14. File Handling

1) Introduction to File I/O in Java (java.io package)

Answer:

In Java, file input and output (I/O) is handled through the java.io package, which provides classes for reading from and writing to files, as well as manipulating data streams and byte data. The **java.io** package includes classes and interfaces for various types of I/O operations, both at the byte and character levels.

---

1. Streams in Java

In Java, I/O operations are performed using **streams**. A stream is a sequence of data, which can be either an **input stream** (for reading data) or an **output stream** (for writing data).

- **Byte Streams**: Handle raw binary data (e.g., images, audio, etc.).
- **Character Streams**: Handle data in the form of characters (text data).

---

2. Byte Streams (InputStream and OutputStream)

Byte streams provide a mechanism for reading and writing binary data. They are used for handling **binary files** (e.g., images, audio files, etc.).

- **InputStream**: Abstract class for reading byte data from a source.
- **OutputStream**: Abstract class for writing byte data to a destination.

2) FileReader and FileWriter Classes

Answer:

The **FileReader** and **FileWriter** classes are part of the **java.io** package and are used for reading and writing text data to and from files. These classes are **character-based streams**, meaning they are designed to handle **characters** (not bytes) and are ideal for text files.

3) BufferedReader and BufferedWriter

Answer:

The **BufferedReader** and **BufferedWriter** classes are part of the **java.io** package and are used for efficient reading and writing of text data. These classes provide **buffered I/O**, meaning they

read or write data in larger chunks, thus improving performance compared to using unbuffered streams (like FileReader and FileWriter).

- **BufferedReader** is used for reading text from an input stream (e.g., a file).
- **BufferedWriter** is used for writing text to an output stream (e.g., a file).

The key advantage of buffered I/O is that it minimizes the number of I/O operations by reading and writing larger blocks of data at once, rather than character by character or byte by byte.

4) Serialization and Deserialization

Answer:

Serialization is the process of converting an object into a byte stream so that it can be easily stored in a file, transferred over a network, or saved in a database. **Deserialization**, on the other hand, is the process of converting the byte stream back into the original object.

Java provides built-in mechanisms for serializing and deserializing objects using the **Serializable** interface and **ObjectOutputStream** and **ObjectInputStream** classes.

==========
==========
==========
==========