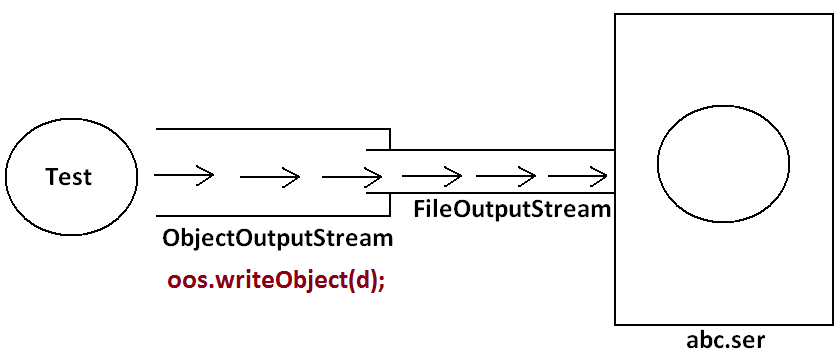
## **Serialization: (1.1 v)**

1. The process of saving (or) writing state of an object to a file is called serialization
2. but strictly speaking it is the process of converting an object from java supported form to either network supported form (or) file supported form.
3. By using FileOutputStream and ObjectOutputStream classes we can achieve serialization process.
4. **Ex**: big ballon

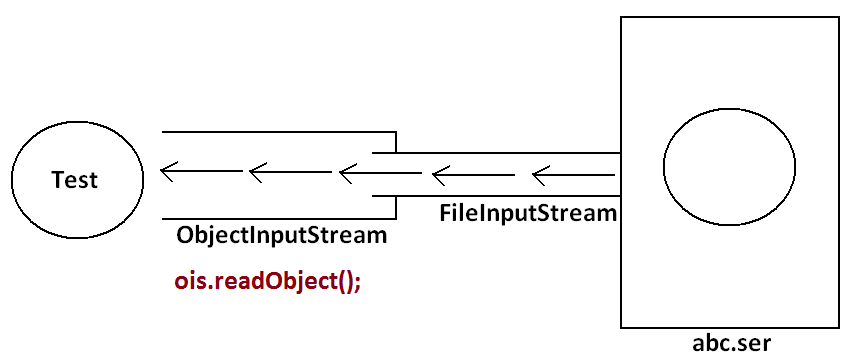
**Diagram:**

****

## **De-Serialization:**

1. The process of reading state of an object from a file is called DeSerialization
2. but strictly speaking it is the process of converting an object from file supported form (or) network supported form to java supported form.
3. By using FileInputStream and ObjectInputStream classes we can achieve DeSerialization.

**Diagram:**

****

**Steps to perform Serialization:**

**— --------------------------------------**

**1. Prepare Serializable Object:**

**In general, all Java objects are not Serializable, only the objects which are implementing java.io.Serializable marker interface are eligible for Serialization.**

**java.io.Serializable marker interface will provide some abilities to the object at runtime, that is, making it eligible for Object Serialization.**

**public class Employee implements Serializable{**

**}**

**Employee emp1 = new Employee();**

**2. Make read target file by Using FileOutputStream:**

**FileOutputStream fos = new FileOutputStream(“E:/documents/emp.txt”);**

**3. Create ObjectOutputStream with FileOutputStream: ObjectOutputStream oos = new ObjectOutputStream(fos);**

**4. Write Serializable Object to ObjectOutputStream: oos.writeObject(emp1);**

**When we write Serializable object to ObjectoutputStream , ObjectOutputStream performs Serialization over the provided object, ObjectoutputStream will send the serialized data to the FIleOutputStream, where FileOutputStream will send that Serialized data to the specified File.**

**Steps to perform Deserialization**

**—-----------------------------------**

**1. Create FileInputStream for getting Deserialized data from flat file. FileInputStream fis = new FileInputStream(“E:/documents/emp.txt”);**

**2. Create ObjectInputStream with FileInputStream: ObjectinputStream ois = new ObjectinputStream(fis);**

**Note: Here ObjectinputStream will get Serialized data from flat file, Objectinp[utStream will perform the real deserialization and generate an Object on the basis of data.**

**3. Read Deserialized Object from ObjectInputSteam public Object readObject()**

**EX: Employee emp2 = (Employee)ois.readObject();**

**Example program**

**--------------------**

**package com.codegnan.serilization;**

**import java.io.FileInputStream;**

**import java.io.FileOutputStream;**

**import java.io.ObjectInputStream;**

**import java.io.ObjectOutputStream;**

**import java.io.Serializable;**

**// Define a Serializable class Employee**

**class Employee implements Serializable {**

**private static final long *serialVersionUID* = 1L; // Ensure compatibility during serialization**

**private int empId;**

**private String empName;**

**private float empSalary;**

**private String empLocation;**

**// Constructor**

**public Employee(int empId, String empName, float empSalary, String empLocation) {**

**this.empId = empId;**

**this.empName = empName;**

**this.empSalary = empSalary;**

**this.empLocation = empLocation;**

**}**

**// Method to display employee details**

**public void getEmpDetails() {**

**System.*out*.println("Employee ID: " + empId);**

**System.*out*.println("Employee Name: " + empName);**

**System.*out*.println("Employee Salary: " + empSalary);**

**System.*out*.println("Employee Location: " + empLocation);**

**}**

**}**

**public class Main {**

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

**// Serialization: Writing an Employee object to a file**

**FileOutputStream fileOutputStream = new FileOutputStream("E:/emp.txt");**

**ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);**

**// Creating an Employee object**

**Employee employee1 = new Employee(111, "malli", 50000.0f, "Hyd");**

**System.*out*.println("Employee Details before Serialization:");**

**employee1.getEmpDetails();**

**// Writing the Employee object to file**

**objectOutputStream.writeObject(employee1);**

**System.*out*.println("Serialized Employee Object: " + employee1);**

**System.*out*.println();**

**// Close output streams after writing**

**objectOutputStream.close();**

**fileOutputStream.close();**

**// Deserialization: Reading an Employee object from a file**

**FileInputStream fileInputStream = new FileInputStream("E:/emp.txt");**

**ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);**

**// Reading the Employee object from file**

**Employee employee2 = (Employee) objectInputStream.readObject();**

**System.*out*.println("Employee Details after Deserialization:");**

**employee2.getEmpDetails();**

**System.*out*.println("Deserialized Employee Object: " + employee2);**

**// Close input streams after reading**

**objectInputStream.close();**

**fileInputStream.close();**

**}**

**}**

**o/p**

**Employee Details before Serialization:**

**Employee ID: 111**

**Employee Name: malli**

**Employee Salary: 50000.0**

**Employee Location: Hyd**

**Serialized Employee Object: com.codegnan.serilization.Employee@56cbfb61**

**Employee Details after Deserialization:**

**Employee ID: 111**

**Employee Name: malli**

**Employee Salary: 50000.0**

**Employee Location: Hyd**

**Deserialized Employee Object: com.codegnan.serilization.Employee@69d9c55**

# **Transient**

**Transient modifier:**

**1. Transient is the modifier applicable only for variables but not for methods and classes.**

**2. While serialization if we don't want to serialize the value of a particular variable to meet the security constraints then we should declare that variable with transient modifier.**

**3. During the serialization jvm ignores the original value of the transient variable and save default value that is transient means "not to serialize".**

**4. Static variables are not part of object state hence serialization concept is not applicable for static variables duo to this declaring a static variable as transient there is no use.**

**import java.io.FileInputStream;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**import java.io.ObjectInputStream;**

**import java.io.ObjectOutputStream;**

**import java.io.Serializable;**

**//Define the ATMCard class**

**public class ATMCard implements Serializable {**

**private static final long *serialVersionUID* = 1L;**

**private String cardNumber; // This will be serialized**

**private transient String pin; // This will not be serialized**

**private transient String expiryDate; // This will not be serialized**

**private transient String securityCode; // This will not be serialized**

**private String cardHolderName; // This will be serialized**

**// Constructor**

**public ATMCard(String cardNumber, String pin, String expiryDate, String securityCode, String cardHolderName) {**

**this.cardNumber = cardNumber;**

**this.pin = pin;**

**this.expiryDate = expiryDate;**

**this.securityCode = securityCode;**

**this.cardHolderName = cardHolderName;**

**}**

**// Getters and setters**

**public String getCardNumber() {**

**return cardNumber;**

**}**

**public void setCardNumber(String cardNumber) {**

**this.cardNumber = cardNumber;**

**}**

**public String getPin() {**

**return pin;**

**}**

**public void setPin(String pin) {**

**this.pin = pin;**

**}**

**public String getExpiryDate() {**

**return expiryDate;**

**}**

**public void setExpiryDate(String expiryDate) {**

**this.expiryDate = expiryDate;**

**}**

**public String getSecurityCode() {**

**return securityCode;**

**}**

**public void setSecurityCode(String securityCode) {**

**this.securityCode = securityCode;**

**}**

**public String getCardHolderName() {**

**return cardHolderName;**

**}**

**public void setCardHolderName(String cardHolderName) {**

**this.cardHolderName = cardHolderName;**

**}**

**@Override**

**public String toString() {**

**return "ATMCard [cardNumber=" + cardNumber + ", pin=" + pin + ", expiryDate=" + expiryDate + ", securityCode="**

**+ securityCode + ", cardHolderName=" + cardHolderName + "]";**

**}**

**// Main method for testing serialization and deserialization**

**public static void main(String[] args) {**

**ATMCard originalCard = new ATMCard("1234567812345678", "1234", "12/25", "567", "John Doe");**

**// Serialize the object**

**try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("atmcard.ser"))) {**

**out.writeObject(originalCard);**

**System.*out*.println("Object serialized: " + originalCard);**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**// Deserialize the object**

**ATMCard deserializedCard = null;**

**try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("atmcard.ser"))) {**

**deserializedCard = (ATMCard) in.readObject();**

**System.*out*.println("Object deserialized: " + deserializedCard);**

**} catch (IOException | ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**}**

**}**