Java I/O Stream

جامعة طرابلس - كلية تقنية المعلومات

د. عبدالحميد الواعر

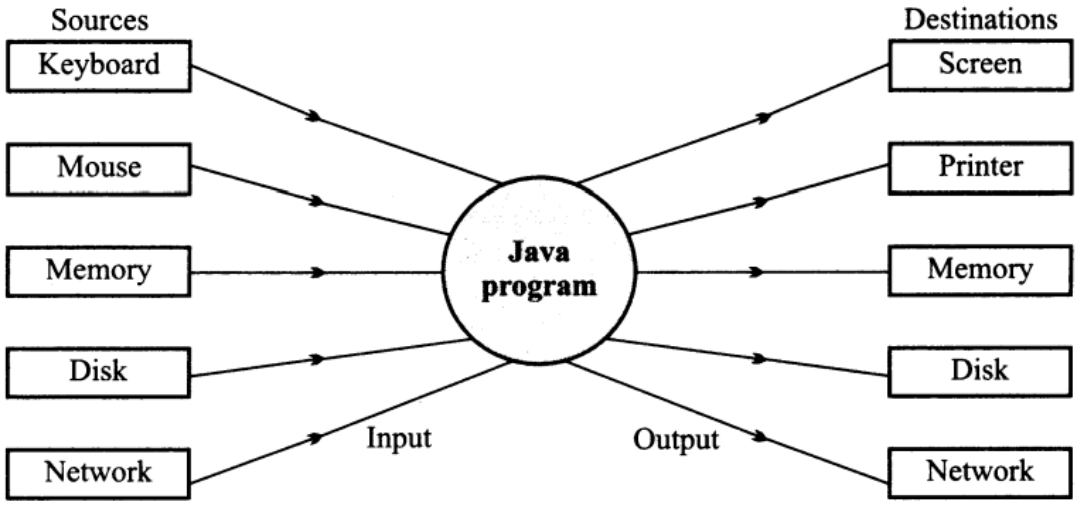
Java I/O stream

Java I/O stream

- تقوم البرامج في لغة جافا بقراءة البيانات من data sources(مثل ملف ، لوحة المفاتيح) وكتابة النتائج إلى data sink (مثل الشاشة ، ملف).
- عمليات الادخال والاخراج في لغة جافا يتم التعامل معها بما يعرف ب Stream.
- Streamهو عبارة عن تدفق للبيانات بشكل متسلسل ومستمر في أتجاه واحد.
- Java I/O Stream يدعم استخدام أنواع متعددة من البيانات مثل :
- Primitive data types
- Objects

Java I/O stream

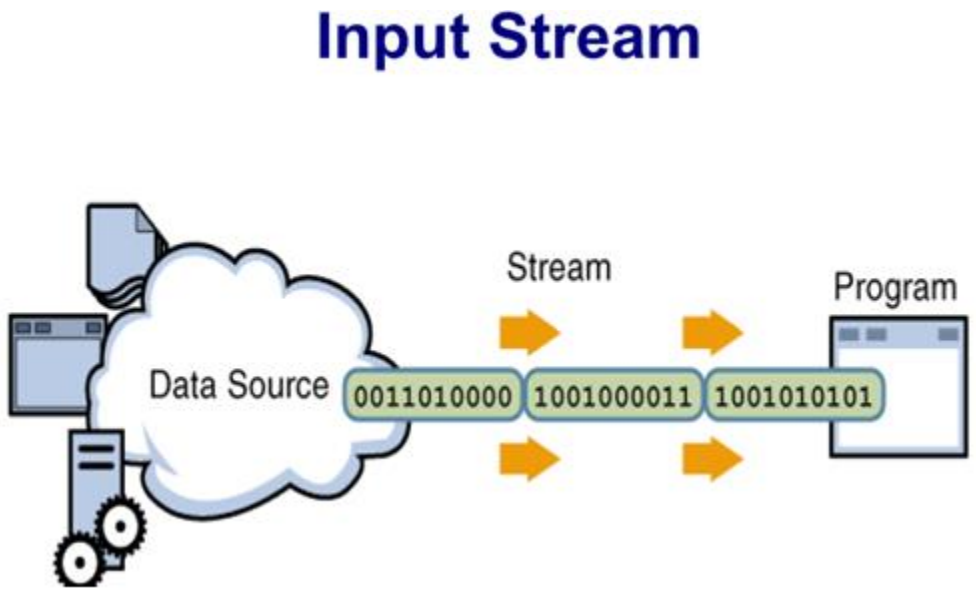
- Stream عبارة عن سلسلة من البيانات تنتقل من مكان إلى أخر.
- I/O Stream يمكن أن يمثل input source أو output destination.
- I/O Stream يمكن أن يستخدم في تمثيل أنواع مختلفة من sources.



Relationship of Java program with I/O devices

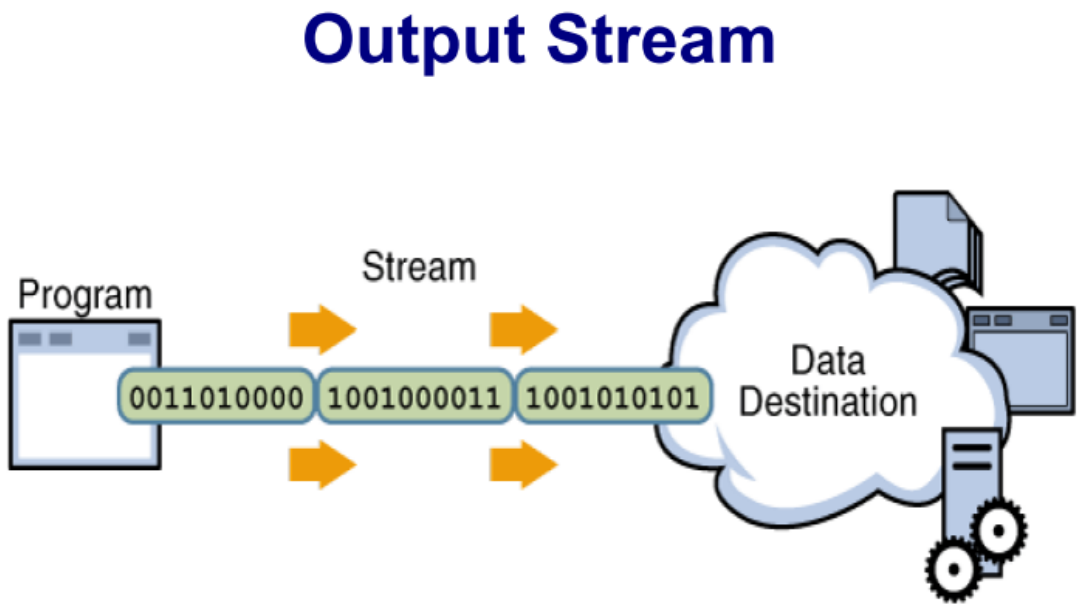
Input stream

- يستخدم في قراءة البيانات من source بشكل متتالي عنصر بعد الاخر.



Output stream

- يستخدم في كتابة البيانات إلى destination بشكل متتالي عنصر بعد الاخر.



Input stream

- يمكن القراءة باستخدام هذا النوع من stream .
- يستخدم في abstract classes التالية:
  - The InputStream Class
  - The Reader Class

Output stream

- يمكن الكتابة باستخدام هذا النوع من stream.
- يستخدم في abstract classes التالية:
  - The OutputStream Class
  - The Writer Class

الخطوات الثلاث للتعامل مع Stream

- فتح input/output streamوربطهما مع source/destinationمن خلال تكوين I/O Stream المناسب.
- القراءة من input stream حتى نهاية البيانات أو الكتابة في output stream.
- أغلاق input/output stream.

أنواع Stream

1. Byte streams

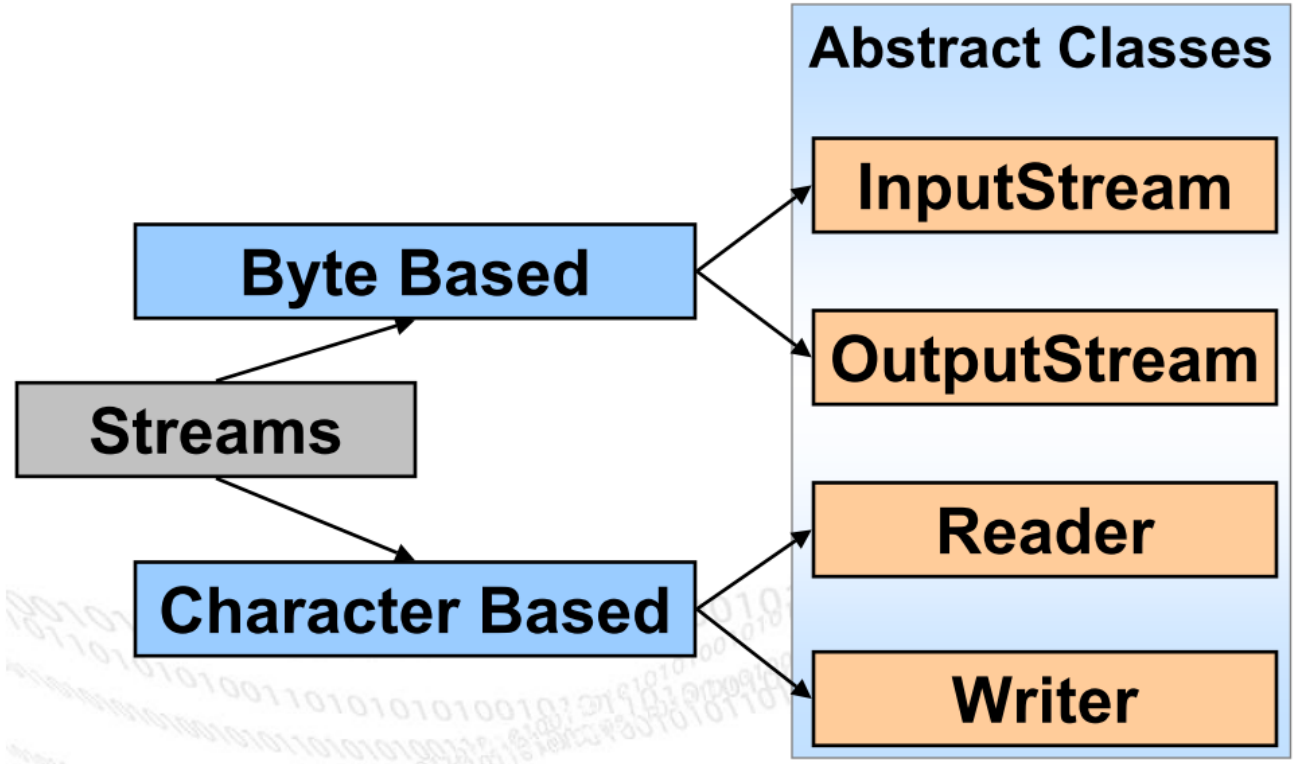
- هو Stream الذي يستعمل بايت واحد في تدفق البيانات وعادة يستخدم في حالة Binary data.
- يستخدم فيه abstract classes التالية:
  - **The InputStream Class**
  - **The OutputStream Class**

أنواع Stream

2. Character streams

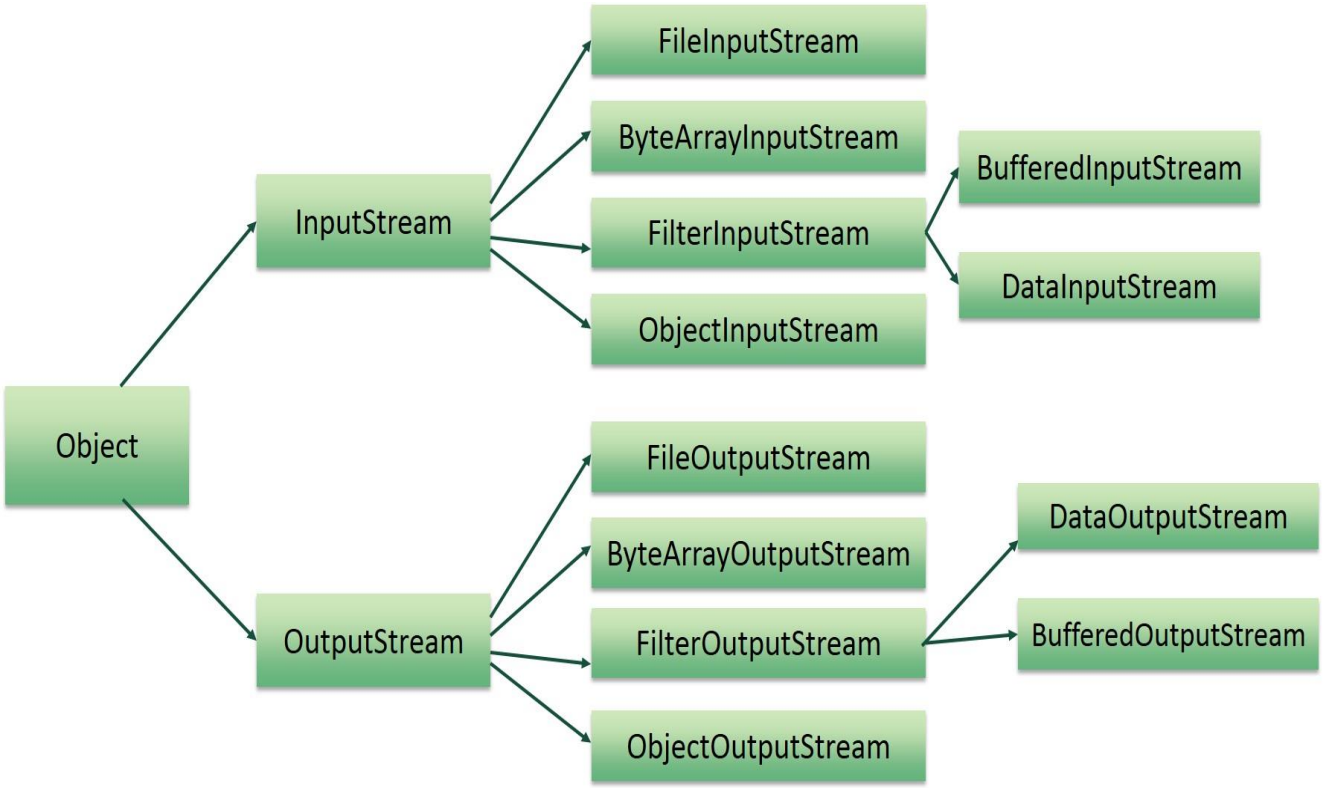
- هو Stream الذي يستعمل في Unicode characters في تدفق البيانات والتي عادة تستخدم في حالة Text data.
- يستخدم فيه abstract classes التالية:
  - **The Reader class**
  - **The Writer class**

أنواع Stream

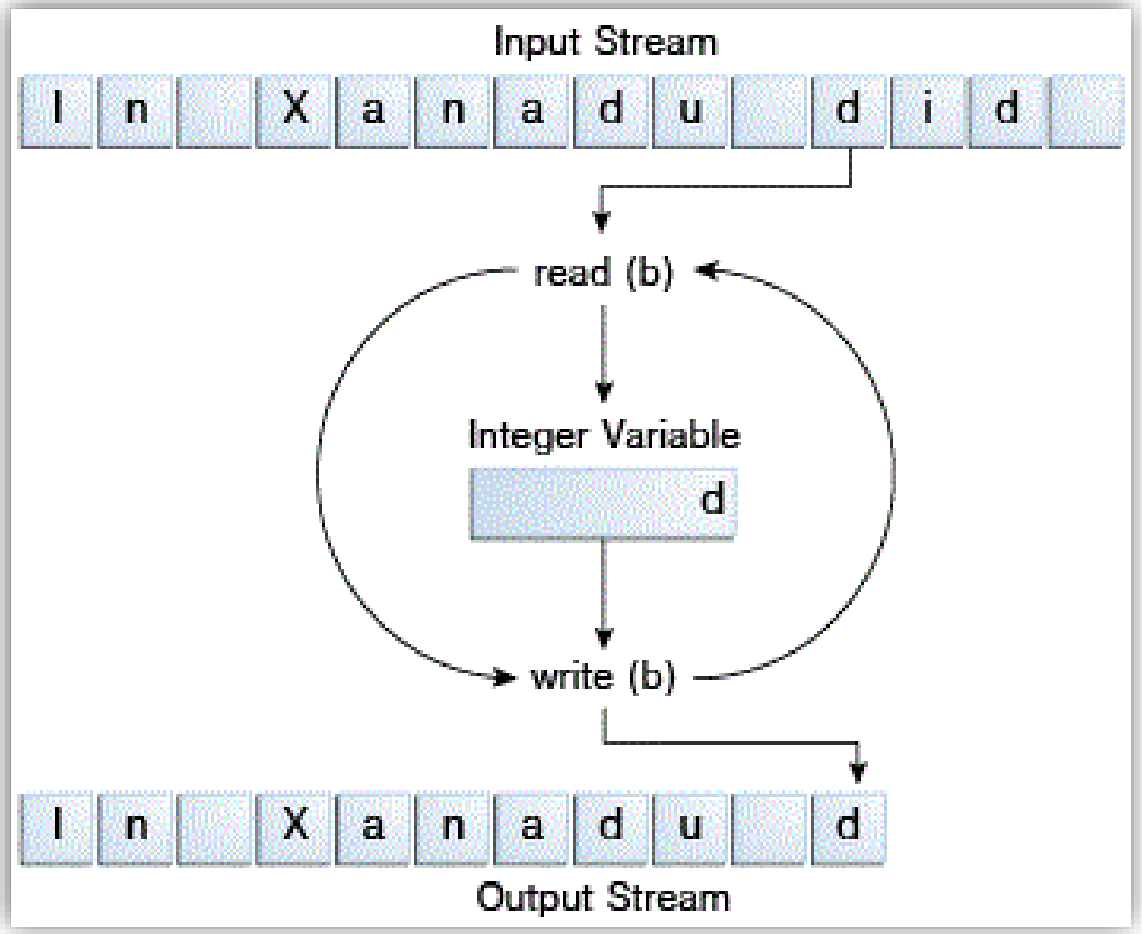


Byte stream

- يستخدم في قراءة وكتابة البيانات في شكل Bytes.
- كل أنواع byte streams المستخدمة يتم اشتقاقها من Super classes:
  - **InputStream**
  - **OutputStream**
- الشكل التالي يبين بعض من أنواع Byte stream.:



مثال:


Input Stream
I n X a n a d u d i d
read (b)
Integer Variable
d
write (b)
I n X a n a d u d
Output Stream

InputStream Class

`public int read(-) throws IOException`
An overloaded method, which also has three versions like that of the *Reader* class. Reads bytes.
`public abstract int read()` - Reads the next byte of data from this stream.
`public int read(byte[] bBuf)` - Reads some number of bytes and stores them in the *bBuf* byte array.
`public abstract int read(char[] cbuf, int offset, int length)` - Reads up to *length* number of bytes and stores them in the byte array *bBuf* starting at the specified *offset*.
`public abstract void close() throws IOException`
Closes this stream. Calling the other *InputStream* methods after closing the stream would cause an *IOException* to occur.

OutputStream Class

`public void write(-) throws IOException`
An overloaded method for writing bytes to the stream. It has three versions:
`public abstract void write(int b)` – Writes the specified byte value *b* to this output stream.
`public void write(byte[] bBuf)` – Writes the contents of the byte array *bBuf* to this stream.
`public void write(byte[] bBuf, int offset, int length)` – Writes *length* number of bytes from the *bBuf* array to this stream, starting at the specified *offset* to this stream.
`public abstract void close() throws IOException`
Closes this stream and releases any system resources associated with this stream. Invocation of other methods after calling this method would cause an *IOException* to occur.
`public abstract void flush()`
Flushes the stream (i.e., bytes saved in the buffer are immediately written to the intended destination).

FileInputStream class

- يستخدم في قراءة البيانات في شكل Bytes من ملف.

java.io

Class FileInputStream

java.lang.Object
    java.io.InputStream
        java.io.FileInputStream

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

FileOutputStream class

- يستخدم في أنشاء ملف وكتابة البيانات في شكل Bytes.

java.io

Class FileOutputStream

java.lang.Object
    java.io.OutputStream
        java.io.FileOutputStream

Constructor and Description
`FileOutputStream(File file)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(File file, boolean append)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(FileDescriptor fdObj)` Creates a file output stream to write to the specified file descriptor, which represents an existing connection to an actual file in the file system.
`FileOutputStream(String name)` Creates a file output stream to write to the file with the specified name.
`FileOutputStream(String name, boolean append)` Creates a file output stream to write to the file with the specified name.

مثال:

- البرنامج التالي يقوم بعمل نسخة من ملف باستخدام الدالة read() و الدالة write(int d).

```
public class FileInputStreamExample {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String img = "image1.jpg";
        File fileIn = new File(path + img);
        try {
            FileInputStream fis = new FileInputStream(fileIn);
            FileOutputStream fos = new FileOutputStream(path + "copy of " + img);
            int d;
            while ((d = fis.read()) != -1) {
                fos.write(d);
            }
            fis.close();
            fos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

مثال:

- البرنامج التالي يقوم بعمل نسخة من ملف باستخدام الدالة read(byte[] a) و الدالة write(byte[] a).

```
public class FileInputStreamBExample {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String img = "image1.jpg";
        File fileIn = new File(path + img);
        try {
            FileInputStream fis = new FileInputStream(fileIn);
            FileOutputStream fos = new FileOutputStream(path + "copy of " + img);
            int d;
            byte [] a = new byte[1000];
            while ((d = fis.read(a)) != -1) {
                fos.write(a);
            }
            fis.close();
            fos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

مثال:

- البرنامج التالي يقوم بعمل نسخة من ملف باستخدام الدالة read(byte[] b, int off, int len) و الدالة write(byte[] b, int off, int len).

```
public class FileInputStreamBLExample1 {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String img = "image1.jpg";
        File fileIn = new File(path + img);
        try {
            FileInputStream fis = new FileInputStream(fileIn);
            FileOutputStream fos = new FileOutputStream(path + "copy of " + img);
            int d=0;
            byte [] a = new byte[1000];
            while ((d = fis.read(a)) != -1) {
                fos.write(a,0,d);
            }
            fis.close();
            fos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

BufferedInputStream class

- يستخدم internal buffer لتحسين عملية قراءة البيانات من stream.

Class BufferedInputStream

java.lang.Object
    java.io.InputStream
        java.io.FilterInputStream
            java.io.BufferedInputStream

Constructor and Description
`BufferedInputStream(InputStream in)` Creates a `BufferedInputStream` and saves its argument, the input stream `in`, for later use.
`BufferedInputStream(InputStream in, int size)` Creates a `BufferedInputStream` with the specified buffer size, and saves its argument, the input stream `in`, for later use.

BufferedOutputStream class

- يستخدم internal buffer لتحسين عملية كتابة البيانات من stream.

Class BufferedOutputStream

java.lang.Object
    java.io.OutputStream
        java.io.FilterOutputStream
            java.io.BufferedOutputStream

Constructor and Description
`BufferedOutputStream(OutputStream out)` Creates a new buffered output stream to write data to the specified underlying output stream.
`BufferedOutputStream(OutputStream out, int size)` Creates a new buffered output stream to write data to the specified underlying output stream with the specified buffer size.

مثال:

```
public class BufferedInputStreamExample {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String img = "image1.jpg";
        File fileIn = new File(path + img);
        try {
            FileInputStream fis = new FileInputStream(fileIn);
            FileOutputStream fos = new FileOutputStream(path + "copy of " + img);
             BufferedInputStream bis = new BufferedInputStream(fis);
             BufferedOutputStream bos = new BufferedOutputStream(fos);
            int d;
            while ((d = bis.read()) != -1) {
                bos.write(d);
            }
            fis.close();
            fos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Data Stream

- تستخدم كل من DataInputStream & DataOutputStream classes مع اي InputStream أو OutputStream لتقوم بعمل فلترة لل Stream ليقوم بعملية الادخال والاخراج حسب نوع البيانات المطلوبة مثل int أو String.

DataInputStream Class

Class DataInputStream

java.lang.Object
    java.io.InputStream
        java.io.FilterInputStream
            java.io.DataInputStream

Constructor and Description
DataInputStream(InputStream in) Creates a DataInputStream that uses the specified underlying InputStream.

DataInputStream Class

Methods

Modifier and Type	Method and Description
int	**read**(byte[] b) Reads some number of bytes from the contained input stream and stores them into the buffer array b.
int	**read**(byte[] b, int off, int len) Reads up to len bytes of data from the contained input stream into an array of bytes.
boolean	**readBoolean**() See the general contract of the readBoolean method of DataInput.
byte	**readByte**() See the general contract of the readByte method of DataInput.
char	**readChar**() See the general contract of the readChar method of DataInput.
double	**readDouble**() See the general contract of the readDouble method of DataInput.
float	**readFloat**() See the general contract of the readFloat method of DataInput.
long	**readLong**() See the general contract of the readLong method of DataInput.
short	**readShort**() See the general contract of the readShort method of DataInput.
int	**readUnsignedByte**() See the general contract of the readUnsignedByte method of DataInput.
int	**readUnsignedShort**() See the general contract of the readUnsignedShort method of DataInput.
String	**readUTF**() See the general contract of the readUTF method of DataInput.

DataOutputStream Class

Class DataOutputStream

java.lang.Object
    java.io.OutputStream
        java.io.FilterOutputStream
            java.io.DataOutputStream

Constructor and Description
`DataOutputStream(OutputStream out)` Creates a new data output stream to write data to the specified underlying output stream.

DataOutputStream Class

Methods

Modifier and Type	Method and Description
void	**flush**() Flushes this data output stream.
int	**size**() Returns the current value of the counter `written`, the number of bytes written to this data output stream so far.
void	**write**(byte[] b, int off, int len) Writes `len` bytes from the specified byte array starting at offset `off` to the underlying output stream.
void	**write**(int b) Writes the specified byte (the low eight bits of the argument `b`) to the underlying output stream.
void	**writeBoolean**(boolean v) Writes a `boolean` to the underlying output stream as a 1-byte value.
void	**writeByte**(int v) Writes out a `byte` to the underlying output stream as a 1-byte value.
void	**writeBytes**(String s) Writes out the string to the underlying output stream as a sequence of bytes.
void	**writeChar**(int v) Writes a `char` to the underlying output stream as a 2-byte value, high byte first.
void	**writeChars**(String s) Writes a string to the underlying output stream as a sequence of characters.
void	**writeDouble**(double v) Converts the double argument to a `long` using the `doubleToLongBits` method in class `Double`, and then writes that `long` value to the underlying output stream as an 8-byte quantity, high byte first.
void	**writeFloat**(float v) Converts the float argument to an `int` using the `floatToIntBits` method in class `Float`, and then writes that `int` value to the underlying output stream as a 4-byte quantity, high byte first.
void	**writeInt**(int v) Writes an `int` to the underlying output stream as four bytes, high byte first.
void	**writeLong**(long v) Writes a `long` to the underlying output stream as eight bytes, high byte first.
void	**writeShort**(int v) Writes a `short` to the underlying output stream as two bytes, high byte first.
void	**writeUTF**(String str) Writes a string to the underlying output stream using **modified UTF-8** encoding in a machine-independent manner.

مثال:

```
public class DataOuputStreamExample {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String img = "MyData.out";
        File fileIn = new File(path + img);
        try {
            DataOutputStream dos = new DataOutputStream(
                                    new BufferedOutputStream(
                                        new FileOutputStream(fileIn)));
            dos.writeInt(15);
            dos.writeDouble(3.14);
            dos.writeBoolean(true);
            dos.writeUTF("Ahmad");
            dos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

مثال:

```
public class DataInputStreamExample {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String img = "MyData.out";
        File fileIn = new File(path + img);
        try {
            DataInputStream dos = new DataInputStream(
                                  new BufferedInputStream(
                                        new FileInputStream(fileIn)));
            System.out.println(dos.readInt());
            System.out.println(dos.readDouble());
            System.out.println(dos.readBoolean());
            System.out.println(dos.readUTF());
            dos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

ObjectInputStream & ObjectOutputStream

- تستخدم كل من ObjectInputStream & ObjectOutputStream classes في عملية حفظ وأسترجاع objects ليتم استعمالها مرة أخرى.

- classالمراد حفظ objects الناتجة عنها يجب أن تقوم بعمل implementation ل Serializable interface.

مثال:

```
public class Student implements Serializable{
    private int Id;
    private String name;
    private String address;

    public Student(int Id, String name, String address) {
        this.Id = Id;
        this.name = name;
        this.address = address;
    }

    @Override
    public String toString() {
        return "Student{" + "Id=" + Id + ", name=" + name + ", address=" + address + '}';
    }

}
```

```java
public class ObjectOutputStreamExample {

    public static void main(String[] args) {
        String path = "d:\\test\\";
        String f = "Students.dat";
        Student s = new Student(12345678, "Ahmad", "Tripoli");
        try {
            ObjectOutputStream oos = new ObjectOutputStream(
                                     new FileOutputStream(path + f));
            oos.writeObject(s);
            oos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}
```

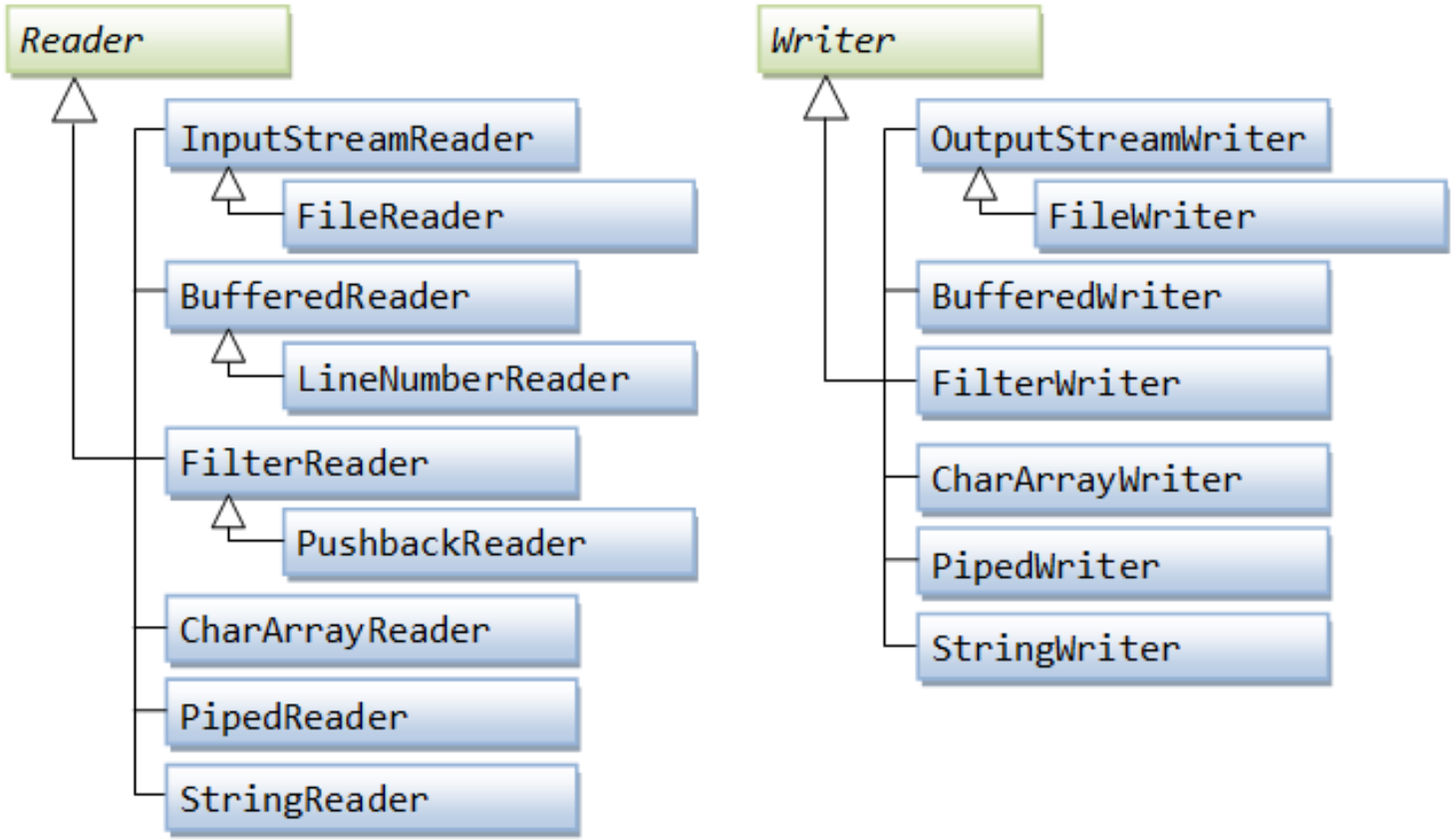
```java
public class ObjectInputStreamExample {

    public static void main(String[] args) {
        String path = "d:\\test\\";
        String f = "Students.dat";
        try {
            ObjectInputStream oos = new ObjectInputStream(
                                        new FileInputStream(path + f));
            Student s = (Student) oos.readObject();
            System.out.println(s);
            oos.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
    }

}
```

Character streams

- هو Stream الذي يستعمل Unicode characters في تدفق البيانات والتي عادة تستخدم في حالة Text data.
- يستخدم فيه abstract classes التالية:
  - **The Reader class**
  - **The Writer class**



Reader Class

`public int read(-) throws IOException`
An overloaded method, which has three versions. Reads character(s), an entire character array or a portion of a character array.
`public int read()` - Reads a single character.
`public int read(char[] cbuf)` - Reads characters and stores them in character array *cbuf*.
`public abstract int read(char[] cbuf, int offset, int length)` - Reads up to *length* number of characters and stores them in character array *cbuf* starting at the specified *offset*.
`public abstract void close() throws IOException`
Closes this stream. Calling the other *Reader* methods after closing the stream would cause an *IOException* to occur.

`public void mark(int readAheadLimit) throws IOException`
Marks the current position in the stream. After marking, calls to reset() will attempt to reposition the stream to this point. Not all character-input streams support this operation.
`public boolean markSupported()`
Indicates whether a stream supports the mark operation or not. Not supported by default. Should be overidden by subclasses.
`public void reset() throws IOException`
Repositions the stream to the last marked position.

Writer Class

`public void write(-) throws IOException`
An overloaded method with five versions:
`public void write(int c)` – Writes a single character represented by the given integer value.
`public void write(char[] cbuf)` – Writes the contents of the character array *cbuf*.
`public abstract void write(char[] cbuf, int offset, int length)` – Writes *length* number of characters from the *cbuf* array, starting at the specified *offset*.
`public void write(String str)` – Writes the string *string*.
`public void write(String str, int offset, int length)` – Writes *length* number of characters from the string *str*, starting at the specified *offset*.
`public abstract void close() throws IOException`
Closes this stream after flushing any unwritten characters. Invocation of other methods after closing this stream would cause an *IOException* to occur.
`public abstract void flush()`
Flushes the stream (i.e., characters saved in the buffer are immediately written to the intended destination).

FileReader Class

Constructor and Description
`FileReader(File file)` Creates a new `FileReader`, given the `File` to read from.
`FileReader(FileDescriptor fd)` Creates a new `FileReader`, given the `FileDescriptor` to read from.
`FileReader(String fileName)` Creates a new `FileReader`, given the name of the file to read from.

FileWriter Class

Constructor and Description
`FileWriter(File file)` Constructs a FileWriter object given a File object.
`FileWriter(File file, boolean append)` Constructs a FileWriter object given a File object.
`FileWriter(FileDescriptor fd)` Constructs a FileWriter object associated with a file descriptor.
`FileWriter(String fileName)` Constructs a FileWriter object given a file name.
`FileWriter(String fileName, boolean append)` Constructs a FileWriter object given a file name with a boolean indicating whether or not to append the data written.

مثال:

```
public class FileReaderExample {
    public static void main(String[] args) {
        String path = "d:\\test\\";
        String f = "test.txt";
        File fileIn = new File(path + f);
        try {
            FileReader fr = new FileReader(fileIn);
            FileWriter fw = new FileWriter(path + "copy of " + f);
            int d;
            while ((d = fr.read()) != -1) {
                System.out.println(d);
                fw.write(d);
            }
            fr.close();
            fw.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}
```

BufferedReader Class

Constructor and Description
`BufferedReader(Reader in)` Creates a buffering character-input stream that uses a default-sized input buffer.
`BufferedReader(Reader in, int sz)` Creates a buffering character-input stream that uses an input buffer of the specified size.

BufferedWriter Class

Constructor and Description
`BufferedWriter(Writer out)` Creates a buffered character-output stream that uses a default-sized output buffer.
`BufferedWriter(Writer out, int sz)` Creates a new buffered character-output stream that uses an output buffer of the given size.

مثال:

```
public class BufferedReaderExample {
    public static void main(String[] args) {
         String path = "d:\\test\\";
        String f = "test.txt";
        File fileIn = new File(path + f);
        try {
            FileReader fr = new FileReader(fileIn);
            FileWriter fw = new FileWriter(path + "copy of " + f);
            BufferedReader br = new BufferedReader(fr);
            BufferedWriter bw = new BufferedWriter(fw);
            int d;
            while ((d = br.read()) != -1) {
                System.out.println(d);
                bw.write(d);
            }
            br.close();
            bw.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}
```

مثال:

- البرنامج التالي يقوم بعمل نسخة من ملف مع أستخدام الدالة newLine() .

```
public class ReadLineExample {
    public static void main(String[] args) {
          String path = "d:\\test\\";
        String f = "test.txt";
        File fileIn = new File(path + f);
        try {
            FileReader fr = new FileReader(fileIn);
            FileWriter fw = new FileWriter(path + "copy of " + f);
            BufferedReader br = new BufferedReader(fr);
            BufferedWriter bw = new BufferedWriter(fw);
            String d;
            while ((d = br.readLine()) != null) {
                System.out.println(d);
                bw.write(d);
                bw.newLine();
            }
            br.close();
            bw.close();
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}
```