

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 ► 1.- Introducción a Entrada/Saída en Java

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

1.- Introducción a Entrada/Saída en Java

Con **entrada/saída** nos referimos ao xeito de *enviar información* dende unha aplicación hacia o exterior (saída), ou de *aceptar información* dende o exterior hacia a aplicación (entrada). Esta información pode orixinarse en diversas fontes, por exemplo:

- Teclado
- Ficheiros en Disco
- Conexións de Rede
- Unha tarxeta de captura de vídeo
- Unha Webcam
- Unha tarxeta capturadora de sonido
- etc. etc.

Tamén é posible enviar información hacia diversos destinos como:

- Unha tarxeta de vídeo (á pantalla)
- Un ficheiro en disco
- Unha conexión de rede
- Unha tarxeta de sonido
- etc. etc.

En Java, esto está asociado ao concepto de **fluxo (Stream)**. Un *Stream* é un fluxo de datos dende a aplicación hacia o exterior (**OutputStream**) ou do exterior hacia a aplicación (**InputStream**). O manexo dos Streams se realiza mediante clases do paquete **java.io**, que heredan das clases **InputStream** ou **OutputStream**.

A máquina virtual Java crea no arranque dous obxectos *Stream* en atributos estáticos da clase **System**, que estarán dispoñibles durante a execución da aplicación :

- **System.in** - Obxecto **InputStream**, asociado ao teclado
- **System.out** - Obxecto **PrintStream**, asociado á tarxeta de vídeo (pantalla)
- **System.err** - Obxecto **PrintStream**, asociado á saída de erros estándar, normalmente a pantalla.

Evidentemente, en calqueira aplicación se poden crear novos obxectos **InputStream** e **OutputStream** asociados ao fluxo de datos que se queira.

As clases **InputStream** e **OutputStream** ofrecen unha funcionalidade moi básica, polo que normalmente non se empregan directamente, se non que se utilizan a través de outras clases que nos proporcionan a funcionalidade que precisamos segundo o uso que se lle queira dar ao Stream. Un exemplo é a clase **PrintStream** que nos proporciona

Si observamos a documentación de *PrintStream*, veremos que hereda de *OutputStream*, ademáis de ver na lista de constructores un constructor que permite crear obxectos *PrintStream* a partir de calqueira *OutputStream* que se lle suministre. Por exemplo, a máquina virtual Java, crea un obxecto *PrintStream* a partir dun *OutputStream* asociado á tarxeta de vídeo (pantalla), de xeito que a saída mediante o método ***println*** irá a pantalla.

Sería perfectamente posible crear obxectos *PrintStream* asociados a outros tipos de fluxos de saída (como unha conexión de rede, ou un ficheiro en disco), e enviar información mediante os seus métodos, como ***println***.

Outro exemplo xa coñecido é a clase *Scanner*.

A clase *Scanner* nos proporciona métodos para ler liñas, Integer, String, Float... etc dun fluxo de entrada. Si observades a súa API no enlace anterior, veredes que é posible construír un obxecto *Scanner* pasándolle como parámetro un fluxo de entrada. Si o fluxo de entrada está asociado ao teclado (como *System.in*), o fluxo que se leerá será o que ven do teclado do sistema.

Última modificación: Luns, 19 de Decembro do 2016, 11:46

