compact1, compact2, compact3
java.util.stream

Interface IntStream

All Superinterfaces:
AutoCloseable, BaseStream<Integer,IntStream>

```
public interface IntStream
extends BaseStream<Integer,IntStream>
```

A sequence of primitive int-valued elements supporting sequential and parallel aggregate operations. This is the `int` primitive specialization of `Stream`.

The following example illustrates an aggregate operation using `Stream` and `IntStream`, computing the sum of the weights of the red widgets:

```
int sum = widgets.stream()
                 .filter(w -> w.getColor() == RED)
                 .mapToInt(w -> w.getWeight())
                 .sum();
```

See the class documentation for `Stream` and the package documentation for java.util.stream for additional specification of streams, stream operations, stream pipelines, and parallelism.

Since:
1.8

See Also:
Stream, java.util.stream

Nested Class Summary

Nested Classes

Modifier and Type	Interface and Description
static interface	**IntStream.Builder** A mutable builder for an IntStream.

Method Summary

All Methods Static Methods Instance Methods Abstract Methods Default Methods

Modifier and Type	Method and Description
boolean	**allMatch**(**IntPredicate** predicate) Returns whether all elements of this stream match the provided predicate.
boolean	**anyMatch**(**IntPredicate** predicate) Returns whether any elements of this stream match the provided predicate.
DoubleStream	**asDoubleStream**() Returns a DoubleStream consisting of the elements of this stream, converted to double.
LongStream	**asLongStream**() Returns a LongStream consisting of the elements of this stream, converted to long.
OptionalDouble	**average**() Returns an OptionalDouble describing the arithmetic mean of elements of this stream, or an empty optional if this stream is empty.
Stream<Integer>	**boxed**() Returns a Stream consisting of the elements of this stream, each boxed to an Integer.
static **IntStream.Builder**	**builder**() Returns a builder for an IntStream.
<R> R	**collect**(**Supplier**<R> supplier, **ObjIntConsumer**<R> accumulator, **BiConsumer**<R,R> combiner) Performs a **mutable reduction** operation on the elements of this stream.
static **IntStream**	**concat**(**IntStream** a, **IntStream** b) Creates a lazily concatenated stream whose elements are all the elements of the first stream followed by all the elements of the second stream.
long	**count**() Returns the count of elements in this stream.
IntStream	**distinct**() Returns a stream consisting of the distinct elements of this stream.
static **IntStream**	**empty**() Returns an empty sequential IntStream.
IntStream	**filter**(**IntPredicate** predicate) Returns a stream consisting of the elements of this stream that match the given predicate.
OptionalInt	**findAny**() Returns an **OptionalInt** describing some element of the stream, or an empty OptionalInt if the stream is empty.
OptionalInt	**findFirst**() Returns an **OptionalInt** describing the first element of this stream, or an empty OptionalInt if the stream is empty.
IntStream	**flatMap**(**IntFunction**<? extends **IntStream**> mapper) Returns a stream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element.
void	**forEach**(**IntConsumer** action) Performs an action for each element of this stream.
void	**forEachOrdered**(**IntConsumer** action) Performs an action for each element of this stream, guaranteeing that each element is processed in encounter order for streams that have a defined encounter order.
static **IntStream**	**generate**(**IntSupplier** s) Returns an infinite sequential unordered stream where each element is generated by the provided IntSupplier.
static **IntStream**	**iterate**(int seed, **IntUnaryOperator** f) Returns an infinite sequential ordered IntStream produced by iterative application of a function f to an initial element seed, producing a Stream consisting of seed, f(seed), f(f(seed)), etc.
PrimitiveIterator.OfInt	**iterator**() Returns an iterator for the elements of this stream.
IntStream	**limit**(long maxSize) Returns a stream consisting of the elements of this stream, truncated to be no longer than maxSize in length.

IntStream	map(IntUnaryOperator mapper) Returns a stream consisting of the results of applying the given function to the elements of this stream.
DoubleStream	mapToDouble(IntToDoubleFunction mapper) Returns a DoubleStream consisting of the results of applying the given function to the elements of this stream.
LongStream	mapToLong(IntToLongFunction mapper) Returns a LongStream consisting of the results of applying the given function to the elements of this stream.
<U> Stream<U>	mapToObj(IntFunction<? extends U> mapper) Returns an object-valued Stream consisting of the results of applying the given function to the elements of this stream.
OptionalInt	max() Returns an OptionalInt describing the maximum element of this stream, or an empty optional if this stream is empty.
OptionalInt	min() Returns an OptionalInt describing the minimum element of this stream, or an empty optional if this stream is empty.
boolean	noneMatch(IntPredicate predicate) Returns whether no elements of this stream match the provided predicate.
static IntStream	of(int... values) Returns a sequential ordered stream whose elements are the specified values.
static IntStream	of(int t) Returns a sequential IntStream containing a single element.
IntStream	parallel() Returns an equivalent stream that is parallel.
IntStream	peek(IntConsumer action) Returns a stream consisting of the elements of this stream, additionally performing the provided action on each element as elements are consumed from the resulting stream.
static IntStream	range(int startInclusive, int endExclusive) Returns a sequential ordered IntStream from startInclusive (inclusive) to endExclusive (exclusive) by an incremental step of 1.
static IntStream	rangeClosed(int startInclusive, int endInclusive) Returns a sequential ordered IntStream from startInclusive (inclusive) to endInclusive (inclusive) by an incremental step of 1.
OptionalInt	reduce(IntBinaryOperator op) Performs a **reduction** on the elements of this stream, using an **associative** accumulation function, and returns an OptionalInt describing the reduced value, if any.
int	reduce(int identity, IntBinaryOperator op) Performs a **reduction** on the elements of this stream, using the provided identity value and an **associative** accumulation function, and returns the reduced value.
IntStream	sequential() Returns an equivalent stream that is sequential.
IntStream	skip(long n) Returns a stream consisting of the remaining elements of this stream after discarding the first n elements of the stream.
IntStream	sorted() Returns a stream consisting of the elements of this stream in sorted order.
Spliterator.OfInt	spliterator() Returns a spliterator for the elements of this stream.
int	sum() Returns the sum of elements in this stream.
IntSummaryStatistics	summaryStatistics() Returns an IntSummaryStatistics describing various summary data about the elements of this stream.
int[]	toArray() Returns an array containing the elements of this stream.

Methods inherited from interface java.util.stream.BaseStream

close, isParallel, onClose, unordered

Method Detail

filter

IntStream filter(IntPredicate predicate)

Returns a stream consisting of the elements of this stream that match the given predicate.

This is an intermediate operation.

Parameters:
predicate - a non-interfering, stateless predicate to apply to each element to determine if it should be included

Returns:
the new stream

map

IntStream map(IntUnaryOperator mapper)

Returns a stream consisting of the results of applying the given function to the elements of this stream.

This is an intermediate operation.

Parameters:
mapper - a non-interfering, stateless function to apply to each element

Returns:
the new stream

mapToObj

<U> Stream<U> mapToObj(IntFunction<? extends U> mapper)

Returns an object-valued Stream consisting of the results of applying the given function to the elements of this stream.

This is an intermediate operation.

Type Parameters:
U - the element type of the new stream

Parameters:
mapper - a non-interfering, stateless function to apply to each element

Returns:
the new stream