List<Integer> l = Arrays.asList(1,2,3,4);

l.forEachI(n -> sout(n)); //better way to do this is first create a stream then use the method as once you consumed the stream you can’t reuse use it and list is not changed as a stream is created.

\*\*\*\*

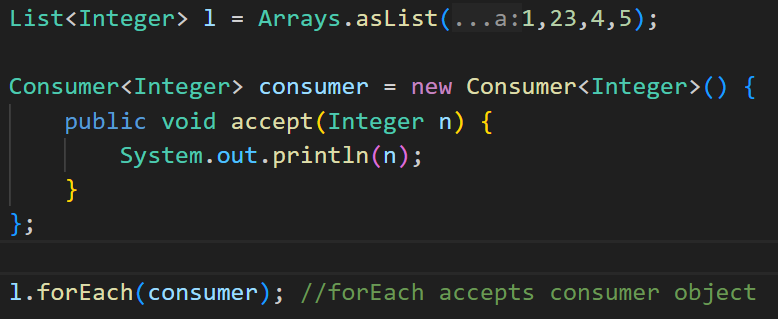
ForEach does not returns anything .

Map returns a new stream.

Sorted returns a new stream.

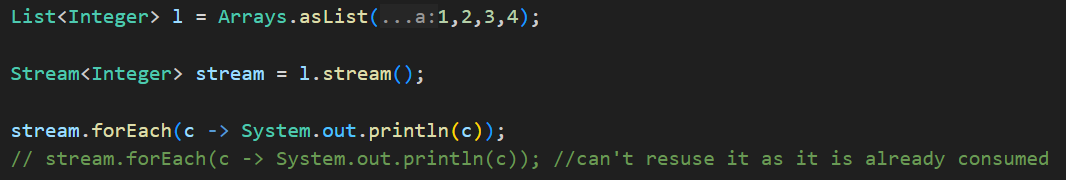
Filter returns a new stream.

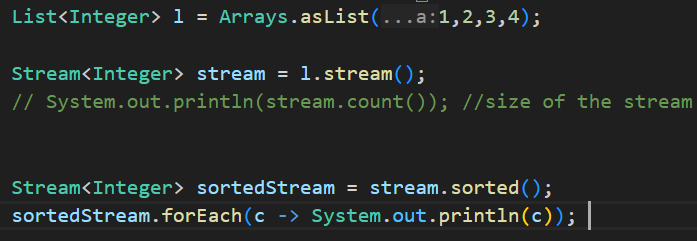
Reduce returns one single value.



A computer screen shot of a computer code

Description automatically generated



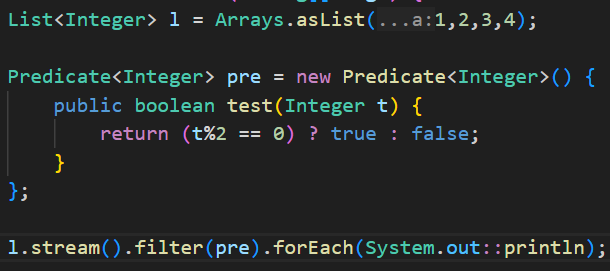


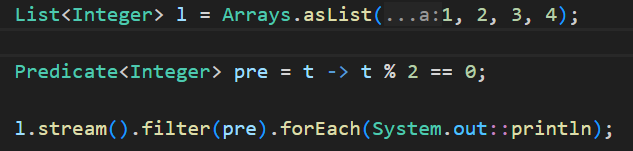
A screen shot of a computer program

AI-generated content may be incorrect.

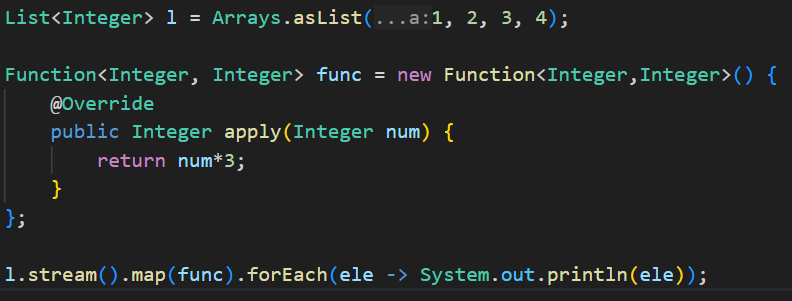
\*\*Decode Filter:

Filter accepts a functional Interface know as Predicate, Predicate have an abstract method called test. Filter is a method in stream Interface.

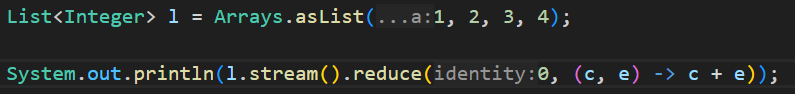


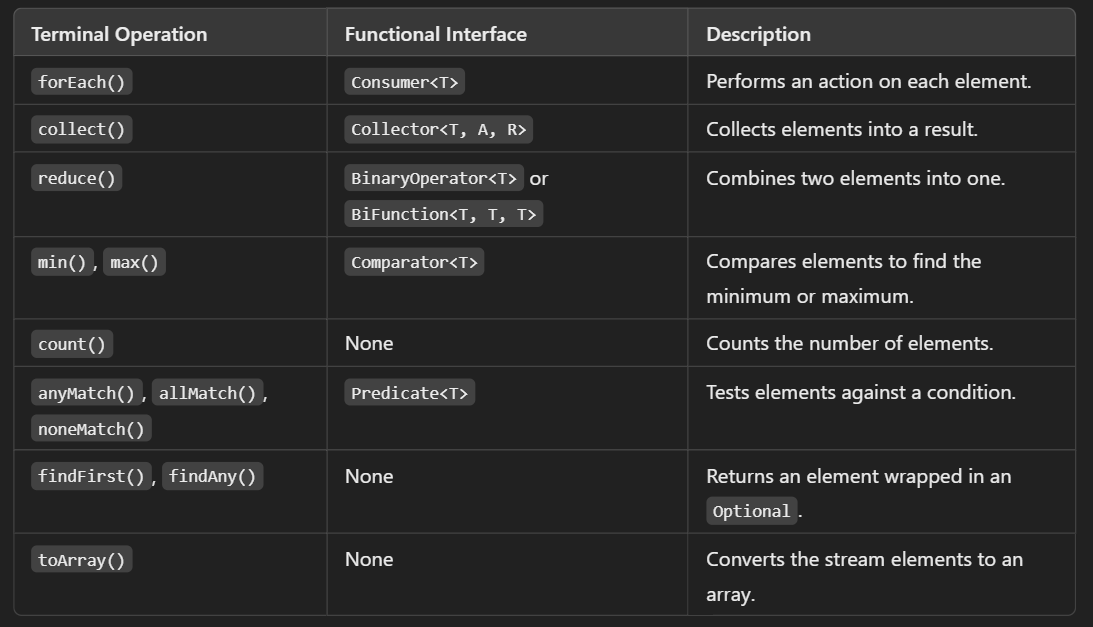


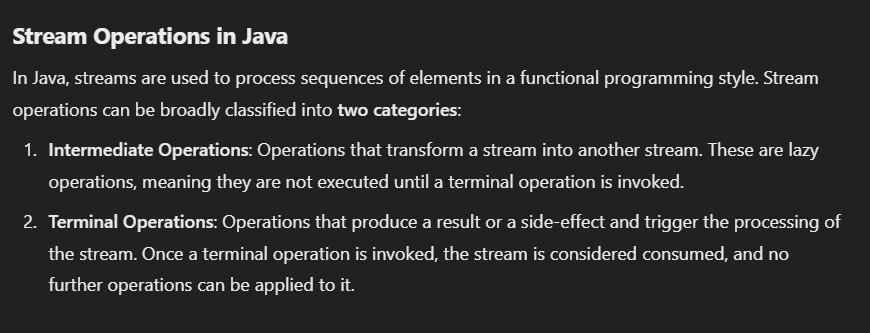
\*\*Decode Map



\*\*Reduce Method

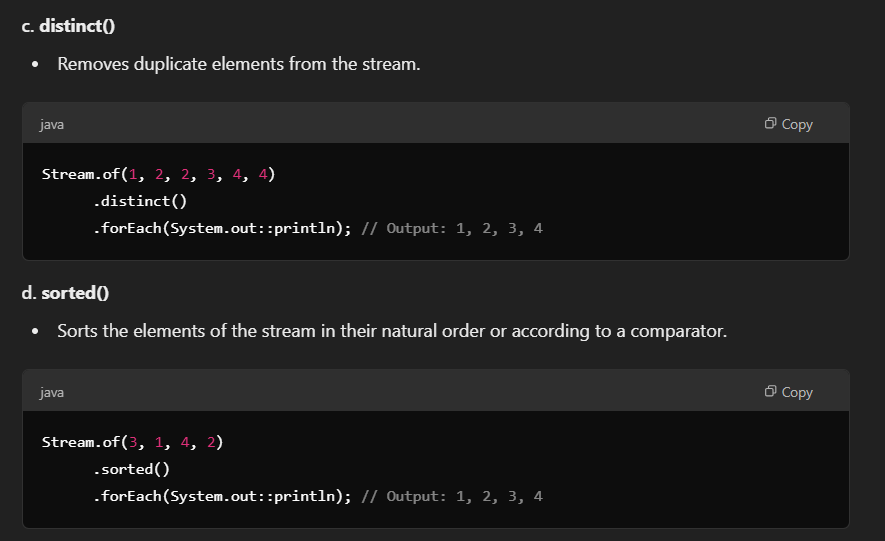






A screenshot of a computer program

AI-generated content may be incorrect.



A screenshot of a computer program

AI-generated content may be incorrect.

flatMap(list -> list.stream())

flatMap(Collection::stream)

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

