Stream API in Java

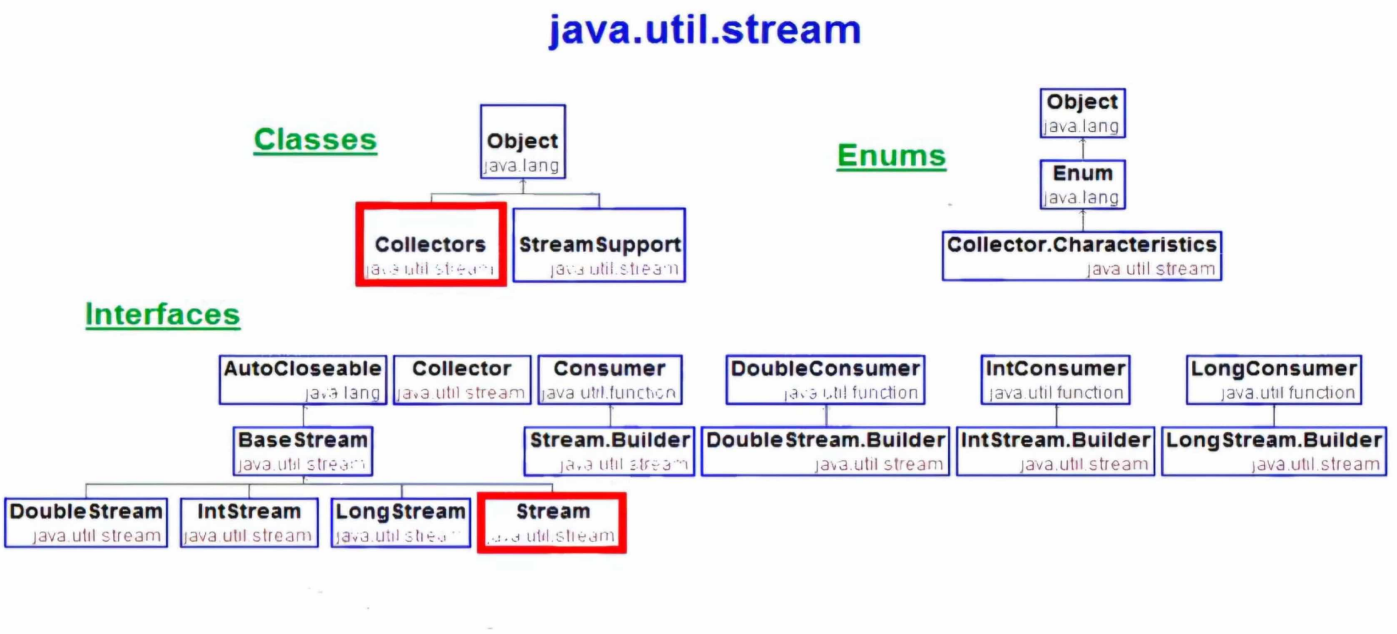
=> Stream API :-

-Stream API is related to "Collection Framework" or "Group of Objects"

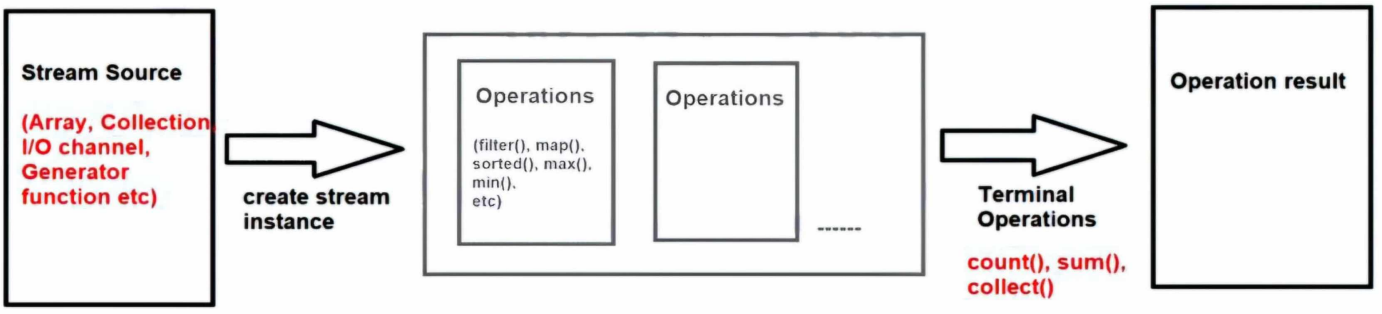
-Note : Stream API is not related to Java IO Stream, it is totally different from IO stream

-Stream API was introduced in Java SE 8 version

➔ Hierarchy of Stream API :-



➔ Stream API Operations Lifecycle :-



1. Get stream from the source and create stream instance

2. Perform operations(0,1 or many operations) such as filter(), sort(), map() etc and transform it into another stream

3. Perform terminal operation such as count(), sum() etc and produces the result

=> Stream Interface :-

-> Stream is an interface which is present in java.util.stream package

-> Syntax :

```
public interface Stream
{
    forEach();
    filter();
    map();
    collect();
    sorted();
    min();
    max();
    count();
 //some static methods
    of() { - }
    empty() { - }
    builder() { - }
```

```java
// How to get Stream instance

public class Test1
{
    public static void main(String[] args)
    {
        ArrayList<Integer> al=new ArrayList<>();    //Stream source
        al.add(10);
        al.add(20);
        al.add(30);
        al.add(40);
        al.add(50);

        //------1st way to get Stream instance-----
//        Stream<Integer> s1=al.stream();
//        s1.forEach(e->System.out.println(e));

        //------2nd way to get Stream instance-----
//        Stream s2=Stream.of(al);
//        s2.forEach(e->System.out.println(e));

        //------3rd way to get Stream instance-------
        //Stream s3=Stream.of(100,200,300,400,500);
//        Stream s3=Stream.of("Arun", "Rohit", "amit", "Sumit", "Divas");
//        s3.forEach(e->System.out.println(e));

        //------4th way to get Stream instance----
//        int[] arr={600,700,800,900,1000};
//        IntStream s4=Arrays.stream(arr);
//        s4.forEach(e->System.out.println(e));

        //------5th way to get Stream empty instance-----
//        Stream s5=Stream.empty();

        //-----6th way to get Stream instance-----
//        Stream s6=Stream.builder().build();
    }
}
```

```java
//WAP to get all the even numbers in an arraylist
//WAP to count all the even numbers in an ArrayList

public class Test2
{
    public static void main(String[] args)
    {
        ArrayList<Integer> al=new ArrayList<>();
        al.add(10);
        al.add(20);
        al.add(30);
        al.add(40);
        al.add(55);

        //-----------------using collections and simple logic--------------------
//        for(int i:al)
//        {
//            if(i%2==0)
//            {
//                System.out.println(i);
//            }
//        }

        //-----------------using stream api (first part)-----------------------------
//        Stream<Integer> s=al.stream();
//        List l=s.filter(e->e%2==0).collect(Collectors.toList());
//        System.out.println(l);

        //-----------------using stream api (second part)-----------------------------
//        Stream<Integer> s=al.stream();
//        s.filter(e->e%2==0).forEach(e->System.out.println(e));


        //-----------------using stream api (third part)-----------------------------
        //al.stream().filter(e->e%2==0).forEach(e->System.out.println(e));


        //-------------count even numbers---------------------------
        System.out.println(al.stream().filter(e->e%2==0).count());




        //method chaining
//        String name="Arun";
//        String s1=name.concat("java");
//        String s2=s1.toUpperCase();
//        int l=s2.length();
//
//        int leng=name.concat("java").toUpperCase().length();
//        System.out.println(leng);
    }
}
```

==========

```java
//WAP to convert all the names to uppercase

public class Test4
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();
        al.add("deepak");
        al.add("rahul");
        al.add("amit");
        al.add("deepesh");
        al.add("ankit");

        Stream<String> s=al.stream();
        s.map(e->e.toUpperCase()).forEach(e->System.out.println(e));
    }
}
```

```java
// WAP to sort the list

public class Test5
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();
        al.add("deepak");
        al.add("rahul");
        al.add("amit");
        al.add("deepesh");
        al.add("ankit");

        Stream<String> s=al.stream();
        //s.forEach(e->System.out.println(e));
        //s.sorted().forEach(e->System.out.println(e));

        s.sorted((x,y)->-x.compareTo(y)).forEach(e->System.out.println(e));
    }
}
```

=> Task :-

1. WAP to print all the numbers which are greater than 30

2. WAP to print all the string values whose length is greater than 5
   (Ramesh, deepesh, Arun, rahul, ankit) -> Ramesh, deepesh

2. WAP to print all the names whose name starts with d

3. WAP to get the first character of each name

4. WAP to replace the a charcter with z character

5. WAP to sort an arraylist containg integer values

6. WAP to get Minimum and Maximum element in an ArrayList

7. WAP to print the sum of all the even numbers in an ArrayList

// WAP to print the sum of all the even numbers in an ArrayList

```
public class Test7
{
    public static void main(String[] args)
    {
        ArrayList<Integer> al=new ArrayList<>();
        al.add(1);
        al.add(4);
        al.add(2);
        al.add(5);
        al.add(3);

        Stream<Integer> s=al.stream();
        //List l=s.filter(e->e%2==0).collect(Collectors.toList());

//          Stream<Integer> s2=s.filter(e->e%2==0);          //get all even number streams
//          IntStream is=s2.mapToInt(Integer :: intValue);  //convert the Stream into IntStream
//          System.out.println(is.sum());                   //sum all the numbers present in IntStream

        int res=s.filter(e->e%2==0).mapToInt(Integer :: intValue).sum();
        System.out.println(res);


//          int sum=0;
//          Iterator<Integer> itr=l.iterator();
//          while(itr.hasNext())
//          {
//              sum=sum+itr.next();
//          }
//          System.out.println(sum);
    }
}
```

```
//WAP to print all the numbers which are greater than 30
//(functional interfaces, lambda expression, Predefined Functional Interfaces, Stream API)

public class Test1
{
    public static void main(String[] args)
    {
        ArrayList<Integer> al=new ArrayList<>();
        al.add(10);
        al.add(20);
        al.add(30);
        al.add(40);
        al.add(50);

        al.stream().filter(e->e>30).forEach(e->System.out.println(e));

        /*-----------------------------------------------------
        Stream<Integer> s=al.stream();

        Predicate<Integer> p=(e)->
                  {
                      return e>30;
                  };

        Consumer<Integer> c=(e)->
                        {
                            System.out.println(e);
                        };

        s.filter(p).forEach(c);
               ---------------------------------------*/

        //Stream<Integer> s=al.stream();
        //s.filter(e->e>30).forEach(e->System.out.println(e));


    }
}
```

// WAP to print all the string values whose length is greater than 5

```
public class Test2
{
    public static void main(String[] args)
    {
        ArrayList<String> al=new ArrayList<>();
        al.add("ArunSharma");
        al.add("amit");
        al.add("rahul");
        al.add("deepesh");
        al.add("kamal");

        Stream<String> s=al.stream();
        //s.filter(e->e.length()>5).forEach(e->System.out.println(e));
        List l=s.filter(e->e.length()>5).collect(Collectors.toList());
        System.out.println(l);
    }
```

// WAP to print all the names whose name starts with d

```
public class Test3
{
    public static void main(String[] args)
    {
        ArrayList<String> al=new ArrayList<>();
        al.add("deepak");
        al.add("amit");
        al.add("rahul");
        al.add("deepesh");
        al.add("kamal");

        Stream<String> s=al.stream();
        s.filter(e->e.startsWith("x")).forEach(e->System.out.println(e));
    }
```

// WAP to get the first character of each name
//WAP to replace the 'a' charcter with 'z' character

```
public class Test4
{
    public static void main(String[] args)
    {
        ArrayList<String> al=new ArrayList<>();
        al.add("deepak");
        al.add("amit");
        al.add("rahul");
        al.add("deepesh");
        al.add("kamal");

        Stream<String> s=al.stream();
        //s.map(e->e.charAt(0)).forEach(e->System.out.println(e));

//          String name="deepak";
//          System.out.println(name.replace("a", "z"));
        s.map(e->e.replace("a", "z")).forEach(e->System.out.println(e));
    }
```

//WAP to sort an arraylist containg integer values

```
public class Test5
{
    public static void main(String[] args)
    {
        ArrayList<Integer> al=new ArrayList<>();
        al.add(10);
        al.add(40);
        al.add(20);
        al.add(50);
        al.add(30);

        Stream<Integer> s=al.stream();
        //s.sorted().forEach(e->System.out.println(e));
        s.sorted((x,y)->-x.compareTo(y)).forEach(e->System.out.println(e));
    }
```

//WAP to get Minimum and Maximum element in an ArrayList

```
public class Test6
{
    public static void main(String[] args)
    {
        ArrayList<Integer> al=new ArrayList<>();
        al.add(10);
        al.add(40);
        al.add(20);
        al.add(50);
        al.add(30);

        Stream<Integer> s=al.stream();
//          int min=s.min((x,y)->x.compareTo(y)).get();
//          System.out.println("Minimum Element is : "+min);

        int max=s.max((x,y)->x.compareTo(y)).get();
        System.out.println("Maximum element is : "+max);
    }
```

Method Reference:

Method reference is used to refer the method of functional interface

```
@FunctionalInterface
interface I1
{
    void m1();
}
class A
{
    public void m2()
    {
        System.out.println("i am m1() method in class A");
    }
}

public class Test1
{
    public static void main(String[] args)
    {
        A ob=new A();
        I1 i=ob::m2;     //  :: -> Method reference
        i.m1();    //--> now internally it is calling m2()
    }
}
```

```java
interface I2  1 usage
{
    void m1();
}
class A2  1 usage
{
    static void m2()  no usages --> static method
    {
        System.out.println("hiiii");
    }
}
public class Test2
{
    public static void main(String[] args)
    {
        I2 i=A2::m2;    //since m2 is static
        i.m1();
    }
}
```

```java
interface I3  1 usage
{
    void m1(String a);  -> method with parameter
}
class A3  2 usages
{
    public void m2(String a)
    {
        System.out.println("hello : "+a);
    }
}

public class Test3
{
    public static void main(String[] args)
    {
        A3 ob=new A3();
        I3 i=ob::m2;
        i.m1("Arun");
    }
}
```

Constructor reference:

```
interface I4  1 usage
{
    void m1();
}
class A4  1 usage
{
    A4()  no usages
    {
        System.out.println("constructor...!!");
    }
}
public class Test4
{
    public static void main(String[] args)
    {
        //new A4();
        I4 i=A4::new;
        i.m1();   -> calling m1() method but constructor of A4 gets
    }
}
```