Javarevisited

Blog about Java programming language, FIX Protocol, Tibco Rendezvous and related Java technology stack.

Top 10 Java Serialization Interview Questions and Answers

What is Serialization in Java

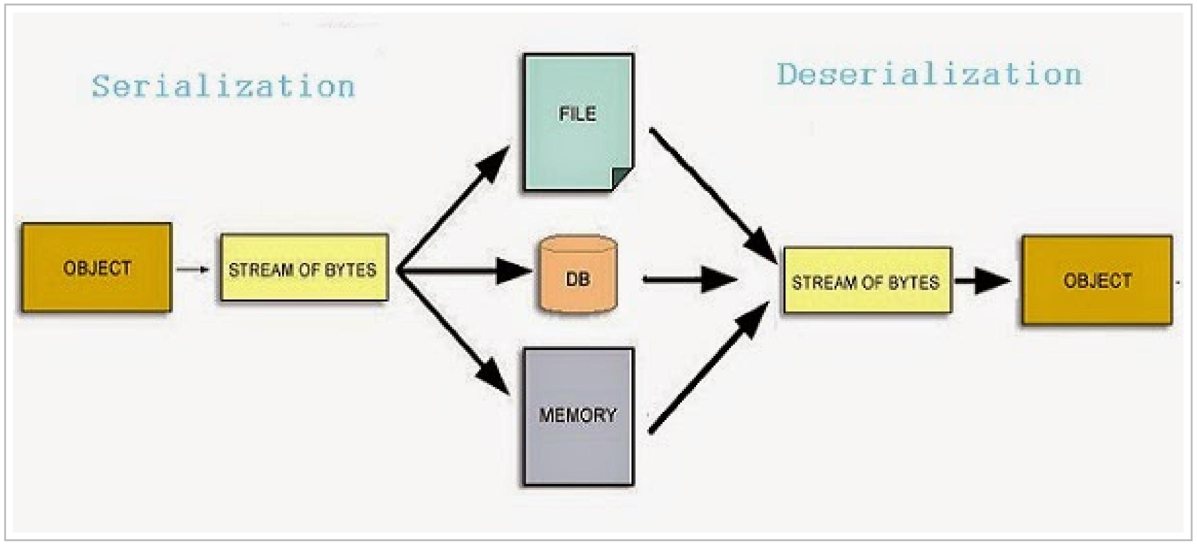
Java Serialization is one of important concept but it's been rarely used as persistence solution and developer mostly overlooked Java serialization API. As per my experience Java Serialization is quite an important topic in any core Java interview, In almost all the interview I have faced there is one or two Java serialization questions and I have seen interview where after few question on serialization candidate start feeling uncomfortable because of lack of experience in this area. They don't know How to serialize object in Java or they are not familiar with any Java Serialization example to explain, forget about questions like Difference between transient and volatile variable or Difference between Externalizable and Serializable in Java. In this article we will question from both beginner and advanced level, which can be equally beneficial to freshers, new comers and senior Java developers with some years of Java development experience.

Gmail for Work
Look more professional with custom Gmail from Google
Start free trial

It's time to go home
This festive season, enjoy unbelievably low fares starting from just ₹ 1352. Book your ticket on goIndiGo.in and get packing to home, sweet, home!

10 Interview questions on Serialization in Java

Most commercial project uses either database or memory mapped file or simply flat file for there persistence requirement and only few of them rely on serialization process in Java. Anyway this post is not a Java serialization tutorial or how to serialize object in java but about interview questions around serialization mechanism and Serialization API, Which is worth to have a look before going for any Java or J2EE interview and surprising yourself with some unknown contents. for those who are not familiar about java Serialization "Java serialization is the process which is used to serialize object in java by storing object's state into a file with extension .ser and recreating object's state from that file, this reverse process is called deserialization.



The Java Serialization API provides a standard mechanism for developers to handle object serialization using Serializable and Externalizable interface. By the way this article is in continuation of my previous article Top 20 design pattern interview questions, and 10 Interview questions on Singleton Pattern in Java So here we go.

What is Serialization in Java

Object Serialization in Java is a process used to convert Object into a binary format which can be persisted into disk or sent over network to any other running Java virtual machine; the reverse process of creating object from binary stream is called deserialization in Java. Java provides Serialization API for serializing and deserializing object which includes java.io.Serializable, java.io.Externalizable, ObjectInputStream and ObjectOutputStream etc. Java programmers are free to use default Serialization mechanism which Java uses based upon structure of class but they are also free to use there own custom binary format, which is often advised as Serialization best practice, Because serialized binary

Paytm
Reebok
Women's Footwear at Best Prices

Shop Now
*T&C Apply

Categories
programming (230)
coding (88)
interview questions (42)
jsp-servlet (33)
database (28)
linux (28)
Eclipse (19)

format becomes part of Class's exported API and it can potentially break Encapsulation in Java provided by private and package-private fields. This pretty much answer the question What is Serialization in Java.

How to make a Java class Serializable?

Making a class Serializable in Java is very easy, Your Java class just needs to implements `java.io.Serializable` interface and JVM will take care of serializing object in default format. Decision to making a Class Serializable should be taken concisely because though near term cost of making a Class Serializable is low, long term cost is substantial and it can potentially limit your ability to further modify and change its implementation because like any public API, serialized form of an object becomes part of public API and when you change structure of your class by implementing addition interface, adding or removing any field can potentially break default serialization, this can be minimized by using a custom binary format but still requires lot of effort to ensure backward compatibility. One example of How Serialization can put constraints on your ability to change class is SerialVersionUID. If you don't explicitly declare SerialVersionUID then JVM generates its based upon structure of class which depends upon interfaces a class implements and several other factors which is subject to change. Suppose you implement another interface than JVM will generate a different SerialVersionUID for new version of class files and when you try to load old object object serialized by old version of your program you will get `InvalidClassException`.

Question 1) What is the difference between Serializable and Externalizable interface in Java?

This is most frequently asked question in Java serialization interview. Here is my version Externalizable provides us `writeExternal()` and `readExternal()` method which gives us flexibility to control java serialization mechanism instead of relying on Java's default serialization. Correct implementation of `Externalizable` interface can improve performance of application drastically.

Followers

Question 2) How many methods Serializable has? If no method then what is the purpose of Serializable interface?

Serializable interface exists in java.io package and forms core of java serialization mechanism. It doesn't have any method and also called Marker Interface in Java. When your class implements `java.io.Serializable` interface it becomes Serializable in Java and gives compiler an indication that use Java Serialization mechanism to serialize this object.

Question 3) What is serialVersionUID? What would happen if you don't define this?

One of my favorite question interview question on Java serialization. SerialVersionUID is an ID which is stamped on object when it get serialized usually hashcode of object, you can use tool `serialver` to see `serialVersionUID` of a serialized object . SerialVersionUID is used for version control of object. you can specify serialVersionUID in your class file also. Consequence of not specifying serialVersionUID is that when you add or modify any field in class then already serialized class will not be able to recover because serialVersionUID generated for new class and for old serialized object will be different. Java serialization process relies on correct serialVersionUID for recovering state of serialized object and throws `java.io.InvalidClassException` in case of serialVersionUID mismatch, to learn more about serialversionuid see this article.

Question 4) While serializing you want some of the members not to serialize? How do you achieve it?

Another frequently asked Serialization interview question. This is sometime also asked as what is the use of transient variable, does transient and static variable gets serialized or not etc. so if you don't want any field to be part of object's state then declare it either static or transient based on your need and it will not be included during Java serialization

Join this site
with Google Friend Connect

Members (3564) More »

Already a member? Sign in

MAX LIFE INSURANCE

Replay

₹1 Crore + ₹40,000
Lump sum Guaranteed Monthly income for 10 years

At only ₹9,200 per annum

MAX LIFE ONLINE TERM PLAN

GET FREE E-QUOTE

Male Non Smoker 28 years, 30 year term. Disclaimer

A Max India and MS Joint Venture

Recommended Best Practices

- 3 Method and Constructor overloading best practices
- How string in switch case internally works in Java 7?
- Top 10 Programming Best Practices to name variables and methods
- How SSL, HTTPS and Certificates works in Java Application?
- 7 tips while dealing with password in Java?
- Introduction of How Android works for Java Programmers
- How substring() method works in Java?

process.

Question 5) What will happen if one of the members in the class doesn't implement Serializable interface?

One of the easy question about Serialization process in Java. If you try to serialize an object of a class which implements Serializable, but the object includes a reference to an non- Serializable class then a '`NotSerializableException`' will be thrown at runtime and this is why I always put a *SerializableAlert* (comment section in my code) , one of the code comment best practices, to instruct developer to remember this fact while adding a new field in a Serializable class.

Follow by Email

Email address...

Submit

Question 6) If a class is Serializable but its super class in not, what will be the state of the instance variables inherited from super class after deserialization?

Java serialization process only continues in object hierarchy till the class is Serializable i.e. implements Serializable interface in Java and values of the instance variables inherited from super class will be initialized by calling constructor of Non-Serializable Super class during deserialization process. Once the constructor chaining will started it wouldn't be possible to stop that , hence even if classes higher in hierarchy implements Serializable interface , there constructor will be executed. As you see from the statement this Serialization interview question looks very tricky and tough but if you are familiar with key concepts its not that difficult.

Question 7) Can you Customize Serialization process or can you override default Serialization process in Java?

The answer is yes you can. We all know that for serializing an object

`ObjectOutputStream.writeObject (saveThisobject)` is invoked and for reading object `ObjectInputStream.readObject()` is invoked but there is one more thing which Java Virtual Machine provides you is to define these two method in your class. If you define these two methods in your class then JVM will invoke these two methods instead of applying default serialization mechanism. You can customize behavior of object serialization and deserialization here by doing any kind of pre or post processing task. Important point to note is making these methods private to avoid being inherited, overridden or overloaded. Since only Java Virtual Machine can call private method integrity of your class will remain and Java Serialization will work as normal. In my opinion this is one of the best question one can ask in any Java Serialization interview, a good follow-up question is why should you provide custom serialized form for your object?

Question 8) Suppose super class of a new class implement Serializable interface, how can you avoid new class to being serialized?

One of the tricky interview question in Serialization in Java. If Super Class of a Class already implements Serializable interface in Java then its already Serializable in Java, since you can not unimplemented an interface its not really possible to make it Non Serializable class but yes there is a way to avoid serialization of new class. To avoid Java serialization you need to implement `writeObject()` and `readObject()` method in your Class and need to throw `NotSerializableException` from those method. This is another benefit of customizing java serialization process as described in above Serialization interview question and normally it asked as follow-up question as interview progresses.

Question 9) Which methods are used during Serialization and DeSerialization process in Java?

This is very common interview question in Serialization basically interviewer is trying to know; Whether you are familiar with usage of `readObject()`, `writeObject()`,

Blog Archive

- ► 2015 (122)
- ► 2014 (106)
- ► 2013 (136)
- ► 2012 (217)
- ▼ 2011 (144)
  - ► December (28)
  - ► November (14)

`readExternal()` and `writeExternal()` or not. Java Serialization is done by java.io.ObjectOutputStream class. That class is a filter stream which is wrapped around a lower-level byte stream to handle the serialization mechanism. To store any object via serialization mechanism we call `ObjectOutputStream.writeObject(saveThisobject)` and to deserialize that object we call `ObjectInputStream.readObject()` method. Call to `writeObject()` method trigger serialization process in java. one important thing to note about `readObject()` method is that it is used to read bytes from the persistence and to create object from those bytes and its return an Object which needs to be type cast to correct type.

Question 10) Suppose you have a class which you serialized it and stored in persistence and later modified that class to add a new field. What will happen if you deserialize the object already serialized?

It depends on whether class has its own serialVersionUID or not. As we know from above question that if we don't provide serialVersionUID in our code java compiler will generate it and normally it's equal to hashCode of object. by adding any new field there is chance that new serialVersionUID generated for that class version is not the same of already serialized object and in this case Java Serialization API will throw java.io.InvalidClassException and this is the reason its recommended to have your own serialVersionUID in code and make sure to keep it same always for a single class.

11) What are the compatible changes and incompatible changes in Java Serialization Mechanism?

The real challenge lies with change in class structure by adding any field, method or removing any field or method is that with already serialized object. As per Java Serialization specification adding any field or method comes under compatible change and changing class hierarchy or UN-implementing Serializable interfaces some under non

Ads by Google
► Serialization in Java
► Java Database
► Java Free

compatible changes. For complete list of compatible and non compatible changes I would advise reading Java serialization specification.

12) Can we transfer a Serialized object vie network?

Yes you can transfer a Serialized object via network because Java serialized object remains in form of bytes which can be transmitter via network. You can also store serialized object in Disk or database as Blob.

13) Which kind of variables is not serialized during Java Serialization?

This question asked sometime differently but the purpose is same whether Java developer knows specifics about static and transient variable or not. Since static variables belong to the class and not to an object they are not the part of the state of object so they are not saved during Java Serialization process. As Java Serialization only persist state of object and not object itself. Transient variables are also not included in java serialization process and are not the part of the object’s serialized state. After this question sometime interviewer ask a follow-up if you don't store values of these variables then what would be value of these variable once you deserialize and recreate those object? This is for you guys to think about .

Copyright by Javin Paul 2010-2015. Powered by Blogger.

More questions:

133+ Java Interview Questions from Last 4 years (answers)
50+ Java Mulithreading Questions from Last 2 years (see more)
50+ Programming Phone Interview Questions with answers (read more)

Recommended Books

Java Programming Interview Exposed
Cracking the Coding Interview: 150 Programming Questions

Bose SoundLink On-Ear Bluetooth...

Rs. 21,038.00 New Deals Every Day

Amazon India

You might like:

- How Volatile in Java works? Example of volatile keyword in Java
- How to work with transient variable in java (an example tutorial)
- 10 Best Practices to Follow while writing Code Comments
- Difference between transient and volatile keyword in Java

Recommended by

Posted by Javin Paul G+1 +158 Recommend this on Google

Labels: core java , core java interview question , Serialization

Location: United States

46 comments :

Anonymous said...

Dude I am not completely agree with your statement "Serialization is one of important concept but it's been rarely used as persistence solution" I also work for Investment bank and we use Serialization to persist our equity trading application . We used to persist our Orders and Executions in serialized format and recover from them when our trading application restarts.

April 17, 2011 at 6:28 AM

Anand said...

@ Anonymous - though serialization isnt used extensively because of the arrival of technologies like Hibernate, it is still used in many legacy systems and believe it or not - it is very powerful.

@Javin - Thanks for the comments mate. Some more questions on Java Serialization Click Here

April 18, 2011 at 6:38 AM

JustGettingBy said...

Comprehensive list! One aspect of serialization not covered here is the use of readResolve/writeReplace.

Judicious use of these methods goes a long way accomodating:
i) changes to class signature where a serialversionUUID is declared and readObject/writeObject are not defined or updated

ii) you want class instances to masquerade as something totally different. e.g. a LineItem is serialized as a Hashtable (aka key/value pairs)

April 18, 2011 at 7:40 PM

Marcelo de Castro said...

I liked the post, but I have one observation: The interface Serializable is in the java.io package instead of the java package as stated Question 2.

April 20, 2011 at 6:09 AM

Javin @ FIX Protocol Tutorials said...

@JustGettingBy , Thanks for your valuable comments and adding value into this blog post. yes readResolve()is quite important because its gives kind of flexibility you need to keep your Singleton as Singleton during serialization and desrialization etc.

April 20, 2011 at 6:57 AM

Javin @ Basic Tutorials on FIX Protocol said...

@Marcelo de Castro , your observation is absolutely correct. thanks for pointing that out. Serializable interface is indeed in the java.io package and not in java.lang package.

Thanks

April 20, 2011 at 7:00 AM

Anonymous said...

Great job buddy.......

July 5, 2011 at 11:43 AM

Anonymous said...

serialization, serializable or externalizable all these are just are too complex to understand

August 1, 2011 at 2:31 AM

Anonymous said...

For question number 8, is it not simpler to make variables of derived class as TRANSIENT? This way it will not be serialized.

August 30, 2011 at 8:41 AM

Swaranga said...

Good one. Do write about the Externalizable interface. Could be a good read. To find out more about customizing java serialization read Customizing Java Serialization [Part 1] and Customizing Java Serialization

[Part 2].

September 14, 2011 at 5:01 AM

Travel Trails said...

If class A,B,C,D are such that B extends A, C extends B and so on and if class A and D implents serializable. Then will constructor of class A,B,C will be executed? This question comes to my mind after reading answers to Q6 and Q8. Can you please clarify ?

November 5, 2011 at 11:39 AM

Javin @ Java Enum Tutorial said...

@Travel Trails, As SuperClass A itself implements Serializable so all of its subClass inherit that and they will remain Serializable

November 7, 2011 at 6:49 AM

Sumeet Kataria said...

The answer to question asked by writer of this page to us:
Question:
13th question's part:
After this question sometime interviewer ask a follow-up if you don't store values of these variables then what would be value of these variable once you deserialize and recreate those object?
Answer:
We get the default value of the object as I tested. Example: In case of static String or transient String, you will get NULL value of variable after deserializaton :)

November 12, 2011 at 7:10 AM

Riya said...

Great collection of serialization interview questions. you have almost covered everything in serialization in Java, its not just good for interview but also from learning Java serialization or do I say a good Java Serialization tutorial, keep working.

December 2, 2011 at 2:01 AM

java learner said...

thanks for your guide the question as is

how to stop the serialization without using transient and static key words.

so i got the answer from your question Q1.

once again thanks you

January 2, 2012 at 9:19 AM

Bindu said...

fantastic questions, I learn a lot many things about Java Serialization by going through these interview questions.

January 15, 2012 at 5:47 PM

Jagadish.mk said...

fantastic questions, I learn a lot many things about Java Serialization by going through these interview questions ,,externalization also good it controles serialization very good...

January 17, 2012 at 4:35 AM

Deepak Kumar Modi said...

Dear writer,
I want to make my whole package and hences classes not Serializable. I have tried to write "writeObject()", it doesn't work.

This method in ObjectOutputStream is declared as final, so you can't override.
public final void writeObject(Object obj) throws IOException;

Also if I make private, how we call that: If you call like:

BaseClass out=new BaseClass(new FileOutputStream("HoldSerial.txt"));
out.writeObject(b);

By making a constructor in your class which takes OutputStream as parameter, is not going to work as people will write:

ObjectOutputStream out=new ObjectOutputStream(new FileOutputStream("HoldSerial.txt"));
out.writeObject(b);

Can you write this syntax...

March 18, 2012 at 8:02 PM

Javin @ ejb interview questions answers said...

Hi Deepak, Serializable is an interface and work class by class, you can not make whole package serializable or non serializable. If you are extending from a class which is serializable and you want to make your sub class non serializable just add writeObject() and readObject() and throw NonSerializableException, let me know if you face any issue doing it or if you are facing altogether different issue.

March 19, 2012 at 5:22 AM

rithik said...

Isn't What is Serialization in Java should be first question ?
What is Serialization in Java?
Serialization in Java is a process to store Object's state into binary format to store on persistent storage like File system or sending over network. Serialization is handled by JVM itself by marker interface Serializable which says that object of this class can be Serialized. Java also provides several construct to control Serialization like Externalizable interface, transient variables etc.

August 2, 2012 at 8:42 PM

JavaRocks said...

Nice article, it cleared all my doubts regaring Serialization. However, I have a concern.

This is regarding question no 6.
You mentioned that "if classes higher in hierarchy implements Serializable interface , there constructor will be executed."

If the parent class implements Serializable, then all the child class will automatically be Serializable, and hence, in that case the constructor will never be called, as the entire class hierarchy is Serializable now.

For example :

```
class Z implements Serializable {

}

class A extends Z {
public int i = 50;
}

class B extends A implements Serializable {
public void changeVal(int val) {
i = val;
}
}
```

So, in the above example, irrespective of mentioning class B Serializable, it is always Serializable.
Similarly, even if we did not mention Class A as Serializable, even this class is Serializable.

August 24, 2012 at 5:15 AM

srikanth said...

Nice article, explanation is good.
I was having one question in my mind as this serialization is the mechanism that JVM takes care of serializing the object. So there must be some code written where it is trying to serialize the object (correct me if I am wrong),

Where can we find that code which is doing this serialization ?

November 29, 2012 at 11:15 PM

Gnanam said...

What is serialization in java? Pls explain with example?

December 25, 2012 at 9:28 PM

Shariq Bharat said...

I am regular reader of your blog & find it far ahead of other blogs in terms of approach for explanation of java concepts.
I have one confusion. Serialization can be obtained by implementing serializable interface & this interface dont have any method. So java do all serialization thing. Explain me how? Will java call any other class/method & how this interface is different from user defined interfaces.

March 13, 2013 at 3:31 AM

Javin @ String to Integer Java said...

@Shariq Bharat, Thanks for kind word. By the way you have raised a good question, How does Serialization works, if there is no method in Serializable interface. Well, As you might know Serializable is a marker interface, they are instruction for compiler or runtime. By implementing Serialiazable interface, Developer ask Java to take care of saving state of that object. JVM which is responsible for actual work i.e. reading object's state and converting into Serialized format, uses default serialization process, if not modified using Externalizable interface. Actual Saving of object is done by using ObjectOutputStream. I will try to explain this in more detail, may be in a blog post.

March 13, 2013 at 4:07 AM

Java Geek said...

You cannot serialize Thread object as well !!! Please add it in your tutorial.

April 6, 2013 at 6:44 AM

Ashish said...

Can someone explain me what happen to those fields which are not marked for serialization and those fields which are inside method.

For example--

```
public String Method m1()
{
String name="abc";
return name;
}
```

Can this name be serialized , if not then why ?

May 28, 2013 at 4:51 AM

Uday Nalam said...

Hi This is Uday,

7) Can you Customize Serialization process or can you override default Serialization process in Java?

For this question you answered that,
we can declare our own readObject,writeObject methods to do our serialization.
In that case 'readObject' and 'writeObject' methods are from 'ObjectInputStream' and 'ObjectOutputStream' classes and these methods are final methods.
Then how can we override ?

Please clarify me.

July 20, 2013 at 4:42 AM

Anonymous said...

Hi:

A very informative article. Keep up the good work.

August 3, 2013 at 10:46 PM

Abhijeet Muneshwar said...

Answer to the questions asked in the answer of 13th questions is:
null

i.e.
Question: If you don't store values of these variables then what would be value of these variable once you deserialize and recreate those object?
Answer: null

September 19, 2013 at 3:35 AM

Anonymous said...

Hey Javin,

I was going thru the que 6 and found
"Once the constructor chaining will started it wouldn't be possible to stop that , hence even if classes higher in hierarchy implements Serializable interface , there constructor will be executed".

So the question, how does this possible? If a super class is serializable then sub class will be automatically. :)

October 1, 2013 at 10:44 AM

Ramakrishna Ravipati said...

can we get the original value of transient variable while de serializing the object.

January 29, 2014 at 12:22 AM

Patit Pawan Karmakar said...

Its a very good article thank you very much..........the last question you asked i tried................
------>>>>what would be value of these variable once you deserialize and recreate those object?
It give null output for the transient variables.......so is it true or not ?
Please ensure me...........

February 18, 2014 at 7:12 AM

Alps said...

For transient field, its the default value of the data type, for Object type it would be null.
For static field, the value would be the same as defined.

February 23, 2014 at 5:33 AM

Anonymous said...

Hi Javin:

A very nice blog. keep it up and keep adding more. Follow-up questions (comments) are also very educative.

bye,

March 10, 2014 at 10:30 PM

Anonymous said...

2) class A{
a1;
a2;
a3
}

after serelization
class B{
a1;
a2;
a3;
a4;

}
will it be possible

May 27, 2014 at 1:26 AM

Anonymous said...

Hi Sir,

Can we keep our subclass non-serializable if the super class implements Serializable ?
Please let me know.
Thanks in advance :)
Regards
Aman.

May 27, 2014 at 4:16 AM

Anonymous said...

Feeling great after reading this awesome & exhaustive list of questions on Serialization.....Many thanks :)

June 18, 2014 at 10:04 AM

Chandan Kalita said...

Can u pls explain Q6 in details?
If a class is Serializable but its super class in not, what will be the state of the instance variables inherited from super class after deserialization?

August 7, 2014 at 2:06 PM

MANDAR VAIDYA said...

Hi,
When you declared variable as transient that variable will not serialize but how i can check whether particular variable is serialize or not.so that i can assure that variable not serialized when used with transient

Thanks

November 16, 2014 at 6:02 AM

Anonymous said...

This is regarding question 8. I tried this at HOME :) and serialization seems a special case. I had class B extend class A which was serializable while B wasnt. when I tried to serialize B I got the exception that B is not serializable, basically we dont need to implement any method specifically and throw NotSerializableException for scenario 8.
public class ParentA implements Serializable{

public class IntermediateB extends ParentA {

object.writeObject(fileoutB);

Seems like serialization is a special case for inheritance
Exception in thread "main" java.io.NotSerializableException: java.io.FileOutputStream

April 13, 2015 at 12:12 AM

Anonymous said...

Many People have asked that readObject() & writeObject() are final methods from ObjectInputStream & ObjectOutputStream so how can we Override it?
The answer to that is, you are confused about Method Overriding.
First of all Our Class is never Extending neither ObjectInputStream, nor ObjectOutputStream, so it can't be called Overriding.
Check back with your Overriding Concepts.

Secondly when we do define these methods a.k.a. readObject() & writeObject(), acc. to me, we are directing JVM to use our Defined methods for Serializing/De-serializing the Object of this Class, rather than using the ObjectInputStream & ObjectOutputStream Class methods & thats how we achieve Customization in Serialization.

BTW this post is really informative & very good coverage of Serialization Topic
If there would have been some questions bout readResolve() writeReplace(), it would have been the Best

July 10, 2015 at 4:08 AM

Javin Paul said...

@Anonymous, Thanks for your comments.

You are correct, You don't need to override readObject() from ObjectInputStream and writeObject() from ObjectOutputStream for serializing objects, but yes its confusing.

Whole Serialization concept is little bit secret because JVM does most of the things without telling you directly. You mark your class as Serializable which is a signal to JVM that it can Serialize that object using default Serialization mechanism.

Regarding readResolve() and writeReplace(), sure will add few questions as they are really important especially while keeping a Singleton class singleton during de-serialization.

July 11, 2015 at 12:10 AM

Anonymous said...

How String object is read or write without using Serializable interface.?

October 4, 2015 at 9:08 AM

Anonymous said...

Great list of Java serialization questions, for more Java Serialization interview questions from last 5 years, see this list of [133 Java interview questions]

October 21, 2015 at 7:34 PM

Ajay Singh said...

Regarding Question no 8: Java serialization process only continues in object hierarchy till the class is Serializable i.e. implements Serializable interface in Java and values of the instance variables inherited from super class will be initialized by calling constructor of Non-Serializable Super class during deserialization process. Once the constructor chaining will started it wouldn't be possible to stop that , hence even if classes higher in hierarchy implements Serializable interface , there constructor will be executed.

The above statement is wrong.

Eg :
```
class A {
}

class B extends A implements Serializable {
}

class C extends B {
}

class D extends C implements Serializable {
}
```

No, as per the above statement, only D will be persisted via Serialization, and for class C and above, the constructed will be invoked.

Well as their super parent (class B) is already implementing Serializable interface, there is really no need to declare it explicitly for the child class, as this behavior is already Inherited.

The only constructor that will be called in the above example is for class A.

December 3, 2015 at 1:23 AM

Post a Comment

Enter your comment...

Comment as: Unknown (Go ▼

Sign out

Publish Preview Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)