DAY-66

-------

OBJECT GRAPH in Serialization:

------------------------------

Whenver we are serializing the object, the set of all the other objects which are reachable from that object will be serialized automatically

this group of objects is called as object graph.

In object graph every object should be serializable, if not we will get exception.

// example-8 OBJECT GRAPH W.R.T SERIALIZATION

----------------------------------------------

import java.io.\*;

class Car implements Serializable

{

Bike b= new Bike();

}

class Bike implements Serializable

{

Truck t = new Truck();

}

class Truck implements Serializable

{

int i = 20;

}

class ObjectGraphSerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

FileOutputStream fos = new FileOutputStream("xyz.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("xyz.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

Car c1 = (Car)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(c1.b.t.i);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

20

\*/

CUSTOMIZED SERIALIZATION:

------------------------

Note: during serialization there may be chance of lose of data because of transient keyword

// example-9 Customized SERIALIZATION and De-Serialization

------------------------------------------------------------

import java.io.\*;

class Account implements Serializable

{

String UserName = "sagar";

transient String pwd = "8954";

}

class CustSerializeDemo

{

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

{

// creation of object

Account ac =new Account();

// code for serialization

FileOutputStream fos = new FileOutputStream("lmn.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(ac);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("lmn.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

Account ac1 = (Account)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(ac1.UserName);

System.out.println(ac1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

20

\*/

we can implement customized serialiation using two ways:

1. private void writeObject(ObjectOutputStream os) throws Exception

--> this method will be executed automatically during the time of serialization.Any activity that need to be performed during serialization must be

declared under this method.

2. private void readObject(ObjectInputStream is) throws Exception

--> this method will be executed automatically during the time of De-serialization.Any activity that need to be performed during De-serialization

must be declared under this method.

Note: while performing serialization the above methods must be declared in the class of wihich object we are going to serilaize.

eg: declare this method in car class not in serilizationDemo class.

// example-10 Customized SERIALIZATION and De-Serialization

------------------------------------------------------------

import java.io.\*;

class Account1 implements Serializable

{

String UserName = "sagar";

transient String pwd = "8954";

// encrypting the data during serialization

private void writeobject(ObjectOutputStream os) throws Exception

{

os.defaultWriteObject();

String epwd="6732"+pwd;

os.writeObject(epwd);

}

// decrypting the data during de-serialization

private void readobject(ObjectInputStream is) throws Exception

{

is.defaultReadObject();

String epwd =(String)is.readObject();

pwd = epwd.substring(4);

}

}

class CustSerializeDemo2

{

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

{

// creation of object

Account1 ac =new Account1();

// code for serialization

FileOutputStream fos = new FileOutputStream("uvw.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(ac);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("uvw.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

Account1 ac1 = (Account1)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(ac1.UserName);

System.out.println(ac1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagar

null

\*/

SERIALIZATION W.R.T INHERITANCE:

---------------------------------

CASE-1

-------

example-11

--> As per case-1 even though the child class doesnt implement serilizeable we can serialize child class object if the parent class implements

serilizable interface.

hence in case serilizeable nature is inherited from parent to child.

// example-11 Inheritance in serialization..!

-----------------------------------------------

import java.io.\*;

class Bike implements Serializable

{

int i =10;

}

class KTM extends Bike

{

int j=20;

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

{

// creation of object

//Bike b = new Bike();

KTM k = new KTM();

// code for serialization

FileOutputStream fos = new FileOutputStream("abc.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(k);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("abc.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

KTM k1 = (KTM)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(k1.i);

System.out.println(k1.j);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

10

20

\*/

CASE-2

-------

example-12

// example-12 Inheritance in serialization..!

-----------------------------------------------

import java.io.\*;

class Bike //implements Serializable

{

int i =10;

}

class KTM extends Bike implements Serializable

{

int j=20;

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

{

// creation of object

//Bike b = new Bike();

KTM k = new KTM();

// code for serialization

FileOutputStream fos = new FileOutputStream("abc.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(k);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("abc.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

KTM k1 = (KTM)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(k1.i);

System.out.println(k1.j);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

10

20

\*/

--> Even though parent class doesnt implement serializeable still we can serialize child class object if child class implements serialize interface.

ie, to serialize child class parent class need not to be serializable

example-13

// example-13 Inheritance in serialization..!

------------------------------------------------

import java.io.\*;

class Bike

{

int i =10;

}

class KTM extends Bike implements Serializable

{

int j=20;

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

{

// creation of object

//Bike b = new Bike();

KTM k = new KTM();

k.i=888;

k.j=999;

// code for serialization

FileOutputStream fos = new FileOutputStream("abc.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(k);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("abc.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

KTM k1 = (KTM)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(k1.i);

System.out.println(k1.j);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

10

999

\*/

--> At the time of serilization jvm will check if any variable is inheriting from non-serilizable parent or not.

In case if any variable is inheriting from non-serializable parent then jvm ignore modified value and saves its original value

example-14

// example-14 Inheritance in serialization..!

-----------------------------------------------

import java.io.\*;

class Bike implements Serializable

{

int i =10;

}

class KTM extends Bike implements Serializable

{

int j=20;

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

{

// creation of object

//Bike b = new Bike();

KTM k = new KTM();

k.i=888;

k.j=999;

// code for serialization

FileOutputStream fos = new FileOutputStream("abc.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(k);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("abc.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

KTM k1 = (KTM)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(k1.i);

System.out.println(k1.j);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

888

999

\*/

--> At the time of serilization jvm will check if any variable is inheriting from non-serilizable parent or not.

In case if any variable is inheriting from serializable parent then jvm ignore original value and saves its newly modified value

example-15

// example-15 Inheritance in serialization..!

------------------------------------------------

import java.io.\*;

class Bike implements Serializable

{

int i =10;

Bike()

{

System.out.println("Bike constructor..!");

}

}

class KTM extends Bike implements Serializable

{

int j=20;

KTM()

{

System.out.println("KTM constructor..!");

}

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

{

// creation of object

//Bike b = new Bike();

KTM k = new KTM();

k.i=888;

k.j=999;

// code for serialization

FileOutputStream fos = new FileOutputStream("abc.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(k);

System.out.println("serialization is successfully completed...!");

// code for De-serialization

FileInputStream fis = new FileInputStream("abc.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

KTM k1 = (KTM)ois.readObject();

System.out.println("De-serialization is successfully completed...!");

System.out.println(k1.i);

System.out.println(k1.j);

}

}

/\*

output:

-------

Bike constructor..!

KTM constructor..!

serialization is successfully completed...!

De-serialization is successfully completed...!

888

999

\*/

--> At the time of serilization jvm will check if any variable is inheriting from non-serilizable parent or not.

In case if any variable is inheriting from non-serializable parent then jvm executes its zero-argument constructor of parent class.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------