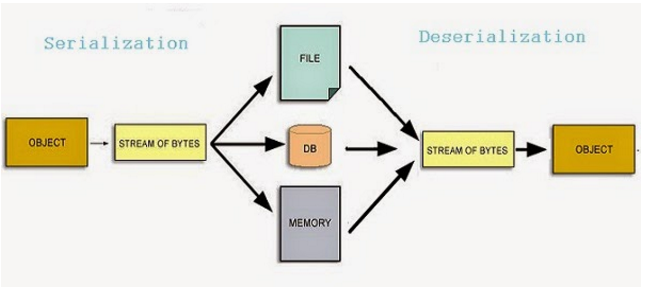
* **Serialization**

****

**Serializable**

1. **What is Serialization? IMP**

Using serialization we can save or transfer the state of an object by converting it to a byte stream.

This can be done in java by implementing Serialiazable interface.

Serializable interface it gives compiler an indication that use Java Serialization mechanism to serialize this object.

1. **Say I have an object I want to serialize it but my class does not implement serializable interface what will happen?**

To serialize any object, class should implement serializable interface. If class does not extend serializable interface but if we try to serialize it, we get an exception

java.io.NotSerializableException

Sample program is there in the project otherCoreJava-> serialization.samplecode

1. **How many methods Serializable has? If no method then what is the purpose of Serializable interface?**

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.

1. **Why is Serialization required? What is the need to Serialize? IMP**

To send state of one or more object’s state over the network through a socket.

To save the state of an object in a file.

An object’s state needs to be manipulated as a stream of bytes.

1. **If I do not make my class serializable won’t it be persisted? IMP**

It will persist the object.

But serialization breaks the object into a series (hence the name) of bytes which can be passed across a network.

https://coderanch.com/t/471056/java/dont-implement-serializable-interface

1. **What is the Difference between Serializable and Externalizable Interfaces? IMP**

**Serializable** is a marker interface therefore you are not forced to implement any methods.

**Externalizable** contains two methods writeExternal() and readExternal() which must be implemented.

**Serializable** interface provides a inbuilt serialization mechanism to you which can be in-efficient at times.

**Externilizable** interface provides two methods using these methods we can enhance the performance of specific object serialization based on application needs.

One can drastically improve the application performance by implementing the Externalizable interface correctly. However there is also a chance that you may not write the best implementation, so if you are not really sure about the best way to serialize, I would suggest your stick to the default implementation using Serializable interface.

1. **When will you use Serializable or Externalizable interface? Why?**

**Serializable**: we have small objects and we need to serialize all most all attributes we can use serialization. The attributes that are not required to be serialized can be marked as transient.

**Externalizable:** we have a big Java object with hundreds of attributes and you want to serialize only a few dynamically selected attributes then we should go for Externalizable interface,writeExternal method to selectively serialize the chosen attributes

1. **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 **transient** or **static** based on your need and it will not be included during Java serialization process.

1. **Why static member variables are not part of java serialization process? IMP**

Serialization is applicable on objects or primitive data types only, but static members are class level variables, therefore, different objects of same class have same value for static member.

So, serializing static member will consume unnecessary space and time.

Also, if modification is made in static member by any of the object, it won’t be in sync with other serialized object’s value.

1. **What happens to static members when the object is deserialized? IMP this is the answer but discuss with others about the answer.**

All static members gets default value/or the initial value assigned once they are desrialized.

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

‘NotSerializableException’ will be thrown at runtime.

Good practice:

Suppose if you are writing serializable class in java then it’s good to put a alert stating that any new fields addition in this class must implement serializable interface in java or make it transient.

1. **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? THIS IS DIFFICULT QUESTION NEED TO STUDY SERIALIZATION IN DEPTH TO UNDRRSTAND**

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.

1. Can you Customize Serialization process or can you override default Serialization process in Java?

**Yes you can**.

In case of default serialization ObjectOutputStream.writeObject (saveThisobject) is invoked and for reading object ObjectInputStream.readObject. 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.

Note: 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?**

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

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.

1. **Which methods are used during Serialization and DeSerialization process in Java?**

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

1. **What are the compatible changes and incompatible changes in Java Serialization Mechanism?**

Compatable changes: adding any field or method comes under compatible change.

Incompatible changes: changing class hierarchy or UN-implementing Serializable interfaces.

1. **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.

1. **Which kind of variables is not serialized during Java Serialization?**

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.

Transient variables are also not included in java serialization process and are not the part of the object’s serialized state

1. If you don't store values of these variables then what would be value of these variable once you deserialize and recreate those object?

http://stackoverflow.com/questions/1929130/what-is-the-value-of-static-variables-after-deserializing-an-object

**serialVersionUID**

1. **What is serialVersionUID? What would**

When the object gets serialized a unique id is assigned to the object this id is usually hashcode.

1. **What would happen if you don't define serialVersionUID?**

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.

1. 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.

1. **What is the use of transient Keyword in Java? IMP**

<http://www.fromdev.com/2012/06/15-java-serialization-interview.html> : Need to refer this for more questions.

1. I have class and multiple data members I want not to serialize a member which is not transient and not static how can i do it?