**File I/O Interview Questions**

**Q. What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?**

**Ans.** The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented.

**Q. What an I/O filter?**

**Ans.** An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

**Q. What is the difference between the File and RandomAccessFile classes?**

**Ans.** The File class encapsulates the files and directories of the local file system. The RandomAccessFile class provides the methods needed to directly access data contained in any part of a file.

**Q. Given a text file, input.txt, provide the statement required to open**

**Ans.** this file with the appropriate I/O stream to be able to read and process this file.

**Q. What value does readLine() return when it has reached the end of a file?**

**Ans.** The readLine() method returns null when it has reached the end of a file.

**Q. What is it object serialization ?**

**Ans.** Serialization is a way to convert objects (including complex data structures such as lists and

trees) into a stream of bytes.

**Q. How many methods do you implement if implement the Serializable Interface?**

**Ans.** The Serializable interface is just a "marker" interface, with no methods of its own to implement.

Are there any other 'marker' interfaces?

java.rmi.Remote

java.util.EventListener

**Q. What is transient variable?**

**Ans.** Transient variable can't be serialized. For example if a variable is declared as transient in a Serializable class and the class is written to an ObjectStream, the value of the variable can't be written to the stream instead when the class is retrieved from the ObjectStream the value of the variable becomes null.

**Q. What is Externalizable?**

**Ans.** Externalizable is an Interface that extends Serializable Interface. And sends data into Streams in Compressed Format.

It has two methods, writeExternal(ObjectOuputout) and readExternal(ObjectInput in).

This interface allows us to customize the output.

**Q. What is the difference between Serializable and Externalizable interface in Java?**

**Ans.** 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.

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

**Ans.**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. When your class implements Serializable interface it becomes Serializable in Java and gives compiler an indication that use Java Serialization mechanism to serialize this object.

**Q. What is serialVersionUID? What would happen if you don't define this?**

**Ans.** 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.

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

**Ans.** 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 process.

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

**Ans.** 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.

**Q. 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?**

**Ans.** 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.

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

**Ans.** 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 or 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.

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

**Ans.** 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 throwNotSerializableException from those method. This is another benefit of customizing java serialization process as described in above question and normally it asked as follow-up question as interview progresses.

**Q. Which methods are used during Serialization and DeSerialization process in java?**

**Ans.** This is quite a common question basically interviewer is trying to know that whether you are familiar with usage of readObject (), writeObject (), 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 casted on correct type.

**Q. 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?**

**Ans.** 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 SerializationAPI 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.

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

**Ans.** 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 unimplementing Serializable interfaces some under non compatible changes. For complete list of compatible and non compatible changes I would advise reading java serialization specification.

**Q. Can we transfer a Serialized object vie network?**

**Ans.** Yes you can transfer a Serialized object via network because java serialized object remains in form of bytes which can be transmitter via network.

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

**Ans.** 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 :)

**Q. What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?**

**Ans.** The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented.

**Q. What an I/O filter?**

**Ans.** An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

**Q. What is serialization?**

**Ans.** Serialization

Serialization is a mechanism of writing the state of an object into a byte stream. It is mainly used in Hibernate, JPA, EJB etc. The reverse operation of the serialization is called deserialization. The String class and all the wrapper classes implements Serializable interface by default.

Advantage of Serialization

It is mainly used to travel object's state on the network.

About Serializable interface

Serializable is a marker interface(have no body). It is just used to "mark" Java classes which support a certain capability. It must be implemented by the class whose object you want to persist. Let's see the example given below:

ObjectOutputStream class:

An ObjectOutputStream is used to write primitive data types and Java objects to an OutputStream.Only objects that support the java.io.Serializable interface can be written to streams

Commonly used Constructors:

1) public ObjectOutputStream(OutputStream out) throws IOException {}creates an ObjectOutputStream that writes to the specified OutputStream.

Commonly used Methods:

1) public final void writeObject(Object obj) throws IOException {}write the specified object to the ObjectOutputStream.

2) public void flush() throws IOException {}flushes the current output stream.

**Q. What is Deserialization?**

**Ans.** Deserialization is the process of reconstructing the object from the serialized state.It is the reverse operation of serialization.

ObjectInputStream class:

An ObjectInputStream deserializes objects and primitive data written using an ObjectOutputStream.

Commonly used Constructors:

1) public ObjectInputStream(InputStream in) throws IOException {}creates an ObjectInputStream that reads from the specified InputStream.

Commonly used Methods:

1) public final Object readObject() throws IOException, ClassNotFoundException{}reads an object from the input stream.

**Q. What is transient keyword?**

**Ans.** If you define any data member as transient,it will not be serialized.

**Q. What is Externalizable?**

**Ans.** Externalizable interface is used to write the state of an object into a byte stream in compressed format.It is not a marker interface.

**Q. What is the difference between Serializable and Externalizable interface?**

**Ans.** Serializable is a marker interface but Externalizable is not a marker interface.When you use Serializable interface, your class is serialized automatically by default. But you can override writeObject() and readObject() two methods to control more complex object serialization process. When you use Externalizable interface, you have a complete control over your class's serialization process.