300+ [LATEST] Java Serialization Interview Questions and Answers

Q1. Can A Serialized Object Be Trmitted Through Network?

Yes, a Serialized object can be trmitted via the network as Java serialized object remains in form of bytes. Serialized objects can also be stored in Disk or database as Blob.

Q2. How To Prevent A Child Class From Being Serialized When It's Parent Class Already Implements Serializable Interface?

When the parent implements serializable then the child Class also be serializable although it doesn't implement Serializable interface.

As a workaround, implement writeObject() and readObject() method in the child class and throw NotSerializableException from those methods.

Q3. Give Few Examples Of Incompatible Changes In Java Serialization?

- Changing implementation from Serializable to Externalizable interface,
- Deleting an existing Serializable field,
- Changing non-trient field to trient, non-static to static,
- Changing the field type,
- Updating the class package.
- Modifying the writeObject() / readObject() method, we must not modify these methods, though adding or removing is a compatible change.

Q4. How To Create A Serializable Class In Java?

Implement java.io.Serializable interface in the Java class and JVM will allow to serialize its objects in default serialization format.

Q5. Which Method Is Used During Serialization And Deserialization Process In Java?

Java Serialization is done by java.io.ObjectOutputStream class, 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.

Q6. What Would Happen If The Serialversionuid Of An Object Is Not Defined?

If you don't define serialVersionUID in your serilizable class, Java compiler will make one by creating a hash code using most of your class attributes and features. When an object gets serialized, this hash code is stamped on the object which is known as the SerialVersionUID of that object. This ID is required for the version control of an object. SerialVersionUID can be specified in the class file also. In case, this ID is not specified by you, then Java compiler will regenerate a SerialVersionUID based on updated class and it will not be possible for the already serialized class to recover when a class field is added or modified. Its recommended that you always declare a serialVersionUID in your Serializable classes.

Q7. What Are The Alternatives To Serialization? If Serialization Is Not Used, Is It Possible To Persist Or Trfer An Object Using Any Other Approach?

In case, Serialization is not used, Java objects can be serialized by many ways, some of the popular methods are listed below:

- Saving object state to database, this is most common technique used by most applications. You can use ORM tools (e.g. hibernate) to save the objects in a database and read them from the database.
- Xml based data trfer is another popular mechanism, and a lot of XML based web services use this mechanism to trfer data over network. Also a lot of tools save XML files to persist data/configurations.
- JSON Data Trfer – is recently popular data trfer format. A lot of web services are being developed in JSON due to its small footprint and inherent integration with web browser due to JavaScript format.

Q8. Does Setting The Serialversionuid Class Field Improve Java Serialization Performance?

Declaring an explicit serialVersionUID field in your classes saves some CPU time only the first time the JVM process serializes a given Class. However the gain is not significant, In case when you have not declared the serialVersionUID its value is computed by JVM once and subsequently kept in a soft cache for future use.

Q9. Why Does Serialization Not Save The Value Of Static Class Attributes? Why Static Variables Are Not Serialized?

The Java variables declared as static are not considered part of the state of an object since they are shared by all instances of that class. Saving static variables with each serialized object would have following problems.

- It will make redundant copy of same variable in multiple objects which makes it in-efficient.
- The static variable can be modified by any object and a serialized copy would be stale or not in sync with current value.

Q10. How Do I Serialize A Collection In Java?

All standard implementations of collections List, Set and Map interface already implement java.io.Serializable. However ensure all the objects added in collection are Serializable as well.

Q11. What Is A Serial Version Uid (serialversionuid) And Why Should I Use It? How To Generate One?

The serialVersionUID represents your class version, and you should change it if the current version of your class is not backwards compatible with its earlier versions. This is extract from Java API Documentation

The serialization runtime associates with each serializable class a version number, called a serialVersionUID, which is used during deserialization to verify that the sender and receiver of a serialized

object have loaded classes for that object that are compatible with respect to serialization.

Most of the times, we probably do not use serialization directly. In such cases, I would suggest to generate a default serializable uid by clicking the quick fix option in eclipse.

Q12. What Are Trient Variables? What Role Do They Play In Serialization Process?

The trient keyword in Java is used to indicate that a field should not be serialized. Once the process of de-serialization is carried out, the trient variables do not undergo a change and retain their default value. Marking unwanted fields as trient can help you boost the serialization performance. Below is a simple example where you can see the use of trient keyword.

```
class MyVideo implements Serializable

{

private Video video;

private trient Image thumbnailVideo;

private void generateThumbnail()

{

// Generate thumbnail.

}

private void readObject(ObjectInputStream       inputStream)

throws IOException, ClassNotFoundException

{
```

```
inputStream.defaultReadObject();

generateThumbnail();

}

}
```

Q13. How To Improve Java Serialization Performance?

Java serialization performance directly rely on the number and size of attributes in the Java object. To improve the performance,

- Mark the unwanted or non Serializable attributes as trient.
- Save only the state of the object, not the derived attributes.Some times, serializing them can be expensive.
- Serialize attributes only with NON-default values.
- Use Externalizable interface and implement the readExternal and writeExternal methods to dynamically identify the attributes to be serialized.

Q14. How To Generate A Serialversionuid In Java?

There are 3 ways to create a serialVersionUID value.

Using serialVer command bundled with JDK, pass the serializable class name as command parameter to get its version identifier.

Using Eclipse IDE, hover at the class and from the context menu choose Add default serial version ID, or Add generated serial version ID.

Assign your own value and postfix with 'L'.

private static final long serialVersionUID = 19L;

Q15. List Few Alternatives To Java Serialization?

Saving object state to database using ORM tools,

Xml based data trfer,

and JSON Data Trfer.

Q16. What Are The Ways To Speed Up Object Serialization? How To Improve Serialization Performance?

- The default Java Serialization mechanism is really useful, however it can have a really bad performance based on your application and business requirements. The serialization process performance heavily depends on the number and size of attributes you are going to serialize for an object. Below are some tips you can use for speeding up the marshaling and un-marshaling of objects during Java serialization process.

- Mark the unwanted or non Serializable attributes as trient. This is a straight forward benefit since your attributes for serialization are clearly marked and can be easily achieved using Serialzable interface itself.
- Save only the state of the object, not the derived attributes. Some times we keep the derived attributes as part of the object however serializing them can be costly. Therefore consider calcualting them during de-serialization process.
- Serialize attributes only with NON-default values. For examples, serializing a int variable with value zero is just going to take extra space however, choosing not to serialize it would save you a lot of performance. This approach can avoid some types of attributes taking unwanted space. This will require use of Externalizable interface since attribute serialization is determined at runtime based on the value of each attribute.
- Use Externalizable interface and implement the readExternal and writeExternal methods to dynamically identify the attributes to be serialized. Some times there can be a custom logic used for serialization of various attributes.

Q17. What Happens When One Of The Members Of The Class Does Not Implement Serializable Interface?

At runtime NotSerializableException is thrown when try to serialize the class when one of the members does not implement serializable interface.

Q18. Give More Examples Of Compatible Changes In Java Serialization?

- Adding a new field will not affect serialization. The newly added field will be set to its default values when the object of an older version of the class is unmarshaled.
- Changes In access modifiers such as private, public, protected or default is compatible since they are not reflected in the serialized object stream.
- Changing a trient field to a non-trient field, static to non-static are compatible changes.

- Adding or removing writeObject()/readObject() methods.

Q19. Is Externalizable A Marker Interface In Java?

No. It has two methods readExternal and writeExternal to be implemented.

Q20. How Do I Prevent Deserialization Process Creating Another Instance Of Singleton Class?

Use readResolve method to return the same instance of a class, rather than creating a new instance.

Q21. What Happens When A Class Does Not Define Serialversionuid In Java?

If the serialVersionUID is not defined, then after any modification made in class, we won?t be able to deSerialize existing objects for same class because serialVersionUID generated by Java compiler for the modified class will be different from the old serialized object. Deserialization process will fail by throwing java.io.Invalid Class Exception.

Q22. Which Variables Are Not Serialized During Java Serialization?

Static and trient variables cannot be serialized.

Q23. Does Having Serialversionuid Variable Improve Java Serialization Performance?

Declaring a serialVersionUID field in a Java class saves CPU time the first time the JVM process serializes a given Class. However the performance gain is not very significant, In case when you have not declared the serialVersionUID its value is computed by JVM once and subsequently kept in a soft cache for future use.

Q24. Why Do We Need Externalizable Interface?

Externalizable Interface allows us to control serialization mechanism which may help improve application performance.

Q25. What Are The Compatible And Incompatible Changes In Java Serialization Mechanism?

Adding new field or method is a compatible change and changing class hierarchy or UN-implementing Serializable interface are non compatible changes.

Q26. Is It Possible To Customize The Serialization Process? How Can We Customize The Serialization Process?

Yes, the serialization process can be customized. When an object is serialized, objectOutputStream.writeObject (to save this object) is invoked and when an object is read, ObjectInputStream.readObject () is invoked. What most people do not know is that Java Virtual Machine provides you with an option to define these methods as per your needs. Once this is done, these two methods will be invoked by the JVM instead of the application of the default serialization process. Classes that require special handling during the serialization and deserialization process must implement special methods with these exact signatures:

private void writeObject(java.io.ObjectOutputStream out)

throws IOException

private void readObject(java.io.ObjectInputStream in)

throws IOException, ClassNotFoundException;

private void readObjectNoData()

throws ObjectStreamException;

Q27. Is The Constructor Invoked When A Java Object Is De-serialized?

If the class implements Serializable, the constructor is not called during deserialization process. However, if the class implements Externalizable, the constructor is called during deserialization process.

Q28. Difference Between Readresolve And Readobject Methods In Java Serialization?

For Serializable and Externalizable classes, the readResolve method allows a class to replace/resolve the object read from the stream before it is returned to the caller. By implementing the readResolve method, a class can directly control the types and instances of its own instances being deserialized. This method is called when ObjectInputStream has read an object from the stream and is preparing to return it to the caller.

For serializable objects, the readObject method allows a class to control the deserialization of its own fields and held responsible for restoring the state of the class.

Q29. How To Serialize A Collection In Java? How To Serialize A Arraylist, Hashmap Or Hashset Object In Java?

All standard implementations of collections List, Set and Map interface already implement java.io.Serializable. All the commonly used collection classes like java.util.ArrayList, java.util.Vector, java.util.Hashmap, java.util.Hashtable, java.util.HashSet, java.util.TreeSet do implement Serializable. This me you do not really need to write anything specific to serialize collection objects. However you should keep following things in mind before you serialize a collection object – Make sure all the objects added in collection are Serializable. – Serializing the collection can be costly therefore make sure you serialize only required data isntead of serializing the whole collection. – In case you are using a custom implementation of Collection interface then you may need to implement serialization for it.

Q30. What Is Trient Variable In Java?

Trient variables are not included in the serialization and are not the part of the object's serialized state. A trient variable can be created by specifying trient keyword.

Q31. Is Serializable A Marker Interface In Java?

Yes. It has no methods.

Q32. What Is Serialversionuid In Java?

Every time an object is serialized the Java serialization mechanism automatically computes a hash value called serialVersionUID. ObjectStreamClass's computeSerialVersionUID() method passes the class name, sorted member names, modifiers, and interfaces to the secure hash algorithm (SHA), which returns a hash value.The serialVersionUID is also called suid.

Q33. Is Constructor Of Parent Class Called During Deserialization Process Of Child Class?

If superclass implements Serializable – constructor is not called while if the superclass doesn't implement Serializable – constructor is called during DeSerialization process of child class.

Q34. How Can A Sub-class Of Serializable Super Class Avoid Serialization? If Serializable Interface Is Implemented By The Super Class Of A Class, How Can The Serialization Of The Class Be Avoided?

In Java, if the super class of a class is implementing Serializable interface, it me that it is already serializable. Since, an interface cannot be unimplemented, it is not possible to make a class non-serializable. However, the serialization of a new class can be avoided. For this, writeObject () and readObject() methods should be implemented in your class so that a Not Serializable Exception can be thrown by these methods. And, this can be done by customizing the Java Serialization process. Below the code that demonstrates it

class MySubClass extends SomeSerializableSuperClass {

```
private void writeObject(java.io.ObjectOutputStream out)

throws IOException {

throw new NotSerializableException(“Can not serialize this class”);

}

private void readObject(java.io.ObjectInputStream in)

throws IOException, ClassNotFoundException {

throw new NotSerializableException(“Can not serialize this class”);

}

private void readObjectNoData()

throws ObjectStreamException; {

throw new NotSerializableException(“Can not serialize this class”);

}

}
```

Q35. Is There Any Limit On The Size Of A Serialized Object?

Yes, it is limited by the amount of memory your JVM can allocate to the creation and maintenance of your object. As for writing it out to disk, it is limited by the maximum file size of the underlying OS. Linux, as an example, has a 2GB limit on one file.

Q36. Do Primitive Data Types Involve In Serialization?

Yes, all the primitive data types are part of serialization.

Q37. Do We Need To Implement Any Method When Using Serializable Interface?

No. Serializable interface is a marker interface.

Q38. Why Static Variables Are Not Serialized In Java?

The static variables are class level variables and are not the part of the object state so they are not saved as the part of serialized object.

Q39. What Changes Are Compatible And Incompatible To The Mechanism Of Java Serialization?

This is one of a difficult and tricky questions and wering this correctly would mean you are an expert in Java Serialization concept. In an already serialized object, the most challenging task is to change the structure of a class when a new field is added or removed. As per the specifications of Java Serialization, addition of any method or field is considered to be a compatible change whereas changing of class hierarchy or non-implementation of Serializable interface is considered to be a non-compatible change. You can go through the Java serialization specification for the extensive list of compatible and non-compatible changes. If a serialized object need to be compatible with an older version, it is necessary that the newer version follows some rules for compatible and incompatible changes. A compatible change to the implementing class is one that can be applied to a new version of the class, which still keeps the object stream compatible with older version of same class.

Some Simple Examples of compatible changes are:

- Addition of a new field or class will not affect serialization, since any new data in the stream is simply ignored by older versions. the newly added field will be set to its default values when the object of an older version of the class is un marshaled.
- The access modifiers change (like private, public, protected or default) is compatible since they are not reflected in the serialized object stream.

- Changing a trient field to a non-trient field is compatible change since it is similar to adding a field.
- Changing a static field to a non-static field is compatible change since it is also similar to adding a field.

Some Simple Examples of incompatible changes are:

- Changing implementation from Serializable to Externalizable interface can not be done since this will result in the creation of an incompatible object stream.
- Deleting a existing Serializable fields will cause a problem.
- Changing a non-trient field to a trient field is incompatible change since it is similar to deleting a field.
- Changing a non-static field to a static field is incompatible change since it is also similar to deleting a field.
- Changing the type of a attribute within a class would be incompatible, since this would cause a failure when attempting to read and convert the original field into the new field.
- Changing the package of class is incompatible. Since the fully-qualified class name is written as part of the

Java serialization is one of the most commonly misunderstood areas. Many developers still think its only used for saving objects on the file system.

Hope you found this list useful. Can you think of a questions that is not part of this list? Please don't forget to share it with me in comments section & I will try to include it in the list.

Q40. What Is The Value Of A Trient Variable After Deserialization?

It will be set to its default value. For example, an int trient variable will be set to zero.

Q41. List Few Differences Between Serializable And Externalizable In Java?

Serializable:

- Serializable has its own default serialization process, we just need to implement Serializable interface. We can customize default serialization process by defining following methods in our class, readObject() and writeObject().**Note:** We are not overriding these methods, we are defining them in our class.
- This is a marker interface, does not have any methods.
- Constructor is not called during deSerialization.

Externalizable:

- Override writeExternal() and readExternal() for serialization process to happen when implementing Externalizable interface.
- This has 2 methods readExternal and writeExternal, hence it is not a marker interface.
- Constructor is called during deSerialization.

Q42. When To Use Serializable Vs Externalizable?

Use of trient keyword enables selective attribute serialization, however, use of Externalizable interface can be really effective in some cases when you have to serialize only some dynamically selected attributes of a large object.

Q43. Difference Between Serializable And Externalizable Interface In Java?

Externalizable provides writeExternal() and readExternal() methods which gives flexibility to override java serialization mechanism instead of using on default serialization.

Q44. What Is Serialization In Java?

Java serialization is the process by which Java objects are serialized by storing object's state into a file with extension .ser. Restoring object's state from that file is called deserialization.

Object Serialization converts Java object into a binary format which can be persisted to disk or sent over network to other JVM.

---- >> Below Are The Related Posts Of Above Questions :::

------>>[MOST IMPORTANT]<<------

1. **250+ TOP MCQs on Serialization – 2 and Answers**
2. **250+ TOP MCQs on Serialization in Java and Answers**
3. **300+ [MOSK ASKED] Accenture Java Interview Questions and Answers**
4. **300+ [LATEST] Java Programmer Interview Questions and Answers**
5. **300+ [LATEST] Java Abstraction Interview Questions and Answers**
6. **300+ [LATEST] Java Webdynpro Interview Questions and Answers**
7. **300+ [LATEST] Java Hadoop Developer Interview Questions and Answers**
8. **300+ [LATEST] Java Developer Interview Questions and Answers**
9. **250+ TOP MCQs on Serialization and Answers**
10. **300+ [LATEST] Java Inheritance Interview Questions and Answers**
11. **250+ TOP MCQs on Serialization – 1 and Answers**
12. **300+ [LATEST] Java Applet Interview Questions and Answers**
13. **300+ [LATEST] Java Programming With Oops Concepts Interview Questions and Answers**
14. **300+ [LATEST] Java Persistence Api Interview Questions and Answers**
15. **300+ [LATEST] Java.lang Package Interview Questions and Answers**
16. **300+ [LATEST] Java.util Interview Questions and Answers**
17. **300+ [LATEST] Ericsson Java Interview Questions and Answers**

18. **300+ [LATEST] Aricent Java Interview Questions and Answers**
19. **250+ TOP MCQs on Serialization & Deserialization and Answers**
20. **300+ [LATEST] Java & Ejb & Spring Interview Questions and Answers**