Stream API

04/07/2024

- Stream API is used to process collections of objects.
A stream in Java is a sequence of objects that supports various methods which can be pipelined to produce the desired result.
* Stream API is a way to express and process collections of objects.
* Enable us to perform operations like filtering, mapping, reducing and sorting.

Why Stream API:

* enhances the usability of Java collection types, making it easy to iterate and perform tasks against each element in the collection
* Supports sequential and parallel aggregate operations.
* Stream API allows developers process data in a declarative way.

Key points:

* Stream defines many operations, which can be grouped in two categories
* Intermediate operations.
* Terminal operations
* Stream operations that can be connected are called intermediate operations.
* They can be connected together because their return type is a Stream.
* Operations that close a stream pipeline are called terminal operations.
* Intermediate operations are "lazy

Intermediate Operations:

* When an operation on a stream further produces another stream as a result, we call it an intermediate operation.
* As intermediate operations return another stream as a result, they can be chained together to form a pipeline of operations.

* As the word "intermediate" suggest, these operations doesn't give end result.
* They just convert one stream to another stream.
* For example: map(), filter(), distinct(), sorted(), limit(), skip()

Terminal operations:

* The operations which return non-stream values such as primitive or object are called terminal operations.
* Furthermore, unlike intermediate operations, we can't chain them together.
* They produce the end result.
* Once a terminal operation completes, the stream is no longer valid.
* Hence, we can't use that stream again.
* For example: forEach(), toArray(), reduce(), collect(), min(), max(), count(), anyMatch(), allMatch(), noneMatch(), findFirst(), findAny()
* Stream processing consists of a source of zero or more intermediate operations followed by a terminal operation.
* Each intermediate operation returns a new stream. The terminal operation returns something other than a stream.