1. **What is Input and Output Stream in Java?**

Ans - A stream can be described as a continuous series of data. The InputStream is harnessed for extracting data from a source, whereas the OutputStream is employed to deposit data into a target location.

1. **What are the methods of OutputStream?**

Ans –

write(): This function is used to send the designated byte to the output stream.

write(byte[] array): This method writes the bytes from the provided array into the output stream.

flush(): By invoking this action, all data residing in the output stream is compelled to be written to the intended destination.

close (): This operation terminates and concludes the output stream.

1. **What is serialization in Java?**

Ans - Serialization involves transforming an object into a sequence of bytes, enabling its transmission across a network or its storage within a file or database. In Java, achieving serialization is accomplished by implementing the Serializable interface.

1. **What is the Serializable interface in Java?**

Ans - Within Java, the Serializable interface serves as a marker, devoid of methods. Its purpose lies in tagging classes as eligible for serialization, indicating that instances of these classes can be transformed into byte streams.

1. **What is deserialization in Java?**

Ans - Deserialization involves reversing the process: converting a sequence of bytes back into an object instance. This step is executed subsequent to an object's serialization.

1. **How is serialization achieved in Java?**

Ans - Java accomplishes serialization through the implementation of the Serializable interface. When an object is serialized, its state undergoes conversion into a byte stream, subsequently viable for transfer across networks or storage in files or databases.

1. How is deserialization achieved in Java?

Ans - Java achieves deserialization by consuming a byte stream and utilizing it to reconstruct the initial object instance. This procedure involves invoking the readObject() method from an ObjectInputStream instance.

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

Ans - By designating member variables as static or transient, you exclude them from the serialization process.

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

Ans –

* File
* RandomAccessFile
* FileInputStream
* FileReader
* FileOutputStream
* FileWriter

1. **What is Difference between Externalizable and Serialization interface?**

Ans - The difference between the Externalizable and Serializable interfaces lies in the level of control and customization they offer during the serialization process:

**Serializable Interface:**

* Standard Java serialization mechanism.
* No need to implement specific methods.
* Automatically serializes all non-transient, non-static fields.
* Limited customization; no direct control over serialization process.

**Externalizable Interface:**

* Provides more control over serialization.
* Requires implementation of readExternal() and writeExternal() methods.
* Allows selective serialization of fields, providing control over which data to serialize.
* Can lead to more efficient serialization for specific use cases.