**Core Java - Assignment**

**Module - 1**

1. **File Handling :**

**Theory : -**

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

-> File I/O (Input/Output) in Java is a fundamental aspect of programming that allows you to read from and write to files. The java.io package provides a wide range of classes and methods to handle file operations. Here's an introduction to some of the key classes and concepts in the java.io package.

-> FileReader : Reads character files.

-> FileWriter : Writes character files.

-> FileInputStream: Reads raw byte streams from a file.

-> FileOutputStream: Writes raw byte streams to a file.

1. FileReader and FileWriter Classes

-> FileReader Class : The FileReader class is used to read character files. It extends the InputStreamReader class and provides methods to read characters, arrays of characters, and lines of text.

-> FileWriter Class : The FileWriter class is used to write character files. It extends the OutputStrearüriter class and provides methods to write characters, arrays of characters, and strings.

1. BufferedReader and BufferedWriter

-> BufferedReader Class : The BufferedReader class is used to read text from an input stream efficiently. It buffers the characters, so it can read large chunks of data at once, reducing the number of I/O operations.

Key Methods

-> read() : Reads a single character.

-> read(char[] cbuf, int off, int len) : Reads characters into a portion of an array.

-> readLine() : Reads a line of text.

-> close() : Closes the stream and releases any system resources associated with it.

-> BufferedWriter Class : The BufferedWriter class is used to write text to an output stream efficiently. It buffers the characters, so it can write large chunks of data at once, reducing the number of I/O operations.

Key Methods

-> write(int c) : Writes a single character.

-> write(char[] cbuf, int off, int len) : Writes a portion of an array of characters.

-> write(String s, int off, int len) : Writes a portion of a string.

-> newLine() : Writes a line separator.

-> flush() : Flushes the stream.

-> close() : Closes the stream and releases any system resources associated with it.

1. Serialization and Deserialization

-> Serialization : Serialization is the process of converting an object into a byte stream. This byte stream can then be saved to a file or sent over a network.

-> Serializable: A marker interface that indicates that a class can be serialized.

-> ObjectOutputStream: A class used to write objects to an output stream.

-> Deserialization : Deserialization is the process of converting a byte stream back into an object. This

allows the object to be reconstructed from the byte stream.

-> Serializable: A marker interface that indicates that a class can be deserialized.

-> ObjectlnputStream: A class used to read objects from an input stream.