**Externalization**

* **Introduction:**

In serialization everything taken care by JVM and programmer doesn’t have any control.

In Serialization it is always possible to save entire object to the file. And it is not possible to save part of the object, which may create performance problems.

To overcome this problem, we should go for externalization.

The main advantage of externalization over serialization is everything taken care by programmer and JVM doesn’t have any control.

Based on our requirement, we can save either total object or part of the object, which improves performance of the system.

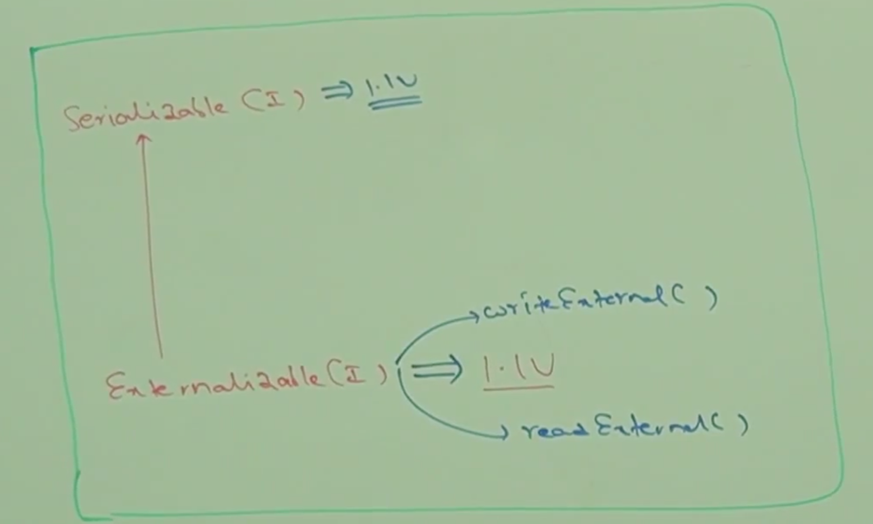
To provide externalizable ability for any Java object compulsory the corresponding class should implement Externalizable interface.

Externalizable interface has two methods:

writeExternal();

readExternal();

Externalizable is the child interface of Serializable.



* **Example:**

public void writeExternal(ObjectOutput out) throws IOException;

This method will be executed automatically at the time of serialization.

Within this method we have to write code to save required variables to the file.

public void readExternal(ObjectInput in)throws IOException, CNFE;

This method will be executed automatically at the time of deserialization.

Within this method we have to write code to read required variables from the file and assign to current object.

But strictly speaking at the time of deserialization JVM will create a separate new object by executing “public no argument constructor”. On that object JVM will call readExternal();

Hence every Externalizable implemented class should compulsory contain public no argument constructor, otherwise we will runtime exception saying: InvalidClassException

* **Example:**

import java.io.\*;

public class ExternalizableDemo implements Externalizable{

String s;

int i;

int j;

public ExternalizableDemo(){

System.out.println(“public no-arg constructor”);

}

public ExternalizableDemo(String s, int i, int j){

this.s = s;

this.i = i;

this.j = j;

}

public void writeExternal(ObjectOutput out)throws IOException{

out.writeObject(s);

out.writeInt(i);

}

public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException{

s = (String) in.readObject();

i = in.readInt();

}

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

ExternalizableDemo t1 = new ExternalizableDemo(“durga”, 10, 20);

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

ObjectOutputStream oos = new ObjectOuputStream(fos);

oos.writeObject(t1);

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

ObjectInputStream ois = new ObjectInputStream(fis);

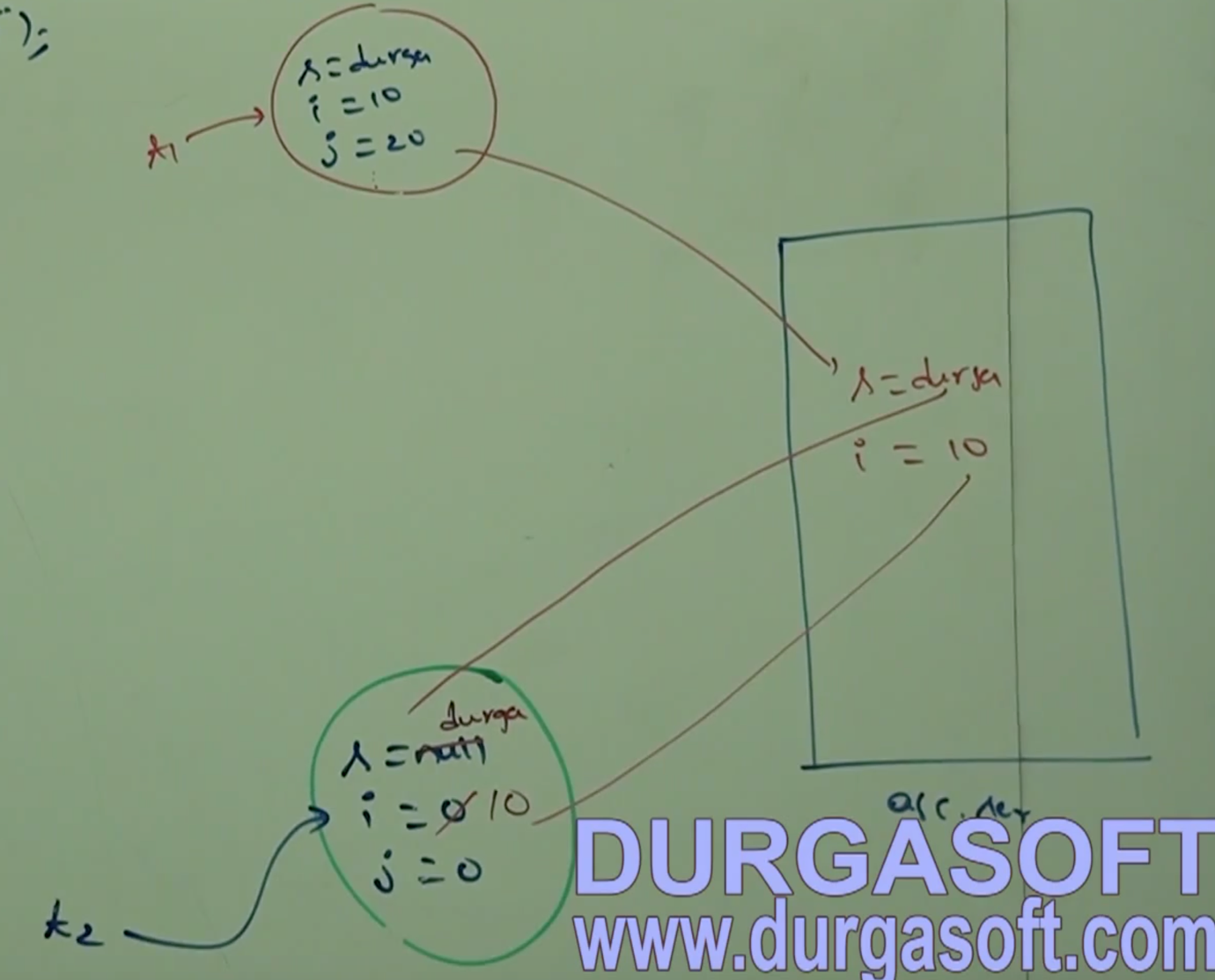
ExternalizableDemo t2 = (ExternalizableDemo)ois.readObject();

System.out.println(t2.s+”…”+t2.i+”…”+t2.j);

}

}

* **Diagrammatic Representation:**



If the class implements Serializable then total object will be saved to the file. In this case output is

durga…10…20

If the class implements Externalizable then only required variable will be saved to the file. In this case output is

public no-arg constructor

durga…10…0

* **Note:**

In Serialization transient keyword will play role. But in externalization transient keyword won’t play any role, of course “transient” keyword not required in externalization.

* **Differences between Serialization and Externalization:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Serialization** | **Externalization** |
| 1 | It is meant for default Serialization | It is meant for customized serialization. |
| 2 | Here, everything takes care by JVM and programmer doesn’t have any control. | Here, everything takes care by programmer and JVM doesn’t have any control. |
| 3 | In this case it always possible to save total object to the file and it is not possible to save part of the object. | Based on requirement we can save either total object or part of the object. |
| 4 | Relatively performance is low. | Relatively performance is high. |
| 5 | It is the best choice if we want to save total object to the file. | It is the best the choice if we want to save part of the object to the file. |
| 6 | Serializable doesn’t contain any methods and it is a marker interface. | Externalizable interface contains two methods, writeExternal() and readExternal() and hence it is not a marker interface. |
| 7 | Serializable implemented class not required to contain “public no-argument constructor” | Externalizable implemented class should compulsory contain “public no-argument constructor” otherwise we will get runtime exception saying: Invalid class exception. |
| 8 | transient keyword will play role in serialization | transient keyword won’t play any role in externalization, of course it won’t be required. |