Exterlization

- Externalization is another way to achive serialization.

Problems with Serialization.

- Here we implement 'Externalizable' ~~internal~~ interface which has implemented 'Serializable' interface.
'Externalizable' is not Marker Interface.

Program (ExternExample.java)

```
import java.io.*;

class Car implements Externalizable
{
 static int age;
   String name;
int year;
```

```
// madotory public no-arg Constructor

public ~~class()~~ Car() {
    Sop ("default");
}

Car(String n, int y) {
 {
   name=n;
   year=y;
  age=10;
 }


// mandatory writeExternal method
 public void writeExternal(ObjectOutput
                                  out)
         throws IOExcep^n {
   {
   Sop("writeExternal");
   out.writeObject(name);
   out.writeInt(year);
   out.writeInt(age);
  }


// Mandatory readExternal method
public void readExternal(ObjectInput in
  throws IOExcep^n, ClassNotFoundExcep^n
{
Sop ("readExternal");
```

```
name=(String) in.readObject();
 year = in.read Int();
 age = in.read Int();
}

public String toString()
{
return("Name:"+name+"\n"+"Year:"
        +year+"\n"+"age:"+age);
 }
}

Public class ExternExample{
   psvm()
{
 //create a car Object
 Car car=new Car("Mitsubishi",2009);
 Car newCar=null;

//serialize the car
try
{
FileOutputStream fo=new
        FileOutputStream("tmp");
ObjectO/pStream so = new ObjectO/pStream
                      (fo);
        so.writeObject (car);
         so.flush();
    } catch (Excepn e) { Sop(e); }

//deserialize the car.

  try {
  FileInputStream fi = new FileI/PStream("tmp");
Object I/PStream si = new Object I/P Stream(fi);
       newCar = (Car) si.readObject();
  }
  catch (Excepn e) {Sop(e);}


      Sop("The Original car is");
        Sop(car);
        Sop ("The new car is");
          Sop(newCar);
     }
  }
```

- In this example, class Car implements Externalizable interface which means that car object is ready for serialization

This class has two public methods
-"writeExternal" and "readExternal".
Unlike Serializable interface which
will serialize all the variables
in the object with just by imple-
-menting the interface, here you
have to explicitly mention what
feilds or variable you ~~have~~ want
to serialize and the same is
done in "writeExternal" and "read-
-External" methods. So in the
"ExternExample" class, when you
write "car" ~~obj~~ class object to
the OutputStream, the "write-
-External" method is called and the
data is persisted. The same
applies to "readExternal" method in
the car object i.e, when you read
the "Car" object from the ~~obj~~
ObjectInputStream, "readExternal"
method is called.

2. Also the information about class
description is added to the
stream which includes the
descripⁿ of all the serializable
superclasses, the description of
the class and the instance
data associated with specific
instance of the class. ~~[illegible]~~ Lots

of data and metadata and again performance issue.

3. Now when an Externalizable object is reconstructed, an instance is created first using the public no-arg constructor, then "readExternal" is called.

4. When an object that implements Serializable ~~object~~ interface, is serialized or de-serialized, no constructor of the Object is called and hence ~~any~~ any initialization which is done in the constructor can't be done.

5. Externalization is nothing but Serialization but by ~~Seri~~ implementing Externalizable interface to persist and restore the Object. To externalize your object, you need to implement Externalizable interface that extends Serializable interface. Here only the identity of the class is written in the serialization Stream and it is the responsibility of the class

to save and restore the
Contents of its ~~those~~
instances which means you
will have complete control of
what to serialize and not
to serialize. But ~~to~~ with
serialization the identity of
all the classes, its
superclasses, instance variables
and then the contents for
these items is written to the
serialization stream. But to
externalize an object, you need
a ~~B~~ default public constructor.

0-March-2018

6. Limitations of Serialization:—

(i) File Size is very high.

(ii) Customization due to transient which is not Effective becoz we get "null" in place of transient attributes.

7. One thing that you can do with Externalization is that you can store extra information into objects like STATIC variables and transient variables or you can add more information if you have any business need.

8. Externalization allows you to customize how serialization is done. By implementing externalization you are controlling what get serialized (and what doesnot) as versus default serialization where all non-transient attributes gets serialized.