Serialization interview question and answer in Java

April 1, 2018 SJ Serialization 0

In this article, we will cover some of the interview questions with their justification on Serialization concept in Java

These are most frequently asked interview question from Serialization in Java

Read Serialization concepts in detail

Q) What is Serialization in Java ?

- The process of writing a state of an Object to a file is called Serialization
- In other words, the process of saving an Object's state to a file is called Serialization
- But practically, it is the process of converting & storing Java Object's state from heap memory (in byte stream) to file supported form (in binary format)
- For more details, read Serialization concept with figure and example

Q) Explain De-Serialization process ?

- The process of reading a state of an Object from a file is called De-Serialization
- But practically, it is the process of converting & re-storing Java Object's state into heap memory from file supported form (which is in binary format)
- For more details, read De-Serialization concept with figure and example

Q) What is the need of Serialization ?

- In Java, everything is Object. So, these objects represent state (or data) which resides inside heap memory of JVM machine running on RAM memory
- Whenever JVM shuts, then these states (or data) are lost
- So, to preserve these state (or data) once again JVM restarts, we need serialization to serialize objects in binary format
- In addition, to transfer over network channels Objects need to be converted into binary format. For this purpose also, serialization concept is very useful

Q) What are the most commonly used classes and interfaces for Serialization & De-Serialization ?

- The most commonly used classes and interfaces are,
- java.io.Serializable
- java.io.Externalizable
- java.io.ObjectInputStream
- java.io.ObjectOutputStream

Q) How many methods are present in Serializable interface ?

- Serializable interface is marker interface which means it doesn't contains any abstract methods
- Read more about Serializable interface

Q) Whether we need to implement any method while implementing Serializable interface ?

- As Serializable interface is marker interface which contains no methods therefore there is nothing to be implemented while implementing Serializable interface
- Read more about Serializable interface

Java JDBC: An example to connect MS Access database
Spring JDBC: An example on JdbcTemplate using Annotation
Oracle OSB 12c: Service Callout and Routing Table example
Oracle OSB 12c: Hello World service with both Business and Proxy Service

Q) Why Serializable interface is called Marker interface in Java ?

- Any interface in Java which doesn't contains any methods is known as marker interface
- Likewise, Serializable interface doesn't contain any methods therefore it is also known as Marker interface
- Read more about Serializable interface

Q) Explain serialVersionUID ?

- Whenever we declare any class implementing Serializable interface, then it always prompts/warns to include/declare serialVersionUID which is of type *long*
- Along with serialized object, this variable also gets serialized and during de-serialization both serialVersionUID compared (stored serialVersionUID & value present inside declared class) and then it converted back
- If there is any mismatch, then InvalidClassException is thrown
- **Note:** any changes to class results in change of serialVersionUID. Therefore, it is highly recommended to declare this value instead allowing compiler to generate for us
- Read more about importance of serialVersionUID

Q) State importance of serialVersionUID ?

- This variable is used both during serialization & de-serialization process
- During serialization, this variable also gets serialized along with original object
- Similarly, during de-serialization stored UID and serialVersionUID declared inside class gets compared and then converted back
- If there is any mismatch, then InvalidClassException is thrown
- Read more about importance of serialVersionUID

Q) How can you restrict few variables/attributes to be serialized ?

- Declare variables as *transient* for whichever we want to restrict from taking part in the serialization process
- Read more about transient variable in serialization and de-serialization process

Q) State importance of marking a variable/attribute as transient during Serialization process ?

- Whenever, it is decided that certain variables need not to be serialized then it is marked with transient modifier
- Like for example, transient double empSal;
- Read more about transient variable in serialization and de-serialization process

Q) What will be value of transient variables after de-serialization ?

- After de-serialization transient variables will have default values
- Like, 0 for int data-type, null for String, false for Boolean data-type, etc.
- Read more about transient variable in serialization and de-serialization process

Q) Which modifier stops variables to be serialized ?

- Transient modifier stops variables to be serialized
- During de-serialization, it returns with default value
- Like, 0 for int data-type, null for String, false for Boolean data-type, etc.
- Read more about transient variable in serialization and de-serialization process

Q) Whether static variables take part in serialization process ?

- Static variable won't take part in serialization process i.e.; it won't be serialized along with instance variables
- Instead during de-serialization process, values will be loaded from class that it currently holds (assigned to)
- Read more about static variable in serialization process

Q) What will happen, if class implements Serializable interface and contains reference variable which also implements Serializable interface ?

- Main object along with reference variable gets serialized as both satisfies the condition of implementing Serializable interface
- Read more about this case with example

Q) What will happen, if one of the reference variable doesn't implement Serializable interface ?

- During serialization, when we try to serialize main object which contains reference variable not implementing Serializable interface then *java.io.NotSerializableException* is thrown
- Read more about this case with example

Q) Whether order of serialization & de-serialization need to be same? What happens, if order is missed ?

- *Order of Serialization*is very *important* to know, because we need to follow the *same order while de-serializing* the objects
- If the Order of Serialization is *unknown*, then it may throw *java.lang.ClassCastException*
- To overcome *ClassCastException*, we can 1st check type of object using *instanceOf* operator and then *assign* it to proper class after doing necessary *type-casting*
- **Exception:** iterating through while loop may throw *EOFException*, we need *catch* this exception and *handle* it properly
- Read more about order of serialization & de-serialization

Q) In Parent-child class relationship, if parent implements Serializable interface and child doesn't implements Serializable interface, then whether child class is serializable ?

- For any object to be serialized, corresponding class must be serializable
- Otherwise, *NotSerializableException* will be thrown at run time, although program compiles successfully
- In this scenario, as parent-class is already implementing *Serializable* interface therefore, all extending sub-classes automatically implements *Serializable* interface through inheritance relationship
- Therefore, for above question *child-class serializes successfully*
- For more about this scenario, read Serialization with Inheritance

Q) In Parent-child class relationship, if parent implements Serializable interface and child doesn't implements Serializable interface, then how can we stop child class from serializing ?

- This question is very similar to above one, but here we are asked to stop serialization process
- To stop serialization, override *writeObject();* method explicitly throw "*No Serialization is allowed*" exception

- To understand further with example, read how to stop Serialization

Q) How to stop serialization ?

- To stop serialization, override ***writeObject();*** method explicitly throw "***No Serialization is allowed***" exception
- To understand further with example, read how to stop Serialization

Q) In Parent-child class relationship, if parent doesn't implements Serializable interface but child implements Serializable interface, then what will happen if child class is serialized ?

- We should understand ***is it possible to serializable sub class***, if its ***super class isn't serializable?***
- The answer is ***yes***, because if the condition to serialize any class on the basis of its super classes implementing ***java.io.Serializable*** interface, then ***no class in Java can be serialized***
- **Reason:** *java.lang.Object* is the base class for any class defined in Java, and it ***doesn't*** implements ***java.io.Serializable*** interface
- In that way, it is very well possible to serialize a sub class even if its super class ***doesn't*** implement ***java.io.Serializable*** interface
- But parent-class attributes will be ignored
- For more about this scenario, read Serialization with Inheritance

Q) How can we customize serialization process ?

- To customize serialization process, instead of implementing Serializable interface implement ***java.io.Externalizable*** interface which has 2 methods unlike Serializable which is Marker interface having no method
- These methods are ***writeExternal();*** & ***readExternal();***
- To serialize, use writeExternal(); method and write custom logic
- Similarly, to de-serialize use readExternal(); method and code custom logic
- For more about customization of serialization process, read Externalizable interface

Q) What is the use of Externalizable interface in Java ?

- Externalizable interface allows to write/code custom logic for both serialization & de-serialization
- It has 2 methods namely ***writeExternal();*** & ***readExternal();***
- Use writeExternal(); method to code/write custom serialization logic and readExternal(); method for custom de-serialization process
- Read more about **Externalizable interface**

Q) Difference between Serializable and Externalizable ?

- The main difference between Serializable & Externalizable is,
- Serializable interface allows to serialize compulsorily every attribute, even if users not indented to serialize few variables
- Whereas Externalizable interface allows to write custom logic to store/serialize only those attributes which is intended by users leaving rest of the attributes
- For more about their differences, read **Serializable v/s Externalizable**

Q) How to serialize ArrayList ?

- Rules for Serialization is, serializing object must implement ***java.io.Serializable*** interface
- ArrayList implements ***java.io.Serializable*** interface
- Therefore, we must concentrate on type of objects inside ArrayList and whether they implements serializable interface or not
- If type of objects ArrayList holds is also implements java.io.Serializable interface, then it is very safe to serialize and serialization succeeds as well
- But ArrayList holds object which doesn't implements java.io.Serializable interface, then Serialization fails throwing exception as ***NotSerializableException***
- For more about serialization of ArrayList with example, read **here**

Q) Serialization affects the very basics of Singleton design pattern, how can we overcome to stop class to be serialized ?

- Whenever Object is serialized and then again de-serialized then new object is created which breaks basic principle of Singleton class
- Therefore, it is highly recommended to override ***readResolve();*** method and return same INSTANCE every time. So that, no

new object is created, and it says true to Singleton property
- Read about more ***readResolve();*** **method in Singleton design pattern** which restricts creating new objects

Hope, you found this article very helpful. If you have any suggestion or want to contribute or tricky situation you faced during Interview hours, then share with us. We will include that code here

References:

- http://www.benchresources.net/java/serialization-in-java/
- https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html
- https://docs.oracle.com/javase/7/docs/platform/serialization/spec/serial-arch.html
- https://docs.oracle.com/javase/7/docs/api/java/io/ObjectOutputStream.html
- https://docs.oracle.com/javase/7/docs/api/java/io/ObjectInputStream.html
- https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html
- https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html

Happy Coding !!
Happy Learning !!

How to serialize and de-serialize ArrayList in Java »

Related Posts:

1. **Importance of SerialVersionUID in Serialization**
2. **Singleton Design pattern with Serialization**
3. **How to stop Serialization in Java**
4. **Transient keyword with static variable in Serialization**

DESERIALIZATION INTERVIEW QUESTION JAVA

SERIALIZABLE SERIALIZATION SERIALVERSIONUID

ALSO ON BENCHRESOURCES.NET

Sorting list of objects on multiple fields using

1 comment • 2 years ago

Juan Daniel Ornella — Thank You!!

How to construct an immutable class in Java

2 comments • 2 years ago

BenchResources.Net — Thanks Sachin !!

Java Collection Interview question and answers

2 comments • 2 years ago

Adil Khan — A small request,would be happy to see if you write article on how

Singleton design pattern – restricting all 4 ways of Object

3 comments • 2 years ago

Anil Nivargi — Nice explanation Thanks.....Singleton Design Pattern in Java with examples