

36019475. CSIFC03. MP0485. Programación

**XUNTA DE GALICIA**CONSELLERÍA DE CULTURA, EDUCACIÓN
E ORDENACIÓN UNIVERSITARIA

Páxina principal ► Os meus cursos ► Formación Profesional a Distancia ► Curso 2016-2017 ►
36019475 IES de Rodeira ► CSIFC03 Desenvolvemento de aplicacións web ►
125_36019475_ZSIFC03_MP0485_A ► Unidade didáctica 6 ► 4.- Traballando con Streams Binarios

NAVEGACIÓN

Páxina principal

• Amiña área persoal

Páxinas do sitio

O meu perfil

Curso actual

125_36019475_ZSIFC03_MP0485_A

Participantes

Distincións

Xeral

Unidade didáctica 1

Unidade didáctica 2

Unidade didáctica 3

Unidade didáctica 4

Unidade didáctica 5

Unidade didáctica 6

Orientaciones para el alumnado. PROG06.

Solución a la tarea para PROG06.

PROG06 Guiada.- Almacenando datos.

Actividades presenciales de la UD6 en la tutoría c...

Tarefa 6 - Soluciona da Titoria Presencial

Foro para PROG06.

Mapa conceptual para PROG06.

PROG06 Completa.- Almacenando datos.

Recursos complementarios UD06.

1.- Introducción a Entrada/Saída en Java

Tarefa a Entregar 1

Solución Tarefa 1

2.- Fluxos Binarios e Fluxos de Caracteres

3.- Acceso a Ficheiros

Tarefa a Entregar 2

Solución

4.- Traballando con Streams Binarios

5 - Traballando con Fluxos de Caracteres

Tarefa a Entregar 3

Solución

6 - Traballando con Ficheiros de Acceso Aleatorio ...








Introducción á comunicación con Sockets

Leeweb.java

Chat

Tarea para PROG06.

Solución Alternativa

 Tarefa a Entregar 4 Solución (Versión sen clase para acceso a ficheiros) Solución (Con clase independente para acceder aos ... Tarefa a entregar 5 Solución Tarefa a Entregar 6 Modificacións Propostas Examen de la UD06. Aplicación Hotel Completa

Unidade didáctica 7

Unidade didáctica 8

Unidade didáctica 9

Unidade didáctica 10

Unidade didáctica 11

Os meus cursos

ADMINISTRACIÓN

Administración do curso

Configuración do meu perfil

4.- Traballando con Streams Binarios

En Java podemos distinguir entre fluxos binarios e fluxos de texto. Para traballar con fluxos binarios se utilizan as clases `InputStream` (lectura de datos) e `OutputStream` (escritura de datos), mentras que cos fluxos de texto se empregan clases "Reader" e "Writer".

Como traballar con fluxos binarios puros é complexo, existen varias clases que permiten traballar con fluxos binarios dun xeito máis "humano". As máis importantes son:

`DataInputStream``DataOutputStream``ObjectInputStream``ObjectOutputStream`

Estes obxectos se poden dotar dunha memoria intermedia (Buffer) que proporcionan unha lectura e escritura máis eficiente, xa que as operacións en lugar de realizarse byte a byte, se realizarán en bloques mediante as clases `BufferedInputStream` e `BufferedOutputStream`. Por exemplo, para crear unha lectura de un fluxo de obxectos con buffer, poderíamos facer:

```
InputStream f=null;
try {
    file=new FileInputStream("Ficheiro.dat");
    InputStream buff=new BufferedInputStream(file);
    ObjectInputStream in=new ObjectInputStream(buff);
} catch (Exception e) {
    System.out.println("Error "+e.getMessage());
    e.printStackTrace();
} finally {
    if (file!=null) file.close();
}
```

Neste exemplo, é importante fixarse especialmente no peche na cláusula `finally`, que se executará aínda que falle a apertura do ficheiro.

Unha clase que proporciona métodos cómodos para a saída pode ser `PrintStream`. Esta clase permite enviar ao fluxo de datos representacións de distintos obxectos. Un exemplo de `PrintStream` é `System.out`, que utilizamos para enviar información formateada á consola. `PrintStream` utiliza un buffer para a saída, que é volcado cando se invoca ao método ***flush***, ou cando se escribe un salto de liña no fluxo.

4.1 - Streams de Obxectos

As clases `ObjectInputStream` e `ObjectOutputStream` sirven para tratar con fluxos de entrada e saída de obxectos "serializados", é decir, convertidos a un fluxo de bytes. Para poder facer isto (convertir un obxecto a un fluxo de bytes, e logo, poder converter de novo o fluxo de bytes ao obxecto orixinal) é necesario que tanto o obxecto como os atributos implementen a interfaz ***Serializable***. A maior parte das clases Java xa o fan.

Os interfaces é un tema que se tratará posteriormente, así que digamos que cando defines a clase teredes que engadir 'implements `Serializable`', deste xeito

```
class CCCCCC implements Serializable {  
}
```

ou si é unha clase herdada

```
class CCCCCC extends BBBBBB implements Serializable {  
}
```

Un problema dos `ObjectStreams` é que almacenan unha cabeceira cando abrimos o ficheiro, de modo que si abrimos o ficheiro para engadir nos gardará esa cabeceira no medio do arquivo, fastidiando logo a lectura correcta. Sen embargo, é posible evitar isto mediante unha clase derivada sobrepoñendo o método `writeStreamHeader` (ver solución oficial da Tarefa6)

Última modificación: Mércores, 18 de Xaneiro do 2017, 12:40

