| PROYECTO | CLASES-METODOS | OBSERVACIONES |
|---------------------------|---|--|
| GESTIÓN DE FICHEROS | | |
| VerDir | Constructor => File(String directorioyfichero) Método => String[] list() Método => boolean isFile() Método => boolean isDirectory() | Muestra los ficheros (archivos o directorios) de un determinado directorio que se cargan en un array de elementos tipo String usando el método list() |
| Actividad1_1_Parte1 | Constructor => File(String directorioyfichero) Método => File[] listFiles() | Muestra los ficheros (archivos o directorios) de un determinado directorio que se cargan en un array de elementos tipo File usando el método list() |
| Actividad1_1_Parte2 | Constructor => File(String directorioyfichero) Método => String[] list() | El nombre del directorio a listar se pasa como parámetro al método main |
| VerInf | Constructor => File(String directorioyfichero) Método => String getName () Método => String getPath () y otros métodos que informan de atributos del fichero | Muestra varios atributos de un fichero (nombre, ruta, si se puede leer, etc.) Al especificar la ruta en Windows, al poner el carácter \ hay que añadir el carácter de escape \ por lo que tendré \\. Ejemplo E: \\Mis_documentos _Curso 2021-22\\fichero.txt |
| CrearDir | Constructor => File(String directorioyfichero) Constructor => File(File directorio, String fichero) Método => boolean mkdir() Método => boolean createNewFile() Método => renameTo(File nuevonombre) | Crea un directorio en el directorio actual (el del proyecto) y en ese directorio crea ficheros. Después renombra uno de los ficheros. Debido a que el método createNewFile puede lanzar IOException, se debe incluir en bloque try-catch. |
| RenombrarFichero | Constructor => File(String directorioyfichero) Método => renameTo(File nuevonombre) | Renombrar fichero |
| GESTIÓN DE FLUJOS DE DATO | S DESDE/HACIA FICHEROS DE TEXTO | |
| LeerFicTexto | Constructor declara fichero => File(String directorioyfichero) Constructor flujo de entrada => FileReader(File fichero) Método lee carácter a carácter como un entero => int read() Método cerrar fichero => int close() | Lee uno a uno los caracteres de un fichero de texto y los muestra en pantalla. El método read() devuelve -1 cuando no hay carácter. Debido a que el método read() puede lanzar IOException, se debe incluir en bloque try-catch, pero en este caso, como alternativa al try-catch, en este ejemplo añade throws IOException al método main() |
| Actividad1_2 | Constructor declara fichero => File(String directorioyfichero) Constructor flujo de entrada => FileReader(File fichero) Método lee carácter a carácter como un entero => int read() Método cerrar fichero => int close() | Muestra en la consola el contenido de un fichero de texto que se le pasa al programa como un parámetro. |
| EscribirFicTexto | Constructor declara fichero => File(String directorioyfichero) Constructor flujo de salida borrando => FileWriter(File fichero) Constructor flujo de salida para adjuntar => FileWriter(File fichero, boolean append) Método escribe carácter (pe. "\n") => write(char c) Método escribe array de caracteres => write(char []) Método adjuntar carácter => append(char c) Método cerrar fichero => int close() | Escribe caracteres en un fichero de varias formas. Si no existe el fichero, al crear el flujo de salida se crea el fichero. 1. los caracteres se escriben uno a uno a partir de un String que se convierte en array de caracteres. 2. se escribe un array de caracteres 3. se escribe un String 4. escribe un array de Strings |
| LeerFichTextoBuf | Constructor flujo de entrada ==> BufferedReader(FileReader(File fichero) Método lee línea => String readLine() Método cerrar fichero => int close() | Lee líneas completas de texto de un fichero determinado ubicado en el mismo directorio que el proyecto y las muestra en pantalla. El método readLine() devuelve null cuando no hay línea La lectura se incluiye en bloque try-catch. |
| EscribirFichTextoBuf | Constructor flujo de salida ==> BufferedWriter(FileReader(File fichero) Método escribe línea => Write(String str) Método escribe salto de línea ==> newLine() Método cerrar fichero => int close() | Escribe líneas completas de texto y añade el salto de línea |
| GESTIÓN DE FLUJOS DE DATO | S DESDE/HACIA FICHEROS BINARIOS (ocupan menos espacio en disco, pero no son | legibles por editores de texto) |
| EscribirFichBytes | Constructor flujo salida => FileOutputStream(File file) => FileOutputStream(File file, boolean append) Constructor flujo entrada => FileInputStream(File file) Método escribe un byte => write(int b) Método lee un byte => read(int b) Método cerrar stream => int close() | Escribe bytes en un fichero (si no existe lo crea el constructor) y después los visualiza. Al elegir el constructor del flujo de salida se puede optar por adjuntar datos al fichero ya existente (append = true) El método read() devuelve -1 cuando no hay dato a leer. |

| Learning Chipter Construction flug control > Pick Team state and Table Learning Chipter Learning Chi |
|--|
| Mettodo cerera stream > int close() Sertificial Stream Out Data Out to ut Stream Out Mettodo cerera stream > int close() Sertificial Stream Out Mettodo cerera stream > int close() Sertificial Stream Out Se |
| Constructor fuju called > Data Outs Stream (Outs out Stream out) Metodo excitle dato (según topo de tato) - with 000000000000000000000000000000000000 |
| Data Output/Stream QuitoutStream Output Stream Output |
| Data Output/Stream (Quitout Stream Output Stream Output Metodo cerem stream >int close() Septimbly Metodo cerem stream >int close() Metodo cerem strea |
| Método cerar betano (según tipo de atar) - yurasxxxxxisodata dala |
| Método cerra stream >> int close() Método ce |
| LeerfichData Constructor flujo entrada >= Pasas novest trasas (anoustitrosas sin) Método cerrar stream >> int close) |
| de datos. La lectura de datos debe haceres selando el topo y medicio de datos la lectura de datos debe haceres estando el topo y orden de lo datos su que estan escritos en el fichero. Para comtodar el fin de la lectura de datos utiliza un budad white demonstrativo de lo destos su budad white demonstrativo de lo destos utiliza un budad white demonstrativo de lo destos su demonstrativo de lo destos su demonstrativo de lo destos su demonstrativo de lo destos demonstrativo de lo destos su demonstrativo de lo destos demonstrativo de lo destos su de lo destos demonstrativo de lo destos su de lo de lo des |
| Método cerar stream ⇒ int close() GESTION DE FLUICO DE ONIES DESEMBACIA RICHEROS BINARIOS (guardando todos los atributos del objeto. El objeto debe implementar la interfaz Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble Constructor delas fichiero → File String directorio/fichero) Constructor filipo sintiar > File (interfa Senal abble constructor Senal abble constructor filipo sintiar > File (interfa Senal abble constructor filipo sintiar > File (interfa Senal abble constructor sena |
| Método cerar stream ⇒ int close() GESTION DE FLUICO DE ONIES DESEMBACIA RICHEROS BINARIOS (guardando todos los atributos del objeto. El objeto debe implementar la interfaz Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble concuerte el objeto en una securiono del bita) (interfa Senal abble Constructor delas fichiero → File String directorio/fichero) Constructor filipo sintiar > File (interfa Senal abble constructor Senal abble constructor filipo sintiar > File (interfa Senal abble constructor filipo sintiar > File (interfa Senal abble constructor sena |
| GESTION DE FLUICS DE COSTITUCIS DESCRIACIOS (ENCREOS BINANCIS) (guardando todos los stributos del objeto. El objeto del implamentar la interfaz Servillativa (convierte el objeto en una secumica de bits.) Interfaz Servillativa (convierte el objeto en una secumica de bits.) Interfaz Servillativa (convierte el objeto en una secumica de bits.) Interfaz Servillativa (convierte el objeto en una secumica de bits.) Interfaz Servillativa (convierte el objeto en una secumica de bits.) Interfaz Servillativa (convierte el objeto en una secumica de bits.) Interfaz Servillativa (convierte el objeto en una secumica que implamenta la interfaz Servillativa (convierte el objeto en una secumica que implamenta la interfaz Servillativa (convierte el objeto en la del objeto Peccas que implamenta la interfaz Servillativa (convierte el objeto el ob |
| CESTION DE FLUICS DE CONTROCE PESOF/MACIA FICHEROS RINARIOS (guardando todos los arributos del objeto. El objeto debe implementar la interfaz Serializable conviente el objeto en una secuencia de bista; linterfaz Serializable (Constructor declara fichera se Pile(String directorioyfichera) |
| Extribition |
| Constructor declara fichero > File[String directorioyfichero) |
| Constructor declara fichero > FielString directorioyfichero) Constructor declara fichero = FielString directorioyfichero) Constructor declara fichero |
| Constructor declara fichero => FietString directorioyfichero) |
| Constructor fluju salida > pialoutosuttireaan (Elar, bollean append) |
| Constructor fluju salida > pialoutosuttireaan (Elar, bollean append) |
| SecribirFichAleatorio Constructor delara fichero >> FlieString directorioyfichero Constructor efficience >> FlieString directorioyfichero Constructor delara fichero >> FlieString |
| ExcribirFichObject (permite adjuntar nuevos objeto so surtacto pate the provided paragraphy of the pro |
| ExcribirFichObject (permite adjuntar nuevos objetos af fichero) Método escribro objeto >> mritechianci (thianch obj) Método escribro objeto >> mritechianci (thianch objeto) Directino objeto o |
| Metodo escribo objeto > vritadiotant (dotant obj) Metodo cerrar stream ⇒ int close() Metodo cerrar stream ⇒ int close() Secribification Metodo cerrar stream ⇒ int close() |
| Método cerra stream => int close() LeerFichObject Constructor declara fichero >> File(String directorioyfichero) Constructor flujo entrada => ObiectInputStream in objectOutputStream que crea cabecera, pero si el fichero accidente de la clase Método lee objeto > readObject() Método cerrar stream => int close() Constructor flujo entrada => Object () Método cerrar stream => int close() Constructor declara fichero >> File(String directorioyfichero) Constructor declara fichero sacceso aleatorio => RandomAccoessFile (Tile file) Bitring model mode "tw" Método se de scritura de datos según su tipo => "vritachosite (double v) Vritachosite (double v) Método para aber la posición del puntero » pasek (long pos) Método para posicionarse en un registro >> pasek (long pos) Método se lectura de datos según su tipo => RandomAccoessFile (File file) Bitring model mode "tw" Método sa saber el tamaño del fichero » Energiti (lite) Método para saber la posición del puntero » pasek (long pos) Método sa saber el tamaño del fichero » pasek (long pos) Método sa saber la posición del puntero » pasek (long pos) Método deplasa puntero unos bytes » santolytes (lite) Método deplasa puntero unos bytes » |
| Método cerrar stream => int close() Service Método cerrar stream M |
| Para que no se escriba la cabecra es necesario sobreescrite centeda a cabe cera en un objet coloquetistream que en mi caso lo he hech oreando la clase MiObjectOutputStream que ne mi caso lo he hech oreando la clase MiObjectOutputStream que ne mi caso lo he hech oreando la clase MiObjectOutputStream que ne me caso de la fichero ne vestas er un objet coloqueta cera cabecera, pero si el fichero que vestas er un objet constructor fluje entrada => Constructor declara fichero > File(String directorioyfichero) Método cerrar stream => int close() Sestión De FUJIOS DESOF/HACIA FICHEROS BINARIOS DE ACCESO ALFATORIO EscribirfichAleatorio Constructor crea fichero acceso aleatorio => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => Constructor declara fichero => File(String directorioyfichero) Vericachara (String => Vericachara (String directorioyfichero) Constructor rea fichero acceso aleatorio => Randomacoesa File (File file, String mode) modo "r" Método para posicionarse en un registro => acces (Long yos) Método para posicionarse en un registro => acces (Long yos) Método desplaza puntero unos bytes => accesta (Long yos) Método desplaza puntero unos bytes => accesta (Long yos) Método para posicionarse en un registro => accesta (Long yos) Método para posicionarse en un registro => accesta (Long yos) Método para posicionarse en un registro => accesta (Long yos) Método para posicionarse |
| Que en micaso to be hecho creamdo la clase MiloipictotuptiStream En el programa, Si el fichero no existe se crea un objetto Constructor declara fichero >> File(String directorioyfichero) Lee ilos objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos. Al leer los objetos desde un archivo de datos con el método read/Diject hace un cast a la clase del Objeto, p.e. (Persona) GESTIÓN DE FILIJOS DE DATOS DESDE/HACIA FICHEROS BINARIOS DE ACCESO ALEATORIO EscribirfichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => |
| MilobjectOutputStream En el programa, Si el fichero no existe se crea un objett ObjectOutputStream que crea cabecera, pero si el fichero y estits que crea cabecera, pero si el fichero y estits que crea cabecera, pero si el fichero y estits que crea cabecera, pero si el fichero y estits que un activo de datos. Al leer los objetos estes que la cabe y milobjectOutputStream que no escribe cabecera. Lee los objetos desde un archivo de datos. Al leer los datos con el método readobject hace un cast a la clase del Objeto, pe. (Persona) Método cerrar stream > int close() |
| MilobjectOutputStream En el programa, Si el fichero no existe se crea un objett ObjectOutputStream que crea cabecera, pero si el fichero y estits que crea cabecera, pero si el fichero y estits que crea cabecera, pero si el fichero y estits que crea cabecera, pero si el fichero y estits que un activo de datos. Al leer los objetos estes que la cabe y milobjectOutputStream que no escribe cabecera. Lee los objetos desde un archivo de datos. Al leer los datos con el método readobject hace un cast a la clase del Objeto, pe. (Persona) Método cerrar stream > int close() |
| LeerFichObject Constructor declara fichero >> File(String directorioyfichero) Constructor flujo entrada >> ObjectInputsStream que no escribe cabecera, emobjectObjectOputsStream que no escribe cabecera. |
| Constructor declara fichero => File(String directorioyfichero) Constructor declara fichero => File(String directorioyfichero) Constructor flujo entrada => |
| EerFichObject Constructor declara fichero => File(String directorioyfichero) Constructor flujo entrada => Constructor declara fichero => File(String directorioyfichero) VariateChara (String =) VariateChara (VariateChara (String =) VariateChara (VariateChara (String =) VariateChara (VariateChara (String =) VariateChara (VariateChara (Va |
| LeerFichObject Constructor declara fichero => File(String directorioyfichero) Lee los objetos desde un archivo de datos. Al leer los datos con el método readObject hace un cast a la clase del Objeto, pe. (Persona) |
| LeerFichObject Constructor declara fichero => File(String directorioyfichero) Constructor fluje entrada => ObjectInputStream (InputStream in) Método lee objeto => readObject () Método cerrar stream => int close() Método cerrar stream => int close() Método cerrar stream => int close() GESTION DE FLUIOS DE DATOS DESDE/HACIA FICHEROS BINARIOS DE ACCESO ALEATORIO |
| Persona Constructor flujo entrada => Object InputStream (InputStream in) Método lee objeto >> readObject () Método cerra stream => int close() GESTIÓN DE FLUIOS DE DATOS DESDE/HACIA FICHEROS BINARIOS DE ACCESO ALEATORIO EscribirFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método se escritura de datos según su tipo => writeInt(int v) writeDouble (double v) Método se escritura de datos según su tipo => writeInt(anc v) writeDouble (double v) Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para saber la posición del puntero => mestilePointer () Método para saber la posición del puntero => mestilePointer () Método desplaza puntero unos bytes => akingbytes (int n) Método desplaza puntero unos bytes >> akingbytes (int n) Método para posicionarse en un registro => mest (long pos) EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => meating (n) posicionarse en un registro => meating (n) posicionarse en un fichero aleatorio del que constructor crea fichero acceso aleatorio => meating (n) posicionarse en un fichero aleatorio del que constructor describen del puntero => meating (n) posicionarse en un fichero aleatorio del que constructor declara fichero => file(String directorioyfichero) Constructor crea fichero acceso aleatorio => meating (n) posicionarse en un fichero aleatorio del que constructor declara fichero => file(String directorioyfichero) Constructor crea fichero acceso aleatorio => meating (n) posicionarse en un registro => meating (n) posicionarse en un registro => meating (n) posicionarse en un fichero aleatorio del que constructor declara fichero => file(String directorioyfichero) Constructor crea fichero acceso aleatorio => meating (n) posicionarse en un fichero aleatorio del que conscruttura (tamaño del registro). Calcula la posición y |
| Método lee Objeto = yeadObjeto () Método lee Objeto = yeadObjeto () |
| Método lee objeto => readObject () Método cerrar stream => int close() EscribirFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor rea fichera osceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Métodos de escritura de datos según su tipo => writeChara (String s) writeDouble (double v) Métodos de escritura de datos según su tipo => writeChara (String s) |
| Método cerrar stream => int close() GESTIÓN DE FLUJOS DE DATOS DESDE/HACIA FICHEROS BINARIOS DE ACCESO ALEATORIO EscribirFichAleatorio Constructor crea fichero acceso aleatorio ⇒ RandomAccessFile (File file, String mode) modo "rw" Métodos de escritura de datos según su tipo ⇒ writeChara (String s) writeChara (String s) writeChara (String s) writeChara (String s) writeDouble (double v) Constructor declara fichero => File(String directorioyfichero) RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro ⇒ seek (long pos) Método para saber la posición del puntero ⇒ getFilePointer () Método desplaza puntero unos bytes ⇒ skipBvtes (int n) Métodos de lectura de datos según su tipo ⇒ readint (), readouble (), readobuble (|
| Abre el fichero en modo "rw" e inserta registros con datos de empleado (total 36 bytes): |
| Abre el fichero en modo "rw" e inserta registros con datos de empleado (total 36 bytes): |
| Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" |
| Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" |
| Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Métodos de escritura de datos según su tipo => writeInt (int v) writeChang (String s) writeDouble (double v) Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => acek (long pos) Método desplaza puntero unos bytes => akipBytes (int n) Método desplaza puntero unos bytes => akipBytes (int n) Método desplaza puntero unos bytes => akipBytes (int n) Método para posicionarse en un registro => seek (long pos) RandomAccessFile (File file, String mode) modo "r" EscribirFichAleatorioUnReg Constructor declara fichero => length () Método desplaza puntero unos bytes => akipBytes (int n) Métodos de lectura de datos según su tipo => RandomAccessFile (File file, String mode) modo "r" Constructor declara fichero => length () Método desplaza puntero unos bytes => akipBytes (int n) Métodos de lectura de datos según su tipo => RandomAccessFile (File file, String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro >> seek (long pos) Método para posicionarse en un registro >> seek (long pos) |
| RandomAccessFile (File file, String mode) modo "rw" Apellido (10 caracteres UNICODE, 2 bytes x carácter) Apellido (10 caracteres UNICODE, 2 bytes x carácter) Apellido (10 caracteres UNICODE, 2 bytes x carácter) Nº Departamento (entero = 4 bytes) Salario (double = 8 bytes) Salario (double = 8 bytes) Usa datos se escriben desde un array con elementos de cada uno de los tipos y el identificador es el indice del array + 1. En el caso del apellido para que todos ocupen 10 Constructor crea fichero acceso aleatorio => Realiza la lectura de los registros de un fichero aleatorio. Para la lectura de los debemos controlar la posición del puntero del fichero (método getfile/Fointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método getfile/Fointer), sabiendo el tamaño del fichero se realiza con el método seek(posición). EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor declara fichero => File(S |
| Métodos de escritura de datos según su tipo => writeInt(int v) writeChars (String s) writeDouble (double v) LeerFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => Realiza la lectura de los registros de un fichero aleatorio. Para la lectura de los datos debemos controlar la posición del puntero el metro del fichero (método getfilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor declara fichero => File(String directorioyfich |
| Métodos de escritura de datos según su tipo => writeInt(int v) writeOnare (String s) writeDouble (double v) LeerFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método desplaza puntero unos bytes => stipBytes (int n) Método selectura de datos según su tipo => readint (), readOuble (), readChar () EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| LeerFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => Sacek (long pos) Método selectura de datos según su tipo => readInt (), readObuble (), readObuble (), readChar () Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para saber la posición del guntero => Sacek (long pos) Método desplaza puntero unos bytes => Saciobutes (int n) Método selectura de datos según su tipo => readInt (), readObuble (), readChar () Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => Sacek (long pos) Método selectura de datos según su tipo => Saciobutes (int n) Método desplaza puntero unos bytes => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos según su tipo => Saciobutes (int n) Método selectura de datos debemos controlar la posición y se posiciona con el método seek(posición) Saciobutes (int n) Sac |
| Los datos se escriben desde un array con elementos de cada uno de los tipos y el identificador es el indice del array + 1. En el caso del apellido para que todos ocupen 10 caracteres utiliza un buffer. LeerFichAleatorio |
| LeerFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para saber al posición del puntero => getFilePointer () Método desplaza puntero unos bytes => akipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readConbr () EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para saber al posición del puntero => getFilePointer () Método desplaza puntero unos bytes => akipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readConbr () EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| cada uno de los tipos y el identificador es el indice del array + 1. En el caso del apellido para que todos ocupen 10 caracteres utiliza un buffer. Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de los registros de un fichero aleatorio. Para la lectura de los datos debemos controlar la posición del puntero dentro del fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| array + 1. En el caso del apellido para que todos ocupen 10 Caracteres utiliza un buffer. |
| En el caso del apellido para que todos ocupen 10 caracteres utiliza un buffer. Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readOpuble (), readChar () EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Cons |
| LeerFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método para saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de los registros de un fichero aleatorio. Para la lectura de los datos debemos controlar la posición del puntero dentro del fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño de registro y controlando el tamaño del fichero se realiza con el método seek(posición). EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| LeerFichAleatorio Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de los registros de un fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlar la posición del puntero dentro del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) aleatorio. Para la lectura de los datos debemos controlar la posición del puntero dentro del fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método seek(posición). Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) aleatorio. Para la lectura de los datos debemos controlar la posición del puntero dentro del fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método seek(posición). Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) aleatorio. Para la lectura de los datos debemos controlar la posición del puntero dentro del fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método seek(posición). Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| RandomAccessFile (File file, String mode) modo "r" Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer() Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Controlar la posición del puntero dentro del fichero (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero se realiza con el método seek(posición). (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). (método getFilePointer), sabiendo el tamaño de cada registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). |
| Método para posicionarse en un registro => seek (long pos) Método para saber la posición del puntero => getFilePointer () Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) registro y controlando el tamaño del fichero (método length). El desplazamiento dentro del fichero se realiza con el método seek(posición). Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se po |
| Método para saber la posición del puntero => getFilePointer() Método pasa saber el tamaño del fichero => length() Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt(), readDouble(), readChar() EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| Método para saber la posición del puntero => getFilePointer () Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBvtes (int n) Métodos de lectura de datos según su tipo => readInt (), readDouble (), readChar () EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => conoce su estructura (tamaño del registro). Calcula la RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| Método pasa saber el tamaño del fichero => length () Método desplaza puntero unos bytes => skipBvtes (int n) Métodos de lectura de datos según su tipo => readInt () , readDouble () , readChar () EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| Método desplaza puntero unos bytes => skipBytes (int n) Métodos de lectura de datos según su tipo => readInt(), readDouble(), readChar() EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| Métodos de lectura de datos según su tipo => readInt(), readDouble(), readChar() EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Método para posicionarse en un registro => seek (long pos) |
| readInt(), readDouble(), readChar() EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posición y y |
| EscribirFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) Inserta un registro en un fichero aleatorio del que conoce su estructura (tamaño del registro). Calcula la posición y se posiciona con el método seek(posición) |
| Constructor crea fichero acceso aleatorio => RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) conoce su estructura (tamaño del registro). Calcula la posición y se posición y s |
| RandomAccessFile (File file, String mode) modo "rw" Método para posicionarse en un registro => seek (long pos) posición y se posiciona con el método seek(posición) |
| Método para posicionarse en un registro => seek (long pos) |
| |
| |
| Métodos de escritura de datos según su tipo => |
| |
| <pre>writeInt(int v), writeChars(String s), writeDouble(double v)</pre> |
| LeerFichAleatorioUnReg Constructor declara fichero => File(String directorioyfichero) Lee un registro de un fichero aleatorio. El número del |
| |
| Constructor crea fichero acceso aleatorio => registro se pasa como parámetro al programa. |
| RandomAccessFile (File file String mode) mode "rw" |
| RandomAccessFile (File file, String mode) mode "rw" Método para posicionarse en un registro => seek (long pos) |

| | Métodos de escritura de datos según su tipo => writeInt(int v), writeChars(String s), writeDouble(double v) | |
|-------------------------------------|--|--|
| Actividad1_3_Consulta | | |
| Actividad1 3 Insercion | | |
| Actividad1 4 Modificacion | | |
| Actividad1_4_Borrado | | |
| | => https://www.w3schools.com/xml/xml_whatis.asp | |
| Procesadores XML (Parser) => | https://www.tutorialspoint.com/java_xml/java_xml_parsers.htm E8 => https://docs.oracle.com/javase/8/docs/ | |
| Documentación (API DOM) pa | equete org.w3c.dom => https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/ | package-tree.html |
| Documentación (API SAX) paq | uete org.xml.sax => http://www.saxproject.org/apidoc/org/xml/sax/package-tree | <u>.html</u> |
| Documentación (API XStream |) XStream Core 1.4.18 => http://x-stream.github.io/javadoc/index.html | |
| CrearEmpleadoXml | Construye el procesador o parser DOM => <u>DocumentBuilderFactory</u> () (try -Catch controla ParserConfigurationException) Crea una clase para crear el documento XML => <u>DocumentBuilder</u> () | Crea un fichero XML a partir de los datos de un fichero aleatorio existente, utilizando el parser DOM Recorre los registros del fichero aleatorio donde están |
| | Crea un interfaz para usar el documento DOM => <u>DOMImplementation</u> Para trabajar con el documento XML (en memoria) emplea los métodos del interfaz <u>DOMImplementation</u> | los datos y crea en memoria la estructura de árbol del documento XML (raíz, nodos,) Finalmente convierte el documento XML en un fichero. |
| | Convierte el documento en un fichero => https://docs.oracle.com/javase/tutorial/jaxp/xslt/writingDom.h tml Para realizar la conversión Crea una clase Transformer a partir de TransformerFactory | Comentario: Los constructores DocumentBuilder y TransformerFactory están protegidos. Para usarlos hay que crear antes una instancia de la clase => https://stackoverflow.com/questions/1057221/what-ar-e-practical-uses-of-a-protected-constructor/1057245 |
| LecturaEmpleadoXml | Construye el procesador o parser DOM => DocumentBuilderFactory () | Realiza la lectura de un fichero XML, utilizando el parser DOM. El fichero se procesa con el parser DOM y una vez que está en memoria se emplean los métodos para leer etiquetas, valores y demás elementos del archivo. |
| Actividad1_5 FichPersona.dat | | Lee los datos de un fichero .dat que contiene objetos Persona y a partir de él crea un fichero XML utilizando el parser DOM. |
| UD1_PruebaSax1 alumnos.xml | Construye el procesador o parser SAX => XMLReaderFactory() (XMLReaderFactory está deprecated y se recomienda usar SAXParserFactory) Crea procesador para el fichero => createXMLReader() Crea un gestor de contenido personalizado. Personaliza el procesador SAX al sobrescribir los métodos de DefaultHandler() para tratar los eventos que produce la lectura del fichero (XML) Método asigna gestor al procesador => setContentHandler(ContentHandler handler) Declara el archivo a procesar => InputSource() Método ejecuta el procesador => parse(InputSource input) | Realiza la lectura de un fichero XML, utilizando un parser SAX, en concreto XMLReaderFactory, pero que está deprecated. Los métodos para tratar el archivo se definen sobrescribiendo la clase DefaultHandler https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html |
| UD1_PruebaSax2 alumnos.xml | Idem a UD1_PruebaSax1, pero usando SAXParserFactory en lugar de XMLReaderFactory que está deprecated. | Realiza la lectura de un fichero XML, utilizando un parser SAX, en concreto SAXParserFactory |
| Actividad1_6 Empleados.xml | | Realiza la lectura de un fichero XML, utilizando un parser SAX, en concreto SAXParserFactory |
| EscribirPersonas FichPersona.dat | Añadir archivos JAR al Build path: xtream-1.4.18.jar, kxml2-min-2.3.0.jar Crea el fichero XML, asignando valores a las etiquetas y los datos, empleando la clase XStream Método asigna nombre a la clase del objeto => alias Método asigna nombre a los nombres de un campo => aliasField Método para omitir el nodo raíz => addImplicitCollection Método que genera el fichero XML a partir de la lista de objetos => toXML | Crea una lista de objetos (ListaPersonas) del tipo (clase) Persona, que obtiene leyendo desde un fichero .dat que incluye los datos. Después convierte la lista de objetos (ListaPersonas) en un fichero XML, es decir, serializa objetos Java a XML. Lo hace empleando la librería XStream => https://x-stream.github.io/download.html |
| EscribirPersonas2 | Idem a EscribirPersonas, pero capturando excepciones con bloques try-catch | |

FichPersona.dat

| LagarDagasagas | A Sadir and in a IAD at Duild noth wheeler 1 4 40 in any 2 min 1 4 4 a in | T Dealine le leutoure de confishe de VAAL |
|--------------------------|--|--|
| LeerPersonas | Añadir archivos JAR al Build path: xtream-1.4.18.jar, xpp3_min-1.1.4c.jar | Realiza la lectura de un fichero XML |
| Personas.xml | y xmlpull-1.1.3.1.jar | Deserializa objetos Java a partir de un fichero XML, |
| | Crea el fichero XML, asignando valores a las etiquetas y los datos, empleando | empleando la librería XStream => |
| | la clase XStream | https://x-stream.github.io/download.html |
| | Método asigna nombre a la clase del objeto => alias | 1 |
| | Método asigna nombre a los nombres de un campo => aliasField | Al código del ejemplo del libro, después de crear el |
| | Método para omitir el nodo raíz => addImplicitCollection | objeto XStream con la siguiente línea: |
| | Método que obtiene la lista de objetos a partir del fichero XML => fromXML | XStream xstream = new XStream(); |
| | | hay que añadirle las dos líneas de código siguientes: |
| | | <pre>xstream.allowTypes(new Class[] {ListaPersonas.class}); xstream.allowTypes(new Class[] {Persona.class});</pre> |
| | | para que se puedan crear las clases. |
| C- wattalan | Construction to a construction Construction of Transformation and Property of the Construction of the Cons | |
| Convertidor | Para realizar la conversión Crea una clase Transformer a partir de TransformerFactory | Genera un fichero HTML a partir de un fichero XML y |
| | Para indicar la hoja de estilos usa el método => | otro XSL (https://www.w3.org/Style/XSL/) |
| | newTransformer (Source source) | Ficheros XSL (expresa hojas de estilo en lenguaje XML) |
| | Para realizar la transformación usa el método => | <u>'</u> |
| | transform (Source ficheroXML, Source ficheroHTML) | 1 |
| | | |
| | DE EXCEPCIONES => <u>Fiemplos-tipos de excepciones</u> | 1 |
| UD1_EjemploExcepcion | Clase de la que se crea un objeto al producirse una excepción => Exception | Ejemplo de programa que produce una excepción (al |
| | | dividir por 0 - by zero) y detiene el programa, indicando |
| | | el error en la consola. |
| UD1_EjemploExcepciones | Para capturar la excepción desde el programa se emplea el bloque Try- Catch. | Ejemplo de programa que captura varios tipos de |
| | Es posible anidar bloques Try-Catch | excepción. |
| | La clase Exception si se usa debe ponerse al final de todos los manejadores | 1 |
| | de eventos | |
| UD1_EjemploExcepciones2 | Los métodos de la clase Throwable permiten mostrar información de la | Ejemplo que utiliza varios métodos de la clase |
| | excepción producida => <u>Throwable</u> | Throwable para mostrar información sobre la excepción |
| | | que se produzca |
| TRABAJO CON FICHEROS XML | con JAXB API JAXB => https://javaee.github.io/jaxb-v2/ | - |
| UD1_LecturaJAXB | Se trabaja con el paquete <u>java.xml.bind</u> | Realiza la lectura de un documento XML y lo convierte |
| | Añadir archivos JAR al Build path <mark>: istack-commons-runtime.jar,</mark> | en objetos (deserializa) de una clase definida |
| | javax.activation-api.jar, jaxb-api.jar y jaxb-runtime.jar | previamente por el programador, empleando JAXB |
| | Clase que gobierna la deserialización => <u>Unmarshaller</u> | |
| UD1_EscrituraJAXB | Se trabaja con el paquete <u>iava.xml.bind</u> | Realiza la escritura de un documento XML a partir de |
| | Añadir archivos JAR al Build path: istack-commons-runtime.jar, | los valores de las instancias de una clase de objetos |
| | javax.activation-api.jar, jaxb-api.jar y jaxb-runtime.jar | definidos previamente por el programador |
| | Clase que gobierna la serialización => Marshaller | |
| Actividad1_7 | | Lectura de un fichero de datos que contiene objetos |
| Actividad1_8 | | Creación de fichero XML a partir de fichero de Objetos |
| departamentos.dat | | usando DOM |
| Actividad1_9 | | Creación de fichero XML a partir de fichero de Objetos |
| departamentos.dat | | usando XStream |
| Actividad1_10 | | Conversión de fichero XML a HTML |
| departamentos.xml | | |
| Actividad1_11 | | Lectura de fichero XML empleando JAXB |
| cine.xml | | |
| Actividad1_12 | | Escritura de fichero XML empleando JAXB |
| cine.xml | | |