**Serialization Interview Questions**

Table of Contents

[1. What’s serialization in java? 2](#_Toc478581257)

[2. How to make a java class Serializable? 3](#_Toc478581258)

[3. Difference between transient and volatile in Java? 4](#_Toc478581259)

[4. What’s transient variable in Java? 4](#_Toc478581260)

[5. Difference between Serializable and externalizable in Java? 6](#_Toc478581261)

[6. How many methods Serializable has? If no method then what is the purpose of Serializable interface? 7](#_Toc478581262)

[7. What is serialVersionUID? What would happen if you don't define this? 7](#_Toc478581263)

[8. While serializing you want some of the members not to serialize? How do you achieve it? 8](#_Toc478581264)

[9. What will happen if one of the members in the class doesn't implement Serializable interface? 8](#_Toc478581265)

[10. 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? 9](#_Toc478581266)

[11. 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? 9](#_Toc478581267)

# What’s 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 their own custom binary format, which is often advised as Serialization best practice, Because serialized binary format becomes part of Class's exported API and it can potentially break Encapsulation in Java provided by private and package-private fields.

# 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.

# Difference between transient and volatile in Java?

* + - volatile and transient are two completely different keywords from different areas of Java programming language.
    - transient keyword is used during serialization of Java object while volatile is related to visibility of variables modified by multiple thread during concurrent programming.
    - Only similarity between volatile and transient is that they are less used or uncommon keywords and not as popular as public, static or final.
    - transient keyword is used along with instance variables to exclude them from serialization process. if a field is transient its value will not be persisted. On the other hand volatile keyword can also be used in variables to indicate compiler and JVM that always read its value from main memory and follow happens-before relationship on visibility of volatile variable among multiple thread.
    - transient keyword cannot be used along with static keyword but volatile can be used along with static.
    - transient variables are initialized with default value during de-serialization and there assignment or restoration of value has to be handled by application code.

# What’s transient variable in Java?

* + - In One-word transient keyword is used in serialization process to prevent any variable from being serialized, so if you have any field which is not making sense to serialize, you can simply declare that as transient and it won't be serialized.
    - We know the purpose of transient keyword or having transient variable its make sense to think about which variable should be marked as transient. My rule is that any variable whose value can be calculated from other variables doesn't require to be saved. For example if you have a field called "interest" whose value can be derived from other fields e.g. principle, rate, time etc. then there is no need to serialize it.
    - Another example is of word count, if you are saving article then no need to save word count, because it can be created when article gets deserialized. Another good example of transient keyword is "Logger" since most of the time you have logger instance for logging in Java but you certainly don't want it to serialize correct?
    - Example

|  |
| --- |
| public class **Stock** {      private **transient** Logger logger = Logger.getLogger(Stock.class); **//will not serialized**      private String symbol; //will be serialized      private BigInteger price; //serialized      private long quantity; //serialized  } |

* + - Transient keyword can only be applied to fields or member variable. Applying it to method or local variable is compilation error.
    - Another important point is that you can declare an variable static and transient at same time and java compiler doesn't complain but doing that doesn't make any sense because transient is to instruct "do not save this field" and static variables are not saved anyway during serialization.
    - In similar way you can apply transient and final keyword together to a variable compiler will not complain but you will face another problem of reinitializing a final variable during deserialization.
    - Transient variable in java is not persisted or saved when an object gets serialized.

# Difference between Serializable and externalizable in Java?

* + - Both serializable and extenalizable used to serialize or persist java objects but the way they do is little different.
    - In case of Serializable Java Virtual machine has full control for serializing object while in case of Externalizable, application gets control for persisting objects. writeExternal() and readExternal() method provides complete control on format and content of Serialization process to application which can be leverage to increase performance and speed of serialization process.
    - In case of Serializable, default serialization process is used. While in case of Externalizable custom Serialization process is used which is implemented by application.
    - JVM gives call back to readExternel() and writeExternal() of java.io.Externalizalbe interface for restoring and writing objects into persistence.
    - Externalizable interface provides complete control of serialization process to application.
    - readExternal() and writeExternal() supersede any specific implementation of writeObject and readObject methods.
    - 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.

# 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.

# What is serialVersionUID? What would happen if you don't define this?

* + - 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

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

* + - 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 process.

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

* + - 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) to instruct developer to remember this fact while adding a new field in a Serializable class.

# 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.

# 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.

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

* + - If Super Class of a Class already implements Serializable interface in Java then it’s already Serializable in Java, since you cannot unimplemented an interface it’s 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.

# Which methods are used during Serialization and Deserialization process in Java?

* + - 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.

# 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.
    - 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.

# 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 compatible changes.

# 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 transmitted via network. You can also store serialized object in Disk or database as Blob.

# Which kind of variables is not serialized during Java Serialization?

* + - 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.