**Java 8 Stream API**

**Stream in Java 8** can be defined as a sequence of elements from a source. Streams supports aggregate operations on the elements. The source of elements here refers to a [Collection](https://howtodoinjava.com/java-collections/) or [Array](https://howtodoinjava.com/java-array/) that provides data to the Stream.

Stream keeps the ordering of the elements the same as the ordering in the source. The **aggregate operations** are operations that allow us to express common manipulations on stream elements quickly and clearly.

**2. Creating Streams**

The given below ways are the most popular different ways to build streams from collections.

**2.1. Stream.of()**

In the given example, we are creating a stream of a fixed number of integers.

|  |
| --- |
| public class StreamBuilders  {       public static void main(String[] args)       {           Stream<Integer> stream = Stream.of(1,2,3,4,5,6,7,8,9);           stream.forEach(p -> System.out.println(p));       }  } |

**2.2. Stream.of(array)**

In the given example, we are creating a stream from the array. The elements in the stream are taken from the array.

|  |
| --- |
| public class StreamBuilders  {       public static void main(String[] args)       {           Stream<Integer> stream = Stream.of( new Integer[]{1,2,3,4,5,6,7,8,9} );           stream.forEach(p -> System.out.println(p));       }  } |

**2.3. List.stream()**

In the given example, we are creating a stream from the [List](https://howtodoinjava.com/java-arraylist/). The elements in the stream are taken from the List.

|  |
| --- |
| public class StreamBuilders  {       public static void main(String[] args)       {           List<Integer> list = new ArrayList<Integer>();             for(int i = 1; i< 10; i++){               list.add(i);           }             Stream<Integer> stream = list.stream();           stream.forEach(p -> System.out.println(p));       } |