IO Operation

1. What is Input and Output Stream in Java?

In Java, streams are the sequence of data that are read from the source and written to the destination. An input stream is used to read data from the source. And, an output stream is used to write data to the destination.

2. What are the methods of OutputStream4

- write() - writes the specified byte to the output stream.
- write(byte[] array) - writes the bytes from the specified array to the output stream.
- flush() - forces to write all data present in output stream to the destination.
- close() - closes the output stream.

3. What is serialisation in Java?

Serialisation is the conversion of the state of an object into a byte stream; deserialization does the opposite. Stated differently, serialisation is the conversion of a Java object into a static stream (sequence) of bytes, which we can then save to a database or transfer over a network.

4. What is the Serializable interface in Java?

What is serializable interface in Java? Serializable interface is a marker interface. The marker interface provides a hint to the Java runtime that the implementing class allows itself to be serialized. The runtime will take advantage of this interface to serialize the object.

5. What is deserialization in Java?

Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object. The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

6. How is serialization achieved in Java?

Java provides an automatic serialization mechanism that implements the java. io. Serializable interface for each class object marked as Serializable. The marker instructs the JVM to write a serializable object into the desired output byte stream using the ObjectOutputStream.

7. How is deserialization achieved in Java?

Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object. The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

8. How can you avoid certain member variables of class from getting Serialized?

So, if we don't want any variable to get serialized we need to assign it to transient type. Here, we are creating an object of the employee class and assigning values to the member variables of the class and then serializing the object

9. What classes are available in the Java IO File Classes API?

The following classes are available in the Java IM API and are important to work with files in Java.
File
RandomAccessFile
FileInputStream
FileReader
FileOutputStream
FileWriter

10. What is Difference between Externalizable and Serialization interface ?

Serializable	Externalizable
Serializable is a marker interface i.e. does not contain any method.	Externalizable interface includes two methods writeExternal() and readExternal() which implementing classes MUST override.
Serializable interface passes the responsibility of serialization to JVM and its default algorithm.	Externalizable provides control of serialization logic to the programmer – to write custom logic.
Mostly, default serialization is easy to implement, but has a higher performance cost.	Serialization done using Externalizable, add more responsibility to the programmer but often results in better performance.
It’s hard to analyze and modify class structure because any change may break the serialization.	It’s easier to analyze and modify class structure because of complete control over serialization logic.
Default serialization does not call any class constructor.	A public no-arg constructor is required while using Externalizable interface.