**Serialization – Part-01**

* **Serialization:**

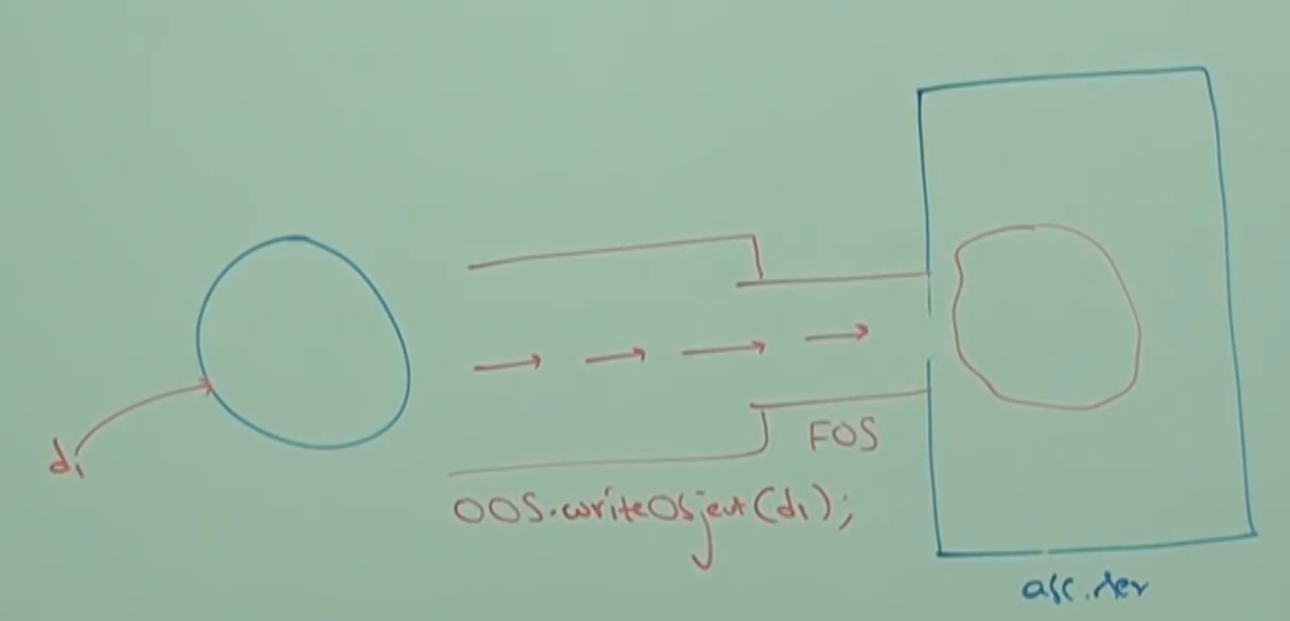
1. Introduction
2. Object Graphs in Serialization
3. Customized Serialization
4. Serialization with respect to Inheritance
5. Externalization.
6. serialVersionUID

* **Introduction:**

Serialization:

The process of writing a state of an object to a file is called serialization. But strictly speaking it is the process of converting an object from Java supported form into either file supported form or network supported form.

By using FileOutputStream and ObjectOutputStream classes we can achieve/implement serialization.

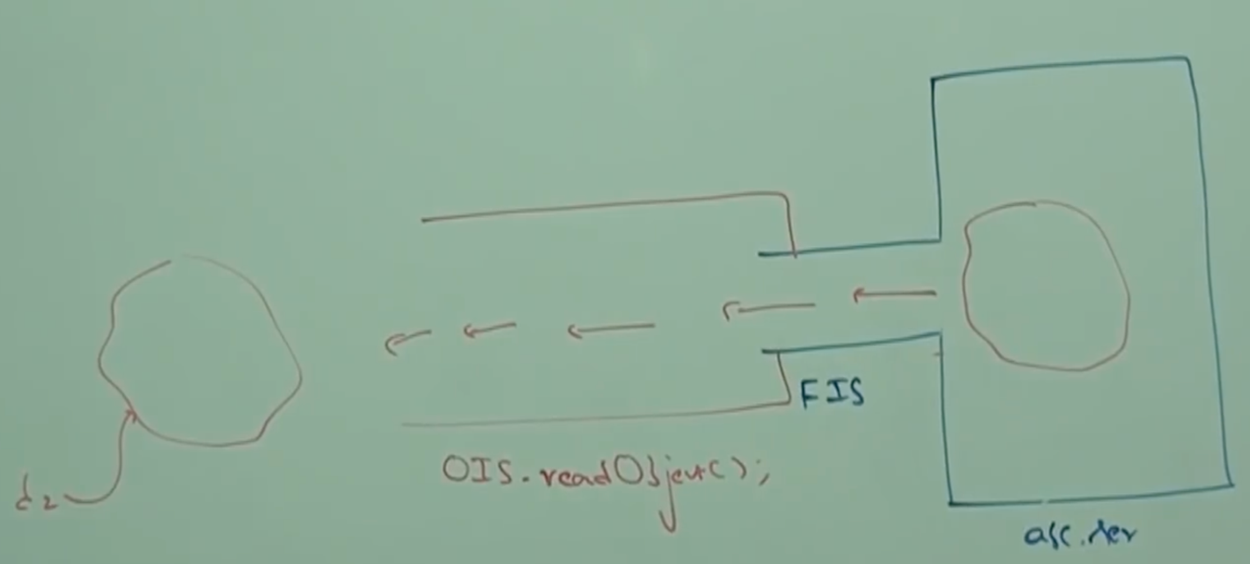


Note: Recall about sending a big balloon from Hyderabad to Bangalore.

Deserialization:

The process of reading state of an object from the file is called deserialization. But strictly speaking it is the process of converting an Object from either file supported form or network supported form into Java supported form.

By using FileInputStream and ObjectInputStream we can implement deserialization.



* **Example:**

import java.io.Serializable;

class Dog implements Serializable{

int i = 10;

int j = 20;

}

class SerializeDemo{

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

Dog d1 = new Dog();

//serialization

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

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(dog);

//deserialization

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

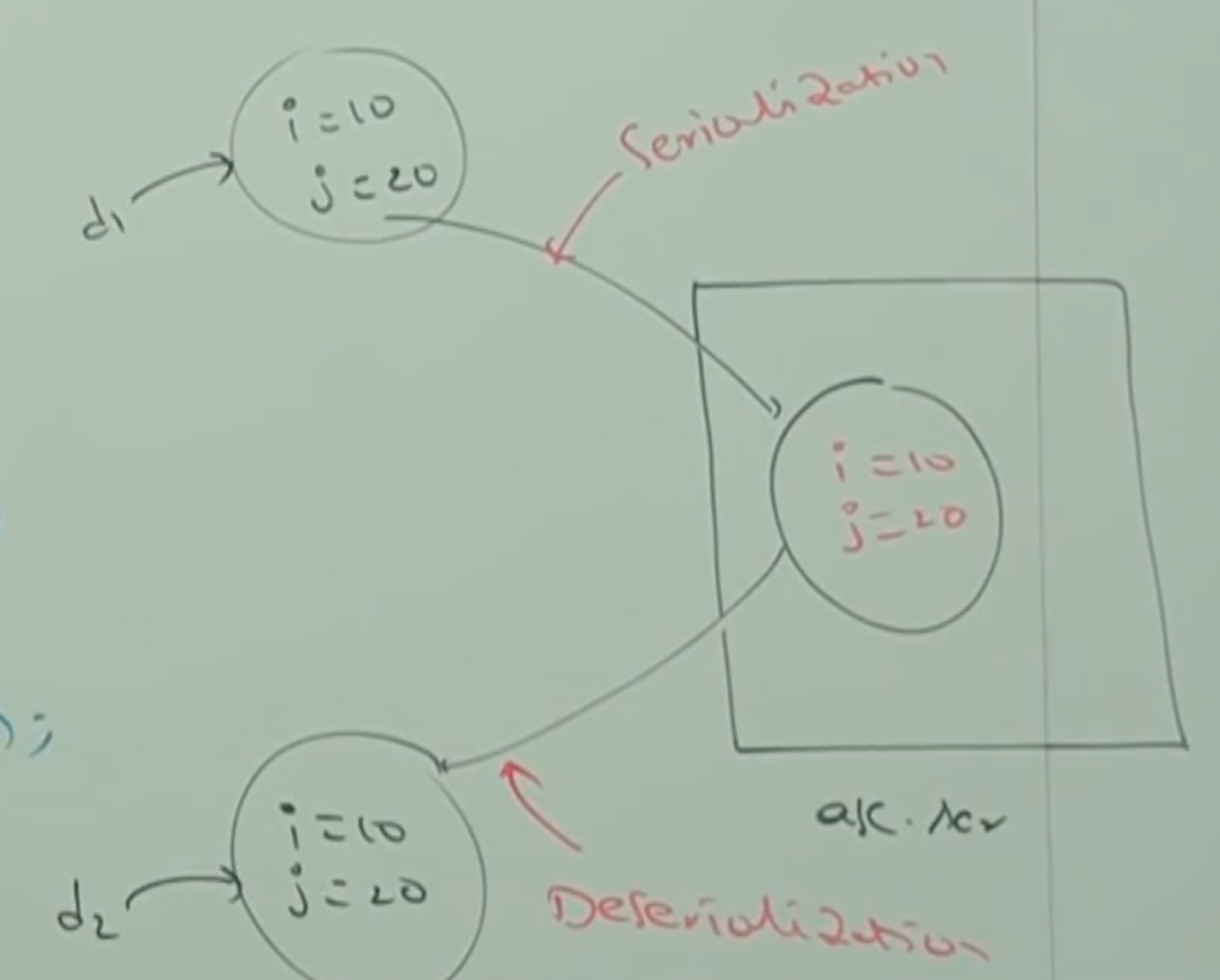
ObjectInputStream ois = new ObjectInputStream(fis);

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

System.out.println(d2.i+”…”+d2.j);

}

}



Note:

1. We can serialize only Serializable objects.
2. An Object is said to be serializable if and only if the corresponding class implements Serializable interface.
3. Serializable interface present in java.io package, and it doesn’t contain any methods. It is a marker interface.
4. If we are trying to serialize non-serializable object then we will get runtime exception saying NotSerilizableException

* **transient keyword:**

transient modifier applicable only for variables but not for methods and classes.

At the time of serialization, if we don’t want to save the value of a particular variable to meet security constraints then we should declare that variable as transient.

While performing serialization JVM ignores the original value of transient variable and saves default value to the file. Hence, transient means not to serialize.

* **transient vs static:**

static variable is not part of object state, and hence it won’t participate in serialization. Due to this declaring static variable as transient there is no use/impact.

* **final vs transient:**

final variables will be participated in serialization directly by the value. Hence, declaring a final variable as transient there is no impact.

* **Summary:**

|  |  |
| --- | --- |
| **Declaration** | **Output** |
| int i = 10  int j = 20 | 10…20 |
| transient int i = 10  int j = 20 | 0…20 |
| transient static int i = 10  transient int j = 20 | 10…0 |
| transient int i = 10  transient final int j = 20 | 0…20 |
| transient static int i = 10  transient final int j = 20 | 10…20 |