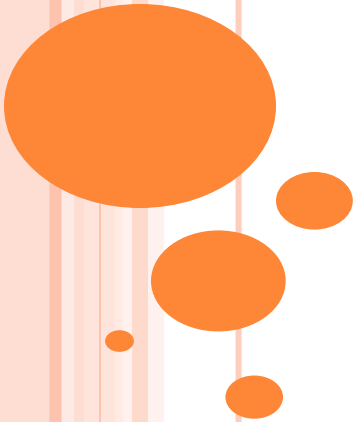
Acceso a ficheros



Fundamentos

- El acceso a ficheros en Java se reduce a una operación de entrada y salida de datos.
- Se utilizan las mismas clases del java.io que se emplean para trasferir datos entre la aplicación y el exterior.
- En lecciones posteriores utilizaremos el paquete java.nio.files con opciones mejoradas para trabajar con ficheros

Clase PrintStream

- Clase por excelencia para envío de datos al exterior
- El atributo estático System.out, contiene un objeto PrintStream que apunta a la pantalla
- Puede crear un objeto PrintStream asociado a otras fuentes:
  - PrintStream(String file). Asociado a un fichero.
  - PrintStream(OutputStream). Asociado a otra fuente.
- Métodos *print y println*

Escritura en un fichero

➢Utilizando PrintStream:

```
String dir="/user/mydata.txt";
try(PrintStream out=new PrintStream(dir)){
    out.println("dato1");
    …
}catch(IOException ex){…}
```

obligatorio capturar la excepción IOException

- Escritura con formato
- Graba los datos en modo sobrescritura
- Si el fichero no existe se crea

➢Utilizando PrintStream y FileOutputStream

Permite realizar la escritura en modo *append*

```
String dir="/user/mydata.txt";
try(FileOutputStream fos=new FileOutputStream(dir, true);
PrintStream out=new PrintStream(fos)){
    out.println("dato1");
    …
}catch(IOException ex){…}
```

Clase BufferedReader

- Clase para lectura de datos desde el exterior
- Dispone de método para lectura de cadenas de caracteres: *readLine()*
- Para crear un BufferedReader necesitamos un objeto Reader intermedio:
  - BufferedReader(Reader reader)
- Posibles implementaciones de Reader:
  - ImputStreamReader para leer del teclado
  - FileReader para lectura desde fichero

Lectura de un fichero

➢Lectura de texto utilizando BufferedReader y FileReader:

```
String dir="/user/mydata.txt";
try(FileReader fr=new FileReader(dir);
BufferedReader br=new BufferedReader(fr)){
    String line;
    while((line=br.readLine())!=null){
        System.out.println(line);
    }
}catch(IOException ex){…}
```

- Lectura de todas las líneas del fichero
- Si el fichero no existe se produce una excepción

Cierre de objetos

- **Los objetos utilizados para escritura y lectura de ficheros se deben cerrar después de utilizarlos:**

cierre clásico en finally:

```
try{
    ps=new PrintStream("c:\\temporal\\datos.txt");
    :
}
catch(IOException ex){
:
}
finally{
    if(ps!=null){
        ps.close();
    }
}
```

Cierre mediante llamada explícita al método **close()**

try con recursos:

```
try(PrintStream  ps=new
PrintStream("c:\\temporal\\datos.txt");)
{

    :
}
catch(IOException ex){
:
}
```

Los objetos creados en try se cierran automáticamente al abandonar el bloque

La clase File

- **Representa una ruta a un fichero o directorio.**

```
File file=new File("/user/mydata.txt");
```

- **Proporciona métodos para obtener información sobre el elemento:**
  - **boolean exists(). Devuelve true si existe**
  - **boolean isFile(). Devuelve true si es un fichero**
  - **boolean isDirectory(). Devuelve true si es un directorio**
  - **boolean delete(). Elimina el elemento. Devuelve true si ha conseguido eliminarlo**