

Variable

1. Block of memory to store data ✓
2. 1 value ✓
3. temporary storage.

Array ✓

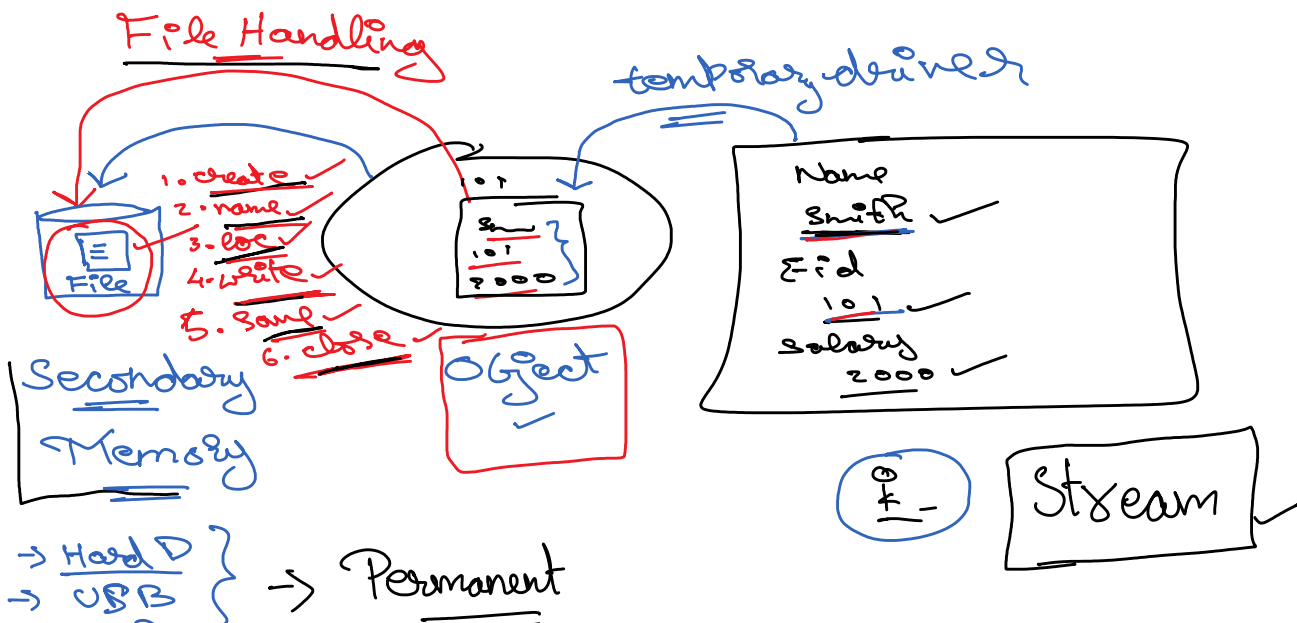
→ temporary

→ Collections { list, set, queue }

file

→ Map → HM, HT, TM }

data, multiple values, temporary



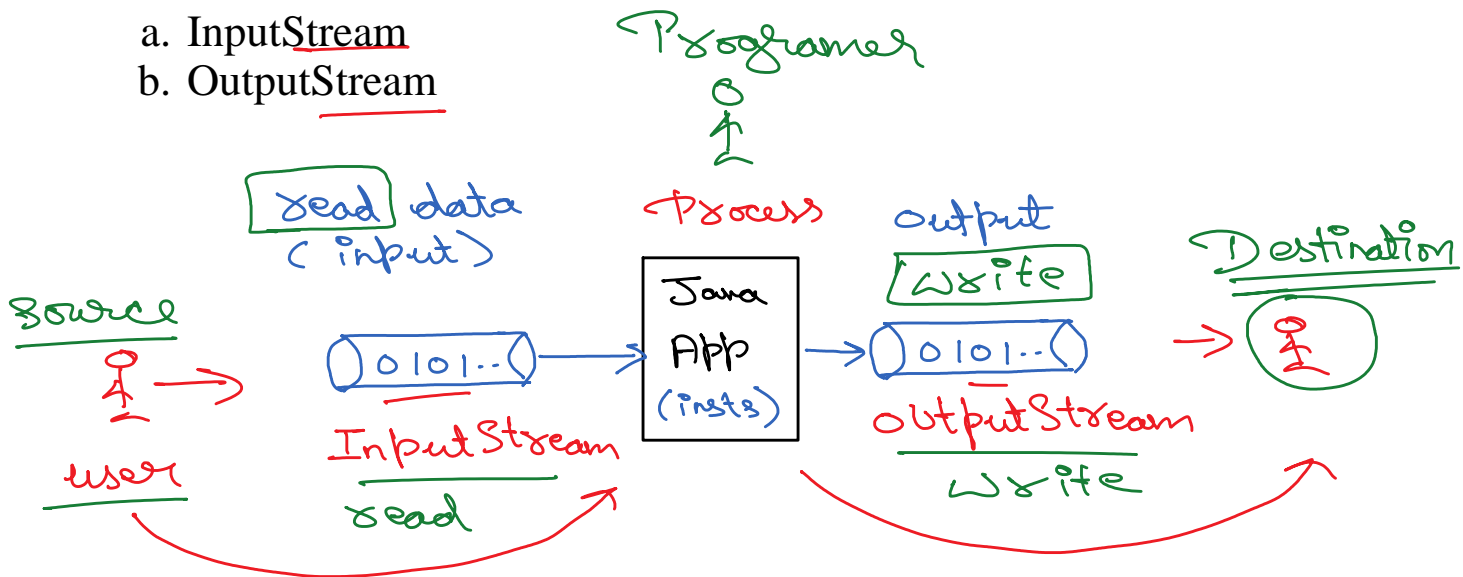
→ Hard D
→ USB
→ CD } → Permanent

Stream :

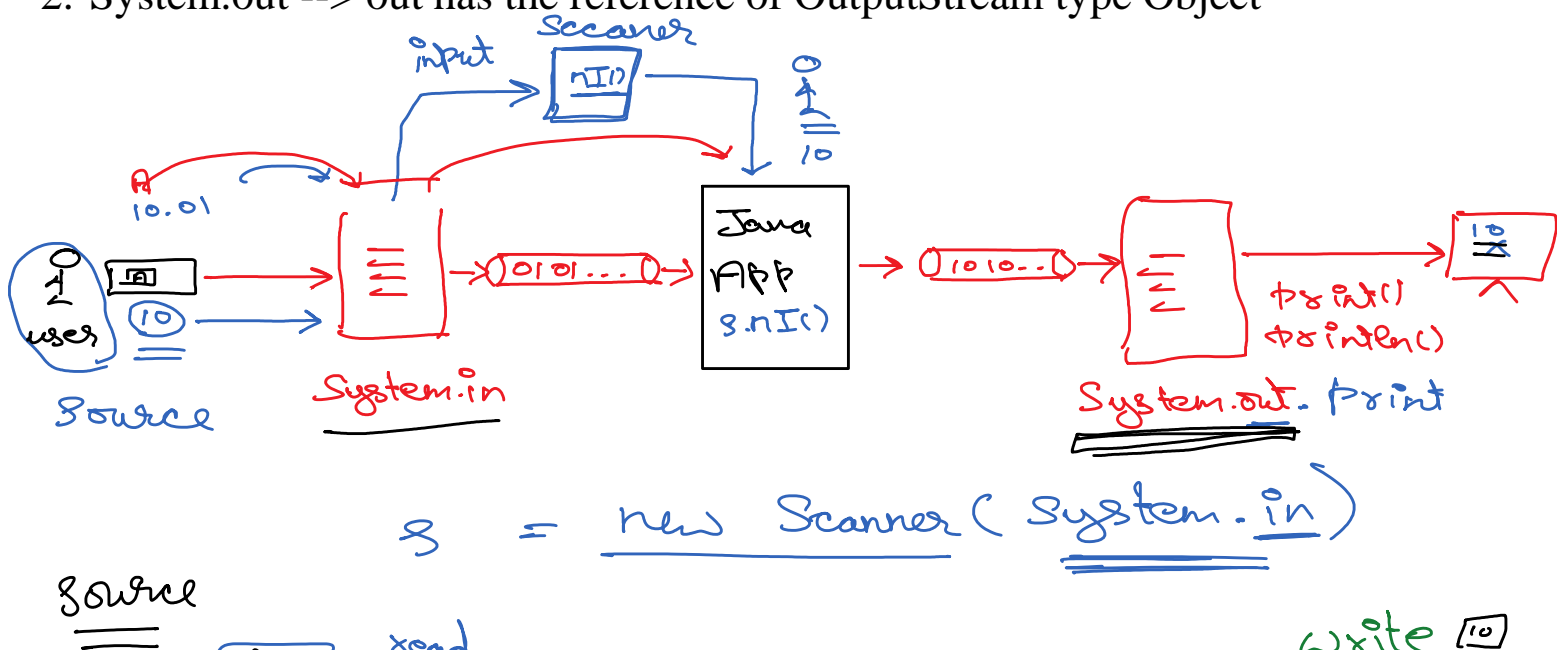
Continuous flow of data in the form of Bytes.

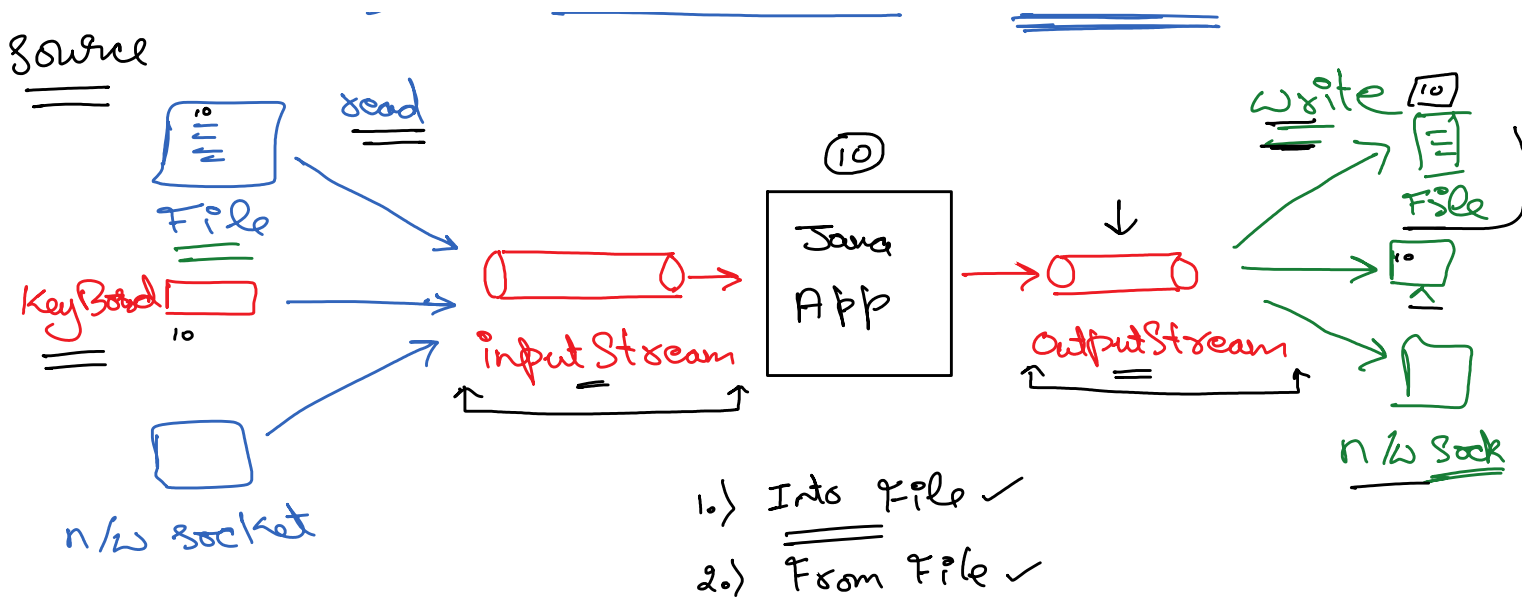
1. Stream is a built-in class Defined in java.io package.
2. Stream class has Two Subclasses :

- a. InputStream
- b. OutputStream



1. System.in ---> in has the reference of InputStream type Object
2. System.out --> out has the reference of OutputStream type Object

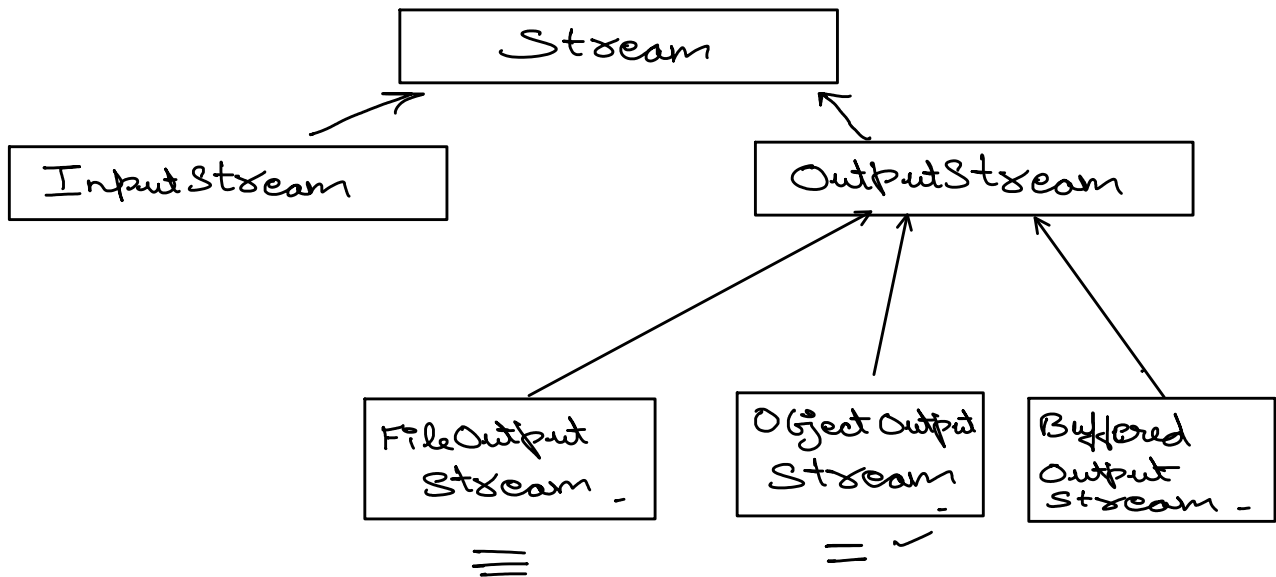




OutputStream :

It helps the programmer to write the stream of data to a specific destination.

Output stream is a subclass of Stream class.



Example:

To write a data to destination as a file we can make use of FileOutputStream.

FileOutputStream :

It is a sub-class of OutputStream, it is used to write data into a file.

Steps to be followed :

Step1 : Create an object of FileOutputStream.

Step2 : Initialize the Object with the location and name of the file.

Note :

1. if the file is not present, FileOutputStream will try to create a new file with the given name in the given location.
2. If the location(path) is incorrect or if operating system of the system does not give permissions to create a file, **in such cases we get** **FileNotFoundException.**

Constructors Of FileOutputStream :

1. FileOutputStream(String path/file_name) throws FileNotFoundException

Step3 : To write data into a file.

We can write data into a file with the help of methods of FileOutputStream.

Methods Of FileOutputStream :

<u>void write(int a) throws IOException</u>	it is used to write a character into a file
<u>void write(byte[] a) throws IOException</u>	it is used to write a multiple characters into a file.

Notes :

1. The method which calls write() must either handle or declare the exception.

Step4 : To close the file.

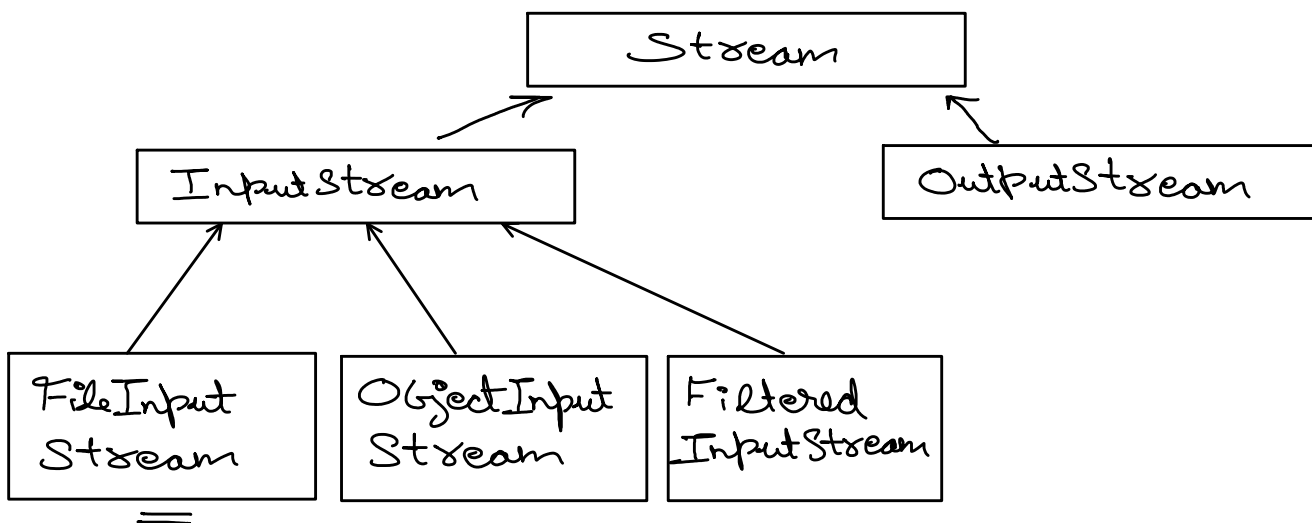
We can close the file with the help of method close()

void close() throws IOException	It is used to close the file
---------------------------------	------------------------------

InputStream :

It helps the programmer to read the data from a specific source, example from a keyboard, file etc ,...

InputStream is a Subclass of Stream class.



FileInputStream :

It is used to read data from a file.

Steps to be Followed :

Step1 : Create an object of FileInputStream.

Step2 : Initialize the object with the path and name of the file.

We perform Step1 & step2 using a constructor.

Constructor of FileInputStream :

FileInputStream(String path/name) throws FileNotFoundException

Note :

1. If the file location is incorrect, or if The OS doesn't give permission to read, then we get FileNotFoundException.

Step3 : To read data from a file which is opened.

We can read the data with the help of Methods of FileInputStream.

int read() throws IOException	this method returns a character from a file in an integer format and moves the cursor the of the file to the next character. If there is no character to be read, then read method returns -1
int read(byte []) IOException	will read all the characters up to the size of the given array.
int available()	it returns the number of bytes of characters present in the file to be read.

Step4 : Close a file

<code>void close() throws IOException</code>	used to close a file
--	----------------------


Examples to write and read data to and from a file :

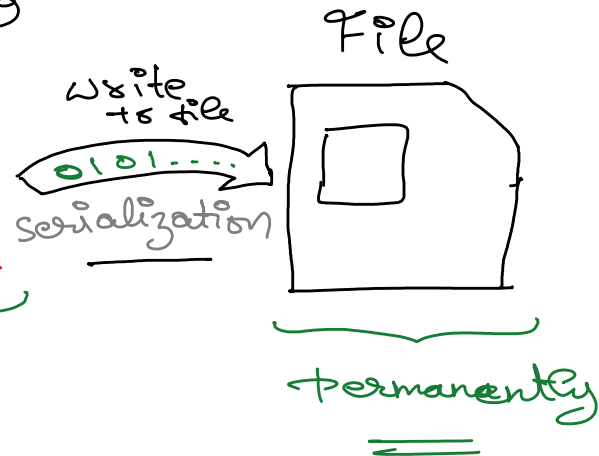
refer : **workspace2/FileHandling/src/io/pack1**


```

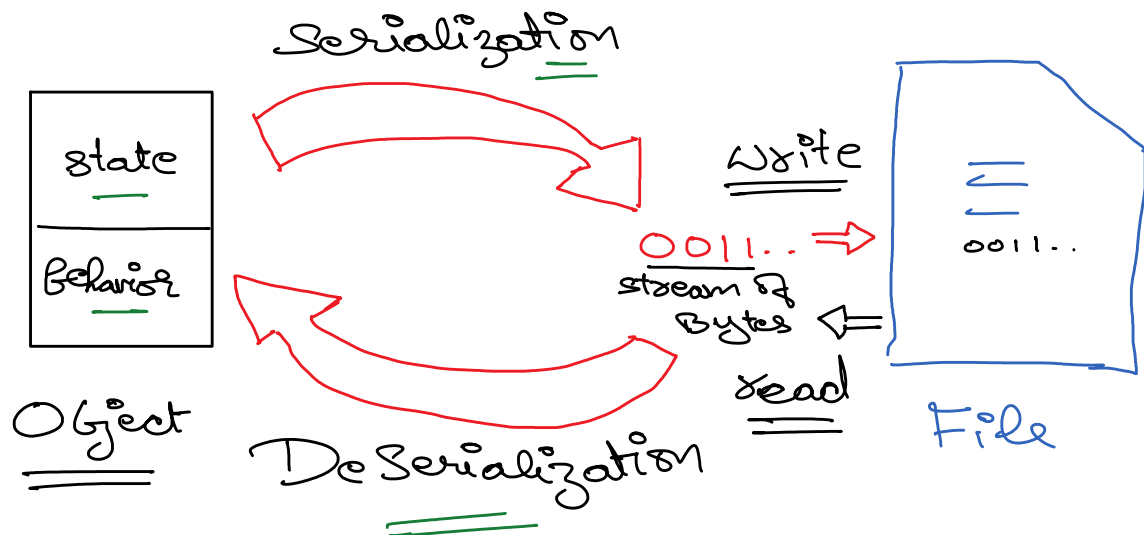
class Emp
{
    int cid ;
    String name ;
    void display()
    {
        //
    }
}

```

 smith
 new Emp() ;



To store write & read an object from a file :



Serialization :

The process of converting the object into a Stream of Bytes is known as Serialization. Serialization is mandatory if the object to be stored in a file.

In java, we have built-in mechanism to perform Serialization, but every

object of java is not eligible for Serialization.

The Object which implements **java.io.Serializable** interface, only those objects are eligible for Serialization.

java.io.Serializable :

It is an interface, which has no any abstract methods.

It is an empty interface.

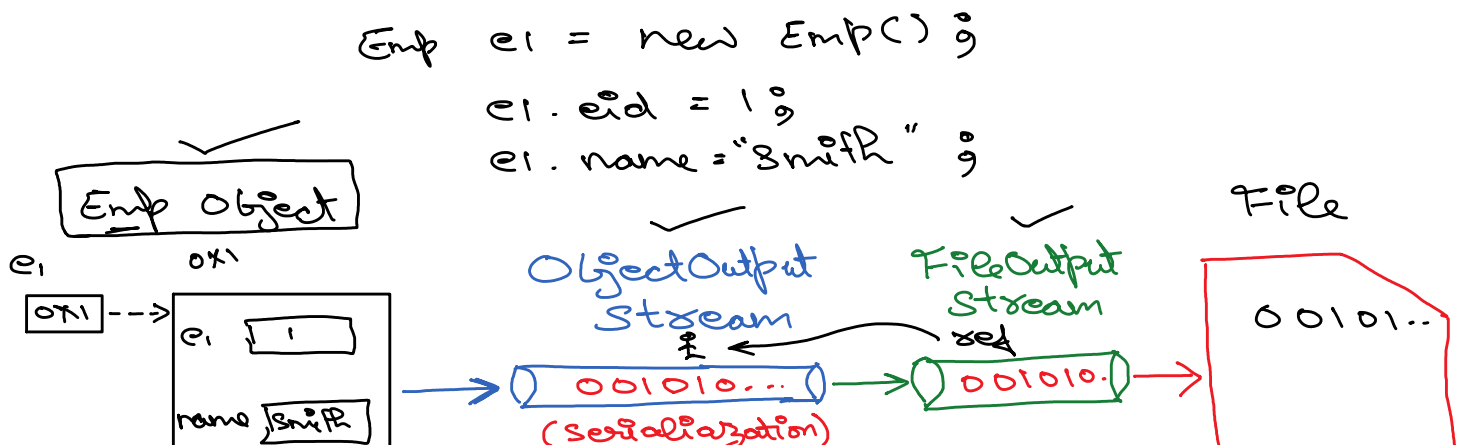
Note :

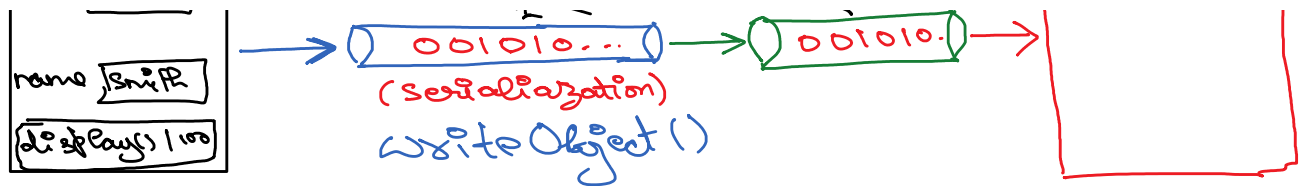
1. It is also known as a marker interface. (an interface which do not have any abstract methods is known as a marker interface)

Example:

```
class Emp implements Serializable
{
    int eid ;
    String name ;
    void <Display () { - - - }
}
```

Emp type object is Serializable. Therefore, we can store them into the file.





To Write an Object into a File :

Step1 :

Create an instance of `FileOutputStream` and initialize with the path/name of the file.

Step2 :

Create an instance of `ObjectOutputStream`, and initialize it with the reference of `FileOutputStream`.

Step3 :

write the object into the file with the help of a method

<code>writeObject(Object)</code>	non-static method of <code>ObjectOutputStream</code> class
---	--

Step4 :

close the file.

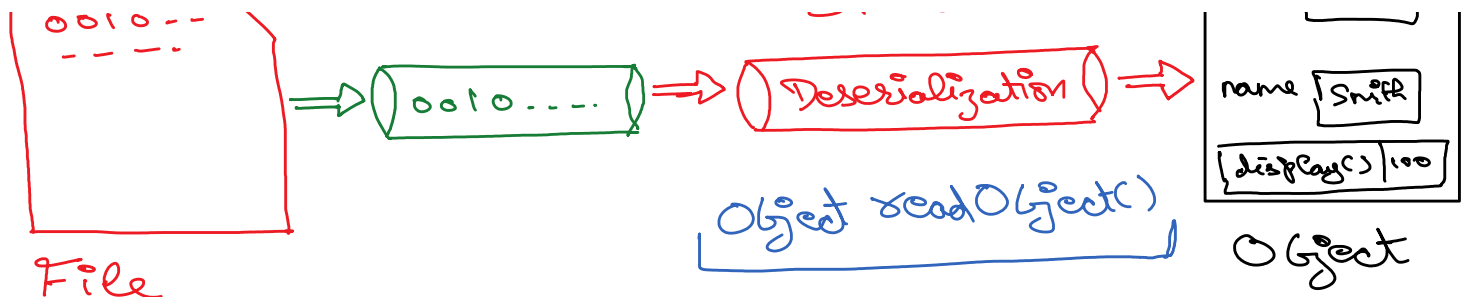
=====

Deserialization :

The process of Converting Stream of bytes into an Object is known as deserialization.

Demo1.txt





To Read An Object From a File :

Step1 :

Create an instance of `FileInputStream` and initialize with the filename/path.

Step2 :

Create an instance of `ObjectInputStream` and initialize with the reference of `FileInputStream`.

Step3 :

read the Object with the help of non-static method of `ObjectInputStream`.

<code>Object readObject()</code>	it detribalizes and returns the instance in Object type
----------------------------------	---

Step4 :

Close the File.

Examples :

Refer:

=====

transient modifier :

1. transient is a keyword.
2. any static or non-static variable prefixed with transient modifier, will not be

serialized, Hence it will not be stored in the File.