day-029-io-operations

1. What is Input and Output Stream in Java?
2. What are the methods Of OutputStream?
3. What is serialization in Java?
4. What is the Serializable interface in Java?
5. What is deserialization in Java?
6. HOW is serialization achieved in Java?
7. How is deserialization achieved in Java?
8. How can you avoid certain member variables of class from getting Serialized?
9. What classes are available in the Java IO File Classes APP
10. What is Difference between Externalizable and Serialization interface.

Answers

1. Input and Output Stream in Java

Input and output streams are used to read and write data to and from a variety of sources, such as files, networks, and other devices. Streams are a fundamental part of Java's I/O system, and they are used by many different classes and libraries.

2. Methods Of OutputStream

The OutputStream class has a number of methods for writing data to an output stream. These methods include:

- write(byte[]): Writes the specified byte array to the output stream.
- write(int): Writes the specified byte to the output stream.
- write(byte[], int, int): Writes the specified number of bytes from the specified byte array to the output stream.
- flush(): Flushes the output stream, ensuring that all data has been written to the underlying device.
- close(): Closes the output stream, releasing any resources that it is using.

3. Serialization in Java

Serialization is the process of converting an object into a sequence of bytes that can be stored or transmitted. This sequence of bytes can then be used to recreate the object at a later time. Serialization is often used to save objects to files or to transmit them over a network.

4. The Serializable interface in Java

The Serializable interface is a marker interface that indicates that an object is eligible for serialization. Objects that implement the Serializable interface can be serialized and deserialized using the ObjectOutputStream and ObjectInputStream classes.

5. **Deserialization in Java**

Deserialization is the process of converting a sequence of bytes into an object. This sequence of bytes can be obtained by reading it from a file or by receiving it over a network. Deserialization is the opposite of serialization, and it is used to recreate objects that have been serialized.

6. **Serialization is achieved in Java**

Serialization is achieved in Java by using the ObjectOutputStream class. The ObjectOutputStream class can be used to serialize any object that implements the Serializable interface. To serialize an object, you create an ObjectOutputStream object and then call the writeObject() method on the object. The writeObject() method writes the object's state to the output stream.

7. **Deserialization is achieved in Java**

Deserialization is achieved in Java by using the ObjectInputStream class. The ObjectInputStream class can be used to deserialize any object that has been serialized. To deserialize an object, you create an ObjectInputStream object and then call the readObject() method on the object. The readObject() method reads the object's state from the input stream and recreates the object.

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

You can avoid certain member variables of a class from getting serialized by marking them with the transient keyword. The transient keyword indicates that the member variable should not be serialized.

9. **What classes are available in the Java IO File Classes APP**

The Java IO File Classes APP contains the following classes:

- File: Represents a file on the file system.
- FileInputStream: Represents an input stream for reading from a file.
- FileOutputStream: Represents an output stream for writing to a file.
- FileWriter: Represents a writer for writing to a file.
- FileReader: Represents a reader for reading from a file.
- FileNotFoundException: Thrown when an attempt is made to open a file that does not exist.
- IOException: Thrown when an I/O error occurs.

10. What is Difference between Externalizable and Serialization interface

The Externalizable interface is similar to the Serializable interface, but it provides more control over the serialization process. The Externalizable interface allows you to specify which fields should be serialized and which fields should not be serialized. The Externalizable interface also allows you to write your own serialization and deserialization methods.