


[Sitemap](#) [Privacy Policy](#) [Terms of Use](#)
[About Us](#) [Write for JBT](#)

JavaBeginnersTutorial


	Core Java	Spring	Hibernate	Adv Java	Oracle	Cheatsheet	Code Base	Example Code
				Contact Me				




Java Beginners Tutorial


google.com/+Javabeginnertutorial

Tutorial for Java Developers



Follow



+ 308



Java Beginners Tutorial


Liked

5.2k likes

Table of Content

[Java Basics:Getting Started with Java](#)
[jdk vs jre vs jvm](#)
[Java Class & Object Tutorial for beginners](#)
[Constructors in Java](#)

Constructors in Java

Variables in Java

Local Variable in Java

Instance Variable in Java

Java Reference Variable

```
public static void main(string args[])
```

Explanation

Write Hello World Application Using Eclipse

Access Modifiers in Java

Non Access Modifiers in Java

Operators in java programming language

Java Statements tutorial for Beginners

Different ways to create an object in Java

this keyword in Java

Java Static Keyword

Java Interface

Overloading

Java Method Override

Java Exceptions Tutorial

Collection in Java

Java Collection Hashmap tutorial

Inner Class

Inheritance

String Builder

Java String Tutorial

[Java serialization concept and Example](#)

[Java serialization concept and Example Part II](#)

[Transient vs Static variable java](#)

[What is the use of serialVersionUID](#)

[Java Thread Tutorial](#)

[Java Array Tutorial](#)

What is the use of serialVersionUID

Jan 9, 2014 • by J Singh • 2 Comments

This entry is part 31 of 33 in the series [Core Java Course](#)

Here I will discuss the importance of the variable serialVersionUID which are used in Serializable classes.

Below is an example that will make you understand the exact use of the variable.

Example Code

Employee.java

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

SerializaitonClass.java (This class will be used to serialize)

```
1
2 package com.jbt;
3
4 import java.io.FileOutputStream;
5 import java.io.IOException;
```

```

6 import java.io.ObjectOutputStream;
7
8 public class SerializaitonClass {
9
10 public static void main(String[] args) {
11     Employee emp = new Employee();
12     emp.firstName = "Vivekanand";
13     emp.lastName = "Gautam";
14
15     try {
16         FileOutputStream fileOut = new FileOutputStream("./employee.txt");
17         ObjectOutputStream out = new ObjectOutputStream(fileOut);
18         out.writeObject(emp);
19         out.close();
20         fileOut.close();
21         System.out.printf("Serialized data is saved in ./employee.txt file");
22     } catch (IOException i) {
23         i.printStackTrace();
24     }
25 }
26 }
27

```

DeserializationClass.java (This class will be used to deserialize)

```

1
2 package com.jbt;
3
4 import java.io.*;
5
6 public class DeserializationClass {
7     public static void main(String[] args) {
8         Employee emp = null;
9         try {
10             FileInputStream fileIn = new FileInputStream("./employee.txt");
11             ObjectInputStream in = new ObjectInputStream(fileIn);
12             emp = (Employee) in.readObject();
13             in.close();
14             fileIn.close();
15         } catch (IOException i) {
16             i.printStackTrace();
17             return;
18         } catch (ClassNotFoundException c) {
19             System.out.println("Employee class not found");
20             c.printStackTrace();
21             return;
22         }
23         System.out.println("Deserializing Employee...");
24         System.out.println("First Name of Employee: " + emp.firstName);
25         System.out.println("Last Name of Employee: " + emp.lastName);
26     }
27 }
28

```

Now execute “SerializationClass.java” and then “DeserializationClass.java”. It will first create a file with Employee object’s state and then while de-serialization it creates object from the same file. Output will be something like below.

```
2 Serialized data is saved in ./employee.txt file
3
```

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

Now let's try and remove "serialVersionUID" variable from Employee.java file and again run "SerializationClass.java" file. It will create "employee.txt" file again with the object's state. Now let's add a new variable in Employee class suppose String Address.

Employee.java

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

Now run "DeserializationClass.java" and see the output.Booom

```
1
2 java.io.InvalidClassException: com.jbt.Employee; local class incompatible: stream classdesc s
3 at java.io.ObjectStreamClass.initNonProxy(Unknown Source)
4 at java.io.ObjectInputStream.readNonProxyDesc(Unknown Source)
5 at java.io.ObjectInputStream.readClassDesc(Unknown Source)
6 at java.io.ObjectInputStream.readOrdinaryObject(Unknown Source)
7 at java.io.ObjectInputStream.readObject0(Unknown Source)
8 at java.io.ObjectInputStream.readObject(Unknown Source)
9 at com.jbt.DeserializationClass.main(DeserializationClass.java:11)
10
```

It will throw an incompatible exception. Because the given class Employee.java was changed in between serialization and de-serialization process. Hence the system failed to identify that it is still the same class. To make our system understand that it is the same class you have to make use of **serialVersionUID** variable inside class.

You can try follow the above steps but keep **serialVersionUID** intact and see the output. De-serialization process will work without any issue.

Bullet Points

- Defining a ***serialVersionUID*** field in serializable class is **not mandatory**.
- If a serializable class has an 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. The value can vary based on compiler implementation. Hence it is advisable to define ***serialVersionUID***.
- It is advised to use private access modifier for ***serialVersionUID***.
- Different class can have same ***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 receiver class and corresponding sender class then ***InvalidClassException*** will be thrown.
- You should use different ***serialVersionUID*** for different version of same class if you want to forbid serialization of new class with old version of same class.

@SuppressWarnings("serial")

If you do not provide serialVersionUID in a class which is supposed to be serialised then compiler will give warning messages about the same. If you want to override this warning you can use given annotation. Once used, compiler will stop complaining about the missing serialVersionUID.

Series Navigation

<< [Transient vs Static variable java](#)

[Java Thread Tutorial](#) >>



[Transient vs Static variable java](#)



[Java Collection Hashmap tutorial](#)

You may also like

jdk vs jre vs jvm
by Gautam



Singleton design pattern in java
by Gautam



Core Java
Java serialization concept and
Example Part II
by Gautam



Core Java
Java serialization concept and
Example
by Gautam

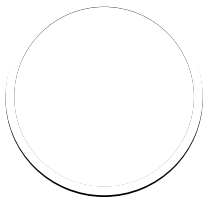


Core Java
Core java interview questions
by J Singh



Core Java
Java equals Method vs ==
Operator
by J Singh

About the author

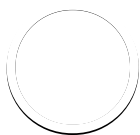


J Singh

Main brain behind this blog/QnA site. She has written all these article as per her personal experience. She has published it mainly to help beginners. These articles might have some problems. So bear with her and let her know if you find any problem in any of the article.

[View all posts](#)

2 Comments

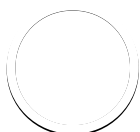


Aditya

Sep 16, 2014 at 11:46 am

Very Nice Xplanation.
Its worth reading.

[Reply](#)



chandra

Jun 8, 2015 at 3:40 am

Nice website ...

Reply

Leave a Comment

Name *

Email *

Website

Comment

Post Comment

Most Popular

Featured Posts

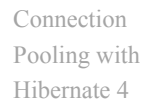
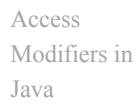
Disclaimer

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.



Java Basics:Getting Started with Java



Concurrency control with Hibernate 4

