behave in the same manner, even if the actual physical devices they are linked to differ



Reading information into a program

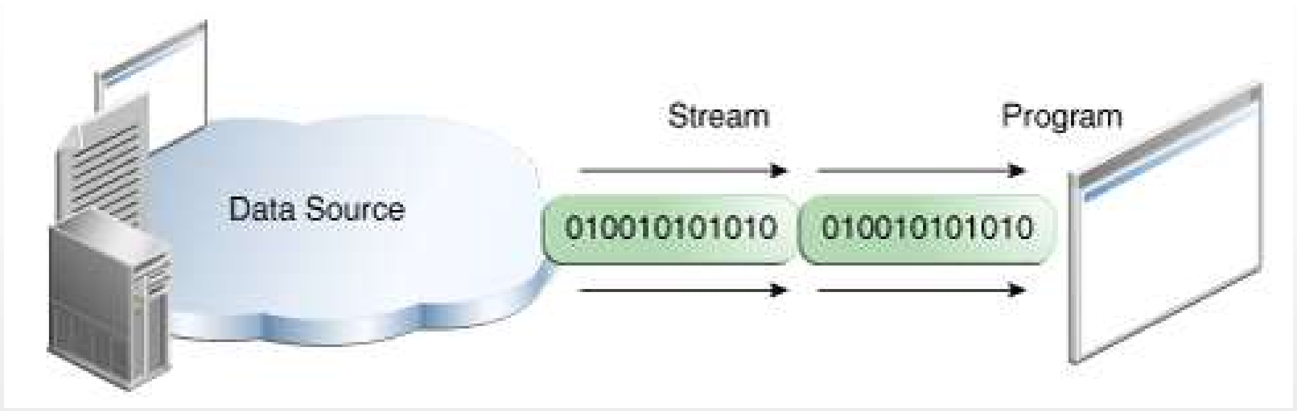


Writing information out of a program

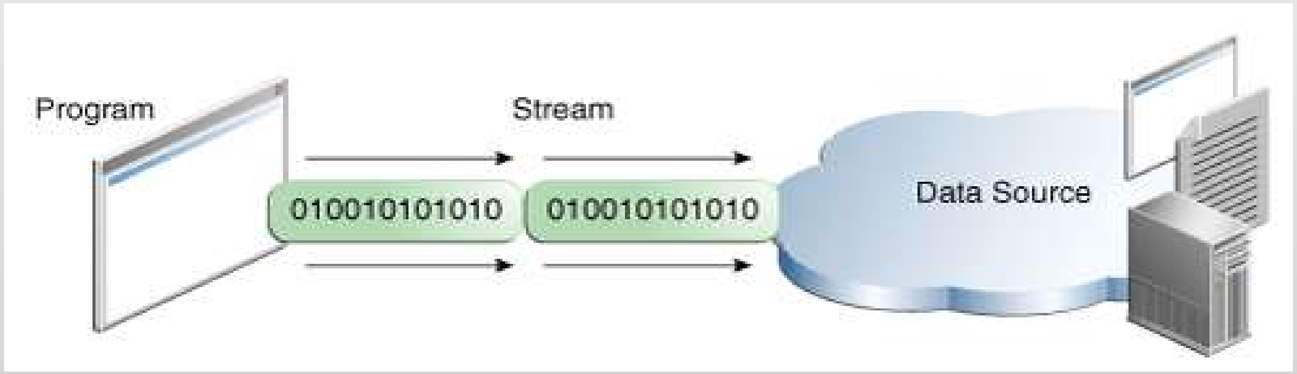
  - The java.io package has a number of classes and interfaces to handle I/O

PresentationPoint

All streams present the same simple model



A program uses an *input stream* to read data from a source, one item at a time

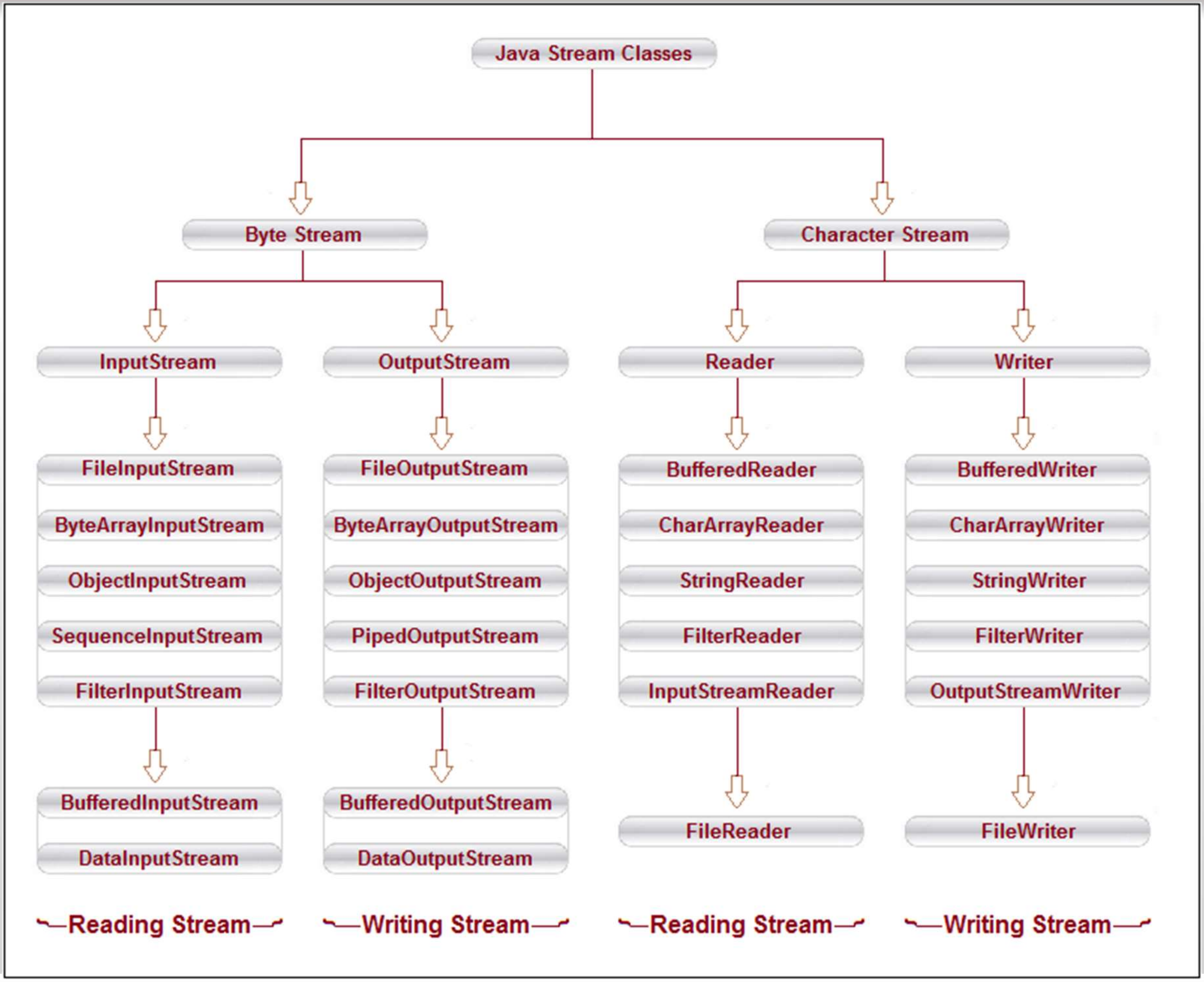


A program uses an *output stream* to write data to a destination, one item at time

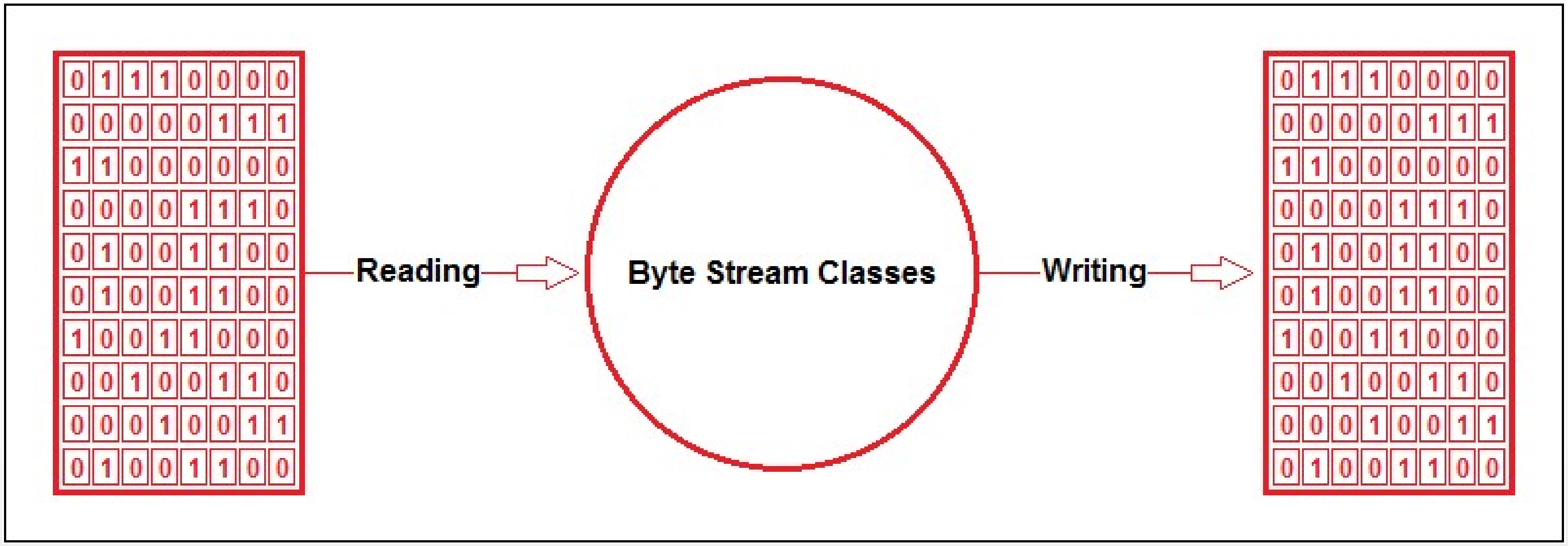
- Some input-output stream will be initialized automatically by the JVM and these streams are available in **System** class as **in**, **out**, and **err** variable.
- **in** reference refers to the default input device, i.e. keyboard.
- **out** and **err** refers to the default output device, i.e. console.

Streams

- **The Stream Classes**
  - Java I/O : built upon four abstract classes (used to build concrete subclasses)
    - InputStream and OutputStream
      - Designed for byte streams
      - Used for working with bytes and binary objects
      - Abstract methods
        - InputStream: read() to read a byte of data
        - OutputStream: write() to write a byte of data
    - Reader and Writer
      - Designed for character streams
      - Used for working with characters and strings
      - Abstract methods
        - Reader: read() to read a character
        - Writer: write() to write a character


Java Stream Classes
Byte Stream
Character Stream
InputStream
OutputStream
Reader
Writer
FileInputStream
ByteArrayInputStream
ObjectInputStream
SequenceInputStream
FilterInputStream
FileOutputStream
ByteArrayOutputStream
ObjectOutputStream
PipedOutputStream
FilterOutputStream
BufferedReader
CharArrayReader
StringReader
FilterReader
InputStreamReader
BufferedWriter
CharArrayWriter
StringWriter
FilterWriter
OutputStreamWriter
BufferedInputStream
DataInputStream
BufferedOutputStream
DataOutputStream
FileReader
FileWriter
Reading Stream
Writing Stream
Reading Stream
Writing Stream

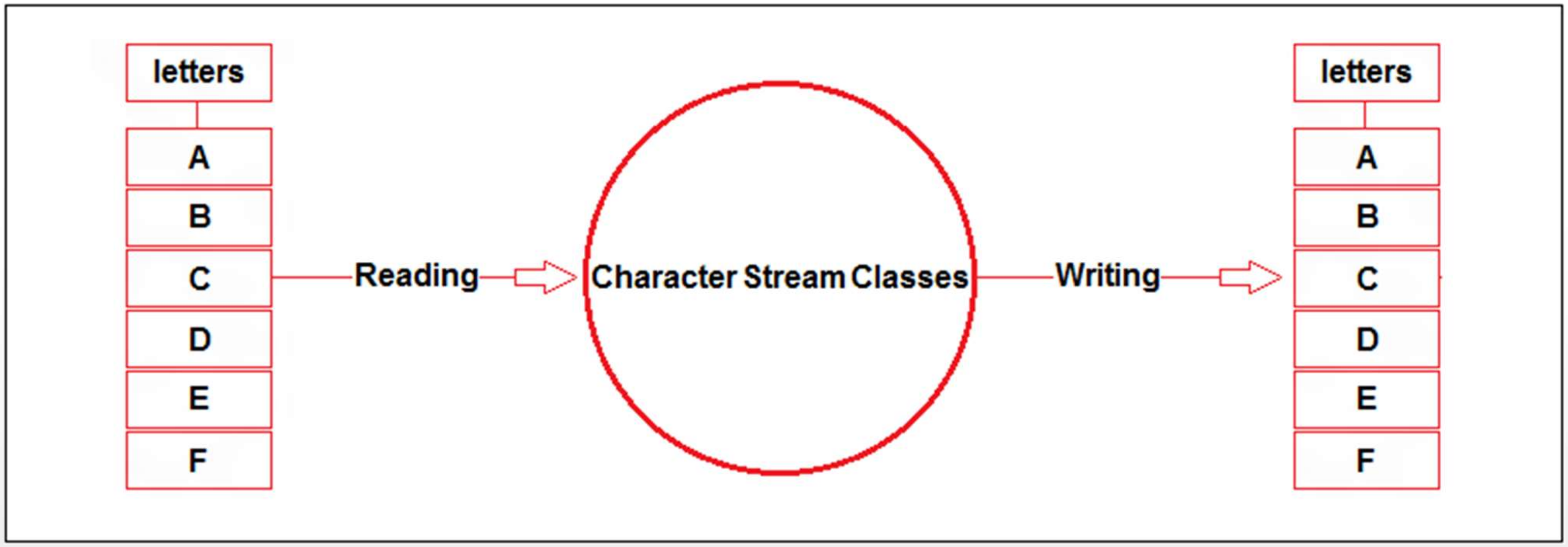
Byte stream



Byte stream classes have been designed to provide functional features for creating and manipulating streams and files for reading and writing bytes(8 bits). Java provides two kinds of byte stream classes: **input stream classes** and **output stream classes**.

use the byte stream classes when working with bytes or other binary objects.

Character streams



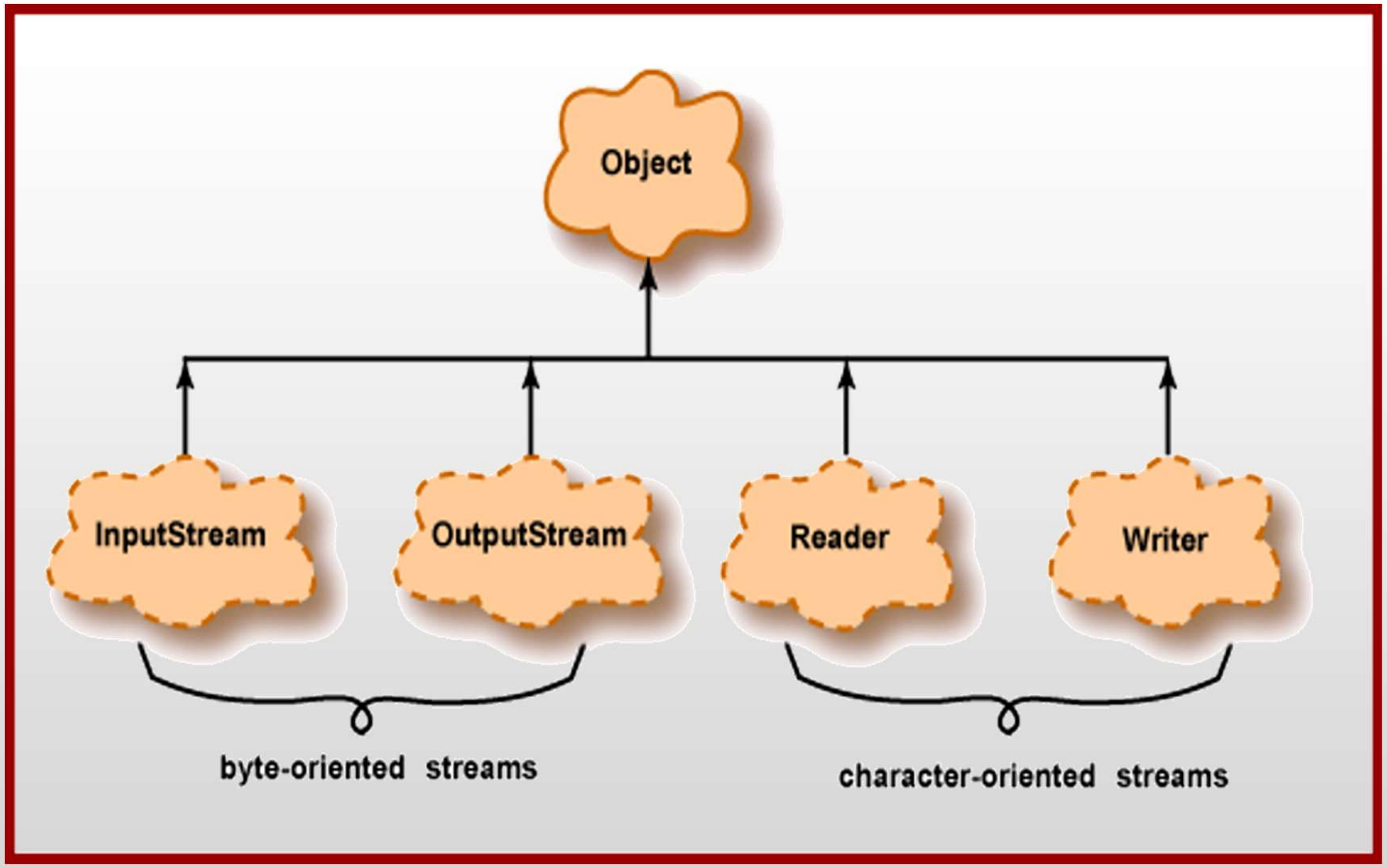
Character streams can be used to read and write **16-bit** Unicode characters. Like byte streams, there are two kinds of character stream classes, namely, **reader stream classes** and **writer stream classes**.

use the character stream classes when working with characters or strings

PresentationPoint

Streams

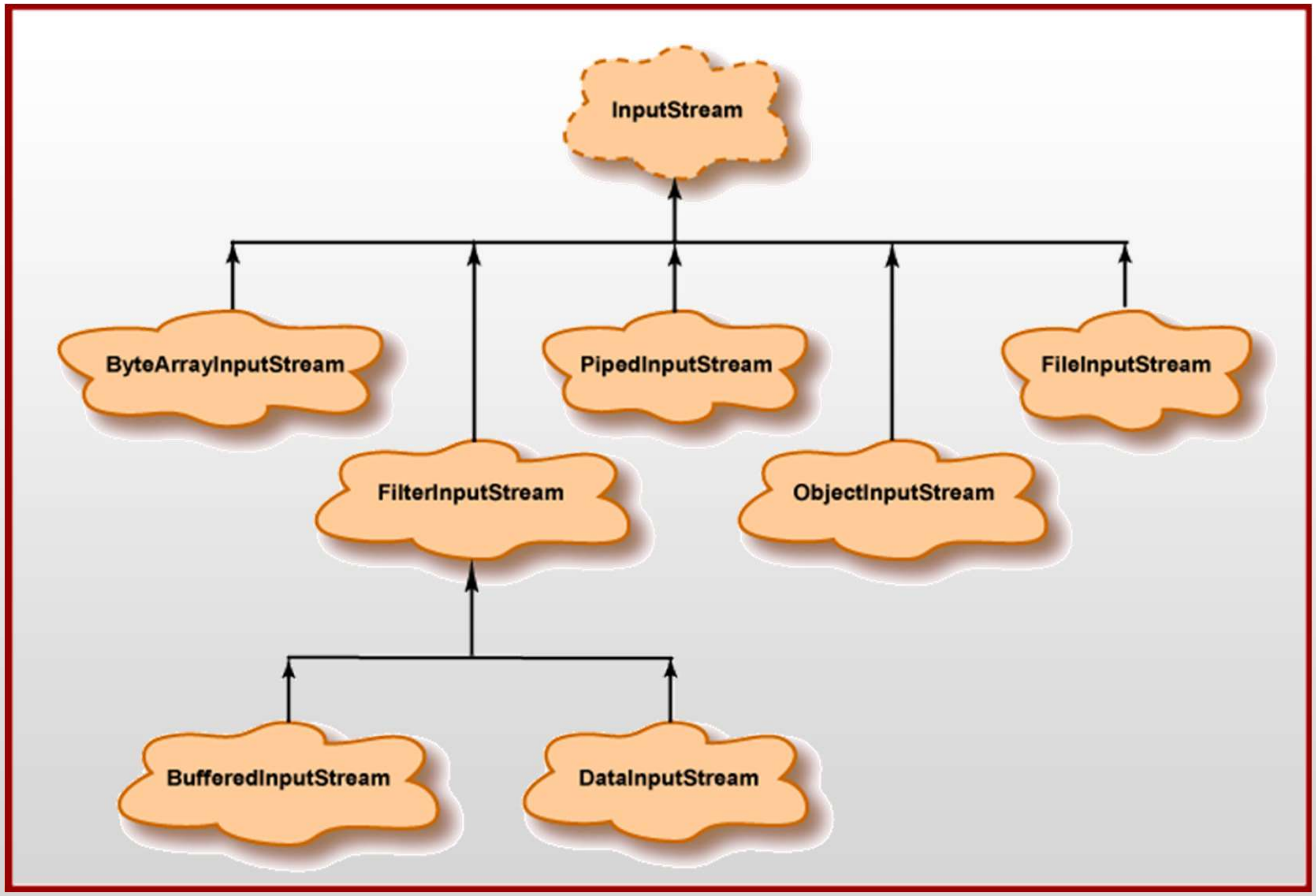
- **The Stream Classes** (Continued …)



PresentationPoint

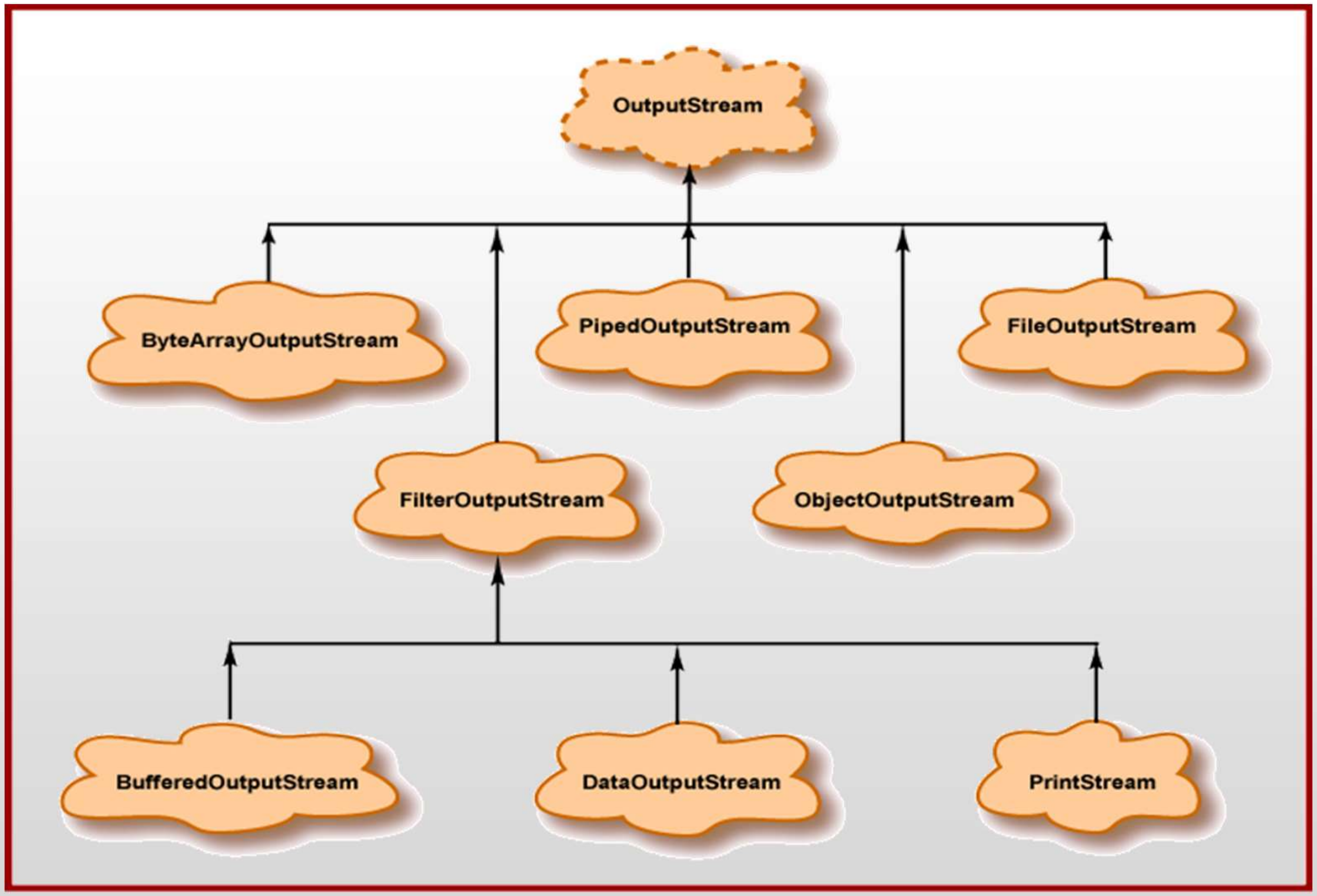
Streams

- **InputStream**



Streams

- **OutputStream**



The Stream Classes

- Java's stream-based I/O is built upon four abstract classes:

  InputStream, **OutputStream**,

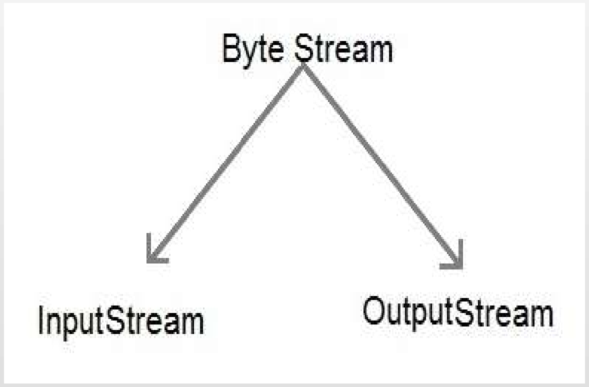
  Reader, **Writer**.

- Used to create several **concrete stream subclasses**.
- Programs **perform their I/O operations through subclasses**.
- The top-level classes define the **basic functionality common to all stream classes**.

The Byte Streams

- Byte stream classes provide a rich environment for handling byte-oriented I/O.

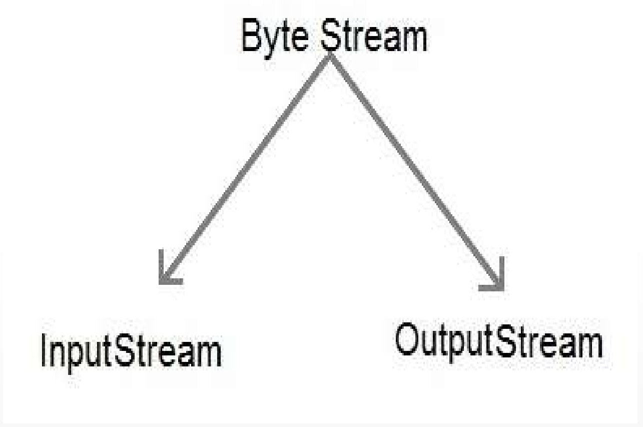
- A byte stream can be used with any type of object, including binary data.



Most of the methods in this class will throw an **IOException** on error conditions.

Methods in InputStream.

int read( )

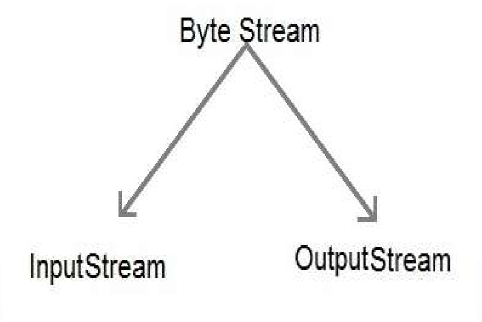


- Returns an integer representation of the next available byte of input.

Methods in InputStream.

int read( )

int read(byte buffer[ ])

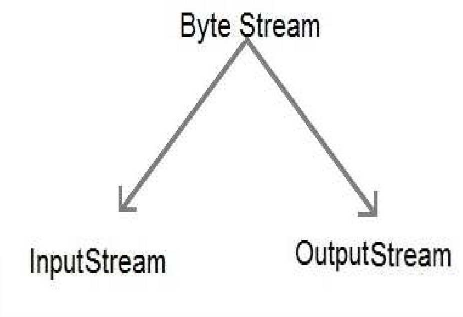


- Attempts to read up to buffer.length bytes into buffer and
- Returns the actual number of bytes that were successfully read.

Methods in InputStream.

int read( )

int read(byte buffer[ ])



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.

Methods in InputStream.

int available( )

- Returns the number of bytes of input currently available for reading.

Methods in InputStream.

int available( )

long skip(long numBytes)

- Ignores (that is, skips) numBytes bytes of input.
- returning the number of bytes actually ignored.

Methods in InputStream.

int available( )

long skip(long numBytes)

void close( )

- Closes the input source. Further read attempts will generate an **IOException**.

Streams

- **The Stream Classes**
  - Java I/O : built upon four abstract classes (used to build concrete subclasses)
    - InputStream and OutputStream
      - Designed for byte streams
      - Used for working with bytes and binary objects
      - Abstract methods

» InputStream:	read()	to read a byte of data
» OutputStream:	write()	to write a byte of data

    - Reader and Writer
      - Designed for character streams
      - Used for working with characters and strings
      - Abstract methods

» Reader:	read()	to read a character
» Writer:	write()	to write a character

Streams

- **InputStream** (Continued …)

Method		**Description**
int	read ( )	✓ Returns an integer representation of the next available byte of input. ✓ Returns −1 when end-of-file is encountered.
int	read (byte buffer[ ])	✓ Attempts to read up to *buffer.length* bytes into *buffer.* ✓ Returns the actual number of bytes successfully read. ✓ Returns −1 when end-of-file is encountered.
int	read (byte buffer[ ], int offset, int numBytes)	✓ Attempts to read up to *numBytes* bytes into *buffer* starting at *buffer*[*offset*]. ✓ Returns the number of bytes successfully read. ✓ Returns −1 when end-of-file is encountered.
int	available ( )	✓ Returns the number of bytes of input currently available for reading.

Methods defined by InputStream

Streams

- **InputStream** (Continued …)

Method		Description
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 remains valid until *numBytes* bytes are read.
void	reset ( )	✓ Resets the input pointer to the previously set mark.
boolean	markSupported ( )	✓ Returns true if mark( )/reset( ) are supported by the invoking stream.
long	skip (long numBytes)	✓ Ignores (that is, skips) *numBytes* bytes of input, returning the number of bytes actually ignored.

Methods defined by InputStream

Streams

- **OutputStream** (Continued …)

Method		**Description**
void	write (int b)	✓ Writes a single byte to an output stream.
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*].
void	close ( )	✓ Closes the output stream. ✓ Further write attempts will generate an IOException.
void	flush ( )	✓ Finalizes the output state so that all buffers are cleared (flushes the output buffers).

Methods defined by OutputStream

Byte Streams

- **FileInputStream**
  - Creates an InputStream to read bytes from a file
  - Constructors (can throw FileNotFoundException)
    - FileInputStream(String filePath)   // full path name of the file

      Ex: FileInputStream f0 = new FileInputStream (“test.txt”);
    - FileInputStream(File fileObj)   // file object describes the file

      Ex: File f = new File (“test.txt”);

      FileInputStream f1 = new FileInputStream(f);
  - FileInputStream overrides six methods of InputStream (except mark() and reset())
    - Attempt to use reset() generates java.io.IOException (mark() and reset() not supported)

Byte Streams

- **FileOutputStream**
  - Creates an OutputStream to write bytes to a file
  - Constructors (can throw IOException in case of read-only files)
    - FileOutputStream(String filePath)   // full path name of the file

      Ex: FileOutputStream f0 = new FileOutputStream ("test.txt");
    - FileOutputStream(File fileObj)   // file object describes the file

      Ex: File f = new File ("test.txt");

      FileOutputStream f1 = new FileOutputStream(f);
    - FileOutputStream(String filePath, boolean *append*)

      Ex: FileOutputStream f2 = new FileOutputStream ("test.txt", true);
    - FileOutputStream(File fileObj, boolean *append*)

      Ex: File f = new File ("test.txt");

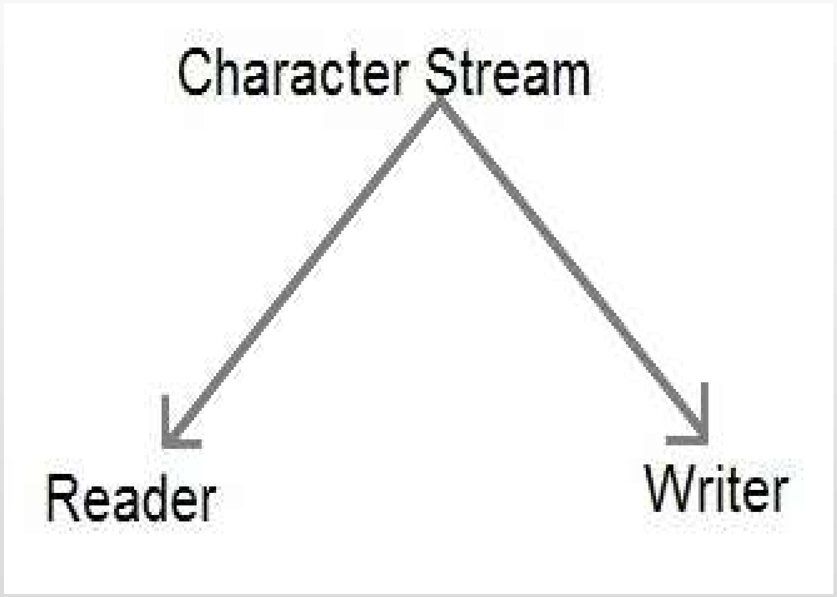
      FileOutputStream f3 = new FileOutputStream(f, false);

Exercise:

- Consider a directory **MIT/MCA** which contains java files, out of which one file is the replica of another file. Write a java program to find those 2 files.

The Character Streams

- Top of the character stream hierarchies are the **Reader** and **Writer** abstract classes.



The Character Streams

- The byte stream classes provide functionality to handle any type of I/O operation, they cannot work directly with Unicode characters.
- **Character streams** : direct I/O **support for characters**. Used for working with characters or strings.

NOTE :

- The character I/O classes were added by the 1.1 release of Java.
- Because of this, we may still find legacy code that uses byte streams where character streams would be more appropriate

Character Streams

- **Reader** (Continued …)
  - Reader – an abstract class – defines streaming character input

Method		Description
int	read ( )	✓ Returns an integer representation of the next available character from the invoking input stream. ✓ Returns –1 when end-of-file is encountered.
int	read (char buffer[ ])	✓ Attempts to read up to *buffer.length* characters into *buffer* and returns the actual number of bytes successfully read. ✓ Returns –1 when end-of-file is encountered.
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. ✓ Returns –1 when end-of-file is encountered.

Methods defined by Reader

Character Streams

- **Reader** (Continued …)

Method		Description
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 remains valid until *numChars* characters are read.
void	reset ( )	✓ Resets the input pointer to the previously set mark.
boolean	markSupported ( )	✓ Returns true if mark( )/reset( ) are supported by the invoking stream.
long	skip (long numChars)	✓ Ignores (that is, skips) *numChars* characters of input, returning the number of characters actually ignored.

Methods defined by Reader

Character Streams

- **Writer** (Continued …)
  - Writer – an abstract class – defines streaming character output

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 output stream. ✓ Returns a reference to the invoking stream.

Methods defined by Writer

Character Streams

- **Writer** (Continued …)

Method		Description
void	write (char ch)	✓ Writes a single character to the invoking output stream.
void	write (char buffer[ ])	✓ Writes a complete array of characters to the invoking output stream.
abstract void	write (char buffer[ ], int offset, int numChars)	✓ Writes a sub-range of *numChars* characters from the array *buffer*, beginning at *buffer*[*offset*] to the invoking output stream.

Methods defined by Writer

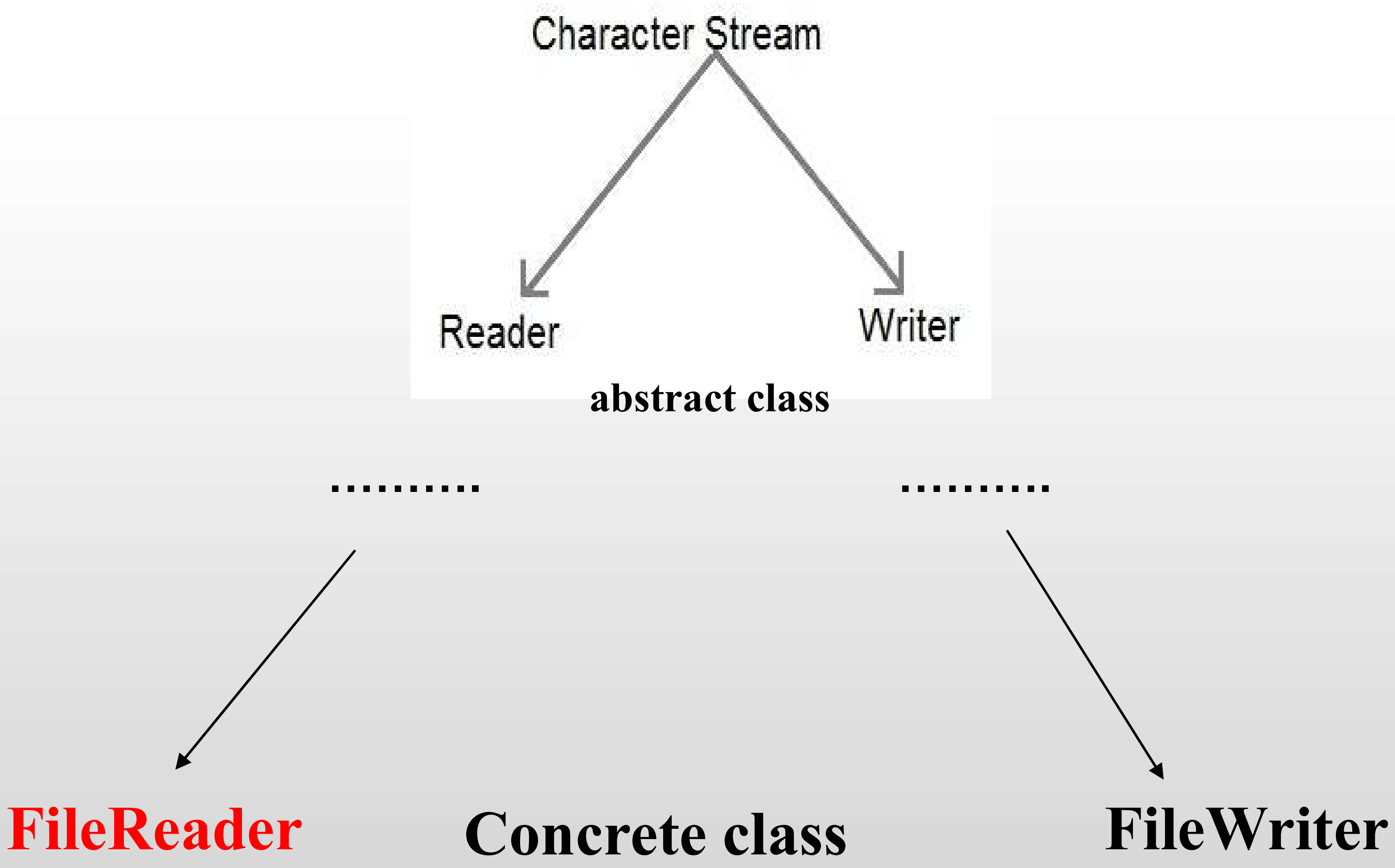
Character Streams

- **Writer** (Continued …)

Method		Description
void	write (String str)	✓ Writes *str* to the invoking output stream.
void	write (String str, int offset, int numChars)	✓ Writes a sub-range of *numChars* characters from the string *str*, beginning at the specified *offset*.
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.

Methods defined by Writer

FileReader


Character Stream
Reader
Writer
abstract class
……….
……….
FileReader
Concrete class
FileWriter

FileReader

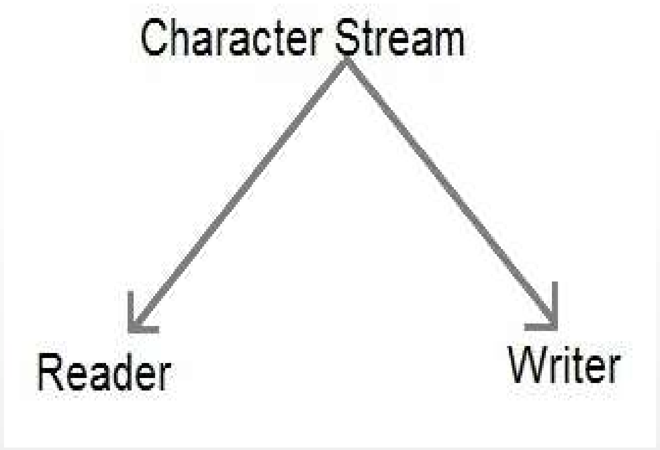
- **FileReader** class creates a **Reader** that can be uses to read the contents of a file.
- Its two most commonly used **constructors** are:

FileReader(String *filePath*)

FileReader(File *fileObj*)

- Either can throw a **FileNotFoundException**.
  - Here**,** ***filePath*** is the full path name of a file, and
  - ***fileObj*** is a **File** object that describes the file.

FileWriter



....

FileReader   FileWriter

FileWriter

- **FileWriter** creates a **Writer** that we can use to write to a file.
- Its most commonly used constructors are:

FileWriter(String *filePath*)

FileWriter(String *filePath*, boolean *append*)

FileWriter(File *fileObj*)

FileWriter(File *fileObj*, boolean *append*)

- They can throw an **IOException**.
- If *append* is **true**, then output is appended to the end of the file.
- Creation of a **FileWriter** is not dependent on the file already existing. **FileWriter** will create the file before opening it for output when you create the object.
- In the case where you attempt to open a read-only file, an **IOException** will be thrown.

Writing

- **void write(int *ch*)**
- **void write(char *buffer*[ ])**
- **void write(char *buffer*[ ], int *offset*, int *numChars*)**
- **void write(String *str*)**

Exercise:

- Consider a directory **MIT/MCA** which contains java files, out of which one file is the replica of another file. Write a java program to find those 2 files.

Byte Streams

- **DataOutputStream**
  - Write primitive data to a stream
  - Constructor
    - DataOutputStream (OutputStream *out*)
      - Creates a data output stream that uses the specified output stream
    - Defines methods to convert values of a primitive type into a byte sequence and then write it to the output stream
      - final void writeDouble (double *value*) throws IOException
      - final void writeBoolean (boolean *value*) throws IOException
      - final void writeInt (int *value*) throws IOException

DataOutputStream- Methods

int size()	It is used to return the number of bytes written to the data output stream.
void write(int b)	It is used to write the specified byte to the underlying output stream.
void write(byte[] b, int off, int len)	It is used to write len bytes of data to the output stream.
void writeBoolean(boolean v)	It is used to write Boolean to the output stream as a 1-byte value.
void writeChar(int v)	It is used to write char to the output stream as a 2-byte value.
void writeChars(String s)	It is used to write string to the output stream as a sequence of characters.
void writeByte(int v)	It is used to write a byte to the output stream as a 1-byte value.
void writeBytes(String s)	It is used to write string to the output stream as a sequence of bytes.
void writeInt(int v)	It is used to write an int to the output stream
void writeShort(int v)	It is used to write a short to the output stream.
void writeShort(int v)	It is used to write a short to the output stream.
void writeLong(long v)	It is used to write a long to the output stream.
void writeUTF(String str)	It is used to write a string to the output stream using UTF-8 encoding in portable manner.
void flush()	It is used to flushes the data output stream.

PresentationPoint

Byte Streams

- **DataInputStream**
  - Read primitive data from a stream
  - Constructor
    - DataInputStream (InputStream *is*)
      - Creates a data input stream that uses the specified input stream
    - Defines methods to read a sequence of bytes from the stream and then convert them into values of a primitive type
      - double readDouble () throws IOException
      - boolean readBoolean () throws IOException
      - int readInt () throws IOException

DataInputStream- Methods

int read(byte[] b)	It is used to read the number of bytes from the input stream.
int read(byte[] b, int off, int len)	It is used to read **len** bytes of data from the input stream.
int readInt()	It is used to read input bytes and return an int value.
byte readByte()	It is used to read and return the one input byte.
char readChar()	It is used to read two input bytes and returns a char value.
double readDouble()	It is used to read eight input bytes and returns a double value.
boolean readBoolean()	It is used to read one input byte and return true if byte is non zero, false if byte is zero.
int skipBytes(int x)	It is used to skip over x bytes of data from the input stream.
String readUTF()	It is used to read a string that has been encoded using the UTF-8 format.

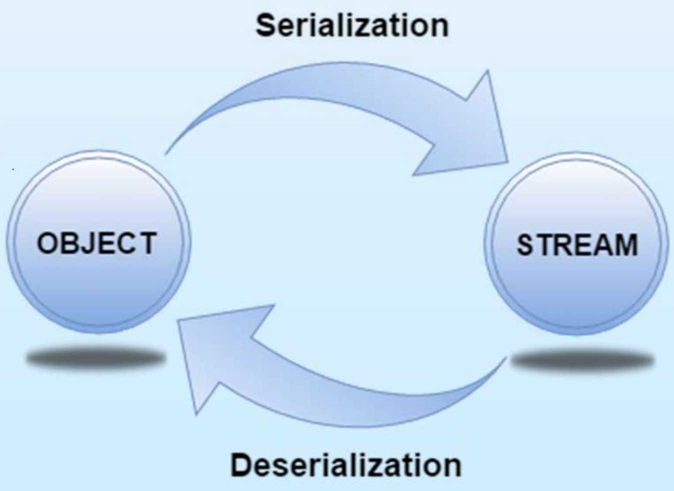
PresentationPoint

Exercises

- Write a java program to find the occurrence of word in a given file

Serialization

- **Object Serialization and Deserialization**
  - Mechanism where characteristics of an object is represented as a sequence of bytes
    - Object’s data
    - Information about the objects type and types of data stored in the object
  - Write serialized object to a persistent storage (disk file)
  - Object can be restored using deserialization



  - JVM independent
    - Object can be serialized in one platform and deserialized in another platform

Serialization

- **Serializable Interface**
  - A built-in interface in java.io package
    - Defines no members
    - Indicates that a class may be serialized
      - If a class is serializable, all its subclasses are also serializable
  - An object of a class that implements this interface can be serialized
  - Variables declared as *transient* or *static* cannot be saved by serialization

Serialization

- **ObjectOutputStream Class**
  - Extends OutputStream class and implements ObjectOutput interface
    - Responsible for writing objects to a stream
  - Constructor

    ObjectOutputStream (OutputStream *out*) throws IOException

    - Argument *out* is the output stream to which serialized object will be written
  - All methods throw IOException

Serialization

- **ObjectOutputStream Class** (Continued …)

Method		**Description**
void	write (byte buffer[ ])	✓ Writes an array of bytes to the invoking stream.
void	write (byte buffer[ ], int offset, int numBytes)	✓ Writes a subrange of *numBytes* bytes from the array *buffer*, beginning at *buffer*[*offset*].
void	write (int b)	✓ Writes a single byte to the invoking stream. ✓ The byte written is the low-order byte of *b*.
void	writeBoolean (boolean b)	✓ Writes a *boolean* to the invoking stream.
void	writeByte (int b)	✓ Writes a *byte* to the invoking stream. ✓ The byte written is the low-order byte of *b*.

Methods defined by ObjectOutputStream Class

Serialization

- **ObjectOutputStream Class** (Continued …)

Method		**Description**
void	writeBytes (String str)	✓ Writes the bytes representing *str* to the invoking stream.
void	writeChar (int c)	✓ Writes a *char* to the invoking stream.
void	writeChars (String str)	✓ Writes the characters in *str* to the invoking stream.
void	writeDouble (double d)	✓ Writes a *double* to the invoking stream.
void	writeFloat (float f)	✓ Writes a *float* to the invoking stream.
void	writeInt (int i)	✓ Writes an *int* to the invoking stream.

Methods defined by ObjectOutputStream Class

Serialization

- **ObjectOutputStream Class** (Continued …)

Method		**Description**
void	writeLong (long l)	✓ Writes a *long* to the invoking stream.
void	writeShort (int i)	✓ Writes a *short* to the invoking stream.
final void	writeObject (Object obj)	✓ Writes *obj* to the invoking stream.
void	close ( )	✓ Closes the invoking stream. ✓ Further write attempts will generate an IOException.
void	flush ( )	✓ Finalizes the output state so that all buffers are cleared (flushes the output buffers).

Methods defined by ObjectOutputStream

Serialization

- **ObjectInputStream Class**
  - Extends InputStream class and implements ObjectInput interface
    - Responsible for reading objects from a stream
  - Constructor

    ObjectInputStream (InputStream *in*) throws IOException

    - Argument *in* is the input stream from which serialized object will be read
  - All methods throw IOException

Serialization

- **ObjectInputStream Class** (Continued …)

Method		Description
int	read ( )	✓ Returns an integer representation of the next available byte of input. ✓ Returns −1 when end-of-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. ✓ Returns −1 when end-of-file is encountered.
boolean	readBoolean ( )	✓ Reads and returns a boolean from the invoking stream.
byte	readByte ( )	✓ Reads and returns a byte from the invoking stream.

Methods defined by ObjectInputStream Class

Serialization

- **ObjectInputStream Class** (Continued …)

Method		Description
char	readChar ( )	Reads and returns a *char* from the invoking stream.
double	readDouble ( )	Reads and returns a *double* from the invoking stream.
float	readFloat ( )	Reads and returns a *float* from the invoking stream.
int	readInt ( )	Reads and returns an *int* from the invoking stream.
long	readLong ( )	Reads and returns a *long* from the invoking stream.
short	readShort ( )	Reads and returns a *short* from the invoking stream.

Methods defined by ObjectInputStream Class

Serialization

 ObjectInputStream Class (Continued …)

Method		Description
final Object	readObject ( )	Reads and returns an object from the invoking stream.
int	available ( )	Returns the number of bytes that are now available in the input buffer.
void	close ( )	Closes the invoking stream. Further read attempts will generate an IOException.

Methods defined by ObjectInputStream Class

Serialization

- **Implementation**
  - Create a class to implement Serializable interface
  - Serialization
    - Create a FileOutputStream to refer to a file named “Student.ser”
    - Create an ObjectOutputStream for this file stream
    - Use *writeObject*() method of ObjectOutputStream to serialize the object
    - close the ObjectOutputStream
  - Deserialization
    - Create a FileInputStream to refer to the file named “Student.ser”
    - Create an ObjectInputStream for this file stream
    - Use *readObject*() method of ObjectInputStream to deserialize the object
    - Close the ObjectInputStream

Transient Keyword

During serialization process, we can protect some sensitive field(like password, PIN) from being stored by placing transient keyword any field.

Example:

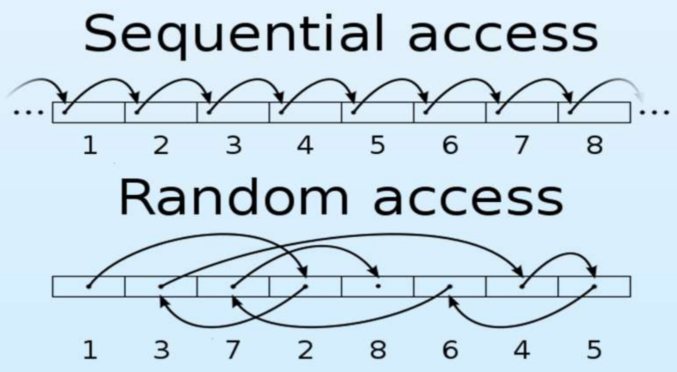
```
Class User{
    String username;
    String email;
    transient String password;

………..
}
```

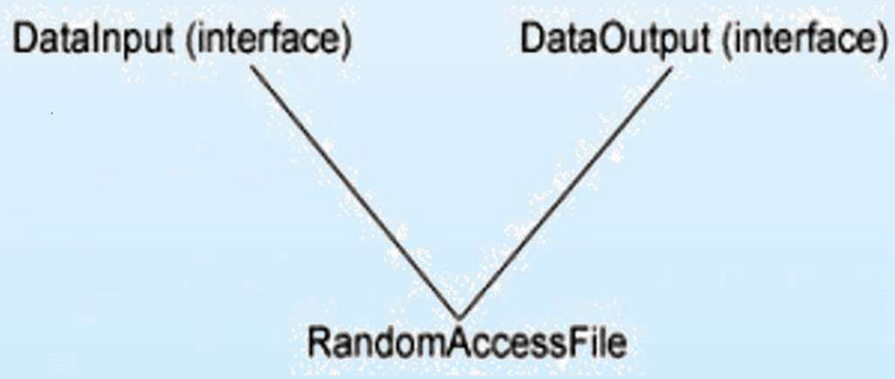
During deserialization, the value of these transient fields will have the default value (null for a JAVA String primitives).

NOTE: Static members are never serialized because they are connected to class not object of class

RandomAccessFile

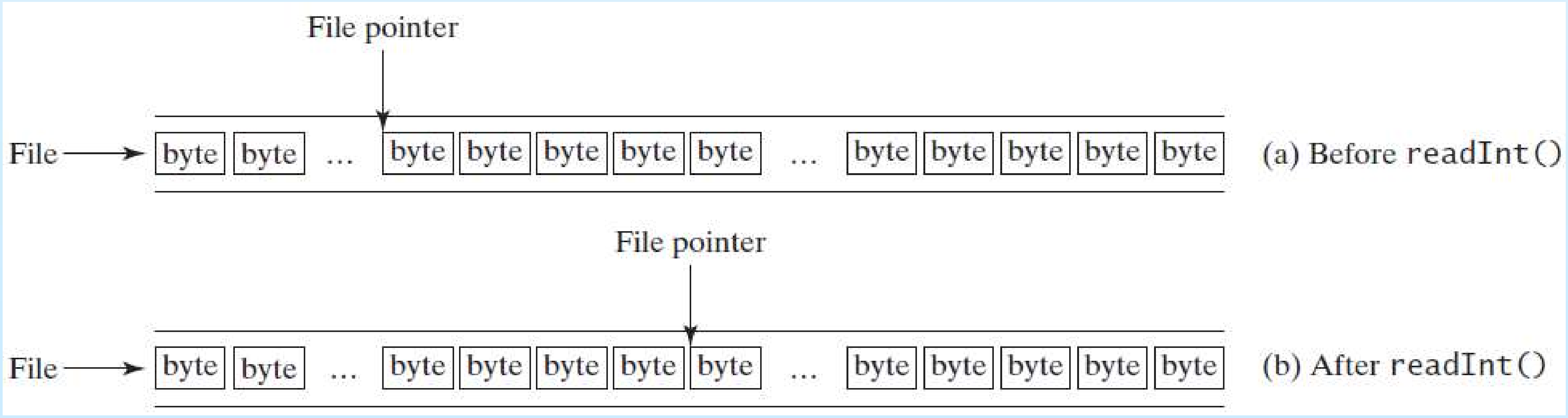
- Random access files are files in which records can be accessed in any order.
- RandomAccessFile class to allow a file to be read from and write to at random locations.





Byte Streams

- **RandomAccessFile**
  - Constructors
    - RandomAccessFile (File *fileObj*, String *access*) throws IOException
    - RandomAccessFile (String *fileName*, String *access*) throws IOException
  - Access modes:
    - 'r' – read only
    - 'rw' – read-write



Byte Streams

- **RandomAccessFile** (Continued …)

Method		Description
void	close ()	Closes the random access file stream and releases any system resources associated with the stream
long	getFilePointer ()	Returns the current offset in the file
long	length ()	Returns the length of the file
void	seek (long pos)	Sets the file-pointer offset, measured from the beginning of the file, at which next read or write occurs
void	setLength (long newLen)	Sets the length of the file
int	skipBytes (int n)	Attempts to skip over *n* bytes of input discarding the skipped bytes

Byte Streams

- **RandomAccessFile** (Continued …)

Method		Description
int	read ()	Reads a byte of data
int	read (byte[] b)	Reads up to *b.length* bytes of data into the array of bytes
int	read (byte[], int off, int len)	Reads up to *len* bytes of data from position *offset* into the array of bytes
void	write (int b)	Writes the specified byte
void	write (byte[] b)	Writes *b.length* bytes of data from the specified byte array
void	write (byte[], int off, int len)	Writes *len* bytes of data from position *offset* from the specified byte array

Byte Streams

- **RandomAccessFile** (Continued …)

Method		**Method**	
boolean	readBoolean ()	void	writeBoolean (boolean b)
byte	readByte ()	void	writeByte (byte b)
char	readChar ()	void	writeChar (char c)
double	readDouble ()	void	writeDouble (double d)
float	readFloat ()	void	writeFloat (float f)
String	readLine ()	void	writeBytes (String str)
int	readInt ()	void	writeInt (int i)
long	readLong ()	void	writeLong (long l)
short	readShort ()	void	writeShort (short s)

END

PresentationPoint