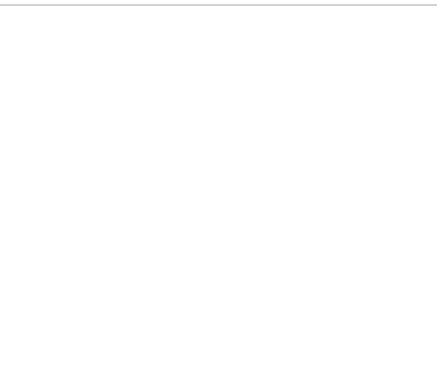



[Sitemap](#) [Privacy Policy](#) [Terms of Use](#)
[About Us](#) [Write for JBT](#)


# JavaBeginnersTutorial

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




**Java Beginners Tutorial**  
[google.com/+Javabeginnertutorial](https://plus.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](#)

---

# Transient vs Static variable java

Dec 29, 2013 • by Gautam • 23 Comments

This entry is part 30 of 33 in the series [Core Java Course](#)

Here i will show you what is the difference between Static and Transient. In any way these two things are completely different with different scope but people use to ask me this question every time so i am mentioning it here. For previous Serialization articles click [here\(Pat I\)](#) and [here\(Part II\)](#).

## Static Variable

Static variables belong to a class and not to any individual instance. The concept of serialization is concerned with the object's current state. Only data associated with a specific instance of a class is serialized, therefore static member fields are ignored during serialization.

## Transient Variable

While serialization if you don't want to save state of a variable. You have to mark that variable as Transient. Environment will know that this variable should be ignored and will not save the value of same.

- **Note\*:** Even concept is completely different and reason behind not saving is different still people get confused about the existence of both. As in both the case variables value will not get saved.

## Difference between these two

Source code : Pay attention to every single code change.

### Employee.java

```
1
2 package com.jbt;
3
4 import java.io.Serializable;
5
6 public class Employee extends superEmployee {
7     public String firstName;
8     private static final long serialVersionUID = 54622236001;
9 }
10
11 class superEmployee implements Serializable{
12     public String lastName;
13     static String companyName;
14     transient String address;
15     static transient String companyCEO;
16 }
17
```

### SerializaitonClass.java

```
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         emp.companyName = "JBT";
15         //Below part needs to be removed in case address field is made final
16         emp.address = "MUM";
17         emp.companyCEO = "ME CEO";
18
19         try {
20             FileOutputStream fileOut = new FileOutputStream("./employee.txt");
21             ObjectOutputStream out = new ObjectOutputStream(fileOut);
22             out.writeObject(emp);
23             out.close();
24             fileOut.close();
25             System.out.printf("Serialized data is saved in ./employee.txt file");
26         } catch (IOException i) {
27             i.printStackTrace();
28         }
29     }
30 }
31
```

### DeserializationClass.java

```
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    System.out.println("Company Name: "+emp.companyName);
27    System.out.println("Company CEO: "+emp.companyCEO);
28    System.out.println("Company Address: "+emp.address);
29    }
30    }
31

```

First Execute “SerializaitonClass” and you will get below output.

```

1
2 Serialized data is saved in ./employee.txt file
3

```

Second execute “DeserializationClass” and you will get below output

```

1
2 Deserializing Employee...
3 First Name of Employee: Vivekanand
4 Last Name of Employee: Gautam
5 Company Name: null
6 Company CEO: null
7 Company Address: null
8

```

As you can see from output only last name value has been saved. Neither Static nor Transient variables value has been saved.

Now i will change the code a little bit and see what happens.

### Employee.java

```

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

```

```
11 class superEmployee implements Serializable {
12 public String lastName;
13 /*
14  * Here i am providing the value of company name, companyCEO and address
15  * while defining these variables.
16  */
17 static String companyName = "TATA";
18 transient String address = "DEL";
19 static transient String companyCEO = "Jayshree";
20 }
21
```

Again execute the same code and see the output

```
1
2 Deserializing Employee...
3 First Name of Employee: Vivekanand
4 Last Name of Employee: Gautam
5 Company Name: TATA
6 Company CEO: Jayshree
7 Company Address: null
8
```

See the output very carefully. Here value of companyName, companyCEO has been saved but not of address. Also check the saved value of these variable.

Company Name: TATA

Company CEO: Jayshree

In both the case value stored here are taken from class(Employee class) and not from Object(emp object). Also companyCEO variable's value is saved even when it is transient. Because static modifier change the behavior of this variable.

## Bullet Point

1. Static variables value can be stored while serializing if the same is provided while initialization.
2. If variable is defined as Static and Transient both, then static modifier will govern the behavior of variable and not Transient.

## Final modifier effect on Serialization

To see the effect of Final modifier i am again changing the code of Employee class.

**Employee.java**

```
1
2 package com.jbt;
```

```

3
4 import java.io.Serializable;
5
6 public class Employee extends superEmployee {
7     public String firstName;
8     private static final long serialVersionUID = 54622236001;
9 }
10
11 class superEmployee implements Serializable {
12     public String lastName;
13     /*
14      * Here i am providing the value of company name,companyCEO and address
15      * while defining these variables.
16      * I am making address as final here
17      */
18     static String companyName = "TATA";
19     transient final String address = "DEL";
20     static transient String companyCEO = "Jayshree";
21 }
22

```

Again execute the code and see the difference. Output for above code would be

```

1
2 Deserializing Employee...
3 First Name of Employee: Vivekanand
4 Last Name of Employee: Gautam
5 Company Name: TATA
6 Company CEO: Jayshree
7 Company Address: DEL
8

```

As you can see now address fields is also saved while serialization because it is now Final.

## Interface and Final

I have seen a scenario when you can serialize variables inside an Interface which is not serialized.

### Employee.java

```

1
2 package com.jbt;
3
4 import java.io.Serializable;
5
6 public class Employee extends superEmployee implements variableConstant{
7     public String firstName;
8     private static final long serialVersionUID = 54622236001;
9 }
10
11 class superEmployee implements Serializable {
12     public String lastName;
13     /*
14      * Here i am providing the value of company name,companyCEO and address
15      * while defining these variables.
16      * I am making address as final here
17      */
18     static String companyName = "TATA";
19     transient final String address = "DEL";
20     static transient String companyCEO = "Jayshree";
21 }
22

```

```

23 interface variableConstant {
24     public static final String education = "MCA";
25 }
26 }
27

```

### DeserializationClass.java

```

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         System.out.println("Company Name: "+emp.companyName);
27         System.out.println("Company CEO: "+emp.companyCEO);
28         System.out.println("Company Address: "+emp.address);
29         System.out.println("Education: "+emp.education);
30     }
31 }
32

```

Execute above code and see the output.

```

1
2 Deserializing Employee...
3 First Name of Employee: Vivekanand
4 Last Name of Employee: Gautam
5 Company Name: TATA
6 Company CEO: Jayshree
7 Company Address: DEL
8 Education: MCA
9

```

Here you can see that education's value is saved. This value is part of an Interface. But as this is constant hence it is saved while serialization.

I think i have covered all possible scenarios. Do let me know in case any particular scenario is not covered in this article. Feel free to say me hi, if you like this article. And yes i didn't qualify the interview. 😊

Series Navigation



&lt;&lt; Java serialization concept and Example Part II

What is the use of serialVersionUID &gt;&gt;



Java serialization concept and  
Example Part II



What is the use of serialVersionUID

---

## 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

Java serialization concept and  
Example  
by Gautam



### Core Java

Core java interview questions  
by J Singh



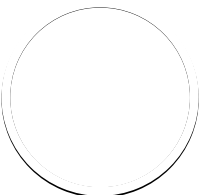
### Core Java

Java equals Method vs ==  
Operator  
by J Singh

---

## About the author

---



### Gautam

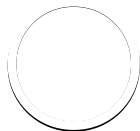
Gautam is Software Professional who works for an MNC Company. And he loves to share his knowledge about Java & related technology here on this blog by writing some article in hope some one will be benefited by this.

[View all posts](#)

---

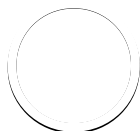
## 23 Comments

---

**ganesh**

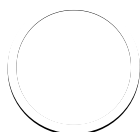
Jan 20, 2014 at 6:35 am

very nice.

[Reply](#)**Anu**

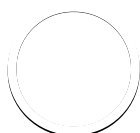
May 11, 2014 at 12:42 pm

thank u ...really useful for me

[Reply](#)**Mohita**

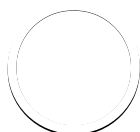
May 16, 2014 at 10:05 am

Great article!

[Reply](#)**Chaitanya V**

Jun 15, 2014 at 7:31 am

Super.... its a nice site...

[Reply](#)**Tirupathi Reddy**

Jul 4, 2014 at 5:58 am

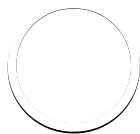
Hi ,

This website is very usefull to know exacting thing .

Regards,  
Tirupathi

Reply

---



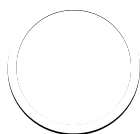
**sandeep**

Aug 4, 2014 at 4:57 pm

really wonderful explaine wid example

Reply

---



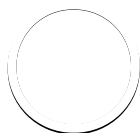
**Hieu Tran**

Aug 14, 2014 at 8:05 am

The static variable(education) aren't saved, because it is a static variable.  
In this case, the line `System.out.println("Education: "+emp.education);` printed out the value  
beavse education's value is loaded from class, not from serialized object

Reply

---



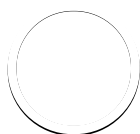
**sirisha**

Sep 18, 2014 at 5:12 pm

Really article is very helpful.  
Thanks a lot.

Reply

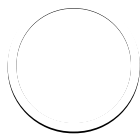
---



**Sarika**

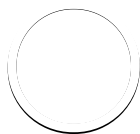
Nov 7, 2014 at 7:53 am

The static variable values are printed because they are accessed along with the class. Static  
variables can be accessed by `classname.staticvariable`.  
`companyname`, `companyCEO`, `education` are all static variable values and are printed because they  
called as `classname.variable`.

[Reply](#)**Ravi**

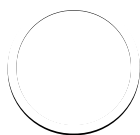
Nov 14, 2014 at 5:34 pm

Nice

[Reply](#)**Anurag**

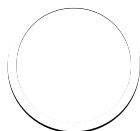
Jan 2, 2015 at 1:02 pm

Similarly final variables are also not serialized and they are picked from class as soon the class loads itself. You can verify the serialized value in a file and you will not found any final or static variable content there.

[Reply](#)**Siva Sankar**

Jan 12, 2015 at 6:14 am

Very good article.

[Reply](#)**tamkanat**

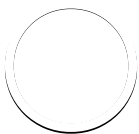
Jan 23, 2015 at 6:01 am

hi,

very nice information.thanks for the effort.

If u dont mind, can u tell y u dint qualified.

wat did they asked u in interview that u couldnt answer?

[Reply](#)**Vivekanand Gautam**

Jan 24, 2015 at 4:52 am

Hi Tamkant,

Thanks for your nice word. I am sorry but i don't remember the questions. After all these years i can say

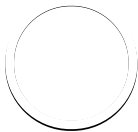
qualifying in an interview is not only dependent on your knowledge but also on Interviewers knowledge.

I have taken so many interviews my self. And you can not judge someone in 30 min.

Better we only concentrate on our knowledge and be confident. Someone will understand you.

And you will qualify the interview someday. 😊

Reply

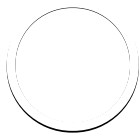


**venkataramareddy**

Mar 29, 2015 at 6:29 pm

very nice..10q

Reply



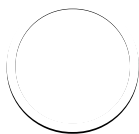
**Naveen Goel**

Feb 3, 2015 at 5:07 pm

Hiii

Do you think there might be some security leak of data while serialization because your data is freely moves into your class and also the outside of the class. So what's better approach to safe guard data is Inner class concept is good.

Reply



**Naveen Goel**

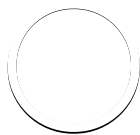
Feb 3, 2015 at 5:09 pm

Hi Vivekananda's,

In case of interface data is not serialized because in interfaces by default all fields are final and as you already explained final fields govern behavior of variables as static.

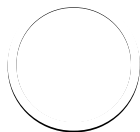
Reply

---

**Shailesh**

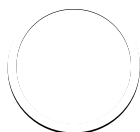
Mar 18, 2015 at 5:20 pm

Nice explanation...!!

[Reply](#)**Rafeeq Mullan**

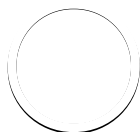
Apr 21, 2015 at 10:13 am

Very Nice information, thanks for sharing.

[Reply](#)**Shilpa Khare**

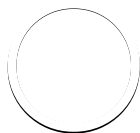
Jul 24, 2015 at 10:18 am

Nice explanation!

[Reply](#)**Bhaskar**

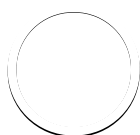
Aug 12, 2015 at 2:01 pm

Very well explained !!

[Reply](#)**Namita**

Aug 13, 2015 at 7:20 am

Well explained... Thanks

[Reply](#)**Balapoorna**

Aug 25, 2015 at 1:25 pm

Hi

Even if the variable inside the interface is not static or final, will it still retain the value of the variable

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.



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



An introduction  
to Python 3



Connection  
Pooling with  
Hibernate 4

Multi-tenancy  
with Hibernate 4

Concurrency  
control with  
Hibernate 4



Auditing with  
Hibernate 4