# Stream In Java

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(Coll
ectors.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.printl
n(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.

Use of Map function

```
import java.util.*;

import java.util.stream.*;


class Main

{

public static void main(String args[])

{

    ArrayListList<Integer> number = Arrays.asList(2,3,4,5);


    ArrayList<Integer> square = number.stream().map(x -> x*x).

                                    collect(Collectors.toList());
```

```
        System.out.println(square);


}
}
```

Use of filter Function

=————————————-

```
import java.util.*;
import java.util.stream.*;


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


        ArrayList<String> names =
                        Arrays.asList("Reflection","Collection","Stream");


ArrayList<String> result = names.stream().filter(s->s.startsWith("S")).
                                collect(Collectors.toList());
System.out.println(result);
```

```
}

}
```

Use of sorted method()

```
import java.util.*;

import java.util.stream.*;


class Main

{

public static void main(String args[])

{


        ArrayList<String> names =

                                Arrays.asList("Reflection","Collection","Stream");


        ArrayList<String> show =

                        names.stream().sorted().collect(Collectors.toList());

    System.out.println(show);



}

}
```

Exampleof forEach()

```
import java.util.*;

import java.util.stream.*;


class Main

{

public static void main(String args[])

{


        ArrayList<Integer> numbers = Arrays.asList(2,3,4,5,2);

        numbers.stream().map(x->x*x).forEach(y->System.out.println(y));

}

}
```

Example of reduce

```
import java.util.*;

import java.util.stream.*;


class Main

{

public static void main(String args[])
```

```
{

        ArrayList<Integer> numbers = Arrays.asList(2,3,4,5,2);



        int even =
            numbers.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);


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