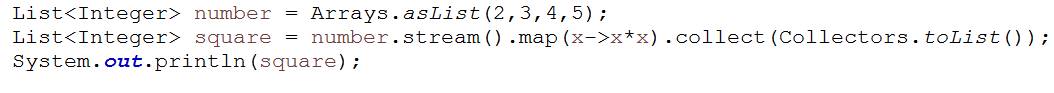
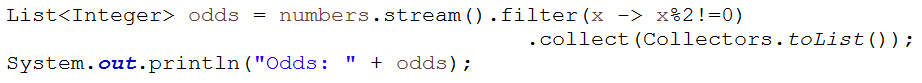
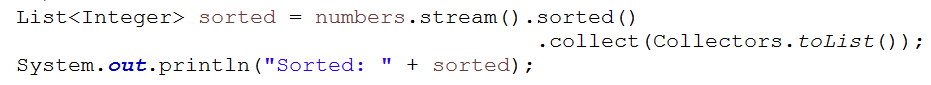
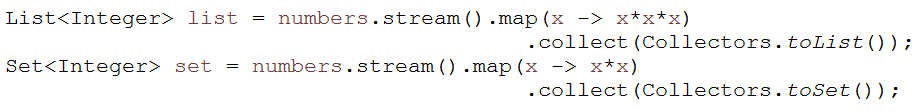
**Stream API**

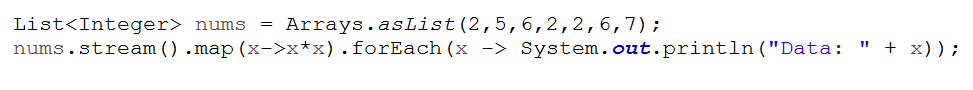
* Stream API is used to process collections of objects.
* A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.
* A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
* Streams don’t change the original data structure, they only provide the result as per the pipelined methods.
* Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

**Intermediate Operations:**

1. **map:** The map method is used to map the items in the collection to other objects according to the Function passed as argument.  
   
2. **filter:** The filter method is used to select elements as per the Predicate passed as argument. 
3. **sorted:** The sorted method is used to sort the stream. 

**Terminal Operations:**

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream. 
2. **forEach:** The forEach method is used to iterate through every element of the stream.



1. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.
2. public class App
3. {
4. public static void main(String[] args) {
5. List<Integer> numbers = Arrays.*asList*(1, 2, 3, 4, 5);
6. int sum = numbers.stream()
7. .reduce(0, (a, b) -> a + b);
8. System.***out***.println("Sum: " + sum);
9. }
10. }

**Output: 15**

**In this example:**

* The stream is created from a list of integers.
* The reduce method is called with 0 as the identity value (the starting value for the sum).
* The accumulator function takes two integers and returns their sum. This function is applied repeatedly across the elements of the stream to produce a single sum value.
* The result is printed to the console.

What are the practical use case of reduce()?

**Summing Elements**

List<Integer> numbers = Arrays.asList(3, 5, 8, 9, 12);

int sum = numbers.stream().reduce(0, Integer::sum);

### ****Finding Maximum or Minimum****

List<Integer> numbers = Arrays.asList(3, 5, 8, 9, 12);

int max = numbers.stream().reduce(Integer.MIN\_VALUE, Integer::max);

int min = numbers.stream().reduce(Integer.MAX\_VALUE, Integer::min);

### ****Concatenating Strings****

List<String> words = Arrays.asList("Java", "Stream", "API");

String concatenated = words.stream().reduce("", (a, b) -> a + " " + b).trim();