Stream In Java

Difficulty Level : Medium Last Updated : 09 Oct, 2019

Introduced in Java 8, the 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.
The features of Java stream are –

- 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.

Different Operations On Streams-
Intermediate Operations:

1. **map:** The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.
   ```
   List number = Arrays.asList(2,3,4,5);
   List square = number.stream().map(x->x*x).collect(Collectors.toList());
   ```
2. **filter:** The filter method is used to select elements as per the Predicate passed as argument.
   ```
   List names = Arrays.asList("Reflection","Collection","Stream");
   List result = names.stream().filter(s-
   >s.startsWith("S")).collect(Collectors.toList());
   ```
3. **sorted:** The sorted method is used to sort the stream.
   ```
   List names = Arrays.asList("Reflection","Collection","Stream");
   List result = names.stream().sorted().collect(Collectors.toList());
   ```

Terminal Operations:

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.
   ```
   List number = Arrays.asList(2,3,4,5,3);
   Set square = number.stream().map(x->x*x).collect(Collectors.toSet());
   ```

2. **forEach:** The forEach method is used to iterate through every element of the stream.

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

3. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.
   The reduce method takes a BinaryOperator as a parameter.

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

   Here ans variable is assigned 0 as the initial value and i is added to it .

Program to demonstrate the use of Stream

```
//a simple program to demonstrate the use of stream in java
import java.util.*;
import java.util.stream.*;

class Demo
{
  public static void main(String args[])
  {

    // create a list of integers
    List<Integer> number = Arrays.asList(2,3,4,5);

    // demonstration of map method
    List<Integer> square = number.stream().map(x -> x*x).
                           collect(Collectors.toList());
    System.out.println(square);

    // create a list of String
    List<String> names =
                Arrays.asList("Reflection","Collection","Stream");

    // demonstration of filter method
    List<String> result = names.stream().filter(s->s.startsWith("S")).
                          collect(Collectors.toList());
    System.out.println(result);

    // demonstration of sorted method
    List<String> show =
            names.stream().sorted().collect(Collectors.toList());
    System.out.println(show);

    // create a list of integers
    List<Integer> numbers = Arrays.asList(2,3,4,5,2);

    // collect method returns a set
    Set<Integer> squareSet =
```

```
            numbers.stream().map(x->x*x).collect(Collectors.toSet());
        System.out.println(squareSet);

        // demonstration of forEach method
        number.stream().map(x->x*x).forEach(y->System.out.println(y));

        // demonstration of reduce method
        int even =
           number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);

        System.out.println(even);
      }
    }
```

Output:

```
[4, 9, 16, 25]
[Stream]
[Collection, Reflection, Stream]
[16, 4, 9, 25]
4
9
16
25
6
```

Important Points/Observations:

1. A stream consists of source followed by zero or more intermediate methods combined together (pipelined) and a terminal method to process the objects obtained from the source as per the methods described.
2. Stream is used to compute elements as per the pipelined methods without altering the original value of the object.

This article is contributed by **Akash Ojha** .If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.