***Common Intermediate Functions***

filter(Predicate) => Filters elements

map(Function) => Transforms elements

flatMap(Function) => Flattens nested streams

distinct() => Removes duplicates

sorted() => Sorts elements

limit(long) => Limits the number of elements

skip(long) => Skips elements

peek(Consumer) => For debugging

***Common Terminal Functions***

forEach(Consumer) => Consumes each element

collect(Collector) => Produces a collection or summary

reduce() => Combines elements into one

count() => Returns count

min(), max() => Returns min/max element

anyMatch(Predicate) => Checks if any match

allMatch(Predicate) => Checks if all match

noneMatch(Predicate) => Checks if none match

toArray() => Converts to array

***Parallel Stream LifeCycle:-***

***1)Source***

***-*** Collection.parallelStream()

- Stream.parallel()

***2️)ForkJoinPool***

***-*** Internally used by parallel streams for task splitting and joining.

- Default pool size = Number of available processors.

***3️)Intermediate Operations***

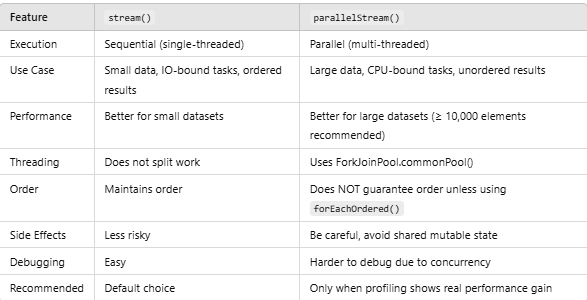
***-*** Same as Stream API: filter(), map(), flatMap(), distinct(), sorted(), etc.

***4️)Terminal Operations***

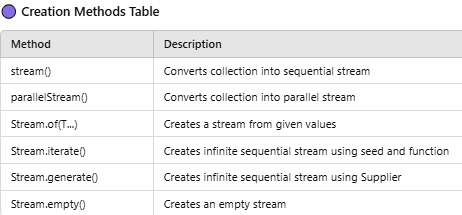
***-*** Same as Stream API: collect(), reduce(), count(), forEach(), etc.

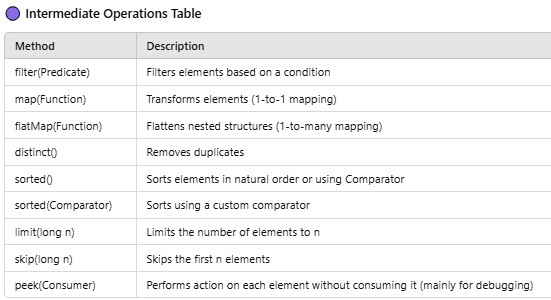
***Notes:***

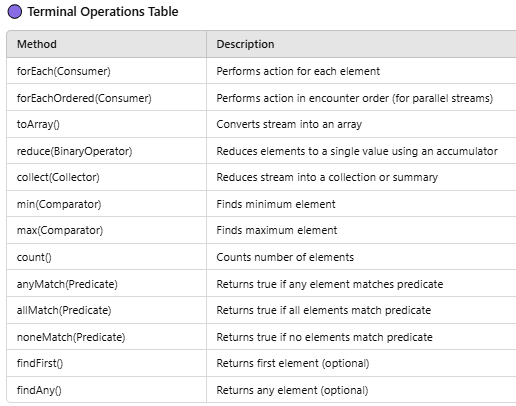
* Parallel streams use multiple threads.
* They split the data, process chunks in parallel, and combine the results.
* Parallel streams may not preserve the encounter order unless specifically instructed.

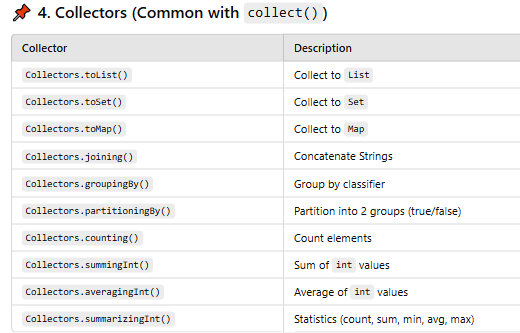












**Notes:**

* **All these methods** are used in both Stream and ParallelStream.
* Only difference:  
  .stream() creates a **sequential stream**  
  .parallelStream() creates a **parallel stream**
* ***Drawbacks and limitations of Stream API:-***

