OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP

DETAIL: FIELD | CONSTR | METHOD

Java SE 22 & JDK 22

SEARCH Q Search

Returns a Gatherer that performs an ordered, reduction-like,

implemented, or for reductions which are intrinsically order-

An operation which executes a function concurrently with a

configured level of max concurrency, using virtual threads.

Returns a Gatherer that gathers elements into windows --

Returns a Gatherer that gathers elements into windows --

encounter-ordered groups of elements -- of a given size, where each subsequent window includes all elements of the previous

window except for the least recent, and adds the next element in

encounter-ordered groups of elements -- of a fixed size.

accumulation -- using the provided functions.

transformation for scenarios where no combiner-function can be

Returns a Gatherer that performs a Prefix Scan -- an incremental

Module java.base Package java.util.stream

SUMMARY: NESTED | FIELD | CONSTR | METHOD

Class Gatherers java.lang.Object

java.util.stream.Gatherers

public final class **Gatherers** extends Object

Gatherers is a preview API of the Java platform. Programs can only use Gatherers when preview features are enabled. Preview features may be removed in a future release, or upgraded to permanent features of the Java platform.

Since: 22

Implementations of Gatherer PREVIEW that provide useful intermediate operations, such as windowing functions, folding functions, transforming elements concurrently, etc.

Returns a Gatherer that gathers elements into windows -- encounter-ordered groups of elements -- of a fixed size. If the stream is empty then no window will be produced. The last

BiFunction<? super R,? super T,? extends R> folder)

BiFunction<? super R,? super T,? extends R> scanner)

subsequent value is obtained by applying the BiFunction to the current value and the next input element, after which the resulting value is produced downstream.

Returns a Gatherer that performs a Prefix Scan -- an incremental accumulation -- using the provided functions. Starting with an initial value obtained from the Supplier, each

Function<? super T,? extends R> mapper)

Description

dependent.

the stream.

Method Summary All Methods Static Methods Concrete Methods Method

Modifier and Type

fold(Supplier<R> initial, BiFunction<? super R,? super</pre> T,? extends R> folder)

mapConcurrent(int maxConcurrency, Function<? super T,?</pre> static <T, R> **Gatherer** PREVIEW <T,?,R> extends R> mapper)

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

scan(Supplier<R> initial, BiFunction<? super R,? super</pre> static <T, R> Gatherer^{PREVIEW}<T,?,R> T,? extends R> scanner)

windowFixed(int windowSize) static <TR> Gatherer^{PREVIEW}<TR,?,List<TR>> static <TR> Gatherer^{PREVIEW}<TR,?,List<TR>>> windowSliding(int windowSize)

Methods declared in class java.lang.Object clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Details windowFixed

Implementation Requirements: Each window produced is an unmodifiable List; calls to any mutator method will always cause UnsupportedOperationException to be thrown. There are no guarantees on the implementation type or serializability of the produced Lists.

duration of evaluation of this operation.

Type Parameters: TR - the type of elements the returned gatherer consumes and the contents of the windows it produces

IllegalArgumentException - when windowSize is less than 1

Stream.of(1,2,3,4,5,6,7,8).gather(Gatherers.windowSliding(2)).toList();

Gatherers.fold(() -> "", (string, number) -> string + number)

Gatherers.scan(() -> "", (string, number) -> string + number)

// will contain: [[1, 2, 3, 4, 5, 6], [2, 3, 4, 5, 6, 7], [3, 4, 5, 6, 7, 8]]

windowSliding

public static <TR> Gatherer^{PREVIEW}<TR,?,List<TR>> windowSliding(int windowSize)

Stream.of(1,2,3,4,5,6,7,8).gather(Gatherers.windowSliding(6)).toList(); **API Note:** For efficiency reasons, windows may be allocated contiguously and eagerly. This means that choosing large window sizes for small streams may use excessive memory for the

List<List<Integer>> windows6 =

duration of evaluation of this operation.

Stream.of(1,2,3,4,5,6,7,8,9)

.findFirst();

initial - the identity value for the fold operation

Stream.of(1,2,3,4,5,6,7,8,9)

T - the type of element which this gatherer consumes

R - the type of element which this gatherer produces

initial - the supplier of the initial value for the scanner

.toList();

.gather(

folder - the folding function

Implementation Requirements:

Each window produced is an unmodifiable List; calls to any mutator method will always cause UnsupportedOperationException to be thrown. There are no guarantees on the implementation type or serializability of the produced Lists. **Type Parameters:**

windowSize - the size of the windows **Returns:**

IllegalArgumentException - when windowSize is less than 1 fold

Example: // will contain: Optional["123456789"] Optional<String> numberString =

Parameters:

Returns:

Throws:

a new Gatherer

// will contain: ["1", "12", "123", "1234", "12345", "123456", "1234567", "12345678", "123456789"] List<String> numberStrings =

Returns: a new Gatherer which performs a prefix scan Throws:

Parameters:

Type Parameters:

mapConcurrent public static <T, R> Gatherer^{PREVIEW}<T,?,R> mapConcurrent(int maxConcurrency,

API Note: In progress tasks will be attempted to be cancelled, on a best-effort basis, in situations where the downstream no longer wants to receive any more elements. **Implementation Requirements:**

If a result of the function is to be pushed downstream but instead the function completed exceptionally then the corresponding exception will instead be rethrown by this method as an instance of RuntimeException, after which any remaining tasks are canceled. **Type Parameters:**

Parameters: maxConcurrency - the maximum concurrency desired mapper - a function to be executed concurrently **Returns:** a new Gatherer

R - the type of output

Throws: IllegalArgumentException - if maxConcurrency is less than 1 NullPointerException - if mapper is null

static <T, R> Gatherer PREVIEW <T,?,R>

window may contain fewer elements than the supplied window size. Example: // will contain: [[1, 2, 3], [4, 5, 6], [7, 8]] List<List<Integer>> windows = Stream.of(1,2,3,4,5,6,7,8).gather(Gatherers.windowFixed(3)).toList(); **API Note:** For efficiency reasons, windows may be allocated contiguously and eagerly. This means that choosing large window sizes for small streams may use excessive memory for the

Parameters: windowSize - the size of the windows **Returns:** a new gatherer which groups elements into fixed-size windows **Throws:**

Returns a Gatherer that gathers elements into windows -- encounter-ordered groups of elements -- of a given size, where each subsequent window includes all elements of the previous window except for the least recent, and adds the next element in the stream. If the stream is empty then no window will be produced. If the size of the stream is smaller than the window size then only one window will be produced, containing all elements in the stream. Example: // will contain: [[1, 2], [2, 3], [3, 4], [4, 5], [5, 6], [6, 7], [7, 8]] List<List<Integer>> windows2 =

TR - the type of elements the returned gatherer consumes and the contents of the windows it produces **Parameters:** a new gatherer which groups elements into sliding windows **Throws:**

public static <T, R> Gatherer^{PREVIEW}<T,?,R> fold(Supplier<R> initial, Returns a Gatherer that performs an ordered, reduction-like, transformation for scenarios where no combiner-function can be implemented, or for reductions which are intrinsically order-dependent. **Implementation Requirements:** If no exceptions are thrown during processing, then this operation only ever produces a single element.

Type Parameters: T - the type of elements the returned gatherer consumes R - the type of elements the returned gatherer produces

NullPointerException - if any of the parameters are null See Also: Stream.reduce(Object, BinaryOperator) scan public static <T, R> Gatherer*PREVIEW<T,?,R> scan(Supplier<R> initial,

Example:

scanner - the function to apply for each element NullPointerException - if any of the parameters are null

An operation which executes a function concurrently with a configured level of max concurrency, using virtual threads. This operation preserves the ordering of the stream.

T - the type of input

Report a bug or suggest an enhancement For further API reference and developer documentation see the Java SE Documentation, which contains more detailed, developer-targeted descriptions with conceptual overviews, definitions of terms, workarounds, and working code examples. Other versions. Java is a trademark or registered trademark of Oracle and/or its affiliates in the US and other countries. Copyright © 1993, 2024, Oracle and/or its affiliates, 500 Oracle Parkway, Redwood Shores, CA 94065 USA. All rights reserved. Use is subject to license terms and the documentation redistribution policy. Modify Cookie Preferences. Modify Ad Choices.