DAY-17

# JAVA:

## STREAM API:

* The Stream API in java is introduced in Java 8.
* And it is a powerful tool for processing collections of objects in a functional style.
* It allows you to perform operations like filtering, mapping, and reducing data in a declarative way.

STREAM:

* A sequence of elements supporting sequential and parallel aggregate operations.
* A stream is composed of bytes.
* They are 3 types:
* System. out => Standard output stream
* System. In => Standard input stream
* System .err => Standard error stream

BENEFITS OF STREAM:

* Concise and readable code.
* Parallel processing with .parallelStream()
* Functional Programming style

Java provides a package in Java 8 called java.util.stream. This package consists of classes, interfaces, and Enum to allow functional-style operations on the elements.

OPERATIONS OVER STREAMS:

* Intermediate Operations
* Terminal Operations
* Short-Circuit Operations

## JAVA I/O:

Java I/O is used to process input and produce output.

Java uses the concept of a stream to make I/O operations

We use java.io package;

STREAMS:

* It represents a sequence of data.
* It has two types:
* Input streams
* Output Streams
* Streams can be categorized into byte streams (I/O streams) and character streams (Readers and Writers).

READERS AND WRITERS:

Readers and Writers are specialized stream classes designed for handling character data.

They provide a convenient way to read from and write to character-based data sources.

OUTPUT STREAM:

It is used to write data to a destination; it may be a file, an array, peripheral device, or a socket.

INPUT STREAM:

It is used to read from a source; it may be a file, an array, peripheral device, or a socket.

BYTE STREAMS (I/O STREAMS):

It is used for handling binary data.

Examples:

1. FileInputStream
2. FileOutputStream
3. BufferedInputStream
4. BufferedOutputStream

CHARACTER STREAMS (I/O STREAMS):

It is used for handling text data.

Examples:

1. FileReader
2. FileWriter
3. BufferedReader
4. BufferedWriter