Table of Content

Java serialization concept and Example Part II

Dec 28, 2013 • by Gautam • Add Comment

This entry is part 29 of 33 in the series Core Java Course

In first part we have discussed about what is serialization and how to do the same in Java. In this article we will discuss some advance topics covering Serialization.

In this article we will use the same codes base we have provided in previous article.

Use of serialVersionUID

You must have seen a variable named "***serialVersionUID***" have been used in source code. There is a specific reason behind using this variable.

serialVersionUID is a version number associated to each serializable class by serialization runtime. This version number is used during deserialization process to verify that the sender and receiver of a serialized object have loaded class for that object which is compatible with respect to serialization.

- Defining a ***serialVersionUID*** field in serializable class is **not mandatory**.
- If a serializable class have explicit ***serialVersionUID*** then this field should be of type ***long*** *and must be static and final.*
- If there is no ***serialVersionUID*** field defined explicitly then serialization runtime will calculate default value for that class. Which can vary based on compiler implementation. Hence it is advisable to

define ***serialVersionUID.***

- It is advised to use private access modifier for ***serialVersionUID.***
- Array classes cannot declare an explicit serialVersionUID, so they always have the default computed value, but the requirement for matching serialVersionUID values is waived for array classes.
- If there is a difference between serialVersionUID of loaded reciever class and corresponding sender class then **InvalidClassException** will be thrown.

Use of Transient

We can save the state of an object using Serializable. But what if i don't want to save state of a field? In this case transient modifier can be used like below. Transient fields state will not be saved while serialization process and default value will be assigned to same variable while de-serialization.

Changing the Employee class with transient variable.

```
1
2  package com.jbt;
3
4  import java.io.Serializable;
5
6  public class Employee implements Serializable
7  {
8     public String firstName;
9     /*
10     * Here transient modifier is used for lastName variable.
11     * This variable's state will not be saved while serialzation.
12     * While De-Serialization process default value will be provide.
13     * null in this case.
14     */
15    transient public String lastName;
16    private static final long serialVersionUID = 54622236001;
17 }
18
```

If you execute the same class(SerializaitonClass & DeserializationClass) output will be different then previous code.

```
1
2  Deserializing Employee...
3  First Name of Employee: Vivekanand
4  Last Name of Employee: null
5
```

As you can see above last name is coming as null because state of that variable was not saved while serialization process.

Class Hierarchy and Serializable

Here i will discuss about the effect of Serializable interface on Class hierarchy. If a class has implemented Serializable interface then state of this class can be saved. But if same class extend another class which didn't implement Serializable interface then Super class's state will not be saved.

To see the difference we will update original Employee class. Now this class will extend another class *superEmployee* . This super class will not implement Serializable interface.

```
1
2  package com.jbt;
3
4  import java.io.Serializable;
5
6  public class Employee extends superEmployee implements Serializable {
7  public String firstName;
8  private static final long serialVersionUID = 54622236001;
9  }
10
11 class superEmployee {
12 public String lastName;
13 }
14
```

If you execute "SerializaitonClass" and "DeserializationClass" one after another then output would be like below

```
1
2  Deserializing Employee...
3  First Name of Employee: Vivekanand
4  Last Name of Employee: null
5
```

Transient vs Static variable

I have written a complete article on this. Please visit here.

Custom Serialization Process

<TO_DO>

Series Navigation

<< Java serialization concept and Example

Transient vs Static variable java >>

Java serialization concept and Example

Transient vs Static variable java

You may also like

Core Java

jdk vs jre vs jvm

by Gautam

Core Java

Singleton design pattern in java

by Gautam

Core Java

Java Collection Hashmap tutorial

by J Singh

Core Java

What is the use of serialVersionUID

by J Singh

Core Java

Core java interview questions

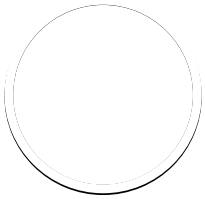
by J Singh

Core Java

Java equals Method vs == Operator

by J Singh

About the author



Gautam

Gautam is Software Professional who works for an MNC Company. And he loves to share his knowledge about Java & related technology here on this blog by writing some article in hope some one will be benefited by this.

View all posts

Leave a Comment

Name *

Email *

Website

Comment

Post Comment

Most Popular

Developing a Spring 3 Framework MVC application step...

Java Basics:Getting Started with Java

Access Modifiers in Java

Operators in java programming language

Variables in Java

Featured Posts

An introduction to Python 3

Connection Pooling with Hibernate 4

Multi-tenancy with Hibernate 4

Concurrency control with Hibernate 4

Auditing with Hibernate 4

Disclaimer

Oracle & Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. Java Beginners Tutorial is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

The Examples & Tutorial provided here are for learning purpose only. I do not warrant the correctness of its content. The risk from using it lies entirely with the user.