

Unidad 6

Entrada/Salida en Java

Programación
1º D.A.M.

1

Contenido

1. Excepciones
2. Entrada/Salida estándar
3. Manejo de ficheros



2

1. Excepciones

3

1. Excepciones

- Mecanismo de control y gestión de errores de ejecución
- Posibilidades de gestión
 - Gestión en el propio método
 - try - catch
 - Paso de la excepción a otro método
 - throws

4

1. Excepciones

```
try{
    // Bloque de código peligroso
}catch(TipoExcepcion1 objEx1){
    // Tratamiento excepciones TipoExcepcion1
}catch(TipoExcepcion2 objEx2){
    // Tratamiento excepciones TipoExcepcion2
}finally{
    // Sentencias comunes (haya o no excepción)
    // Se ejecutan SIEMPRE
    // Usos típicos: cierre de ficheros, conexiones a
    // BD's, etc
}
```

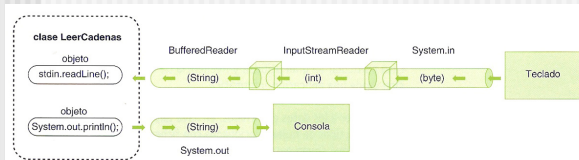
5

2. E/S estándar

1. Entrada estándar
2. Salida estándar

6

2. E/S estándar



7

2.1. Entrada estándar

- **InputStream**
 - Clase abstracta para leer de secuencias de bytes
 - Método `read()`
 - `System.in` lo es, asociado al teclado
- **InputStreamReader**
 - Convierte secuencias de bytes a secuencias de caracteres
 - Requieren un `InputStream`
- **BufferedReader**
 - Memoria temporal para almacenar datos leídos
 - Método `readLine()` para lectura
 - Genera `IOException` (requiere `try-catch` o `throws`)
 - Posterior conversión de `String` al tipo correspondiente

8

2.1. Entrada estándar

- **Teclado**
- **stdin**
- **Lectura de un carácter**

```
char c = System.in.read();
```
- **Lectura de una línea**

```
try{
    BufferedReader teclado = new BufferedReader(new
        InputStreamReader(System.in));
    String cadena = teclado.readLine();
} catch (IOException e){
    System.out.println(e.getMessage());
}
```

9

2.1. Entrada estándar

■ La clase Scanner

- Requiere `import java.util.Scanner;`

```
Scanner teclado      = new Scanner(System.in);

String cadena        = teclado.nextLine();
int entero            = teclado.nextInt();
float real            = teclado.nextFloat();
long largo            = teclado.nextLong();
double real2          = teclado.nextDouble();
boolean booleano     = teclado.nextBoolean();

teclado.close();
```

10

2.2. Salida estándar

■ OutputStream

- Clase abstracta para escribir secuencias de bytes
 - Método `write()`
 - `System.out` lo es, asociado a la pantalla

■ OutputStreamWriter

- Convierte secuencias de bytes a secuencias de caracteres
- Requieren un `OutputStream`

■ BufferedWriter

- Memoria temporal en la que escribir
- Método `write()` para escritura

11

2.2. Salida estándar

■ Salida estándar

- Pantalla
- `stdout`
- Escritura en salida estándar
 - `System.out.print`
 - `System.out.println`

■ Salida estándar de error

- Pantalla
- `stderr`
- Escritura en salida estándar de error
 - `System.err.print`
 - `System.err.println`

12

3. Manejo de ficheros

1. Gestión de ficheros
2. L/E de ficheros
 1. L/E de texto
 1. Escritura de texto
 2. Lectura de texto
 2. L/E de datos
 1. Escritura de datos
 2. Lectura de datos
 3. L/E de objetos
 1. Escritura de objetos
 2. Lectura de objetos
 4. L/E aleatoria
3. Redirección de la E/S estándar

13

3.1. Gestión de ficheros

- Clase `File`, dentro de `java.io`
- Operaciones con el sistema de archivos
 - Creación de archivos
 - Listado de archivos
 - Renombrado de archivos
 - Eliminación
 - Creación de directorios
 - Consulta de directorios
 - Eliminación de directorios
 - ...
- Dificultad por trabajar Java con distintos sistemas

14

3.1. Gestión de ficheros

- File
 - Construcción de objetos
 - `File archivo = new File("/dir1/fich.txt");`
 - Archivo a partir de su ruta absoluta
 - `File directorio = new File("nombre");`
 - Directorio a partir de una ruta relativa
 - `File archivo2 = new File(directorio, "fich2.txt");`
 - Archivo a partir de su ruta y luego su nombre

15

3.1. Gestión de ficheros

■ File

■ Problema con las rutas

• Descripción

- Windows: / ó \
- Sistemas Unix: /

• Soluciones recomendadas

- `FileDialog` de Swing para selección
- Usar variable estática `separatorChar`

```
String ruta = "dir1/dir2/fich.txt";
ruta=ruta.replace('/',File.separatorChar);
```

16

3.1. Gestión de ficheros

■ File

■ Métodos genéricos

- `String toString()`
- `boolean exists ()`
- `boolean canRead()`
- `boolean canWrite()`
- `boolean isHidden()`
- `boolean isAbsolute ()`
- `boolean equals(File f)`
- `String getAbsolutePath()`
- `File getAbsoluteFile()`
- `String getName()`
- `String getParent()`
- `File getParentFile()`
- `boolean setReadOnly()`

17

3.1. Gestión de ficheros

■ File

■ Métodos de directorios

- `boolean isDirectory()`
- `boolean mkdir()`
- `boolean mkdirs()`
- `boolean delete()`
- `String[] list()`
- `static File[] listRoots()`
- `File[] listFiles()`

18

3.1. Gestión de ficheros

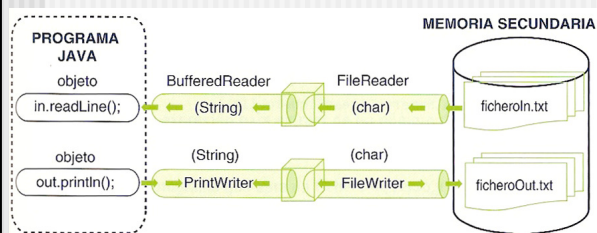
■ File

■ Métodos de archivos

- boolean `isFile()`
- boolean `renameTo(File f)`
- boolean `delete()`
- long `length()`
- boolean `createNewFile()`
- static File `createTempFile(String prefijo, String sufijo)`
- static File `createTempFile(String prefijo, String sufijo, String directorio)`
- void `deleteOnExit()`

19

3.2.1. L/E de ficheros de texto



20

3.2.1.1. Escritura de texto

■ Escritura de texto en un fichero

```
PrintWriter fSalida = new
    PrintWriter(new FileWriter(fichero));
fSalida.println("Texto a escribir");
fSalida.close();
```

- Para añadir al final de un fichero existente
 - ... new FileWriter (fichero, true)
- Véanse otros métodos interesantes, como
 - append
 - flush

21

3.2.1.2. Lectura de texto

■ Lectura de las líneas de un fichero

```

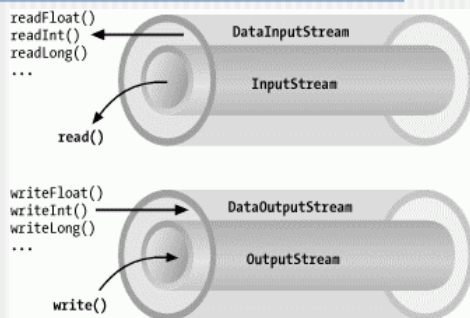
■ BufferedReader fEntrada =
  new BufferedReader(
    new FileReader(fichero));

■ String lin = "";
■ while((lin=fEntrada.readLine())!=null)
  • System.out.println("Leido: " + lin);

```

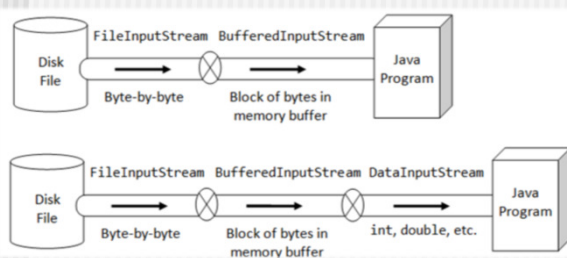
22

3.2.2. L/E de datos



23

3.2.2. L/E de datos



24

3.2.2.1. Escritura de datos

- Proceso
 - Crear objeto `FileOutputStream`
 - A partir de `File` con ruta al archivo a escribir
 - A partir simplemente de la ruta
 - `FileNotFoundException`
 - Posibilidad de añadir booleano al constructor
 - `true` para añadir datos (append)
 - Crear objeto `DataOutputStream`
 - A partir del `FileOutputStream` anterior
 - Usar objeto anterior para escribir
 - Método `writeTipo` (`writeInt`, `writeDouble`, ...)
 - `IOException`
 - Cerrar el archivo
 - Método `close` del objeto `DataOutputStream`

25

3.2.2.1. Escritura de datos

- Ejemplo


```
File f = new File("C:/prueba.dat");
try{
    FileOutputStream fos = new FileOutputStream(f);
    DataOutputStream dos = new DataOutputStream(fos);
    for(int i = 0; i < 10; i++){
        dos.writeInt(i);
    }
    dos.close();
}catch(IOException e){
    System.err.println("Error en la escritura");
}
```

26

3.2.2.2. Lectura de datos

- Proceso
 - Crear objeto `FileInputStream`
 - A partir de `File` con ruta al archivo a escribir
 - A partir simplemente de la ruta
 - `FileNotFoundException`
 - Crear objeto `DataInputStream`
 - A partir del `FileInputStream` anterior
 - Usar objeto anterior para escribir
 - Método `readTipo` (`readInt`, `readDouble`, ...)
 - `IOException`
 - `EOFException`: subtipo generado al llegar a final de fichero
 - Cerrar el archivo
 - Método `close` del objeto `DataInputStream`

27

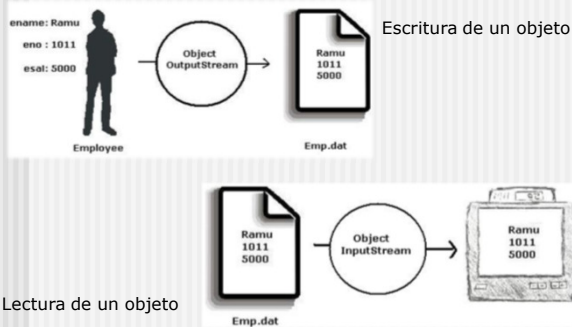
3.2.2.2. Lectura de datos

■ Ejemplo

```
File f = new File("C:/prueba.dat");
boolean finFichero = false;
try{
    FileInputStream fis = new FileInputStream(f);
    DataInputStream dis = new DataInputStream(fis);
    while(!finFichero){
        try{
            int valor = dis.readInt();
            System.out.print(valor);
        }catch (EOFException e){
            finFichero = true;
        }
        dis.close();
    }
} catch (FileNotFoundException e){
    System.err.println("El fichero no existe");
} catch (IOException e){
    System.err.println("Error en la escritura");
}
```

28

3.2.3. L/E de objetos



29

3.2.3.1. Escritura de objetos

■ Proceso

- Crear objeto **FileOutputStream**
 - A partir de **File** con ruta al archivo a escribir
 - A partir simplemente de la ruta
 - **FileNotFoundException**
 - Posibilidad de añadir booleano al constructor
 - **true** para añadir datos (append)
- Crear objeto **ObjectOutputStream**
 - A partir del **FileOutputStream** anterior
- Usar objeto anterior para escribir
 - Método **writeObject**
 - **IOException**
- Cerrar el archivo
 - Método **close** del objeto **ObjectOutputStream**

30

3.2.3.1. Escritura de objetos

■ Ejemplo

```
File f = new File("C:/coches.dat");
try{
    FileOutputStream fos = new FileOutputStream(f);
    ObjectOutputStream oos=new ObjectOutputStream(fos);
    for(int i = 0; i < 10; i++){
        Coche c = new Coche();
        oos.writeObject(c);
    }
    oos.close();
}catch(IOException e){
    System.err.println("Error en la escritura");
}
```

31

3.2.3.2. Lectura de objetos

■ Proceso

- Crear objeto FileInputStream
 - A partir de File con ruta al archivo a escribir
 - A partir simplemente de la ruta
 - FileNotFoundException
- Crear objeto ObjectInputStream
 - A partir del FileInputStream anterior
- Usar objeto anterior para escribir
 - Método readObject
 - IOException
 - EOFException: subtipo generado al llegar a final de fichero
- Cerrar el archivo
 - Método close del objeto ObjectInputStream

32

3.2.3.2. Lectura de objetos

■ Ejemplo

```
File f = new File("C:/coches.dat");
boolean finFichero = false;
try{
    FileInputStream fis = new FileInputStream(f);
    ObjectInputStream ois = new ObjectInputStream(fis);
    while(!finFichero){
        try{
            Coche c = (Coche)ois.readObject();
            System.out.print(c);
        }catch(EOFException e){
            finFichero = true;
        }
    }
    ois.close();
}catch(FileNotFoundException e){
    System.err.println("El fichero no existe");
}catch(IOException e){
    System.err.println("Error en la escritura");
}
```

33

3.2.4. L/E aleatoria

- Acceso a ficheros
 - Secuencial
 - Acceso a una posición pasando por las anteriores
 - Directo o aleatorio
 - Acceso directo a una posición
 - No requiere pasar por posiciones anteriores

34

3.2.4. L/E aleatoria

- RandomAccessFile
 - Para l/e con acceso aleatorio
 - Construcción del objeto
 - Objeto `File`, o nombre del fichero
 - Cadena de permisos
 - "r", "w", "rw"

```
File f = new File("C:/prueba.dat");
RandomAccessFile raf = new RandomAccessFile(f, "rw");
```

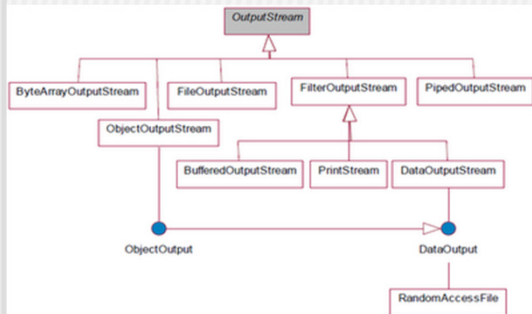
35

3.2.4. L/E aleatoria

- RandomAccessFile
 - Métodos importantes
 - `void seek(long pos)`
 - `long getFilePointer()`
 - `long length()`
 - Métodos de lectura
 - `readBoolean, readByte, readChar, readInt, readDouble, readFloat, readUTF, readLine`
 - Métodos de escritura
 - `writeBoolean, writeByte, writeChar, writeChars, writeInt, writeDouble, writeFloat, writeUTF, writeLine`

36

3.2.4. L/E aleatoria



37

3.3. Redirección de S estándar

■ Redirección de salida estándar

■ Salida estándar = pantalla

- System.out

■ Redirección a un archivo

```

FileOutputStream fos = new FileOutputStream(fichero);
PrintStream ps = new PrintStream(fos);
System.setOut(ps);
  
```

38

3.3. Redirección de S estándar

■ Redirección de salida estándar de error

■ Salida estándar de error = pantalla

- System.err

■ Redirección a un archivo

```

FileOutputStream fos = new FileOutputStream(fichero);
PrintStream ps = new PrintStream(fos);
System.setErr(ps);
  
```

39

Unidad 6

Entrada/Salida en Java

Programación
1º D.A.M.

40
