**What Does “Name-Based Lookup” Mean in Enum Serialization?**

When you serialize an enum in Java, it **is serialized**, but **not like a normal object** with all its fields. Instead, Java uses the **name of the enum constant** (like "INSTANCE") to identify it.

**How Enum Serialization Works**

When Java serializes an enum:

* It does **not** serialize its fields (even if you had some).
* It serializes **only the name of the constant** (like "INSTANCE").
* During **deserialization**, the JVM **does not create a new object** — it uses that name to look up the **existing enum constant** in the enum class.

**This ensures the same enum instance is always returned**, preserving the singleton.

**Example:**

import java.io.\*;

enum MySingleton implements Serializable {

INSTANCE;

int value = 42;

}

public class Test {

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

MySingleton s1 = MySingleton.INSTANCE;

s1.value = 99;

// Serialize

ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("singleton.ser"));

oos.writeObject(s1);

oos.close();

// Reset value

s1.value = 0;

// Deserialize

ObjectInputStream ois = new ObjectInputStream(new FileInputStream("singleton.ser"));

MySingleton s2 = (MySingleton) ois.readObject();

ois.close();

System.out.println(s1 == s2); // true

System.out.println(s2.value); // 0 and not 99

}

}

**Observations:**

* s1 == s2 is true: **Same instance** — singleton preserved.
* But value is 0: Because only the name "INSTANCE" was used; no field values were serialized.

When we say "enum serialization uses name-based lookup", it means:

* Java **does serialize enums** — but only **their name**.
* During deserialization, Java uses that name to **look up the constant** in memory.
* It doesn’t create a **new object**, hence preserving singleton behavior.
* But it also doesn’t persist **field state** unless you handle it manually.

**What If You Want to Preserve Field Values?**

You can **manually handle field serialization** using writeObject and readObject methods:

enum MySingleton implements Serializable {

INSTANCE;

int value = 42;

private void writeObject(ObjectOutputStream oos) throws IOException {

oos.defaultWriteObject();

oos.writeInt(value); // Manually save value

}

private void readObject(ObjectInputStream ois) throws IOException, ClassNotFoundException {

ois.defaultReadObject();

value = ois.readInt(); // Restore value

}

}

Now, the value field will be preserved during serialization/deserialization.

**Summary:**

Enum serialization does **not persist** instance fields unless you **manually serialize them** using writeObject/readObject. By default, only the **enum constant's name** is used to fetch the singleton instance during deserialization.