Streams API

Before we present our solution to this project, it is a good idea to spend some time talking about what Streams API is. Our intention is not provide an in-depth overview about Java 8 streams, but some definitions and the description of the streams operations used in our solution are required. For more details about Streams API, visit the link below:

‘’’’’’’’’’’’’’’’’’’’’’Include link’’’’’’’’’’’’’’’’’’’

The definition of a stream is “a sequence of elements from a source that supports aggregate operations”. That source can be collections, arrays, or I/O resources.

Streams operations are either intermediate or terminal. Intermediate operations return a stream so we can chain multiple intermediate operations (i.e filter and map operations). Terminal operations are located at the end of the pipeline and they either return void or a non-stream result (i.e forEach and collect operations). One of the most important characteristic of intermediate operations is laziness, they are not execute unless a terminal operation exists in the pipeline operation.

Include an example about that.

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

Java Streams are consumable, so there is no way to create a reference to stream for future usage. Since the data is on-demand, it is not possible to reuse the same stream multiple times. Although, we can overcome this limitation using Supplier. We will talk in detail about this later on section …….

Stream operations do the iterations internally over the source elements provided, in contrast to Collections where explicit iteration is required

Streams Operations

Stream operations do the iterations internally over the elements, in contrast to Collections where explicit iteration is required. Streams provides plenty of different operations, but here we will mention only the operations needed for our project in order to simulate SQL statements.

*filter*

This is the most straight-forward mapping operation that we can do in Java using Streams for simulating the WHERE clause in SQL. It allows us to get a stream of elements which satisfy a predicate.

Include an example

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

*map*

With this operation we can modify the elements of a *Stream* applying functions over them and even transform the elements into new objects.

Include an example

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

*flatMap*

If you have a stream where every element contains its own sequence of elements and you want to create a stream of these inner elements, you need the operation *Stream.flatMap*.

Include an example

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

*concat*

This operation allows you to concat or merge two streams.

Include an example

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

*collect*

It performs a reduce operation on the stream elements. *Stream.collect()* works with one argument as collector or three arguments as supplier, accumulator and combiner using lambda expression. Here will discuss an example of *Stream.collect()* with *Collector*.

With Collector we can transform the elements of a stream in a List, Set, Map or String. Collector also allow you to perform aggregate operations such as summation, grouping, joining, and average.

Include an example

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

Relational Algebra

The relational algebra is a *procedural* query language, which means that you must specify the data that you need, and what are the steps to get those data. The relation algebra consists of a set of operations that take one or two relations as input and return a new relation as their result. The relational algebra forms the basis of the widely used SQL query language.

The fundamental operations in the relational algebra are select, project, union, set difference, Cartesian product, and rename. In addition to the fundamental operations, there are several other operations—namely, set intersection, natural join, and assignment. We shall define these operations in terms of the fundamental operations.

The select, project, and rename operations are called unary operations, because

they operate on one relation. The other three operations operate on pairs of

relations and are, therefore, called binary operations.

Select

The **select** operation selects tuples that satisfy a given predicate. We use the

lowercase Greek letter sigma (\_) to denote selection.

Projection

Union

Set-Difference