UD1. Ficheros y Flujos

Repaso Excepciones y Ficheros

Excepciones

```
try{
}catch(NumberFormatException nfe){
        mensaje = "Caracteres no numéricos";
 }catch (ArithmeticException ae) {
        mensaje = "División por cero";
}catch (ArrayIndexOutOfBoundsException aio) {
        mensaje = "Array fuera de rango";
}catch(ClassNotFoundException cnf){
        System.out.println("Error la clase");
}
```

Excepciones (cont.)

```
//Siempre se ejecuta
    }finally{
      System.out.println("finally  Se ejecuta siempre");
    }
//Tratamiento de ficheros
    }catch(FileNotFoundException  fnfe){
      System.out.println("Error  en el fichero");
    }catch(IOException  ioe){
      System.out.println("Error  E/L");
    }finally{
     try{
        os.close();
     }catch(IOException  ioe){
        System.out.println("Error  E/L");
     }catch(NullPointerException  np){
        System.out.println("Error  NullPointer");
      }
    }
```

Ficheros. Clase File

- //Constructor

  new File

- //Métodos

  .mkdir()

  .renameTo()

  .getName()

  .getPath()

  .getParent()

  .canWrite()

  .canRead()

  .length()

  .isDirectory()

  .isFile()

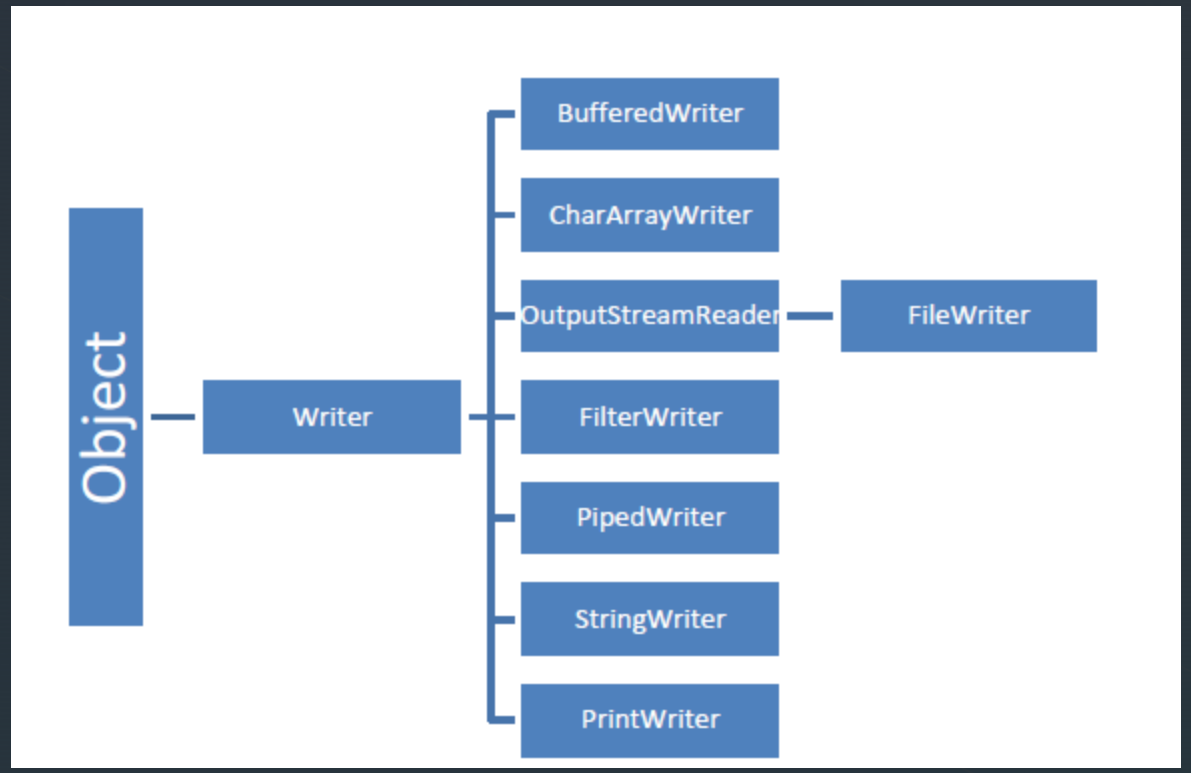
  .exists()

Ficheros. Ficheros Secuenciales

- Texto
  - Escritura

```
FileWriter fw


salida= new PrintWriter()


salida.flush();//garantiza que todos los datos enviados a través del buffer de salida han sido escritos en el fichero

salida.close();// cierra la conexión con el fichero y libera los recursos utilizados por ésta


fw.write()


fw.append()

fw.close()


BufferedWriter bw = new BufferedWriter(fw);


bw.write("Fila número: " +i);

bw.newLine();

bw.close();
```

Ficheros. Ficheros Secuenciales

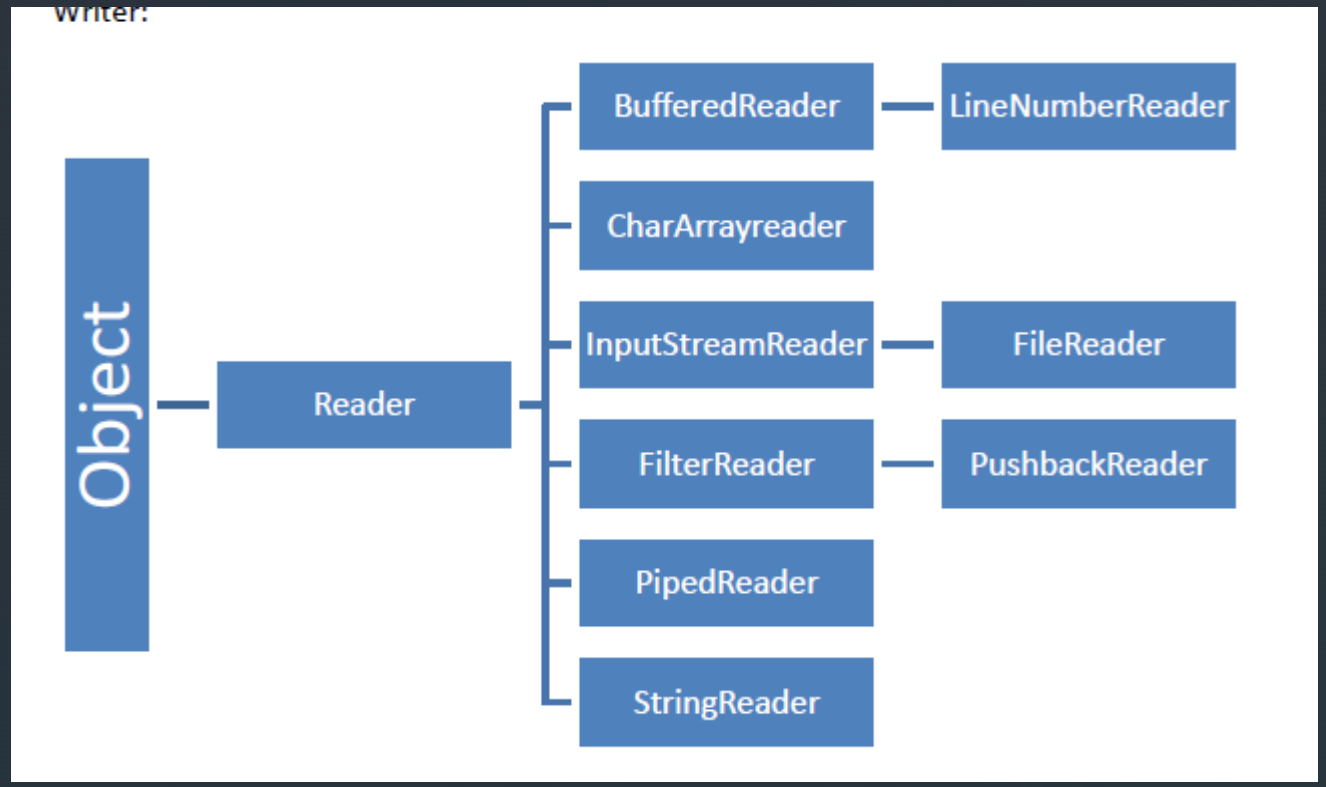


Ficheros. Ficheros Secuenciales

- Texto
  - Lectura

```
fr = new FileReader


br = new BufferedReader(fr); //lee líneas
while((nombre = br.readLine())!= null){
String nombre; // variable donde se recupera la informacion
System.out.println(nombre);
br.close();
//OPCION1
while((caracter = fr.read()) != -1 ){//lee caracter a carácter
//OPCION2
 sc = new Scanner(fr); //mientras no encuentre el final sigue leyendo
                while(sc.hasNext()){
                System.out.println(sc.next());
                }


fr.close();
```

Ficheros. Ficheros Secuenciales



Ficheros. Ficheros Secuenciales

- Datos
  - Lectura

```
flujolectura= new DataInputStream (new FileInputStream(archivo));


while (true){
        System.out.println ("Codigo de Usuario: "+
        flujolectura.readInt());


        System.out.println ("Nombre de Usuario: "+
        flujolectura.readUTF());
    }
flujolectura.close();
```

Ficheros. Ficheros Secuenciales

- **Métodos para Lectura**

boolean readBoolean();

byte readByte();

int readUnsignedByte();

int readUnsignedShort();

short readShort();

char readChar();

int readInt();

long readLong();

floar readFloat();

double readDouble();

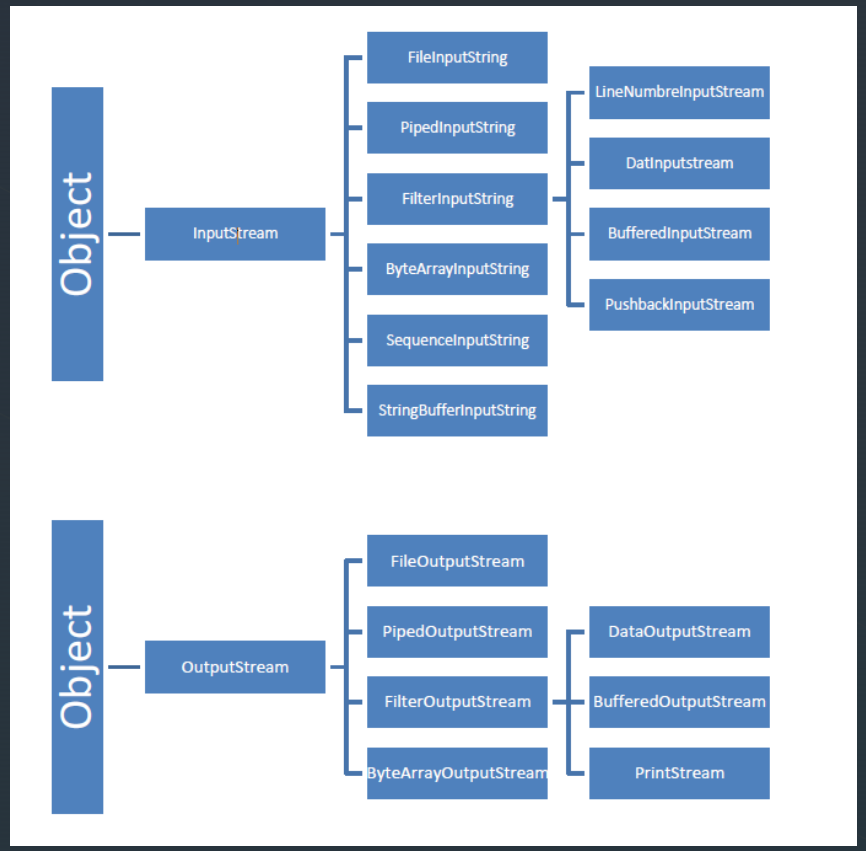
String readUTF();

Ficheros. Ficheros Secuenciales.Flujos

- FileInputStream -Para leer información de un fichero.
- PipedInputStream -Implementa el concepto de “tubería”.
-

Método	Acción
int read()	-Lee un carácter y lo devuelve
int read(char[] buf)	-Lee hasta buf.length caracteres de datos de una matriz de caracteres (buf). Los caracteres leídos del fichero se van almacenando en buf.
int read(char[] buf, int desplazamiento, int n)	- Lee hasta n caracteres de datos en la matriz buf y comenzando por buf [desplazamiento] y devuelve el número de caracteres leídos.

Ficheros. Ficheros Secuenciales. Flujos



Ficheros. Ficheros Secuenciales. Flujos

- FileOutputStream Para enviar información a un fichero.
- PipedOutputStream Cualquier información que se desee escribir aquí acaba
- automáticamente como entrada del PipedInputStream asociado.
- Implementa el concepto de “tubería”.
- **Método** **Función**
- void write(int b) -Escribe un byte
- void write(byte[] b) -Escribe b.length bytes
- void write (byte[] b, int desplazamiento, int n - Escribe n bytes a partir de la matriz de bytes de entrada y comenzando por b [desplazamiento]

Ficheros. Ficheros Secuenciales

- Datos
  - Escritura

```
FileOutputStream fEscritura = null;
DataOutputStream ds = null;


ds.writeInt(array[i]);
ds.writeUTF(nombres[i]);


ds.writeDouble(i);


ds.close();
fEscritura.close();
```

Ficheros. Ficheros Secuenciales

- **Métodos para Escritura**

  void writeBoolean(boolean v);

  void writeByte(int v);

  void writeBytes(String s);

  void writeShort(int v);

  void writeChars(String s);

  void writeChar(int v);

  void writeInt(int v);

  void writeLong(long v);

  void writeFloat(float v);

  void writeDouble(double v);

  void writeUTF(String str);

Ficheros. Ficheros Secuenciales

- Datos. Otros Tipos

```
//leemos un caracter y sino es el final lo escribimos el final del fichero -1
            while((caracter = fr.read()) != -1 ){
            System.out.println((char)caracter);


fr = new FileReader(f);
            sc = new Scanner(fr);
            //mientras no encuentre el final sigue leyendo - array de palabras
            while(sc.hasNext()){
               System.out.println(sc.next());
            }


sc.close();
fr.close();
```

Ficheros. Ficheros Serialización

- Serialización Objetos
  - Escritura

```
fs = new FileOutputStream("Personas.txt");


os = new ObjectOutputStream(fs);//sobreescribe el fichero

mo = new MiObjectOuputStream(fo);//añade objetos a un fichero existente


//instancia del objeto

        Objeto = new

oo.writeObject(objeto)


fo.close();

oo.close();
```

```
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
/* la clase MiObjectOuputStream quedaría con los siguientes métodos*/
public class MiObjectOuputStream extends ObjectOutputStream {


//Constructor sin parámetros
protected MiObjectOuputStream() throws IOException,
SecurityException {
super();



//Constructor que recibe como parámetro un objeto OutputStream
protected MiObjectOuputStream(OutputStream out) throws
IOException {
super(out);
}
/*redefinición del método que escribe la cabecera para que no haga nada en caso de que el fichero ya tenga datos
*/
protected void writeStreamHeader(){
}
}
```

Ficheros. Ficheros Serialización

- Serialización
  - Lectura

```
fi = new FileInputStream("Alumnos.dat");
    oi = new ObjectInputStream(fi);
    while(true){
    //se crea el objeto donde se va a guardar los datos leídos del disco
    clase objeto = (clase) oi.readObject();
    System.out.println(objeto);
    }

  oi.close();
```

Ficheros. Ficheros Aleatorios o Directos

- Escritura

```
puntero = new RandomAccessFile (f,"rw");
puntero.seek((clave-1) * tamanhoRegistro);
        puntero.writeInt(clave);
        puntero.writeUTF(nombre);
        puntero.writeInt(edad);
```

- Lectura

```
puntero = new RandomAccessFile (f,"r");

for (int r=0; r < contadorRegistros; r++) {
        puntero.seek(r* tamanhoRegistro);
        clave=puntero.readInt();
        nombre = puntero.readUTF();
        edad=puntero.readInt()
```

Ficheros. Ficheros Aleatorios o Directos

Métodos:	Función
`long getFilePointer()`	Devuelve la posición actual del puntero del archivo.
`void seek( long k )`	Coloca el puntero del archivo en la posición indicada por **k** (los archivos empieza en la posición 0).
`int skipBytes( int n )`	Intenta saltar n bytes desde la posición actual.
`long length()`	Devuelve la longitud del archivo.
`void setLength( long t)`	Establece a **t** el tamaño del archivo.

Métodos Escritura:	Función
`void write(byte b[], int ini, int len)`	Escribe len caracteres del vector b.
`void write(int i )`	Escribe la parte baja de i (un byte) en el flujo.
`void writeXxx(xxx)`	Escribe el tipo indicado en xxx.

Métodos Lectura:	Función
`xxx readXxx();`	Lee y devuelve el tipo leído
`void readFully( byte b[] );`	Lee bytes del archivo y los almacena en un vector de bytes.
`void readFully( byte b[], int ini, int len )`	Lee len bytes del archivo y los almacena en un vector de bytes.
`String readUTF()`	Lee una cadena codificada con el formato UTF-8.

Ficheros. Ficheros XML

- SAX
- DOM