(Part 12)

Externalization,
Serialisation

When should one go for externalization and when should it be used?

Serialisation, → taken care completely by JVM,
↳ In Serialization it is not possible to partially serialize the object, whole object has to be done, Even if we use ~~[illegible]~~ "transient" Keyword we have read default value of those

Externalisation → Complete Control is with the programmer.
↳ In Externalisation part of the object can be ~~serialize~~ converted to transport supported format. This is not the case with serilisation. Here we can choose which attributes we want to transfer.

Now Imagine a class with ~~[illegible]~~ Lots of Attribute and we only need to convert (2-3) attributes into transport format.
If we do serialisation, unnecessarily we will slowdown down performance. So we should go for externalisation.

Class Account implements Serializeable → Class Gets Serializeable Ability

class Account implements Externalizable → Class Gets Externalizeable Ability

Serializeable → It is a marker Interface, has no methods where total ability will be provided by JVM.

Externalizeable → two methods are there and programmer is responsible for providing functionality

method1 → writeExternal ( )

method2 → readExternal ( )

$$\begin{tikzcd}
\boxed{\text{Serializeable (I)}} \\
\boxed{\text{Externalizeable (I)}} \arrow[u]
\end{tikzcd}$$

{ this is child interface of Serializeable }.

Since it requires programmer and hence because of laziness of programmer it is not so popular.

1. writeExternal ( ) {

    ① Serialization ( ) → Logic to be written

    ② to save required properties to the file

}

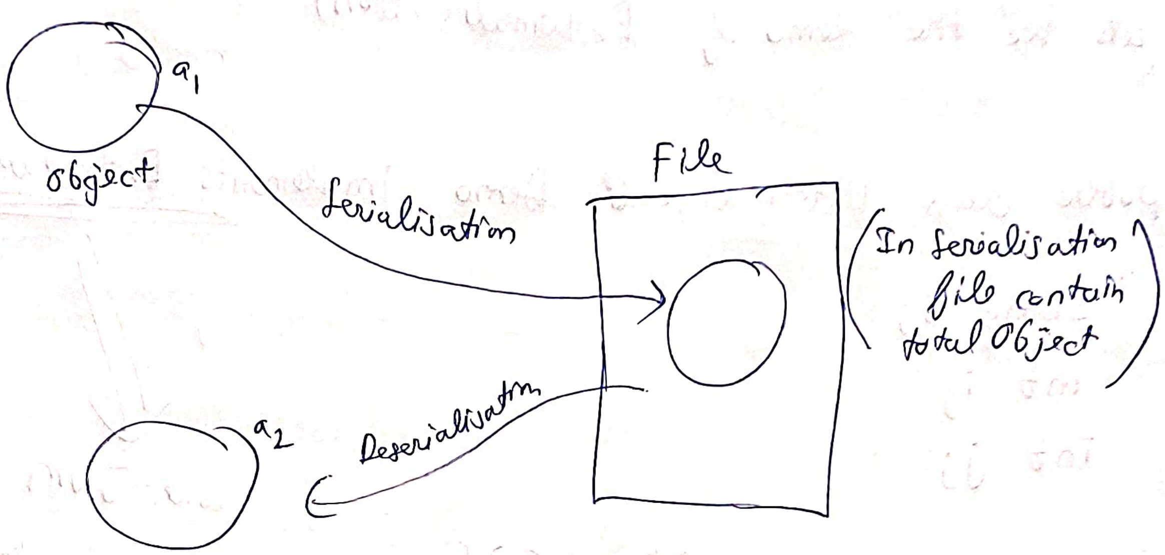
2. readExternal ( ) {

    ① DeSerialisation ( ) → Logic to deserialize

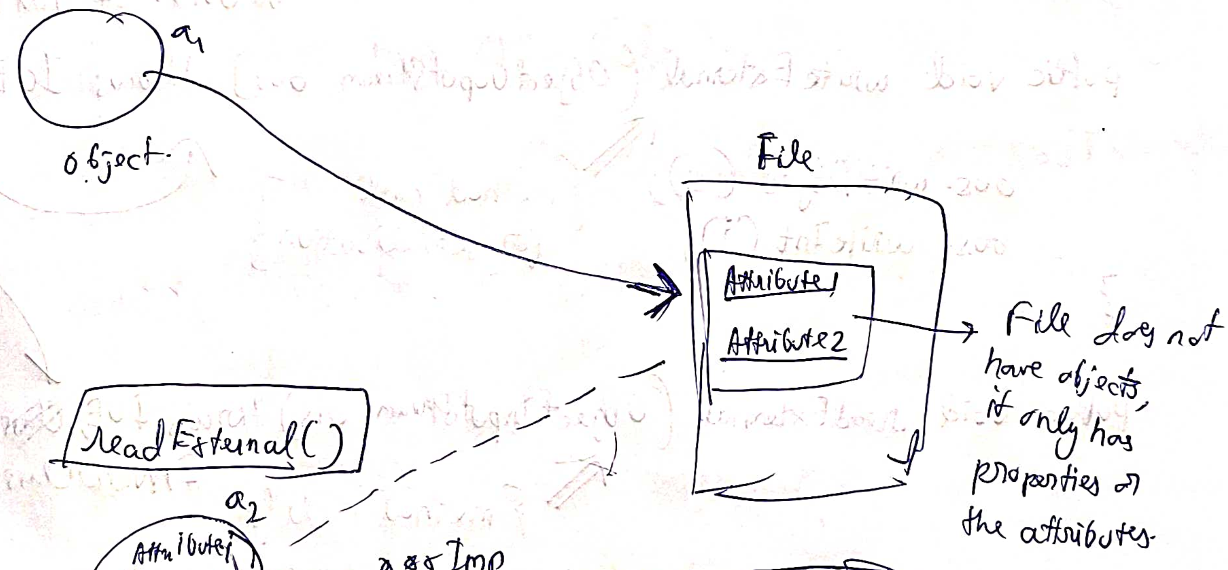
    ② To read the required variables from the file

}

In Serialisation



(In Externalization.)



***Imp

At the time of Deserialisation, JVM creates new Object automatically on this Object a_2, JVM will call read External method.

(***)

To ~~call this~~ create this Object JVM will call the public no args constructor. → If not present RunTime Exception: Invalid Class Exception.

(Part 13)

(Lets see the demo of Externalisation)

```
public class ExternalizeableDemo implements Externalizeable {

    String s;
    int i;
    int j;

    public ExternalizeableDemo ( ) {

        Soot ("public no-arg constructor");
    }

    public void writeExternal (ObjectOuputStream oos) throws IOE {

        oos. writeObject (s);
        oos. writeInt (i);
    }

    public void readExternal (ObjectInputStream ois) throws IOE, ~~Class~~
                                        INvalidClassException {

        s = (String )ois. readObject ( );

        i = ois. readInt ( );

    }.

}
```

When JVM sees that the class implements ~~Serializeable~~, Externalizeable

Whenever serialisation will be done. It will call 2 methods

{ method call for serialisation }

{ method call for externalisation }

```
psvm () {

    ExternalizeDemo t1 = new ExternalizeDemo ( );
     t1. s = " test";
     t1. i = 10;
     t1. j = 20;

      FOS  fos = new FOS( "abc.ser");
      OOS     oos = new OOS( fos);
      oos. writeObject ( t1);

       FIS    fis = new FIS ( "abc.ser");
       OIS     ois =  new OIS ( fis);
       ExternalizeableDemo t2 = (ExternalizeableDemo) ois.readExternal();

         sout( t2.s + "...."+ t2.i + "...." + t2.j);
               ⇓              ⇓              ⇓
              test            10             0 ↗ Since it is
                                                 not part of
}                                                Externalisation
```

During Deserialisation, JVM will call public no-args construction of the Object being created.

(★★★) Imp ~~Use of~~ Transient key-word has no use in Externalisation. If we declare an attribute transient and make it participate in externalization, it will work and data will get transferred.