**STREAM API IN JAVA 8**

Stream is a sequence of objects on which various operations can be applied to produce desired results.

**Intermediate operations** are lazily executed, return stream and could be pipelined while **terminal operations** mark the end of stream and return the result. There could be 0 or more intermediate operations and at least 1 terminal operation.

Streams never change the original data structure. Only provide results as per defined methods.

Stream is not a data structure. They take input from Collections, Arrays and I/O

**Intermediate Operations:**

1. **Map**: maps elements of a collection to other objects as per the predicate passed

List number = Arrays.asList(2,3,4,5);  
List square = number.stream().map(x->x\*x).collect(Collectors.toList());

1. **Filter**: filters the elements from a collection based on the predicate passed

List names = Arrays.asList("Reflection","Collection","Stream");  
List result = names.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());

1. **Sorted**: sorts the elements of the stream

List names = Arrays.asList("Reflection","Collection","Stream");  
List result = names.stream().sorted().collect(Collectors.toList());

**Terminal Operations**:

1. **Collect**: returns the result generated from intermediate operations

List number = Arrays.asList(2,3,4,5,3);  
Set square = number.stream().map(x->x\*x).collect(Collectors.toSet());

1. **forEach**: iterates the stream

List number = Arrays.asList(2,3,4,5);  
number.stream().map(x->x\*x).forEach(y->System.out.println(y));

1. **reduce**: takes a binary operator and returns a single result from a stream of objects

List number = Arrays.asList(2,3,4,5);  
int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);