**SERIALIZATION**

1) Introduction

2) Object graph

3) Serialization with respect to inheritance

4) customized serialization

**Serialization:**

**The process of saving an object to file or the process of sending an object across the network is called serialization** but strictly speaking is the process of converting an object from java supported form to file supported form or network supported form by using **FileOutputStream & ObjectOutputStream** classes we can active serialization

**Deserialization:**

**The process of reading an object from a file or from the network is called deserialization** but strictly speaking deserialization is the process of converting an object from file support form or n/w supported form to java supported form.

By using **FileInputStream, ObjectInputStream** classes we can achieve Deserialization

Import java.io.\*;

class Dog **implements Serialization**

{

int x = 10;

int y = 20;

}

class SerializationDemo

{

public static void main (String[ ] args) throws IOException

{

Dog d1 = new Dog ();

FileOutputStream fos = new FileOutputStream(“abc.ser”);

ObjectOutputStream os = new ObjectOutputStream(fos);

fos.writeObject (d1);

//Deserializaiton

FileInputStream fis = new FileInputStream(“abc.ser);

ObjectInputStream is = new ObjectInputStream(fis);

Dog d2 = (d1)is.readObject ( );

}

}

We can perform serialization, only for serialization object an object is said to serializable if and only if the corresponding class implements Serializable interface.

**Serializable interface present in java.io package and doesn’t contain any method & hence it is marker interface.**

if we are trying to serialize, non-serializable object, we will get runtime exception saying not serializable exception

while performing serialization if we don’t want to send the value of a variable, to meet security constraint, such type of variables we must check with **transient keyword**

**For the transient variable, jvm send default value**, instead of original value transient means not to serializable they won’t participate in serialization process,

1. declaring static variable as transient, there is no impact at all
2. final variable will be participated in the serialization directly by value

Hence declaring final variable as transient there is impact.

**Output:**

int i = 10;, int j = 20; 10 --- 20

int i = 10;

transient int j = 20 10 ---- 0

int i = 10;

transient static int j = 20 10 ---- 20

transient int i = 10;

transient static int j = 20 0 ----- 20

transient final int i= 10

transient static int j = 20 10 ---- 20

**Object Graphs in Serialization:**

whenever we are saving an object to a file, the set of all objects which are reachable from that obj, will be saved automatically this group of objects is nothing but object graph

On the object graph, every obj should be serializable otherwise we will get runtime Exception saying “notserialazable exception”

class Dog implements Serialization{

Cat c = new Cat();

Rat r = new Rat();

}

class Rat implements Serialization {int j = 20;}

class Serialize Demo2{

public static void main(String [] args) throws IOException

{

Dog d1 = new Dog();

FileOutputStream fos = new FileOutputStream (ObjectOutputStream oos = new ObjectOutputStream (for);)

**oos.writeObject (d1);**

System.out.println(d1.c.r.j);

FileInputStream fis = new FileInputStream (“abc.ser”);

ObjectInputStream ois = new ObjectInputStream (fis);

**Dog d2 = (dog)ois.readObject ( );**

System.out.println (d1.c.r.j);

}

}

In the above program, whenever we are trying to serialize dog obj, cat & rat object by default will be saved automatically, because these are part of object graph of dog obj Among Dog, Cat, Rat, if any class is not serialize, we will get runtime exception sayings not serializable exception **hence every object in object graph compulsory should be serializable**

{

Transient Cat c = new Cat ();

}

class Cat

{

int j = 20;

}

class SerializaDemo

{

public static void main(String s[]){

Dog d = new Dog( );

System.out println(d.c.j);

FileOutputStream for = new FileOutputStream (“abc.ser”)

ObjectOutputStream oos = new ObjectOutputStream (for

oos. WriteObject(d);

FileInputStream fis = new FileInputStream(abs ser”)

ObjectInputStream ois = new ObjectInputStream(fis);

Dog d1 = (dog) ois. readObject ( );

System.out.println (d1.c.j); // N.P.E

}

}

In the above prog, before serialization, dog object an provide value, but after deserialization dog object can’t provide j value i.e during deserializaiton, there may be chance of loss of information**, because of transient keyword we can recover this loss of information, by using customized serialization.**

We can achieve customized serialization, by using the following two methods

**1) private void writeObject (OutputStream os) {} this method will be executed automatically by the JVM, at the time of serialization**

**i) private void readObject (InputStream is){} this method will be excuted automatically by the JVM, at the time of deserialiation**

class Cat

{

int j = 20;

}

class Dog implements Serializable

{

Transient Cat c = new Cat ( );

private void writeObject(ObjectOutputStream os) throws Exception

{

os.writeObject ( );

int x = c.j + 200;

os.writeInt (x);

}

private void readObject (ObjectInputStream is ) throws Exception

{

is. readObject ( );

int k = is. readInt ( );

c = new Cat ( );

c.j = k-200;

int k = is.readInt()-200; c = new Cat ();, c.j = k;

}

class SerializationDemo

{

public static void main (String args[])

{

Dog d = new Dog( );

System.out.println(d.c.j); //20

FileOutputStream fos = new FileOutputStream (“abc.ser”);

ObjectOutputStream oos= new ObjectOutputStream(fos);

oos. WritObject(d);

FileInputStream fis = new FileInputStream (“abs .ser”);

ObjectInputStream ois = new ObjectInputStream(fis);

Dog d = (dog) ois.readObject ()

System.out.println (d1.c.j); //20

}

}

**What is the purpose of transient?**

**What is the difference between default serialization & customized serialization?**

**Serialization with Respect to serializable**

Serializable nature is inherited from parent to child i.e. **if the parent is serializable then automatically every child is also by default serializable.**

if the parent class implements serializable then we can serialize child class obj also even though Child doesn’t implement serializable especially

class Animal implements Serializable

{

int i= 10;

}

class Dog extends Animal

{

int j = 20;

}

In this case we can serialize dog obj because parent is already serializable

**Note:**

**Every servlet in java is by default serializable because parent class generic servlet already implements serializable**

Even though parent class is not serializable then we can serialize child class obj.