**Serialization – Part-03 – Customized Serialization – Part01**

* **Customized Serialization:**

During default serialization there may a chance of loss of information because of transient keyword.

Example:

import java.io.\*;

class Account implements Serializable{

String username = “durga”;

String password = “anushka";

}

class CustSerializeDemo{

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

Account a1 = new Account();

System.out.println(userName+”…”+password);

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject();

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

ObjectInputStream ois = new ObjectInputStream(fis);

Account a2 = (Account) ois.readObject();

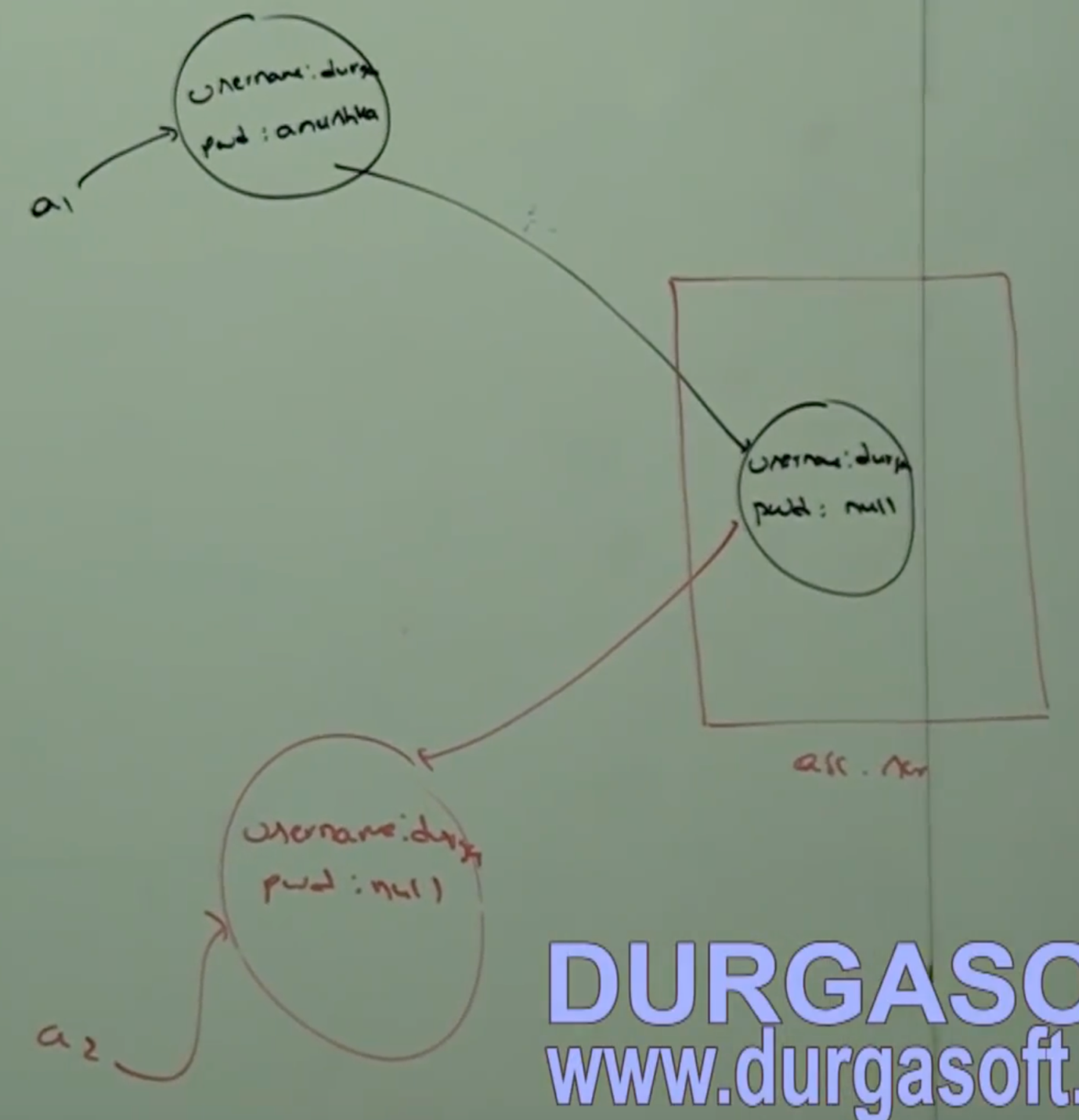
System.out.println(userName+”…”+password”);

}

}

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 variable as transient.

Hence, during default serialization there may be a chance of loss of information because of transient keyword. To recover this loss of information we should go for customized serialization.

****

* **Customized Serialization:**

We can implement customized serialization by using the following two methods.

private void writeObject(ObjectOutputStream os)throws Exception;

This method will be executed automatically at the time of serialization. Hence, at the time of serialization if we want to perform any activity we have to define that in this method only.

private void readObject(ObjectInputStream is) throws Exception;

This method will be executed automatically at the time of deserialization. Hence, at the time of deserialization if we want to perform any activity we have to define that in this method only.

Note:

The above methods are callback methods because these are executed automatically by the JVM.

While performing which object serialization we have to do extra work. In the corresponding class we have to define above methods. For example, while performing Account object serialization, if we require to do extra work, in the Account class we have to define above methods.

Steps to be performed:

Serialization:

Encrypt the information and save/serialize it manually.

Deserialization:

Read the encrypted information.

Decrypt it.

Assign it to the variable.

Note: Recall the story of sending an amount packed/hidden inside the mango box.

* **Customized Serialization Example:**

import java.io.\*;

class Account implements Serializable{

String username=”durga”;

transient String pwd = “anushka”;

private void writeObject(ObjectOutputStream oos) throws Exception{

oos.defaultWriteObjec(); // default serialization

String epwd = “123”+pwd;

oos.writeObject(epwd);

}

private void readObject(ObjectInputStream ois) throws Exception{

ois.defaultReadObject(); // default deserialization

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

pwd = epwd.substring(3);

}

}

class CustSerializeDemo{

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

Account a1 = new Account();

System.out.println(a1.username+”…”+a1.pwd);

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

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

ObjectInputStream ois = new ObjectInputStream(fis);

Account a2 = (Account) ois.readObject();

System.out.println(a2.username+”…”+a2.pwd);

}

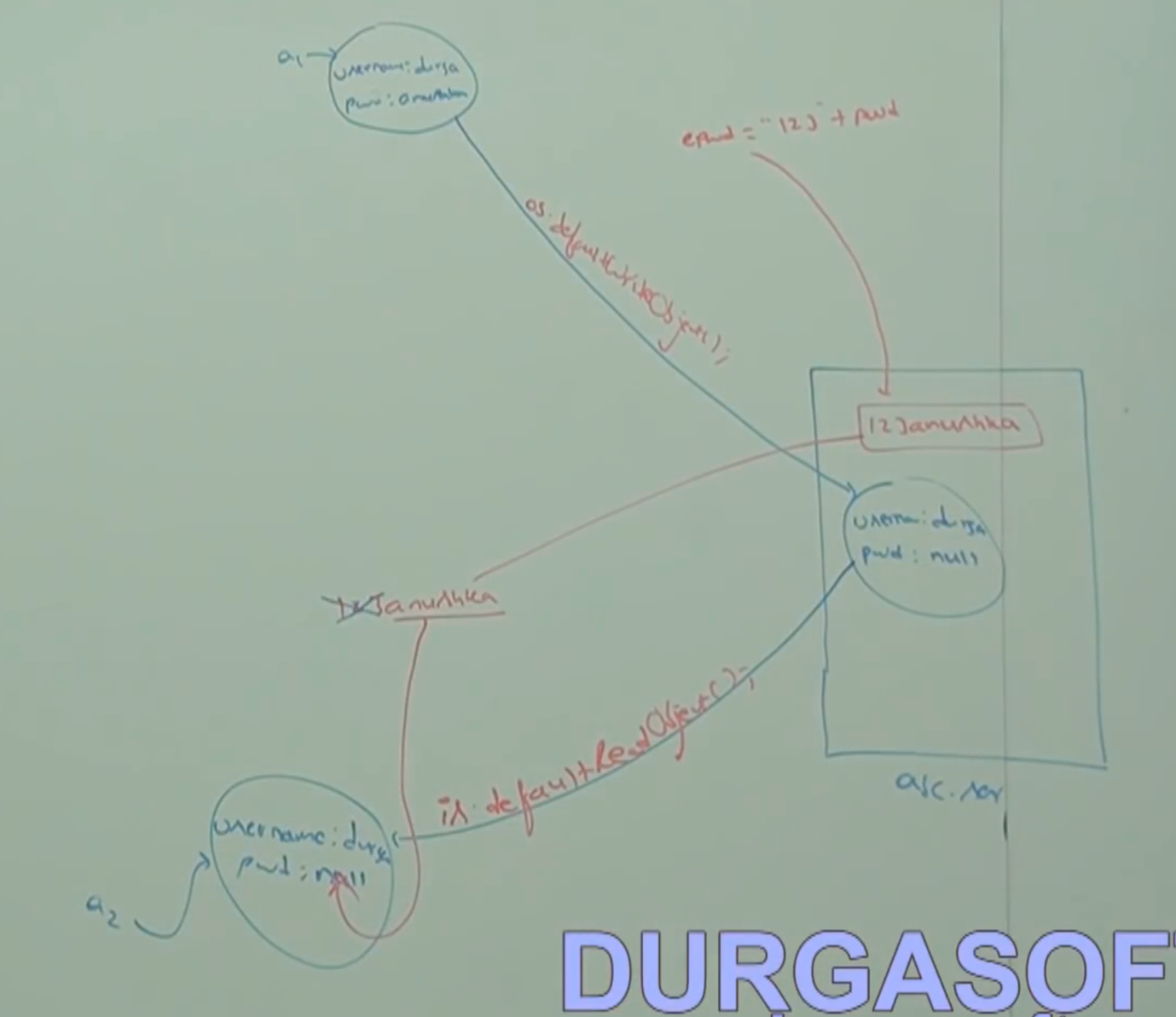
}

**Output:**

durga…anushka

durga…anushka

* **Diagrammatic Representation:**

****

In the above program before serialization and after serialization Account object can provide proper username and password.

Note:

Programmers can’t call private methods directly from outside of the class. But JVM can call private methods directly from outside of the class.

The methods access should be private, if you give other access specifier the customized serialization won’t work.