SERIALIZATION & DE SERIALIZATION PROBLEMS

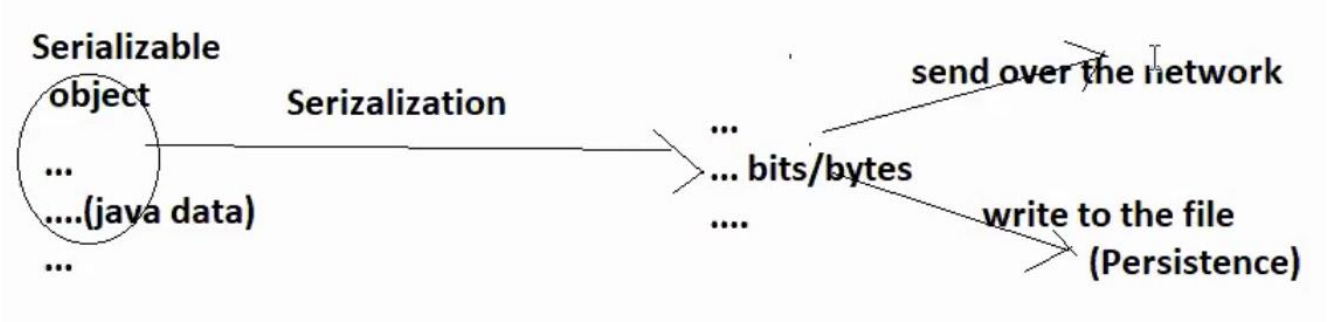
Serialization is the process of converting object data into bits and bytes once object data is converted into bits and bytes specially bits some people will takes those bits and bytes send over the network, some people will takes bit and bytes writes in the file , some people will take those bits and bytes putes into the pockets ….this is not the work of serialization ……**serialization main work is converting object state(data) into bits and bytes.**

Ex:- flipcart.com send card details to paypall as serializable object data

Mobile fames writes top 10 sores to files as serializable object data..

"serialization is also msarker interface by implementing serializable interface then only jvm know ok this object data I have to convert into bits and bytes and send over the network."

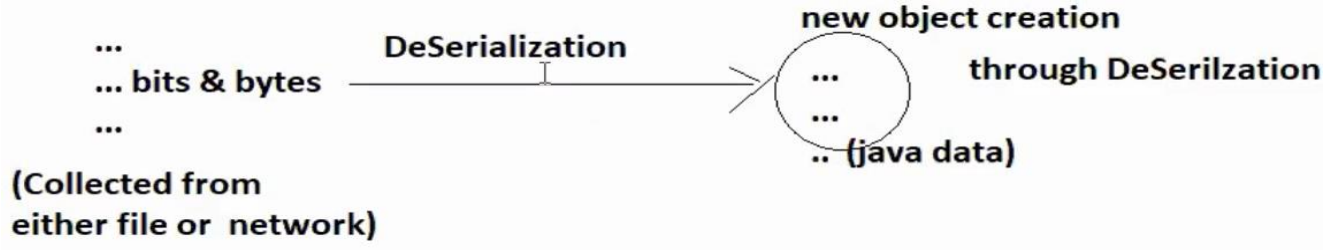
- ⇨ Serializable is posso le only on Serializable objects ex:- JVM converts only Serializable object data into bits and bytes
- ⇨ Implementation class object of java.io.serializable(I) marker interface are called serializable objects.



Desirializable :-

Converting bits and bytes into java data and creating new object having that java data as initial values is called Deserialization

- ⇨ In Deserialization process, constructor will not be executed though object is created because the gathered through Deserialization is already initialized with object so there is no need separated constructor based initialization to that object.



SERIALIZATION & DE SERIALIZATION PROBLEMS

We need to support of stream to write the data of Serialized Object to file and also for Deserialization

These streams of **high level streams**

- ⇨ High level stream internally take the support of LowLevel streams to complete read and write operation of the dest files..
- ⇨ FileINputStream,FileOutputStream and etc are Low level streams these streams can deal with only bytes and chars..
- ⇨ ObjectOutputStreams, ObjectInputStreams, DataOutputStream, DataInoutStream and etc are high level streams these streams can work with diff data types values directly because they will be converted into bytes and chars through low level streams.

Note:- "for serialization object we will use ObjectOutputStream and for deserialization we will use ObjectInputStream create new object with initial state"

SERIALIZATION & DE SERIALIZATION PROBLEMS

Ex:-

```
1. package com.nit.printer;
2.
3. import com.nit.cloneble.UtilsClass;
4.
5. public class Printer extends UtilsClass {
6.
7.       private static Printer instence;
8.       private static long serialVersionUID=56l;
9.
10.            //private constructor
11.            private Printer() {
12.                  System.out.println("0 param constructor ");
13.            }
14.
15.            //static factory method
16.            public static Printer getInstence() {
17.                  if(instence==null) {
18.                        instence=new Printer();
19.                  }
20.                  return instence;
21.            }
22.            public void printData(String data) {
23.                  System.out.println(data);
24.            }
25.
26.            //solving cloning problem
27.            //.by throwing exception (this is the best practice )
28.
29.            /*@Override
30.      public Object clone() throws CloneNotSupportedException {
31.      throw new CloneNotSupportedException("cloning is not allowed");
32.            }*/
33.
34.            //2.by returning existing object
35.            @Override
36.            public Object clone() throws CloneNotSupportedException {
37.
38.                  return instence;
39.            }
40.      }
```

SERIALIZATION & DE SERIALIZATION PROBLEMS

```
1. package com.nit.deserilizable;
2.
3. import java.io.FileInputStream;
4. import java.io.FileNotFoundException;
5. import java.io.FileOutputStream;
6. import java.io.ObjectInputStream;
7. import java.io.ObjectOutputStream;
8. import java.io.Serializable;
9.
10.     public class DeSerializationTest implements Serializable {
11.          public static void serilize(Object obj) {
12.               try {
13.                    ObjectOutputStream oos=new
   ObjectOutputStream(new FileOutputStream("abc.txt"));
14.                    oos.writeObject(obj);
15.                    oos.flush();
16.                    oos.close();
17.                    System.out.println("object serialized ");
18.               } catch (FileNotFoundException e) {
19.                    e.printStackTrace();
20.               }
21.               catch (Exception e) {
22.                    e.printStackTrace();
23.               }
24.          }
25.
26.          public static Object desrialize() {
27.               Object obj1 = null;
28.               try {
29.               ObjectInputStream ios=new ObjectInputStream(new
                  FileInputStream("abc.txt"));
30.                    obj1= ios.readObject();
31.                    System.out.println("Deserialised object");
32.                    ios.close();
33.
34.               } catch (Exception e) {
35.                    // TODO: handle exception
36.               }
37.               return obj1;
38.          }
39.     }
40.
```

SERIALIZATION & DE SERIALIZATION PROBLEMS

```
1. package com.nit.deserilizable;
2.
3. import com.nit.printer.Printer;
4.
5. public class DeserializeTest {
6.
7.       public static void main(String[] args) {
8.             Printer p1=null, p2=null;
9.             p1=Printer.getInstence();
10.                  DeSerializationTest.serilize(p1);
11.                  p2=(Printer) DeSerializationTest.desrialize();
12.                  System.out.println(p1.hashCode()+" "+p2.hashCode());
13.            }
14.      }
```

Output:-

```
0 param constructor
object serialized
Deserialised object
1028566121 1096979270
```

“so here by using deserialization we can also break singleton object ”

==========