Serialization Vs DeSerialization
++++++++++++++++++++++++++++++++
1. Serialization => Converting java object to n/w supported of file supported format.
                      FileOutpuStream,ObjectOutputStream(writeObject(obj))

2. DeSerialization => Converting n/w or file supported format object to java object.
                      FileInputStream,ObjectInputStream(readObject())

Note: When we try to Serialize the object, if the object is not in Serializable format then it would result in "NotSerializableException".
      To make the object Serializable, we need to implement an interface called "Serializable"[Marker Interface].

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.*;

class Dog implements Serializable
{
      Cat c = new Cat();
}
class Cat implements Serializable
{
      Rat r = new Rat();
}
class Rat implements Serializable
{
      int j = 20;
}
public class Test
{
      public static void main(String... args) throws Exception{
            Dog d1 = new Dog();

            System.out.println("Serialization Started...");
            //Performing Serialization
            new ObjectOutputStream(
                  new FileOutputStream("Dog.ser")).writeObject(d1);

            System.out.println("End of Serialization...");

            System.out.println();
            System.out.println("De-Serialization Started...");
            //Performing Serialization
            Dog d2= (Dog)new ObjectInputStream(
                  new FileInputStream("Dog.ser")).readObject();
```

```
            System.out.println("Value of j is :: "+d2.c.r.j);
            System.out.println("End of Serialization...");
      }
}

Output
Serialization Started...
End of Serialization...

De-Serialization Started...
Value of j is :: 20
End of Serialization...

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

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.
    => It is a callback method. 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.

Try for the following data
++++++++++++++++++++++++++

```
class Account implements Serializable{
      String name="sachin";
      transient String password="tendulkar";
      transient int pin=4444;

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

            String epwd="123"+password;//performing encryption
            int epin=1234+pin;//performing encryption


            oos.writeObject(epwd);//write the encrypted data to file(abc.ser)
            oos.writeInt(epin);//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
            int epin=ois.readInt();//performing decryption

            password=epwd.substring(3);//writing the extra data to Object
            pin=epin-1234;//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+"======>"+acc.pin);

            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+"========>"+acc.pin);
            System.out.println("DeSerialization ended");
   }
}
Output
sachin=====> tendulkar======>4444
Serialization Started
Serialization ended
**********************************
DeSerialization Started
sachin=====> tendulkar========>4444
DeSerialization ended


}
```