Stream API in Java :

Introduced in Java 8.

It is Used for fast processing of data over Collection & Array.

Stream is not a data structure.

It represents a sequence of elements from a source.

It supports aggregate operations.

What is Stream interface :

Stream is a predefined interface available in java.util.stream sub package.

It provides various methods for data processing.

It was introduced in Java 8 to represents sequence of elements.

public interface Stream<T> extends BaseStream<T, Stream<T>> {}

Stream interface contains forEach(Consumer<T> method) method which is a terminal operation.

Interfaces which contains forEach() method :

The forEach() method is available in the following interfaces :

1) java.lang.Iterable [forEach(Consumer<T> cons)]

2) java.util.Map [forEach(BiConsumer<K,V> cons)]

3) java.util.stream.Stream [forEach(Consumer<T> cons)]

Creation of Stream for Processing of Data :

1) Collection interface has provided the following two default methods to convert the Collection Object into Stream Object for fast Data processing

   a) default java.util.stream.Stream stream() : It will convert the Collection object into Stream object for data processing in a single thread application.

   b) default java.util.stream.Stream parallelStream() : It will convert the Collection object into Stream object for data processing in a multithread application.

//Program on stream() method :

```
package com.ravi.basic;

import java.util.ArrayList;
import java.util.stream.Stream;

public class StreamCreation
{
    public static void main(String[] args)
    {
           ArrayList<String> listOfCity = new ArrayList<>();
           listOfCity.add("Hyderabad");
           listOfCity.add("Indore");
           listOfCity.add("Pune");
           listOfCity.add("Kolkata");
           listOfCity.add("Haryana");


           //Converting the Collection into Stream
           Stream<String> cities = listOfCity.stream();
           cities.forEach(System.out::println);

    }

}
```

//Program on parallelStream() :

```
package com.ravi.basic;

import java.util.ArrayList;
import java.util.stream.Stream;

public class StreamCreation
{
    public static void main(String[] args)
    {
           ArrayList<String> listOfCity = new ArrayList<>();
           listOfCity.add("Hyderabad");
           listOfCity.add("Indore");
           listOfCity.add("Pune");
           listOfCity.add("Kolkata");
           listOfCity.add("Haryana");


           //Converting the Collection into Stream
           Stream<String> parallelStream = listOfCity.parallelStream();
           parallelStream.forEach(System.out::println);

    }

}
```

Note : Output is not predictable because multiple threads are working on the Stream object

---

2) Arrays class has provided the following methods to convert the array into Stream for fast data processing.

   a) public static IntStream stream(int []arr);
   b) public static LongStream stream(long []arr);
   c) public static DoubleStream stream(double []arr);
   d) public static <T> Stream stream(T[] x)

```
package com.ravi.basic;

import java.util.Arrays;
import java.util.stream.DoubleStream;
import java.util.stream.IntStream;
import java.util.stream.LongStream;
import java.util.stream.Stream;

public class StreamCreation
{
    public static void main(String[] args)
    {
       int x[] = {10,20,30,40};
       IntStream stream = Arrays.stream(x);
       stream.forEach(System.out::println);

       System.out.println("...............");

       long []y = {1L, 2L, 6L, 9L};
       LongStream stream2 = Arrays.stream(y);
       stream2.forEach(System.out::println);

       System.out.println("...............");

       double []z = {1D, 2D, 3D, 4D};
       DoubleStream stream3 = Arrays.stream(z);
       stream3.forEach(System.out::println);

       System.out.println("...............");

       String []cities = {"Hyd", "Indore", "Pune"};
       Stream<String> stream4 = Arrays.stream(cities);
       stream4.forEach(System.out::println);

    }

}
```

3) public static <T> Stream of(T ...values)

It is a static method of Stream interface through which we can create Stream of arrays and Stream of Collection. The return type of this method is Stream interface.

```
package com.ravi.basic;

import java.util.stream.Stream;

public class StreamCreation
{
    public static void main(String[] args)
    {
       Stream<Integer> stream = Stream.of(1,2,3,4,5,6);
       stream.forEach(System.out::println);
    }

}
```

4) How to generate an Infinite Stream :

Stream interface has provided the following two static methods to generate infinite Stream.

a) public static <T> Stream<T> generate(Supplier<? extends T> s) :

* Will generate Infinite Stream.

```
import java.util.stream.Stream;

public class StreamCreation
{
    public static void main(String[] args)
    {
      Stream<Double> stream = Stream.generate(()-> Math.random());   //[0.0 to 0.9]
      stream.forEach(System.out::println);
    }

}
```

b) public static Stream iterate(final T seed, final UnaryOperator<T> x) :

* It will generate Inifinite Stream. Here T seed represents the starting point and UnaryOperator will produce the output same as input type.

```
package com.ravi.basic;

import java.util.stream.Stream;

public class StreamCreation
{
    public static void main(String[] args)
    {
     Stream<Double> iterate = Stream.iterate(51.0, n -> n + 1);
     iterate.limit(10).forEach(System.out::println);
    }

}
```