Agenda :
1. Serialization
2. Deserialization
3. transient keyword
4. static Vs transient
5. transient Vs final
6. Object graph in serialization.
7. customized serialization.
8. Serialization with respect inheritance.
9. Externalization
10. Difference between Serialization & Externalization
11. SerialVersionUID

Serialization
=> The process of saving (or) writing state of an object to a file is called serialization but strictly speaking it is the process of
converting an object from java supported form to either network supported form (or) file supported form.
=> By using FileOutputStream and ObjectOutputStream classes we can achieve serialization process.
|=> writeObject(Object obj)

De-Serialization
=> The process of reading state of an object from a file is called DeSerialization but strictly speaking it is the process of
converting an object from file supported form (or) network supported form to java supported form.
=> By using FileInputStream and ObjectInputStream classes we can achieve DeSerialization.
|=> readObject()

```
import java.io.*;
/*
 public java.io.ObjectOutputStream(java.io.OutputStream) throws
java.io.IOException;
 public java.io.FileOutputStream(java.lang.String) throws
java.io.FileNotFoundException;
 public final void writeObject(java.lang.Object) throws java.io.IOException;

 public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
 public java.io.FileInputStream(java.lang.String) throws
java.io.FileNotFoundException;
 public final java.lang.Object readObject() throws java.io.IOException,
java.lang.ClassNotFoundException;
*/

class Dog implements Serializable
{
	static{
		System.out.println("static block gets executed...");
	}
	Dog(){
		System.out.println("Object is created...");
	}

	int i = 10;
	int j = 20;
}
```

```
class Test
{
      public static void main(String[] args)throws Exception
      {
                  Dog d = new Dog();

                  System.out.println("Serialization started.....");
                  String fileName = "abc.ser";
                  FileOutputStream fos   = new FileOutputStream(fileName);
                  ObjectOutputStream oos = new ObjectOutputStream(fos);
                  oos.writeObject(d);
                  System.out.println("Serialized Object refernce is ::"+d);
                  System.out.println("Serialization ended.....");

                  //To pause the execution till we press some key from keyboard
                  System.in.read();

                  System.out.println("De-Serialization started.....");
                  FileInputStream fis   = new FileInputStream("abc.ser");
                  ObjectInputStream ois = new ObjectInputStream(fis);
                  Object obj=ois.readObject();
                  Dog d1 = (Dog)obj;
                  System.out.println("De-Serialized Object refernce is ::"+d1);
                  System.out.println("De-Serialization ended.....");

      }
      //JVM shutdown now
}

eg#2.
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.Serializable;

class Dog implements Serializable{
       int i=10;
       int j=20;
}

class Cat implements Serializable{
      int i=100;
      int j=200;
}

public class TestApp {
      public static void main(String[] args)throws
IOException,ClassNotFoundException {

            Dog d1=new Dog();
            Cat c1=new Cat();

            System.out.println("serialization started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(d1);
            oos.writeObject(c1);
```

```
            System.out.println("Serialization ended");


            System.out.println("Deserialization started");
            FileInputStream fis=new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            Dog d2=(Dog) ois.readObject();
            Cat c2=(Cat) ois.readObject();
            System.out.println("Deserialization ended");

            System.out.println("Dog object data");
            System.out.println(d2.i+"\t" +d2.j);

            System.out.println("Cat object data");
            System.out.println(c2.i+"\t" +c2.j);

      }
}
```

Output

```
serialization started
Serialization ended
Deserialization started
Deserialization ended
Dog object data
10      20
Cat object data
100     200
```

Note:

1. We can perform Serialization only for Serilizable objects.
2. An object is said to be Serilizable if and only if the corresponding class implements Serializable interface.
3. Serializable interface present in java.io package and does not contain any abstract methods. It is marker interface.
   The required ability will be provided automatically by JVM.
4. We can add any no. Of objects to the file and we can read all those objects from the file but in which order we wrote
   objects in the same order only the objects will come back. That is order is important.
5. If we are trying to serialize a non-serializable object then we will get RuntimeException saying "NotSerializableException".

Transient keyword:

1. transient is the modifier applicable only for variables,but not for classes and methods.
2. While performing serialization if we don't want to save the value of a particular variable to meet security constant such
   type of variable ,then we should declare that variable with "transient" keyword.
3. At the time of serialization JVM ignores the original value of transient variable and save default value to the file .
4. That is transient means "not to serialize".

eg#1.

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
```

```
import java.io.ObjectInputStream;
import java.io.Serializable;


class Dog implements Serializable{
      int i=10;
      transient int j=20;
}

public class TestApp {
      public static void main(String[] args)throws
IOException,ClassNotFoundException {

            Dog d1=new Dog();

            System.out.println("serialization started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(d1);
            System.out.println("Serialization ended");


            System.out.println("Deserialization started");
            FileInputStream fis=new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            Dog d2=(Dog) ois.readObject();
            System.out.println("Deserialization ended");

            System.out.println("Dog object data");
            System.out.println(d2.i+"\t" +d2.j);

      }
}
Output
serialization started
Serialization ended
Deserialization started
Deserialization ended
Dog object data
10      0

static Vs transient :
1. static variable is not part of object state hence they won't participate in
serialization   because of this declaring a static variable as
     transient there is no use.

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.Serializable;

class Dog implements Serializable{
       static transient int i=10;
       int j=20;
}

public class TestApp {
```

```
      public static void main(String[] args)throws
IOException,ClassNotFoundException {

            Dog d1=new Dog();

            System.out.println("serialization started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(d1);

            System.out.println("Serialization ended");


            System.out.println("Deserialization started");
            FileInputStream fis=new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            Dog d2=(Dog) ois.readObject();

            System.out.println("Deserialization ended");

            System.out.println("Dog object data");
            System.out.println(d2.i+"\t" +d2.j);
      }
}
Output
serialization started
Serialization ended
Deserialization started
Deserialization ended
Dog object data
10      20

Transient Vs Final:
1. final variables will be participated into serialization directly by their
values.
     Hence declaring a final variable as transient there is no use.
       //the compiler assign the value to final variable

eg: final int x= 10;
      int y = 20;
      System.out.println(x);// compiler will replace this as System.out.println(20)
becoz x is final.
      System.out.println(y);

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.Serializable;

class Dog implements Serializable{
     int i=10;
     transient final int j=20;
}


public class TestApp {
      public static void main(String[] args)throws
```

```
IOException,ClassNotFoundException {

            Dog d1=new Dog();

            System.out.println("serialization started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(d1);

            System.out.println("Serialization ended");


            System.out.println("Deserialization started");
            FileInputStream fis=new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            Dog d2=(Dog) ois.readObject();

            System.out.println("Deserialization ended");

            System.out.println("Dog object data");
            System.out.println(d2.i+"\t" +d2.j);

      }
}
Output
Serialization started
Serialization ended
Deserialization started
Deserialization ended
Dog object data
10      20

Declaration output
================
1.
int i=10;
int j=20;

output
  10 -------- 20

2.
transient int i=10;
int j=20;

output
   0-------20

3.
transient int i=10;
transient static int j=20;

output
    0---------20

4.
transient final int i=10;
transient int j=20;
```

output
     10-------0

5.
transient final int i=10;
transient static int j=20;
output
    10--------20

Note:
We can serialize any no of objects to the file but in which order we serialized in the same order only we have to deserialize,
if we change the order then it would result in "ClassCastException".

Example :
```
Dog d1=new Dog( );
Cat c1=new Cat( );
Rat r1=new Rat( );

FileOutputStreamfos=new FileOutputStream("abc.ser");
ObjectOutputStreamoos=new ObjectOutputStream(fos);
oos.writeObject(d1);
oos.writeObject(c1);
oos.writeObject(r1);

FileInputStreamfis=new FileInputStream("abc.ser");
ObjectInputStreamois=new ObjectInputStream(fis);
Dog d2=(Dog)ois.readObject();
Cat c2=(Cat)ois.readObject();
Rat r2=(Rat)ois.readObject();
```

=> If we don't know the order of Serialization then we need to use the following code

```
      FileInputStream fis =new FileInputStream("abc.ser");
      ObjectInputStream ois=new ObjectInputStream(fis);

      Object obj=ois.readObject();
      if(obj instanceof Dog){
            Dog d=(Dog)obj;
            //perform operation related to Dog
      }
      if(obj instanceof Cat){
            Cat C=(Cat)obj;
          //perform operation related to Cat
      }
      if(obj instanceof Rat){
             Rat r=(Rat)obj;
           //perform operation related to Rat
      }
```

Object graph in serialization:

1. Whenever we are serializing an object the set of all objects which are reachable from that   object will be serialized
     automatically. This group of objects is nothing but object graph in serialization.
2. In object graph every object should be Serializable otherwise we will get runtime exception saying
    "NotSerializableException".

eg#1.

```
import java.io.Serializable;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.IOException;


class Dog implements Serializable{
      Cat c=new Cat();
}

class Cat implements Serializable{
      Rat r=new Rat();
}

class Rat implements Serializable{
       int i=10;
}
public class Test {
      public static void main(String[] args)throws
IOException,ClassNotFoundException{

            Dog d= new Dog();

            System.out.println("Serialization Started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(d);
            System.out.println("Serialization ended");

            System.out.println("**********************************");

            System.out.println("DeSerialization Started");
            FileInputStream fis= new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            Dog d1=(Dog)ois.readObject();
            System.out.println(d1.c.r.i);
            System.out.println("DeSerialization ended");


   }
}
```

Output

```
======
Serialization Started
Serialization ended
**********************************
DeSerialization Started
10
DeSerialization ended
```

CustomizedSerialization

```
=======================
```

During default Serialization there may be a chance of lose of information due to transient keyword.

example: remember mango and money inside it.

```
eg#1.
import java.io.Serializable;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.IOException;

class Account implements Serializable{
      String name="sachin";
      transient String password="tendulkar";
}
public class Test {
      public static void main(String[] args)throws
IOException,ClassNotFoundException{

            Account acc=new Account();
            System.out.println(acc.name +"=====> "+ acc.password);

            System.out.println("Serialization Started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(acc);
            System.out.println("Serialization ended");

            System.out.println("**********************************");

            System.out.println("DeSerialization Started");
            FileInputStream fis= new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            acc=(Account)ois.readObject();
            System.out.println(acc.name +"=====> "+ acc.password);
            System.out.println("DeSerialization ended");
   }
}
```

=> In the above example before serialization Account object can provide proper username and password.

But after Deserialization Account object can provide only username but not password. This is due to declaring password as transient.

Hence doing default serialization there may be a chance of loss of information due to transient keyword.

=> We can recover this loss of information by using customized serialization.

We can implements customized serialization by using the following two methods.

1. private void writeObject(ObjectOutputStream os) throws Exception.
   => This method will be executed automatically by jvm at the time of serialization.
   => Hence at the time of serialization if we want to perform any extra work we have to define that in this
      method only. (prepare encrypted password and write encrypted password seperate to the file)

2. private void readObject(ObjectInputStream is) throws Exception.
   => This method will be executed automatically by JVM at the time of Deserialization.
   Hence at the time of Deserialization if we want to perform any extra activity

we have to define that in this method only.
      (read encrypted password , perform decryption and assign decrypted password to the current object password variable)

eg#1.

```
import java.io.Serializable;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.IOException;

class Account implements Serializable{
      String name="sachin";
      transient String password="tendulkar";

      private void writeObject(ObjectOutputStream oos)throws Exception{
            oos.defaultWriteObject();//performing default Serialization

            String epwd="123"+password;//performing encryption

            oos.writeObject(epwd);//write the encrypted data to file(abc.ser)


      }
      private void readObject(ObjectInputStream ois)throws Exception{
            ois.defaultReadObject();//performing default Serialization

            String epwd=(String)ois.readObject();//performing decryption

            password=epwd.substring(3);//writing the extra data to Object
      }
}
public class Test {
      public static void main(String[] args)throws
IOException,ClassNotFoundException{

            Account acc=new Account();
            System.out.println(acc.name +"=====> "+ acc.password);

            System.out.println("Serialization Started");
            FileOutputStream fos= new FileOutputStream("abc.ser");
            ObjectOutputStream oos=new ObjectOutputStream(fos);
            oos.writeObject(acc);
            System.out.println("Serialization ended");

            System.out.println("**********************************");

            System.out.println("DeSerialization Started");
            FileInputStream fis= new FileInputStream("abc.ser");
            ObjectInputStream ois=new ObjectInputStream(fis);
            acc=(Account)ois.readObject();
            System.out.println(acc.name +"=====> "+ acc.password);
            System.out.println("DeSerialization ended");
   }
}
```

=> At the time of Account object serialization JVM will check is there any writeObject() method in Account class or not.

=> If it is not available then JVM is responsible to perform serialization(default serialization).
 => If Account class contains writeObject() method then JVM feels very happy and executes that Account class
        writeObject() method.
  => The same rule is applicable for readObject() method also.

```
import java.io.*;
/*
 public java.io.ObjectOutputStream(java.io.OutputStream) throws
java.io.IOException;
 public java.io.FileOutputStream(java.lang.String) throws
java.io.FileNotFoundException;
 public final void writeObject(java.lang.Object) throws java.io.IOException;

 public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
 public java.io.FileInputStream(java.lang.String) throws
java.io.FileNotFoundException;
 public final java.lang.Object readObject() throws java.io.IOException,
java.lang.ClassNotFoundException;

*/

class Account implements Serializable
{
      String userName  = "sachin";
      transient String password = "tendulkar";//loss of information
      transient int pin=4444;//loss of information


        //Write a logic of Serialization
       private void writeObject(ObjectOutputStream oos) throws Exception{
            System.out.println("writeObject method is called....");

            // perform default serialization
            oos.defaultWriteObject();

            // perform encryption on password
            String encypwd  = "123" + password;// 123tendulkar
            int encypin     = 1111  + pin;// 5555

            // write the encrytped data as object to serialized file
            oos.writeObject(encypwd);
            oos.writeInt(encypin);

       }

        //Write a logic of Deserialization
       private void readObject(ObjectInputStream ois) throws Exception{
            System.out.println("readObject method is called....");

            //perform default deserialization
            ois.defaultReadObject();

            //read encrypted data from serialized file
            String encypwd = (String)ois.readObject();
            int encypin    = ois.readInt();
```

```
            // perform decryption and attach it to instance variable
            password = encypwd.substring(3);// tendulkar
            pin      = encypin - 1111;// 4444
       }

}

class Test
{
      public static void main(String[] args)throws Exception
      {
                  Account account =new Account();

                  System.out.println("Serialization started.....");
                  String fileName = "abc.ser";
                  FileOutputStream fos   = new FileOutputStream(fileName);
                  ObjectOutputStream oos = new ObjectOutputStream(fos);
                  oos.writeObject(account);
                  System.out.println("Serialization ended.....");

                  //To pause the execution till we press some key from keyboard
                  System.in.read();

                  System.out.println("De-Serialization started.....");
                  FileInputStream fis   = new FileInputStream("abc.ser");
                  ObjectInputStream ois = new ObjectInputStream(fis);
                  Account acc=(Account)ois.readObject();

                  System.out.println("Username is :: "+acc.userName);
                  System.out.println("Password is :: "+acc.password);
                  System.out.println("Pin      is :: "+acc.pin);
                  System.out.println("De-Serialization ended.....");

      }
      //JVM shutdown now
}

Output
D:\IOOperations>javac Test.java

D:\IOOperations>java Test
Serialization started.....
writeObject method is called....
Serialization ended.....

De-Serialization started.....
readObject method is called....
Username is :: sachin
Password is :: tendulkar
Pin      is :: 4444
De-Serialization ended.....
```