Fundamentos de Computadores y Lenguajes

UNIVERSIDAD DE CANTABRIA

1. Elementos básicos de los lenguajes
2. Instrucciones de control
3. Tipos de datos
4. Métodos
5. Modularidad y Abstracción
6. Tratamiento de errores
7. Entrada/salida simple
8. Entrada/salida con ficheros

Fundamentos de Computadores y Lenguajes

UNIVERSIDAD DE CANTABRIA

7. Entrada/salida con ficheros

- 1. Conceptos básicos
- 2. Entrada/salida de texto
- 3. Entrada/salida de texto con formato
- 4. Entrada/salida binaria

1. Conceptos básicos

La Entrada/Salida de Java se organiza generalmente mediante objetos llamados **`Streams`**

Un **`Stream`** es una secuencia ordenada de datos con un determinado origen o destino

Hay dos tipos de streams:

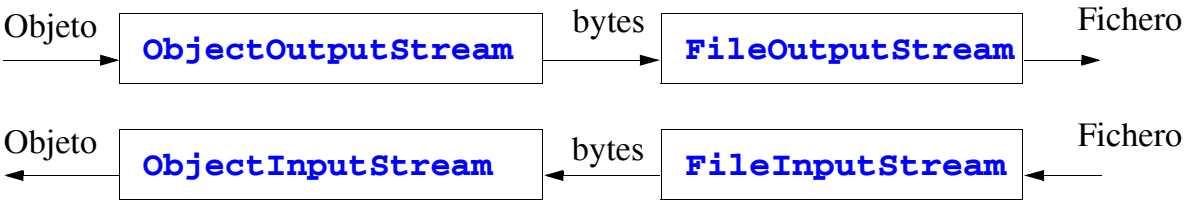
- *de bytes* (binarios): están pensados para ser leídos por un programa
- *de caracteres* (de texto): están pensados para ser leídos y/o creados por una persona

En caso de error, muchos de los métodos de los streams lanzan **`IOException`** y otras excepciones

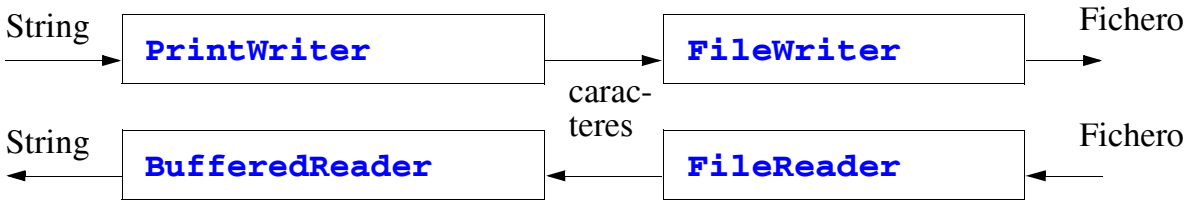
Uso de los Streams

Los streams se usan encadenados. Por ejemplo:

Binarios



De Texto:



Jerarquía de clases ficheros binarios

`OutputStream` -> escritura de ficheros binarios

- **`FileOutputStream`** -> escribe bytes en un fichero
- **`ObjectOutputStream`** -> para escribir objetos y variables en un **`OutputStream`**

`InputStream` -> lectura de ficheros binarios

- **`FileInputStream`** -> lee bytes de un fichero
- **`ObjectInputStream`** -> lee objetos y variables de un **`InputStream`**

Jerarquía de clases ficheros de texto

`Reader` -> lectura de ficheros de texto Unicode

- **`FileReader`**-> lee texto de un fichero
- **`BufferedReader`** -> lee texto (línea a línea) de un **`Reader`**

`Writer` -> escritura de ficheros de texto Unicode

- **`FileWriter`**-> escribe texto en un fichero
- **`PrintWriter`**-> para escribir texto (strings) en un **`Writer`**

Objetos predefinidos

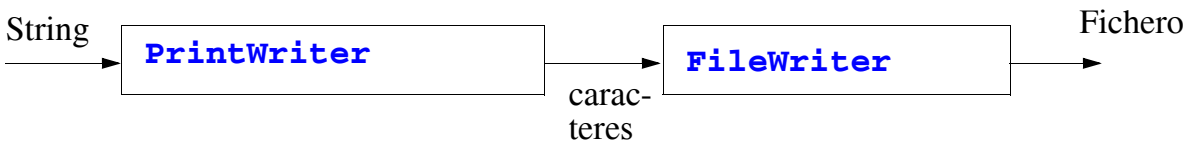
`System.out` es un objeto de la clase `OutputStream` que representa la pantalla

`System.in` es un objeto de la clase `InputStream` que representa el teclado

2. Entrada/salida de texto
Salida de texto

La salida de texto básica sobre ficheros se consigue con la clase `FileWriter`

Descripción	Declaración
Constructor. Requiere el nombre del fichero. Lo crea si no existe	`FileWriter(String s)`
Sincronizar	`void flush()`
Cerrar	`void close()`



Salida de texto (cont.)

La salida de texto formateado es más sencilla con un objeto `PrintWriter`

Descripción	Declaración
Constructor. Requiere un Writer	`PrintWriter(Writer writer)`
Escribir un string	`void print(String str)`
Escribir un string con retorno de línea	`void println(String str)`
Sincronizar	`void flush()`
Cerrar	`void close()`

Ejemplo:

Programa que escribe en un fichero de texto números enteros y reales

```
import java.io.*;

public class EscribeTexto {

  public static void main(String[] args) {

      int i=1, j=2, k=3;
      double x=1.0,y=6.023e23;
```

Ejemplo:

```
      try {
        FileWriter fich = new FileWriter("d1.txt");
        PrintWriter sal = new PrintWriter(fich);
        try {
          sal.println("Enteros "+i+" "+j+" "+k);
          sal.println("Reales "+x+" "+y);
         } finally {
          sal.close();
          fich.close();
         } // try
     } catch (IOException e) {
         System.out.println("Error "+e);
     } // try
   } // main
} // clase
```

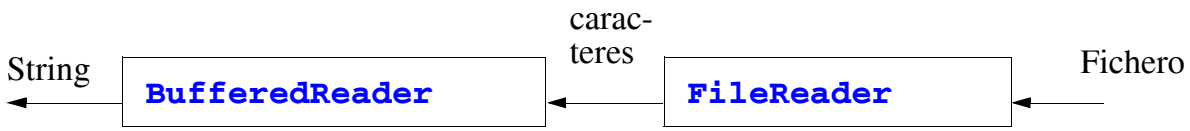

Fichero generado

```
Enteros 1 2 3
Reales 1.0 6.023E23
```

Entrada de texto

La entrada de texto básica sobre ficheros se consigue con la clase `FileReader`, cuyas operaciones son:

Descripción	Declaración
Constructor. Requiere el nombre del fichero. Si no existe lanza FileNotFoundException	`FileReader(String s)throws FileNotFoundException`
Cerrar	`void close()`



Entrada de texto (cont.)

La entrada de texto es más difícil que la salida

Se puede utilizar la clase `BufferedReader`: para leer texto línea a línea

Descripción	Declaración
Constructor. Requiere Un Reader.	`BufferedReader(Reader reader)`
Leer un string. Retorna `null` si se ha llegado al final	`String readLine()`
Cerrar	`void close()`

Luego, cada línea leída se convierte a otros datos con las operaciones `parseInt()`, `parseDouble()`, etc.

Ejemplo

Leer parejas de texto (una línea) y número (otra línea) y escribirlas en pantalla; ejemplo de fichero de entrada:

```
primero
1.0
segundo
3.0
tercero
      4.5
cuarto
8.0
quinto
34R.5
sexto
1.0
```


Ejemplo (cont.)

```
import java.io.*;

public class LeeStrings {

   /**
    * Programa que lee varias parejas de textos y
    * números de un fichero de texto
    */
   public static void main(String[] args) {

      String str,num;
      double x;
      int sep=0;
```

Ejemplo (cont.)

```
try {
  FileReader fich = new FileReader("d2.txt");
 BufferedReader ent = new BufferedReader(fich);
  try {
    do {
      // leer el texto
      str=ent.readLine();
      if (str!=null) {
        // si hay texto escribirlo
        System.out.println("Texto: "+str);
        // leer el número real como texto
        num=ent.readLine();
        if (num==null || num.equals("")) {
           System.out.println("Falta el nº");
        } else {
```


Ejemplo (cont.)

```
          // convertir el texto a numero
          try {
            x=Double.parseDouble(num);
            System.out.println("Numero:"+x);
          } catch (NumberFormatException e) {
            System.out.println ("Error al"+
               "leer numero real: "+num);
          } // try
        } // if
      } // if
    } while (str!=null);
  } finally {
    ent.close();
    fich.close();
} // try
```

Ejemplo (cont.)

```
    } catch (IOException e) {
        System.out.println("Error "+e);
    } // try
  } // main
} // clase
```


Salida para el ejemplo

El programa muestra lo siguiente en la pantalla:

```
Texto: primero
Numero: 1.0
Texto: segundo
Numero: 3.0
Texto: tercero
Numero: 4.5
Texto: cuarto
Numero: 8.0
Texto: quinto
Error al leer numero real: 34R.5
Texto: sexto
Numero: 1.0
```

3. Entrada/Salida de texto con formato

La clase `PrintWriter` dispone de una operación de salida de texto con formato, llamada `printf`

- el objeto `System.out` que representa la pantalla, también
- está copiada del lenguaje C
- el primer parámetro es el string de formato
- luego viene un número variable de parámetros

Ejemplo

```
System.out.printf
    ("%s de %3d años", nombre, edad);
```

Produce la salida (suponiendo nombre="Pedro", edad=18)

```
Pedro de  18 años
```


String de formato

Contiene caracteres que se muestran tal cual

- y especificaciones de formato que se sustituyen por los sucesivos parámetros

Especificaciones de formato más habituales:

`%d`	enteros
`%c`	caracteres
`%s`	string
`%f`	*float* y *double*, coma fija
`%e`	*float* y *double*, notación exponencial
`%g`	*float* y *double*, exponencial o coma fija
`%n`	salto de línea en el formato del sist. operat.
`%%`	el carácter %

String de formato

Puede lanzarse `IllegalFormatException` si el formato no corresponde al parámetro

Después del carácter `%` se puede poner un carácter de opciones:

- `-` alinear a la izquierda
- `0` rellenar con ceros (números sólo)
- `+` poner signo siempre (números sólo)

Especificación de anchura y precisión

Puede añadirse después del "%" (y el carácter de opción si lo hay) la especificación de anchura mínima y/o número de decimales; ejemplos

Parámetros de printf()	Salida
`("Pi= %4.0f %n",Math.PI)`	`Pi=  3`
`("Pi= %4.2f %n",Math.PI)`	`Pi= 3.14`
`("Pi= %12.4f %n",Math.PI)`	`Pi=  3.1416`
`("Pi= %12.8f %n",Math.PI)`	`Pi=  3.14159265`
`("I= %8d %n",18`	`I=  18`
`("I= %4d %n",18)`	`I=  18`
`("I= %04d %n",18)`	`I= 0018`

4. Entrada/salida binaria

Es posible escribir y leer variables y objetos en un fichero binario

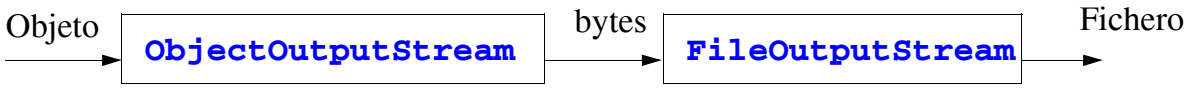
Para poder escribir un objeto, su clase debe

- implementar la interfaz **`Serializable`**
- y definir en la clase un número que sirve para identificar la versión de la clase que se está utilizando
  - hay que cambiar el número si se modifica la clase, para evitar errores

```
import java.io.*;
public class Nombre implements Serializable {
  private static final long serialVersionUID = 10;
    ...
}
```


Salida de objetos y variables

Se usan las clases enlazadas **`FileOutputStream`** y **`ObjectOutputStream`**



Operaciones más habituales: FileOutputStream

Descripción	Declaración
Constructor. Requiere el nombre del fichero. Lo crea si no existe	`FileOutputStream(String s)`
Sincronizar	`void flush()`
Cerrar	`void close()`

Operaciones más habituales: FileOutputStream

Descripción	Declaración
Constructor. Requiere un OutputStream	`ObjectOutputStream (OutputStream out)`
Escribir un booleano	`void writeBoolean(boolean b)`
Escribir un double	`void writeDouble(double d)`
Escribir un int	`void writeInt(int i)`
Escribir un objeto (o un string)	`void writeObject(Object obj)`
Sincronizar	`void flush()`
Cerrar	`void close()`

Al escribir un objeto se escriben también los objetos a los que éste se refiere (y así recursivamente)

Ejemplo:

Añadir una operación de escribir a la clase `Curso` del ejemplo desarrollado en capítulos anteriores

- Previamente ha sido necesario declarar las clases que se van a escribir como `Serializable`

```
import java.io.*;
public class Alumno implements Serializable {
     private static final long serialVersionUID = 10;
   ...
}
import java.io.*;
public class Curso implements Serializable {
   private static final long serialVersionUID = 10;
   ...
}
```

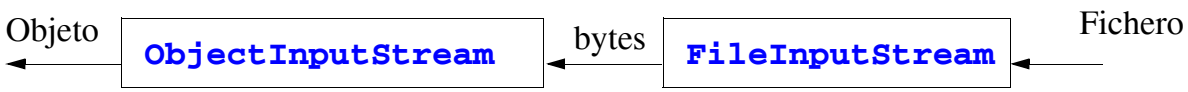

Ejemplo: método a añadir a la clase `Curso`

```
public void escribe(String nombreFichero)
  throws IOException
{
    FileOutputStream fich =
       new FileOutputStream(nombreFichero);
    ObjectOutputStream sal =
       new ObjectOutputStream(fich);
    try {
      sal.writeObject(this); // el objeto actual
    } finally {
      sal.close();
      fich.close();
    } // try
} // escribe
```

Entrada de objetos y variables

Podemos leer variables y objetos de un fichero binario creado con las clases anteriores

Se usan las clases enlazadas `FileInputStream` y `ObjectInputStream`



Operaciones más habituales: FileInputStream

Descripción	Declaración
Constructor. Requiere el nombre del fichero. Si no existe lanza **FileNotFoundException**	`FileInputStream(String s) throws FileNotFoundException`
Cerrar	`void close()`

Operaciones más habituales: ObjectInputStream

Descripción	Declaración
Constructor. Requiere un **InputStream**	`ObjectInputStream(InputStream in)`
Leer un booleano	`boolean readBoolean()`
Leer un double	`double readDouble()`
Leer un int	`int readInt()`
Leer un objeto (y un string)	`Object readObject()`
Cerrar	`void close()`

Al leer un objeto se leen también los objetos a los que éste se refiere y que fueron escritos en el fichero (y así recursivamente)

Lectura de objetos

Al leer un dato de una clase es preciso hacer una conversión de tipo:

```
(NombreClase) objetoStream.readObject()
```

***Ejemplo*: añadir una operación de leer a la clase `Curso` del ejemplo desarrollado en capítulos anteriores:**

- **Previamente ha sido necesario declarar las clases que se van a escribir como `Serializable`**
- **Posteriormente, hay que modificar el programa principal para que**
  - **llame a `lee`() al principio**
  - **llame a `escribe`() al finalizar**

Ejemplo

```
public static Curso lee (String nombreFichero,
    int maxAlumnos)
    throws ClassNotFoundException, IOException
{
    try {
        Curso miCurso;
        FileInputStream fich =
            new FileInputStream(nombreFichero);
        ObjectInputStream ent =
            new ObjectInputStream(fich);
        try {
            miCurso = (Curso) ent.readObject();
        } finally {
          ent.close();
          fich.close();
        } // try
```


Ejemplo

```
        return miCurso;
    } catch (FileNotFoundException e) {
        return new Curso(maxAlumnos);
    } // try
} // lee
```

Ejemplo

```
public class ListaAlumnos {
  public static void main(String[] args) {
    try {
       // leer el fichero
       Curso curso=Curso.lee("primero.dat",100);
       ...
          case SALIR:
          // escribir el fichero
          curso.escribe("primero.dat");
       ...
    } catch (ClassNotFoundException e) {
       ...
    } catch (IOException e) {
       ...
    }
```


UNIVERSIDAD DE CANTABRIA