* Stream API is used to process collections of objects.
* A stream is not a data structure instead it takes input from the Collections.
* Streams don’t change the original data structure, they only provide the result.
* In Streams you can pipeline various operations to get a desired output.

A Stream has different types of Operation:

**Intermediate Operations:**

* Each intermediate operation returns a stream as a result.
* Hence various intermediate operations can be pipelined.

**Terminal Operations:**

* Terminal operations mark the end of the stream and return the result.

**Intermediate Operations:**

**map:** The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.

List<Integer> number = Arrays.*asList*(2,3,4,5);

List<Integer> square = number.stream().map(x->x\*x).collect(Collectors.*toList*());

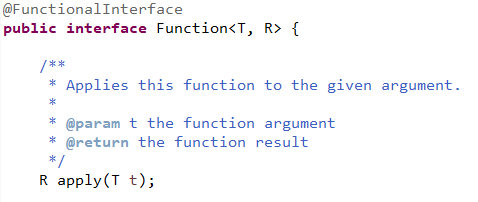
System.***out***.println(square);

**Output:**

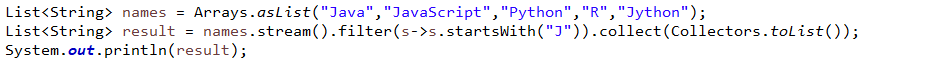
[4, 9, 16, 25]

The map method accepts a Function Interface.



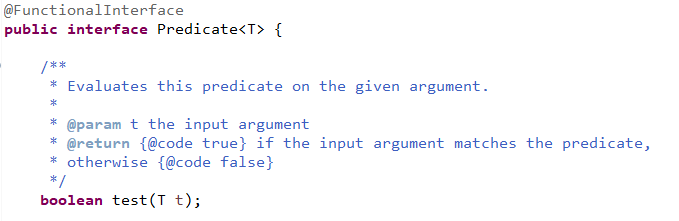


**filter:** The filter method is used to select elements as per the Predicate passed as argument.

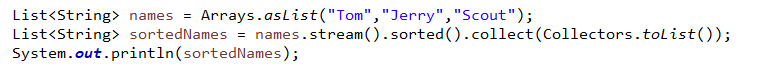


The filter method accepts a Predicate Interface, which is a Functional Interface.





**sorted:** The sorted method is used to sort the stream.



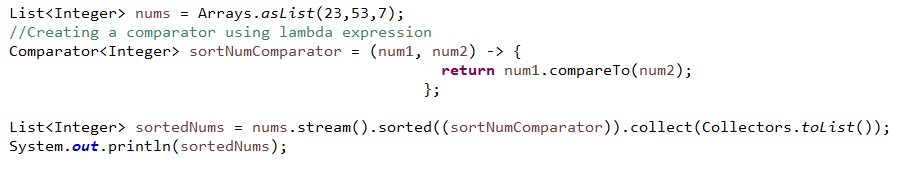
sorted method return Stream.



If you want to sort a stream, you can use the overloaded method of sorted, which accepts a Comparator.

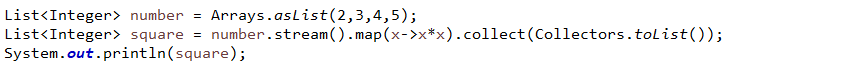


Example:



**Terminal Operations:**

**collect:** The collect method is used to return the result of the intermediate operations performed on the stream.



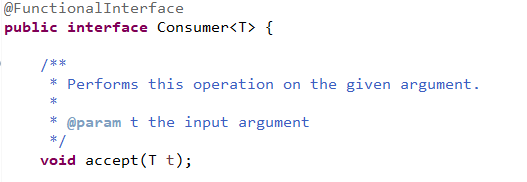
The collect method accepts an Interface of type Collector:



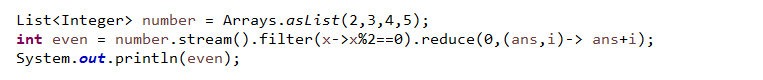
**forEach:** The forEach method is used to iterate through every element of the stream.  


The forEach Method accepts a Functional Interface called Consumer





**reduce:** The reduce method is used to reduce the elements of a stream to a single value.



The reduce method takes a BinaryOperator as a parameter.



So, the BinaryOperator is a Functional Interface



And the abstract method by name “apply” is present inside the BiFunction Functional Interface.

