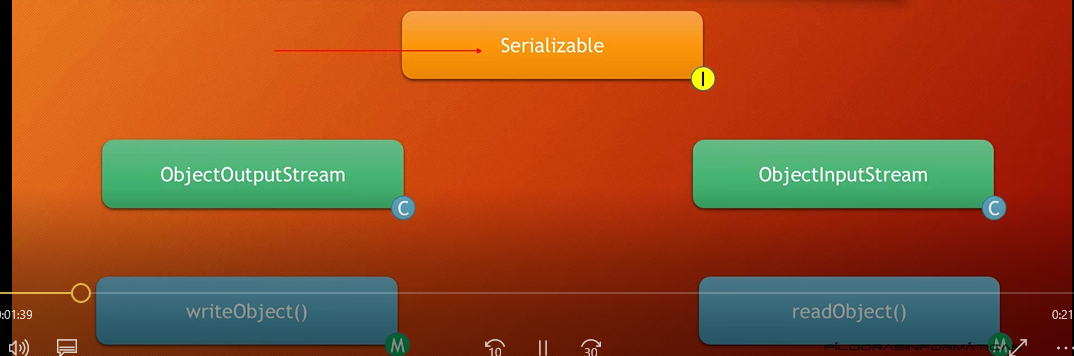
**VIDEO 157**,

SERIALIZACIÓN.

Es convertir un objeto que hayamos creado en java en una sucesión de bytes, con el objetivo de poder almacenar ese objeto en el discoDuro, para en un futuro poder restarurarlo. Y sobre todo poder mover ese archivo através de la red para pasarlo a otros ordenadores y en esos otros ordenadores poder pasarlos a su estado normal, como estaba antes de ser serializado.

Para ello hemos de implementar la interface ‘Serializable’ al objeto que vallamos a ‘Serializar’.



Con la interface ‘Serializable

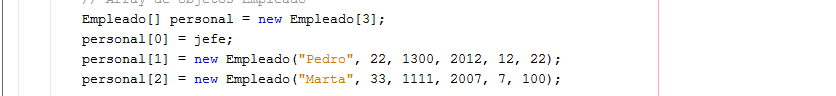
’ le decimos a java que la cl que la implemente es suceptible de ser serializable.

‘OjectOutputStream’ 🡪 es una cl que construye un flujo de datos que permite construir un objeto desde dentro hacia afuera, desde nuestro programa hacia un medio de almacenamiento o hacia un ordenador remoto.

‘OjectInputStream’ 🡪 es una cl que construye un flujo de datos que permite construir un objeto desde fuera hacia fuera.

**EJ con el flujo de datos de la cl** ‘OjectOutputStream’:

Tenemos un Array de objetos ‘Empleado’ y queremos ‘Serializarlo’.



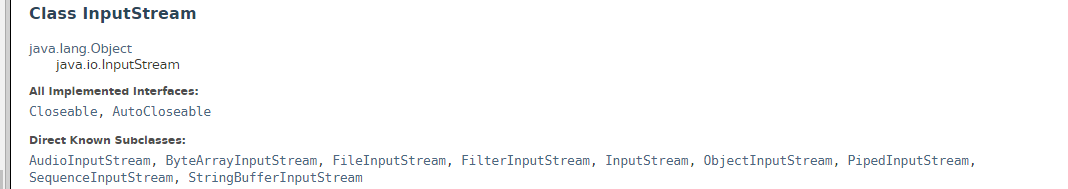
**1º** **Codificamos el flujo de datos, para indicar el lugar donde queremos ubicar los datos**.



**Dato**.

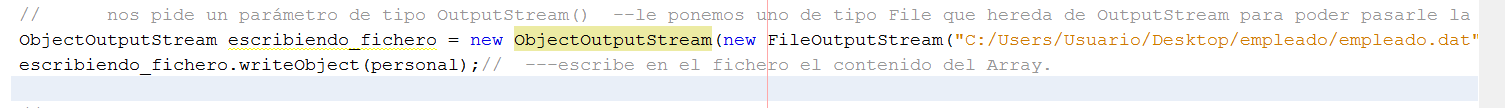
En la api de java vemos que el construtor de ‘[**ObjectInputStream**](file:///C:\Users\Usuario\Desktop\CURSO_JAVA\00%20Todas_Las_Clases_De_Java\api\java\io\ObjectInputStream.html#ObjectInputStream-java.io.InputStream-)([**InputStream**](file:///C:\Users\Usuario\Desktop\CURSO_JAVA\00%20Todas_Las_Clases_De_Java\api\java\io\InputStream.html) in’ pide un objeto de tipo [**InputStream**](file:///C:\Users\Usuario\Desktop\CURSO_JAVA\00%20Todas_Las_Clases_De_Java\api\java\io\InputStream.html) in’ .

**NOS VAMOS A ESTA CL, Y VEMOS LO SIGUIENTE**.

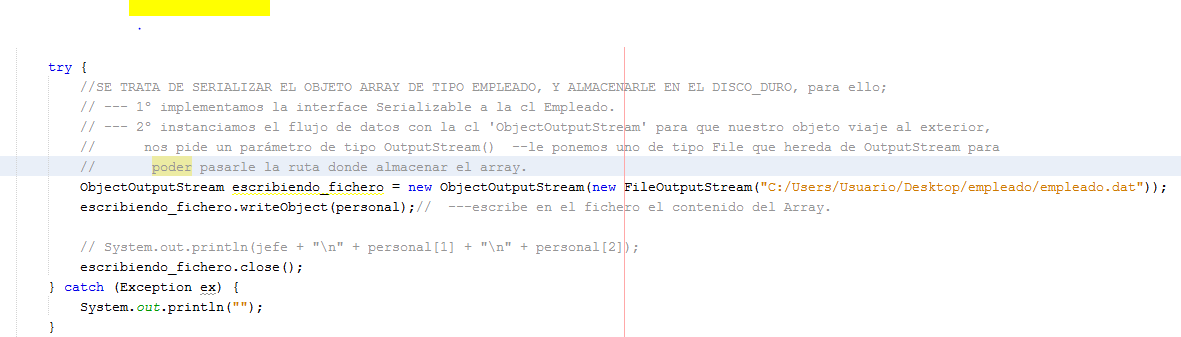


LO QUE NOS INDICA ES QUE PODEMOS PASARLE UN OBJETO DE LAS SUBCLASES COMO PARÁMETRO, Y TOMAMOS LA SUBCLASE ‘FileInputStream’.

**2º** **aplicamoss el mt writeObject() para alamacenar el objeto en el lugar indicado.**



**Todo ello dentro de un try catch**.



**EJ con el flujo de datos de la cl** ‘OjectInputStream’

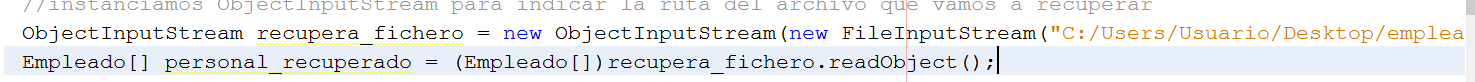
**1º** **Codificamos el flujo de datos, para indicar el lugar DE donde queremos OBTENER los datos**.



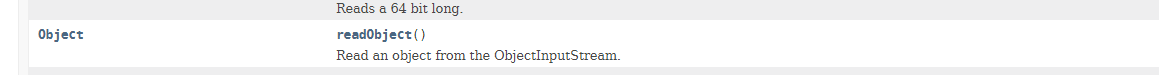
. **Dato**.

En la api de java vemos que EN el construtor DE ‘ObjectInputStream’ ocurre lo mismo que en el costructor de ‘ObjectOutputStream’ ‘’’’???’??

**2º** **Leemos el archivo con el mt ‘readObject()’ y lo almacenamos en una variable del mismo tipo.**



En la api vemos que el mt ‘readObject()’ .

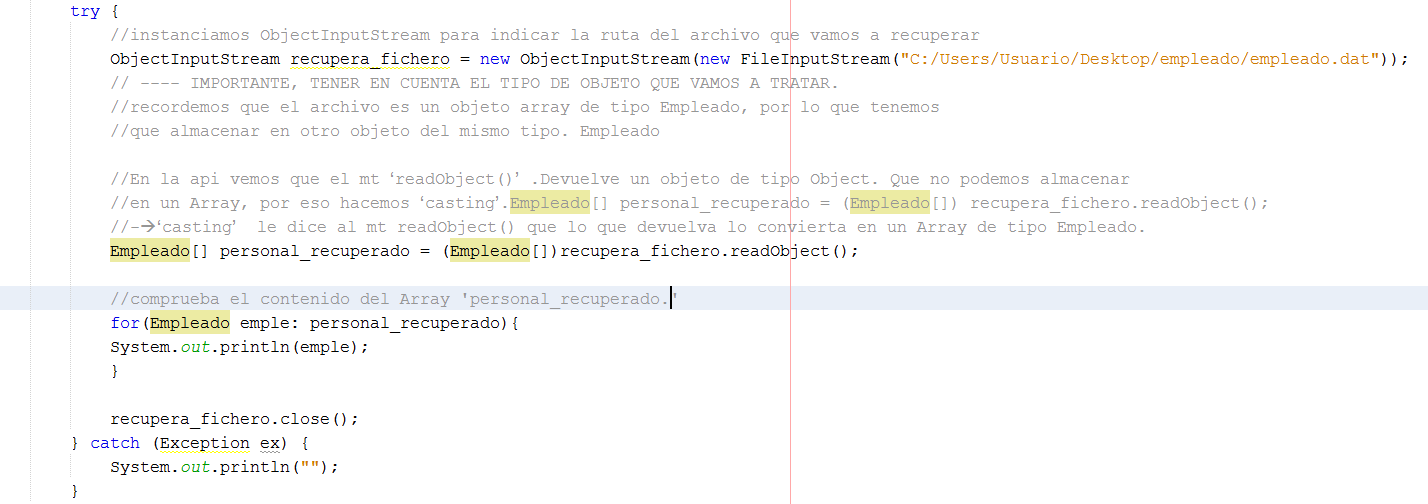


Devuelve un objeto de tipo Object. Que no podemos almacenar en un Array, por eso hacemos ‘**casting**’.

Empleado[] personal\_recuperado = **(Empleado[]) recupera**\_fichero.readObject();

-🡪‘**casting**’ le dice al mt readObject() que lo que devuelva lo convierta en un Array de tipo Empleado.

**Todo ello dentro de un try catch**.



**VIDEO 158**,

Todos los programas de java tiene su huella, su dni. Tiene el nombre de ‘SHA’ consta de 20 bytes.

Para enviar un programa serializado por la red, el emisor y el receptor tienen que tener la misma huella,

Esta huella la crea automáticamente en función del código que tenga el programa, Si el emisor cambia el codigo

La huella del proyeto cambia y el receptor no puede recibir las actualizaciones.

Para evitar esto utilizamos la constante ‘serialVersionUID’, para crear nuestra provia huella y no dejar que java lo haga automáticamente. Podemos darle el valor que queramos, un 1 un 2.

De esta forma la huella no cambia a pesar del cambio de codigo en el programa.

Emisor y receptor tienen que tener la misma constante con el mismo valor.

