**Serialization – Part-04 – Customized Serialization – Part02**

* **Serialization with respect to Inheritance:**

**Case\_01:**

Even though child class doesn’t implement Serializable, we can serialize child class object if parent class implements Serializable interface. That is serializable nature is inherited from parent to child. Hence, if parent is serializable then by default every child is serializable.

Example:

import java.io.\*;

class Animal implements Serializable{

int I = 10;

}

class Dog extends Animal{

int j = 20;

}

class SerializeDemo{

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

Dog d1 = new Dog();

System.out.println(d1.i+”…”+d1.j);

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d1);

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

ObjectInputStream ois = new ObjectInputStream(fis);

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

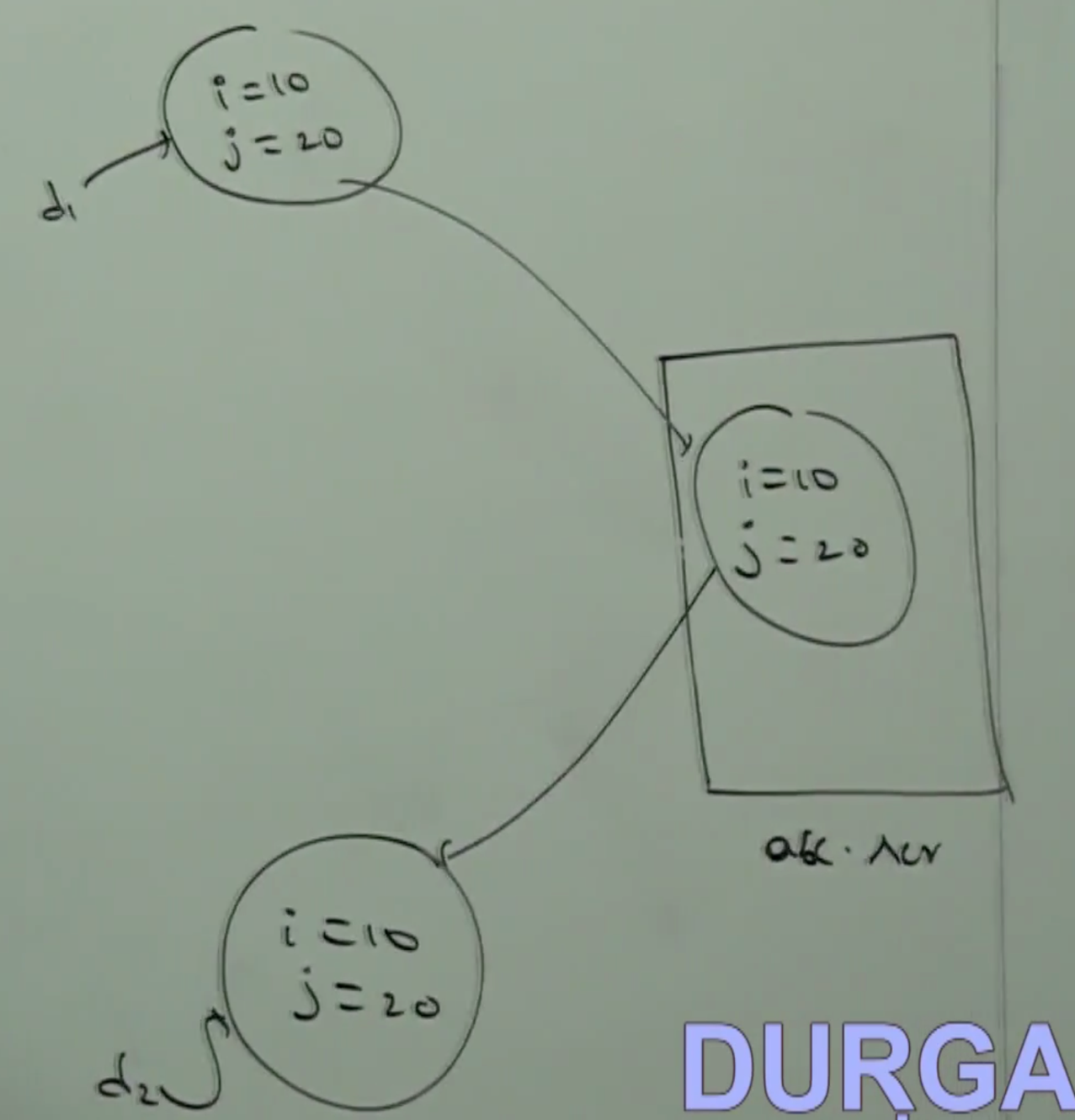
System.out.println(d2.i+”…”+d2.j);

}

}

In the above example, even though doesn’t implement serializable we can serialize dog object, because its parent Animal class implements serializable.

**Diagrammatic Representation:**



Note:

Object class doesn’t implement Serializable interface.

* **Case\_02:**

Even though parent class doesn’t implement Serializable, we can serialize child class object, if child class implements Serializable interface. That is, to serialize child class object, parent class need not be serializable.

At the time of Serialization, JVM will check is any variable inheriting from non-serializable parent or not? If any variable inheriting from non-serializable parent, then JVM ignores original value and save default value to the file.

At the time of deserialization JVM will check is any parent class non-serializable or not? If any parent class is non-serializable then JVM will execute “instance control flow” in every non-serializable parent and share its instance variable to the current object.

While executing instance control flow of non-serializable parent, JVM will always call no-argument constructor. Hence, every non-serializable class should compulsory contain no-argument constructor. It may be default constructor generated by compiler or customized constructor explicitly provided by the programmer. Otherwise we will get Runtime Exception saying:

InvalidClassException

Example:

import java.io.\*;

class Animal{

int I = 10;

Animal(){

System.out.println(“Animal constructor called”);

}

}

class Dog extends Animal implements Serializable{

int j = 20;

Dog(){

System.out.println(“Dog constructor called”);

}

}

class SerializeDemo{

public static void main(String[] args){

Dog d1 = new Dog();

d1.i = 888;

d1.j = 999;

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d1);

System.out.println(“Deserialization started”);

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

ObjectInputStream ois= new ObjectInputStream(fis);

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

System.out.println(d2.i+”…”+d2.j);

}

}

Output:

Animal constructor called

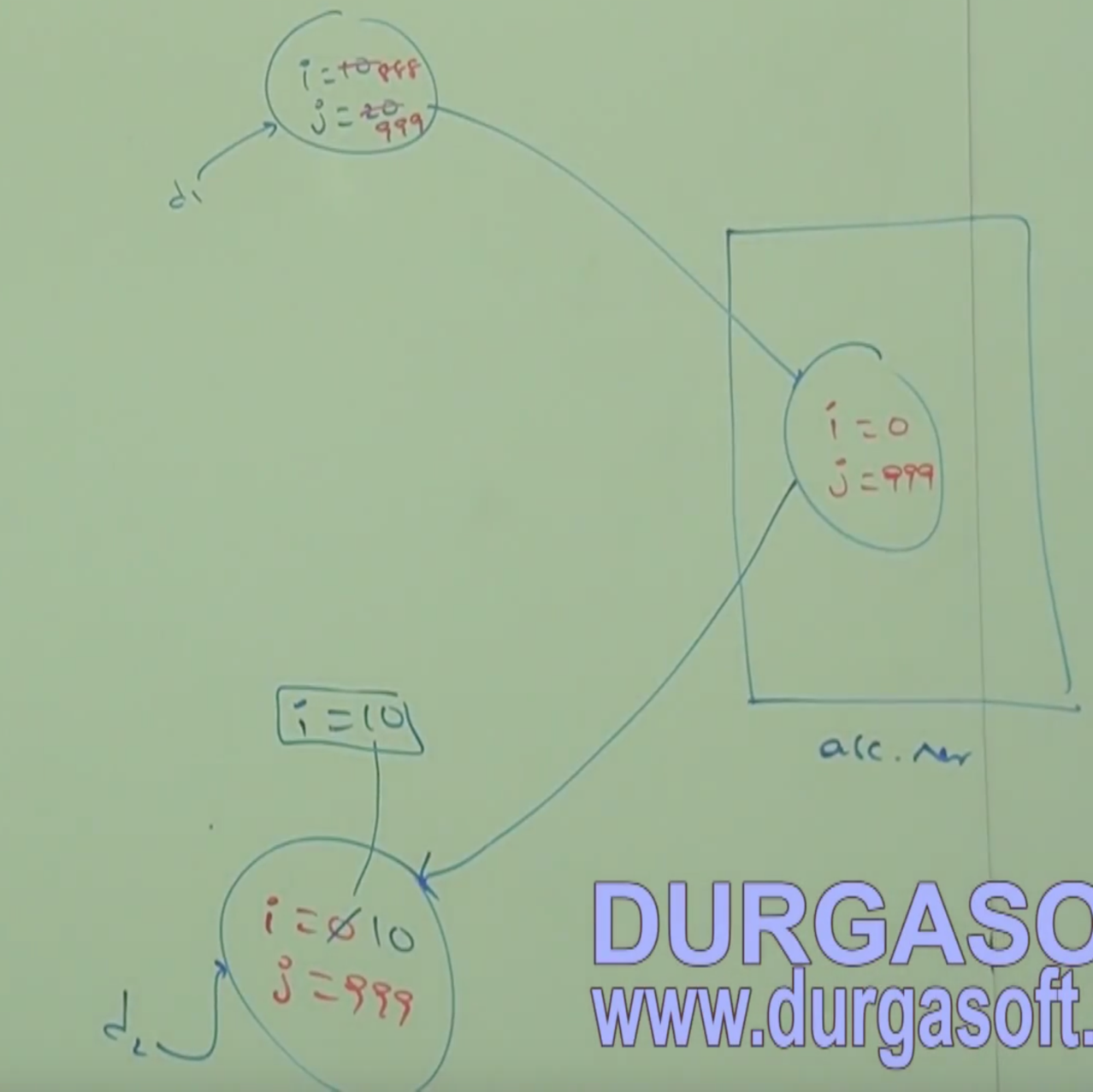
Dog constructor called

Deserialization Started

Animal constructor called

10…999

* **Diagrammatic Representation:**



* **Code for Runtime Exception: InvalidClassException:**

import java.io.\*;

class Animal{

int I = 10;

Animal(int i){

System.out.println(“Animal constructor called”);

}

}

class Dog extends Animal implements Serializable{

int j = 20;

Dog(){

super(10);

System.out.println(“Dog constructor called”);

}

}

class SerializeDemo{

public static void main(String[] args){

Dog d1 = new Dog();

d1.i = 888;

d1.j = 999;

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d1);

System.out.println(“Deserialization started”);

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

ObjectInputStream ois= new ObjectInputStream(fis);

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

System.out.println(d2.i+”…”+d2.j);

}

}

Output: Animal constructor called

Dog constructor called

Deserialization started

RE: InvalidClassException