DAY-65

-------

Serialization:

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

The process of writing state of an object to a file is called Serialization.

It is called as converting a java supported object to file supported or network supported form.

Serialization can be achived using file outputstream and object outputstream.

refer

dia:1

// example-1 serialization

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

import java.io.\*;

class Car implements Serializable

{

int i = 10;

int j = 20;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

}

}

output:

-------

serialization is successfully completed...!

De-serialization:

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

The process of reading state of an object from a file is called De-Serialization.

It is called as converting a file supported or network supported object to java supported object.

De-Serialization can be achived using fileinputstream and objectinputstream

// example-2 De-serialization

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

import java.io.\*;

class Car implements Serializable

{

int i = 10;

int j = 20;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.i);

System.out.println(c1.j);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

10

20

\*/

dia: program diagram

Note:The object in the above program is a normal object hence it can not serialize.Only serializable objects can be serialized.

to make a object serializable in nature we must implement serializeable interface which is a maker interface.

transient keyword:

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

This modifier is applicable only for variable but not for methods and classes.

At the time of serialization if we dont want to save the value of some variables to meet the security constrains then that variable will be declared

as transient.

during the serialization process jvm will ignore the original value and stores default value.

// example-3 serialization and De-serialization without transient keyword

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

import java.io.\*;

class Car implements Serializable

{

String username = "sagarram";

int pwd = 8954;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.username);

System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagarram

8954

\*/

// example-3.1 serialization and De-serialization with transient keyword

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

import java.io.\*;

class Car implements Serializable

{

String username = "sagarram";

transient int pwd = 8954;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.username);

System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagarram

0

\*/

static modifier:

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

static is not a part of object state hence it will not participate in serialization.

Therfore making a value/variable as static as no effects on serialization.

// example-4 serialization and De-serialization with static keyword

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

import java.io.\*;

class Car implements Serializable

{

String username = "sagarram";

static int pwd = 8954;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.username);

System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagarram

8954

\*/

// example-4.1 serialization and De-serialization with transient-static keyword

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

import java.io.\*;

class Car implements Serializable

{

String username = "sagarram";

transient static int pwd = 8954;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.username);

System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagarram

8954

\*/

final v/s transient:

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

final variable will be participated in the serialization directly by value hence declaring the final variable as transient has no effect.

// example-5 serialization and De-serialization with final keyword

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

import java.io.\*;

class Car implements Serializable

{

String username = "sagarram";

final int pwd = 8954;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.username);

System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagarram

8954

\*/

// example-5.1 serialization and De-serialization with transient- final keyword

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

import java.io.\*;

class Car implements Serializable

{

String username = "sagarram";

transient final int pwd = 8954;

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

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

System.out.println(c1.username);

System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

sagarram

8954

\*/

conclusion:

-----------

int i = 10; // 10 20

int j = 20;

int i = 10; // 10 0

transient int j = 20;

transient static int i = 10; // 10 0

transient int j = 20;

transient int i = 10; // 0 20

transient final int j = 20;

transient static int i = 10; // 10 20

transient final int j = 20;

Can we serialize multiple objects ?

--> yes, we can serialize multiple objects in java, but during de-serialization we must fallow the order of serialization.

ie, in which order we have serialized in the same order we must de-serialize.

// example-6 serialization and De-serialization of multiple objects..!

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

import java.io.\*;

class Car implements Serializable

{

}

class Bike implements Serializable

{

}

class Truck implements Serializable

{

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

Bike b = new Bike();

Truck t = new Truck();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

oos.writeObject(b);

oos.writeObject(t);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

Bike b1 = (Bike)ois.readObject();

Truck t1 = (Truck)ois.readObject();

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

//System.out.println(c1.username);

//System.out.println(c1.pwd);

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

\*/

dia:

what if incase if we dont know the order of serialization?

// example-7 De-serialization without knowing the order of serialization..!

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

import java.io.\*;

class Car implements Serializable

{

}

class Bike implements Serializable

{

}

class Truck implements Serializable

{

}

class SerializeDemo

{

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

{

// creation of object

Car c =new Car();

Bike b = new Bike();

Truck t = new Truck();

// code for serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(c);

oos.writeObject(b);

oos.writeObject(t);

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

// code for De-serialization

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

ObjectInputStream ois = new ObjectInputStream(fis);

Object o = ois.readObject();

if (o instanceof Car)

{

Car c1 = (Car)o;

}

else if (o instanceof Bike)

{

Bike b1 = (Bike)o;

}

else if (o instanceof Truck)

{

Truck t1 = (Truck)o;

}

else

{

System.out.println("There is no such objects..!");

}

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

}

}

/\*

output:

-------

serialization is successfully completed...!

De-serialization is successfully completed...!

\*/