Transient keyword with static variable in Serialization

November 3, 2016 SJ Serialization 0

In this article, we will discuss what happens to ***static data member*** when ***transient*** keyword or modifier applied during ***Serialization*** process

This is one of the ***tricky questions*** asked in ***Java interview***

Serialization process

During serialization process i.e.; saving the state of an Object to File, only instance variables will be participated and persisted to file storage

What happens in serialization process, if we declare static data member with transient keyword?

SEARCH TUTORIALS

SEARCH ...

SUBSCRIBE VIA EMAIL

Join 194 other subscribers

Email Address

SUBSCRIBE

POPULAR ARTICLES

JDBC: An example to connect MS Access database in Java 8

- The answer is very simple, only instance variables will be participated in Serialization process
- static variables doesn't participate in Serialization process and also static variables aren't part of Object's state
- So, by declaring static data member with transient doesn't have any impact
- There won't be any compile-time or run-time error

Transient keyword

- Transient keyword or modifier is applicable only for variables
- We can stop persisting specific variable, by declaring transient keyword
- During serialization, JVM ignores the original value of transient variable and saves default value to file
- **Examples:** Customer SSN or password need not to be stored. Hence, it's a good practice to declare those variables as transient
- So whenever we encounter ***transient*** keyword, it means that *not to serialize*

static variable

- A variable declared with static modifier is known as static variable
- Alternatively it is referred as class variable as it belongs to class rather to any specific instance
- Static variable shared among every instance like for example organization name of the employee
- It should be used whenever there is common property for all objects of that class
- Static variables can be accessed directly by class name or interface name instead of creating an instance and then accessing
- Static variables can be accessed from static and non-static methods/blocks using class name or interface name
- Memory allocation for static variables happens at the time of class loading by JVM

Spring JDBC: An example on JdbcTemplate using Annotation
Java JDBC: An example to connect MS Access database
Oracle OSB 12c: Service Callout and Routing Table example
Oracle OSB 12c: Hello World service with both Business and Proxy Service

Demo example on Transient keyword with static data member

For objects to participate in serialization & de-serialization process, corresponding class should implement *java.io.Serializable* interface

Exception: otherwise, run time exception will be thrown stating *NotSerializableException*

Step 1: Create POJO which implements *java.io.Serializable* interface

In Customer POJO, there are 4 member variables with *customerSSN* declared with transient keyword and also 1 static data member called customerCount initialized to 2

transient *customerSSN* –> default value will be saved instead original value
transient static *customerCount* –> won't participate in serialization

Customer.java

```java
package in.bench.resources.serialization;

import java.io.Serializable;

public class Customer implements Serializable {

    // static data member
    static int customerCount = 2;

    // member variables
    int customerId;
    String customerName;
    int customerAge;
    transient int customerSSN;


    // 4-arg parametrized constructor
    public Customer(int customerId, String cust
            int customerAge, int customerSSN) {
        super();
        this.customerId = customerId;
        this.customerName = customerName;
        this.customerAge = customerAge;
        this.customerAge = customerAge;
    }

    // overriding toString() method
    @Override
    public String toString() {
        return "Customer [customerId=" + custom
                + ", customerName=" + customerN
                + ", customerAge=" + customerAg
                + ", customerSSN=" + customerSS
                + ", customerCount=" + customer
    }
}
```

Step 2: Main program to demonstrate serialization/de-serialization

To Serialize: any Object, we can use *ObjectOutputStream* & *FileOutputStream* to write/save to the file (in binary format)

To De-Serialize: any Object, we can use *ObjectInputStream* & *FileInputStream* to read/restore from file (which is in binary format) into Java heap memory

TransientWithStaticDemo.java

```
package in.bench.resources.serialization;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class TransientWithStaticDemo {

    public static void main(String[] args) {

        // create an customer instance using 4-
        Customer serializeCustomer =
                new Customer(103, "AK", 21, 112

        // creating output stream variables
        FileOutputStream fos = null;
        ObjectOutputStream oos = null;

        // creating input stream variables
        FileInputStream fis = null;
        ObjectInputStream ois = null;

        // creating customer object reference
        // to hold values after de-serializatio
        Customer deSerializeCustomer = null;

        try {
            // for writing or saving binary dat
            fos = new FileOutputStream("Custome

            // converting java-object to binary
            oos = new ObjectOutputStream(fos);

            // writing or saving customer objec
            oos.writeObject(serializeCustomer);
            oos.flush();
            oos.close();

            System.out.println("Serialization s
                    + " object saved to Custome

            // reading binary data
            fis = new FileInputStream("Customer

            // converting binary-data to java-o
            ois = new ObjectInputStream(fis);

            // reading object's value and casti
            deSerializeCustomer = (Customer) oi
            ois.close();

            System.out.println("De-Serializatio
                    + " object de-serialized fr
        }
        catch (FileNotFoundException fnfex) {
            fnfex.printStackTrace();
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
        }
        catch (ClassNotFoundException ccex) {
            ccex.printStackTrace();
        }

        // printing customer object to console
        System.out.println("Printing customer v
                + "de-serialized object... \n"
    }
}
```

Output:

```
Serialization success: Customer object saved t

De-Serialization success: Customer object de-ser
from Customer.ser file

Printing customer values from de-serialized obje
Customer [customerId=103, customerName=AK, custc
customerCount=2]
```

Explanation:

During Serialization process,

- In above Customer POJO, customerSSN declared as transient so therefore this is ignored by JVM
- Only Object's state is persisted to file (i.e.; only instance variables)
- Static data member aren't part of Object's state, so this won't be considered
- When we de-serialize, all instance variables without transient keyword will be restored
- But static data member doesn't participated in serialization neither its gets persisted nor restored back from file

References:

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html
https://docs.oracle.com/javase/7/docs/platform/serialization/spec/serial-arch.html
https://docs.oracle.com/javase/7/docs/api/java/io/ObjectOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/ObjectInputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html
http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html#jls-8.3.1.3

Read Also:

- Java Serialization and De-Serialization Tutorial Index
- Serialization and De-Serialization in Java
- Serializable interface
- Transient keyword with Serialization in Java
- Transient keyword with final variable in Serialization
- Serializing a variable with transient modifier or keyword
- Order of Serialization and De-Serialization

- Serialization with Aggregation
- Serialization with Inheritance
- Externalizable interface with example
- Serializable v/s Externalizable
- Importance of SerialVersionUID in Serialization
- Singleton Design pattern with Serialization
- How to stop Serialization in Java
- How to construct a singleton class in a multi-threaded environment in Java
- How to serialize and de-serialize ArrayList in Java

Happy Coding !!
Happy Learning !!

« Transient keyword with final variable in Serialization

Transient keyword with Serialization in Java »

Related Posts:

1. **Transient keyword with Serialization in Java**
2. **Transient keyword with final variable in Serialization**
3. **Serializing a variable with transient modifier or keyword**
4. **Externalizable interface with example**

DESERIALIZATION JAVA SERIALIZABLE
SERIALIZATION TRANSIENT

0 Comments BenchResources.Net Ava

Recommend Share Sort by Best

Start the discussion…

Be the first to comment.

ALSO ON **BENCHRESOURCES.NET**

Apache CXF JAX-WS: Web Service using Top-Down

1 comment • 2 years ago

basanta hota — Any one can help me out same example i tried , even i am not pass any

Singleton design pattern – restricting all 4 ways of Object

3 comments • 2 years ago

Anil Nivargi — Nice explanation Thanks.......Singleton Design Pattern in Java with examples

Jersey 2.x web service using both (JSON + XML) example

2 comments • 2 years ago

BenchResources.Net — Murugesan,With the above setup I never get "bookType"

Spring Application using BeanFactory &

1 comment • 2 years ago

opensourcefeeder — Please include navigation links to next tutorial in sequence.I

Subscribe Add Disqus to your siteAdd DisqusAdd