



Zusammengerafft

Michael Wiedeking · JUG Saxony Day · 27. September 2024

filter



`.filter(□ -> □ == □)`



Stream<T>

```
public Stream<T> filter(Predicate<? super T> predicate);
```

```
var list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
var result = list.stream()
```

```
    .filter(e -> Integer.parseInt(e) % 2 == 0)
```

```
    .toList()
```

```
;
```

```
System.out.println(result)
```

```
// [2, 4, 6, 8, 10]
```

```
static boolean isEven(String s) {  
    try {  
        return Integer.parseInt(s) % 2 == 0;  
    } catch (Exception ex) {  
        return false;  
    }  
}
```

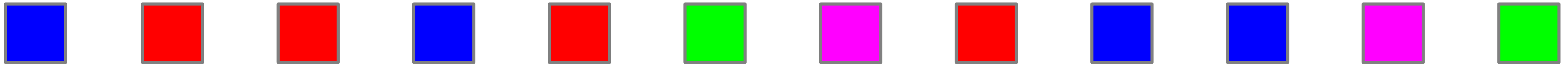
```
var list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F");
```

```
var result = list.stream().filter(Example::isEven).toList();
```

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

```
// [2, 4, 6, 8]
```

map



.map(□ -> ■ => ◊)



Stream<T>

public <R> Stream<R> map(
 Function<? **super** T, ? **extends** R> mapper

);

```
var list = List.<String>of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
var result = list.stream()
```

```
    .map(e -> "_" + e "_")
```

```
    .toList()
```

```
;
```

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

```
// [_1_, _2_, _3_, _4_, _5_, _6_, _7_, _8_, _9_, _10_]
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
int[] result = list.stream()
```

```
    .mapToInt(Integer::parseInt)
```

```
    .toArray()
```

```
;
```

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

```
// [I@7291c18f
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
int[] result = list.stream()
```

```
    .mapToInt(Integer::parseInt)
```

```
    .toArray()
```

```
;
```

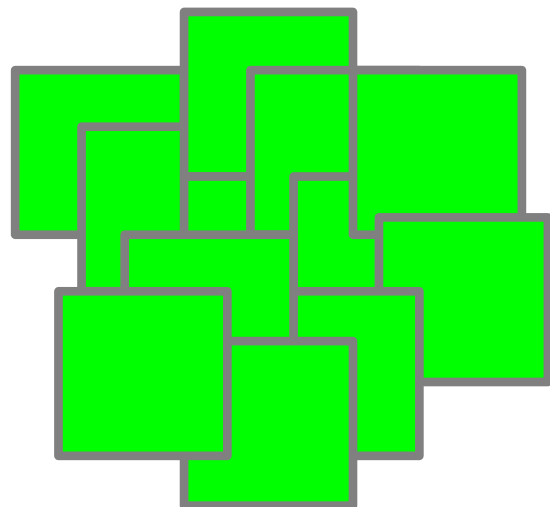
```
System.out.println(Arrays.toString(result));
```

```
// [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

reduce



.reduce((,) ->)



Stream<T>

public Optional<T> reduce(
 BinaryOperator<T> *accumulator*

);

```
var list = List.<String>of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
var result = list.stream()
```

```
    .reduce((e1, e2) -> e1 + e2)
```

```
    .get()
```

```
;
```

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

```
// 12345678910
```


Stream<T>

public <U> U reduce(
 U *identity*,

 BiFunction<U, ? **super** T, U> *accumulator*

);

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .reduce("", (s, e) -> s + e, (s1, s2) -> s1 + s2)
```

```
;
```

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

```
// 12345678910
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .reduce("[", (s, e) -> s + e + "]", (s1, s2) -> s1 + s2)
```

```
;
```

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

```
// [1]2]3]4]5]6]7]8]9]10]
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .parallel()
```

```
    .reduce("[", (s, e) -> s + e + "]", (s1, s2) -> s1 + s2 + "])")
```

```
;
```

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

```
// [1][2][3][4][5]][6][7][8][9][10]]]
```

Stream<T>

public <U> U reduce(
 U *identity*, BiFunction<U, ? **super** T, U> *accumulator*, BinaryOperator<U> *combiner*

);

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .reduce("", (s, e) -> s + e, (s1, s2) -> s1 + s2)
```

```
;
```

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

```
// 12345678910
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .reduce("[", (s, e) -> s + e + "]", (s1, s2) -> s1 + "::" + s2)
```

```
;
```

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

```
// [1]2]3]4]5]6]7]8]9]10]
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .parallel()
```

```
    .reduce("[", (s, e) -> s + e + "]", (s1, s2) -> s1 + "::" + s2)
```

```
;
```

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

```
// [1]::[2]::[3]::[4]::[5]::[6]::[7]::[8]::[9]::[10]
```


collect



.collect((○, □) -> ○□)



Stream<T>

public <R> R collect(
Supplier<R> *supplier*,

BiConsumer<R, ? **super** T> *accumulator*,

BiConsumer<R, R> *combiner*

);

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");  
  
String result = list.stream()  
    .collect(  
        ArrayList<String>::new,  
        ArrayList<String>::add,  
        ArrayList<String>::addAll  
    )  
;  
  
System.out.println(result);  
  
// [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
List<String> list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
String result = list.stream()
```

```
    .collect(
```

```
        ArrayList<String>::new,
```

```
        (l, e) -> l.add("\\" + e + "\\")
```

```
        ArrayList<String>::addAll
```

```
    )
```

```
;
```

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

```
// ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10"]
```

```
interface Collector<T, A, R> {  
  
    Supplier<A>          supplier();  
  
    BiConsumer<A, T>    accumulator();  
  
    BinaryOperator<A>   combiner();  
  
    Function<A, R>      finisher();  
  
    Set<Characteristics> characteristics();  
  
}
```

public static <T, A, R> Collector<T, A, R> of(

Supplier<R> *supplier,*

BiConsumer<R, T> *accumulator,*

BinaryOperator<R> *combiner,*

Function<A, R> *finisher,*

Characteristics ... *characteristics*

)

public static <T, R> Collector<T, R, R> of(

Supplier<R> *supplier,*

BiConsumer<R, T> *accumulator,*

BinaryOperator<R> *combiner,*

Characteristics ... *characteristics*

)

gather

Stream<T>

```
public <R> Stream<R> gather(  
    Gatherer<? super T, ?, R> gatherer  
);
```

```
public interface Gatherer<T, A, R> {  
  
    Supplier<A>                    initializer();  
  
    Integrator<A, T, R>            integrator();  
  
    BinaryOperator<A>             combiner();  
  
    BiConsumer<A, Downstream<? super R>> finisher();  
  
    <RR> Gatherer<T, ?, RR>        andThen(Gatherer<? super R, ?, ? extends RR> that);  
  
}
```

```
public interface Gatherer<T, A, R> {  
    Supplier<A>                               initializer();  
    Integrator<A, T, R>                       integrator();  
    BinaryOperator<A>                        combiner();  
    BiConsumer<A, Downstream<? super R>>    finisher();  
}  
  
public interface Integrator< A, T, R> {  
    boolean integrate(A state, T element, Downstream<? super R> downstream);  
}
```

```
public interface Gatherer<T, A, R> {  
    Supplier<A>  
    Integrator<A, T, R>  
    BinaryOperator<A>  
    BiConsumer<A, Downstream<? super R>>  
}
```

```
    initializer();  
    integrator();  
    combiner();  
    finisher();
```

```
public interface Integrator< A, T, R> {  
    boolean  
    integrate(A state, T element, Downstream<? super R> downstream);  
}
```

```
public interface Downstream<T> {  
    boolean push(T element);  
    boolean isRejecting();  
}
```

map – *again*

```
public interface Gatherer<T, A, R> {  
    Supplier<A>  
    Integrator<A, T, R>  
    BinaryOperator<A>  
    BiConsumer<A, Downstream<? super R>>  
}
```

```
    initializer();  
    integrator();  
    combiner();  
    finisher();
```

```
public interface Integrator< A, T, R> {  
    boolean  
    integrate(A state, T element, Downstream<? super R> downstream);  
}
```

```
public interface Downstream<T> {  
    boolean push(T element);  
    boolean isRejecting();  
}
```

```
public interface Gatherer<T, A, R> {
```

```
Supplier<A> initializer();
```

```
Integrator<A, T, R> integrator();
```

```
BinaryOperator<A> combiner();
```

```
BiConsumer<A, Downstream<? super R>> finisher();
```

```
}
```

```
public interface Integrator< A, T, R> {
```

```
boolean integrate(A state, T element, Downstream<? super R> downstream);
```

```
}
```



```
static <T, R> Gatherer<T, Void, R> of(Integrator<Void, T, R> integrator)
```

```
var list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
var result = list.stream()
```

```
    .gather(Gatherer.of((_, element, downstream) -> { // integrate(Void, String, Downstream<Integer>)
```

```
        downstream.push(Integer.parseInt(element));
```

```
        return true;
```

```
    })))
```

```
    .toList()
```

```
;
```

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

```
// [2, 4, 6, 8]
```

filter – *again*

```
static <T, R> Gatherer<T, Void, R> of(Integrator<Void, T, R> integrator)
```

```
var list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F");
```

```
var result = list.stream()
```

```
    .gather(Gatherer.of((_, element, downstream) -> { // integrate(Void, String, Downstream<String>)
```

```
        if (isEven(element)) {
```

```
            downstream.push(element);
```

```
        }
```

```
        return true;
```

```
    )))
```

```
    .toList()
```

```
;
```

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

```
// [2, 4, 6, 8]
```

n:m

```
var list = List.of("1", "2", "3", "4", "5");  
var result = list.stream()  
    .gather(Gatherer.of((_, element, downstream) -> { // integrate(Void, String, Downstream<Integer>)  
        var n = Integer.parseInt(element);  
        downstream.push(n - 1);  
        downstream.push(n);  
        downstream.push(n + 1);  
        return true;  
    }  
    ))  
    .toList()  
;  
  
System.out.println(result);  
  
// [0, 1, 2, 1, 2, 3, 2, 3, 4, 3, 4, 5, 4, 5, 6]
```

```
var list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");  
var result = list.stream()  
    .gather(Gatherers.windowFixed(3))  
    .toList()  
;  
  
System.out.println(result);  
  
// [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10]]
```

```
var list = List.of("1", "2", "3", "4", "5", "6", "7", "8", "9", "10");
```

```
var result = list.stream()
```

```
    .gather(Gatherers.windowSliding(3))
```

```
    .toList()
```

```
;
```

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

```
// [[1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6], [5, 6, 7], [6, 7, 8], [7, 8, 9], [8, 9, 10]]
```

```
public static <TR> Gatherer<TR, ?, List<TR>> windowFixed(int windowSize) {  
    ...  
}
```



```
class FixedWindow {  
    Object[] window;  
    int at;  
    FixedWindow() {  
        at = 0;  
        window = new Object[windowSize];  
    }  
    ...  
}
```

```
boolean integrate(T element, Downstream<? super List<TR>> downstream) {  
    window[at++] = element;  
  
    if (at < windowSize) {  
        return true;  
    } else {  
        final var oldWindow = window;  
        window = new Object[windowSize];  
        at = 0;  
        return downstream.push(new ArrayList(oldWindow));  
    }  
}
```

```
void finish(Downstream<? super List<TR>> downstream) {  
    if (at > 0 && !downstream.isRejecting()) {  
        var lastWindow = new Object[at];  
        System.arraycopy(window, 0, lastWindow, 0, at);  
        window = null;  
        at = 0;  
        downstream.push(new ArrayList(lastWindow));  
    }  
}
```

```
public static <TR> Gatherer<TR, ?, List<TR>> windowFixed(int windowSize) {  
    return Gatherer.<TR, FixedWindow, List<TR>>ofSequential(  
        FixedWindow::new,  
        Integrator.<FixedWindow, TR, List<TR>>ofGreedy(FixedWindow::integrate),  
        FixedWindow::finish  
    );  
}
```



Fragen!?

Michael Wiedeking · JUG Saxony Day · 27. September 2024



Vielen Dank!

Michael Wiedeking · JUG Saxony Day · 27. September 2024