Module java.base
Package java.lang.foreign

Interface SegmentAllocator

All Known Subinterfaces:

MemorySession[PREVIEW]

Functional Interface:

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

```
@FunctionalInterface
public interface SegmentAllocator
```

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

An object that may be used to allocate memory segments[PREVIEW]. Clients implementing this interface must implement the `allocate(long, long)` method. This interface defines several default methods which can be useful to create segments from several kinds of Java values such as primitives and arrays. This interface is a functional interface: clients can easily obtain a new segment allocator by using either a lambda expression or a method reference.

This interface also defines factories for commonly used allocators:

- `newNativeArena(MemorySession)` creates a more efficient arena-style allocator, where off-heap memory is allocated in bigger blocks, which are then sliced accordingly to fit allocation requests;
- `implicitAllocator()` obtains an allocator which allocates native memory segment in independent, implicit memory sessions[PREVIEW]; and
- `prefixAllocator(MemorySegment)` obtains an allocator which wraps a segment (either on-heap or off-heap) and recycles its content upon each new allocation request.

Passing a segment allocator to an API can be especially useful in circumstances where a client wants to communicate *where* the results of a certain operation (performed by the API) should be stored, as a memory segment. For instance, downcall method handles[PREVIEW] can accept an additional `SegmentAllocator`[PREVIEW] parameter if the underlying foreign function is known to return a struct by-value. Effectively, the allocator parameter tells the linker runtime where to store the return value of the foreign function.

Method Summary

All Methods | Static Methods | Instance Methods | Abstract Methods | Default Methods

Modifier and Type	Method	Description
default `MemorySegment`[PREVIEW]	**`allocate`**`(long bytesSize)`	Allocates a memory segment with the given size.
`MemorySegment`[PREVIEW]	**`allocate`**`(long bytesSize, long bytesAlignment)`	Allocates a memory segment with the given size and alignment constraints.
default `MemorySegment`[PREVIEW]	**`allocate`**`(MemoryLayout`[PREVIEW] ` layout)`	Allocates a memory segment with the given layout.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfAddress`[PREVIEW] ` layout, Addressable`[PREVIEW] ` value)`	Allocates a memory segment with the given layout and initializes it with the given address value.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfByte`[PREVIEW] ` layout, byte value)`	Allocates a memory segment with the given layout and initializes it with the given byte value.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfChar`[PREVIEW] ` layout, char value)`	Allocates a memory segment with the given layout and initializes it with the given char value.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfDouble`[PREVIEW] ` layout, double value)`	Allocates a memory segment with the given layout and initializes it with the given double value.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfFloat`[PREVIEW] ` layout, float value)`	Allocates a memory segment with the given layout and initializes it with the given float value.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfInt`[PREVIEW] ` layout, int value)`	Allocates a memory segment with the given layout and initializes it with the given int value.
default `MemorySegment`[PREVIEW]	**`allocate`**`(ValueLayout.OfLong`[PREVIEW] ` layout,`	Allocates a memory segment with the given layout and initializes it with the given long

	long value)	value.
default MemorySegment[PREVIEW]	**allocate**(ValueLayout.OfShort[PREVIEW] layout, short value)	Allocates a memory segment with the given layout and initializes it with the given short value.
default MemorySegment[PREVIEW]	**allocateArray**(MemoryLayout[PREVIEW] elementLayout, long count)	Allocates a memory segment with the given element layout and size.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfByte[PREVIEW] elementLayout, byte... elements)	Allocates a memory segment with the given layout and initializes it with the given byte elements.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfChar[PREVIEW] elementLayout, char... elements)	Allocates a memory segment with the given layout and initializes it with the given char elements.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfDouble[PREVIEW] elementLayout double... elements)	Allocates a memory segment with the given layout and initializes it with the given double elements.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfFloat[PREVIEW] elementLayout, float... elements)	Allocates a memory segment with the given layout and initializes it with the given float elements.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfInt[PREVIEW] elementLayout, int... elements)	Allocates a memory segment with the given layout and initializes it with the given int elements.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfLong[PREVIEW] elementLayout, long... elements)	Allocates a memory segment with the given layout and initializes it with the given long elements.
default MemorySegment[PREVIEW]	**allocateArray**(ValueLayout.OfShort[PREVIEW] elementLayout, short... elements)	Allocates a memory segment with the given layout and initializes it with the given short elements.
default MemorySegment[PREVIEW]	**allocateUtf8String**(String str)	Converts a Java string into a UTF-8 encoded, null-terminated C string, storing the result into a memory segment.
static SegmentAllocator[PREVIEW]	**implicitAllocator**()	Returns an allocator which allocates native segments in independent implicit memory sessions[PREVIEW].
static SegmentAllocator[PREVIEW]	**newNativeArena**(long arenaSize, long blockSize, MemorySession[PREVIEW] session)	Creates an arena-based allocator used to allocate native memory segments.
static SegmentAllocator[PREVIEW]	**newNativeArena**(long arenaSize, MemorySession[PREVIEW] session)	Creates an arena-based allocator used to allocate native memory segments.
static SegmentAllocator[PREVIEW]	**newNativeArena**(MemorySession[PREVIEW] session)	Creates an unbounded arena-based allocator used to allocate native memory segments.
static SegmentAllocator[PREVIEW]	**prefixAllocator**(MemorySegment[PREVIEW] segment)	Returns a segment allocator which responds to allocation requests by recycling a single segment.

Method Details

allocateUtf8String

default MemorySegment[PREVIEW] allocateUtf8String(String str)

Converts a Java string into a UTF-8 encoded, null-terminated C string, storing the result into a memory segment.

This method always replaces malformed-input and unmappable-character sequences with this charset's default replacement byte array. The CharsetEncoder class should be used when more control over the encoding process is required.

If the given string contains any '\0' characters, they will be copied as well. This means that, depending on the method used to read the string, such as MemorySegment.getUtf8String(long)[PREVIEW], the string will appear truncated when read again.

Implementation Requirements:

the default implementation for this method copies the contents of the provided Java string into a new memory segment obtained by calling `this.allocate(str.length() + 1)`.

Parameters:

`str` - the Java string to be converted into a C string.

Returns:

a new native memory segment containing the converted C string.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfByte[PREVIEW] layout,
                                        byte value)
```

Allocates a memory segment with the given layout and initializes it with the given byte value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfChar[PREVIEW] layout,
                                        char value)
```

Allocates a memory segment with the given layout and initializes it with the given char value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfShort[PREVIEW] layout,
                                        short value)
```

Allocates a memory segment with the given layout and initializes it with the given short value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfInt[PREVIEW] layout,
                                        int value)
```

Allocates a memory segment with the given layout and initializes it with the given int value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfFloat[PREVIEW] layout,
                                        float value)
```

Allocates a memory segment with the given layout and initializes it with the given float value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfLong[PREVIEW] layout,
                                        long value)
```

Allocates a memory segment with the given layout and initializes it with the given long value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfDouble[PREVIEW] layout,
                                        double value)
```

Allocates a memory segment with the given layout and initializes it with the given double value.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(ValueLayout.OfAddress[PREVIEW] layout,
                                        Addressable[PREVIEW] value)
```

Allocates a memory segment with the given layout and initializes it with the given address value. The address value might be narrowed according to the platform address size (see `ValueLayout.ADDRESS`[PREVIEW]).

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout)`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

`value` - the value to be set on the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfByte[PREVIEW] elementLayout,
                                             byte... elements)
```

Allocates a memory segment with the given layout and initializes it with the given byte elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the byte elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfShort[PREVIEW] elementLayout,
                                             short... elements)
```

Allocates a memory segment with the given layout and initializes it with the given short elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the short elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfChar[PREVIEW] elementLayout,
                                             char... elements)
```

Allocates a memory segment with the given layout and initializes it with the given char elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the char elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfInt[PREVIEW] elementLayout,
                                             int... elements)
```

Allocates a memory segment with the given layout and initializes it with the given int elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the int elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfFloat[PREVIEW] elementLayout,
                                             float... elements)
```

Allocates a memory segment with the given layout and initializes it with the given float elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the float elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfLong[PREVIEW] elementLayout,
                                             long... elements)
```

Allocates a memory segment with the given layout and initializes it with the given long elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the long elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(ValueLayout.OfDouble[PREVIEW] elementLayout,
                                             double... elements)
```

Allocates a memory segment with the given layout and initializes it with the given double elements.

Implementation Requirements:

the default implementation for this method calls `this.allocateArray(layout, array.length)`.

Parameters:

`elementLayout` - the element layout of the array to be allocated.

`elements` - the double elements to be copied to the newly allocated memory block.

Returns:

a segment for the newly allocated memory block.

allocate

```
default MemorySegment[PREVIEW] allocate(MemoryLayout[PREVIEW] layout)
```

Allocates a memory segment with the given layout.

Implementation Requirements:

the default implementation for this method calls `this.allocate(layout.byteSize(), layout.byteAlignment())`.

Parameters:

`layout` - the layout of the block of memory to be allocated.

Returns:

a segment for the newly allocated memory block.

allocateArray

```
default MemorySegment[PREVIEW] allocateArray(MemoryLayout[PREVIEW] elementLayout,
                                             long count)
```

Allocates a memory segment with the given element layout and size.

Implementation Requirements:

the default implementation for this method calls `this.allocate(MemoryLayout.sequenceLayout(count, elementLayout))`.

Parameters:

`elementLayout` - the array element layout.

`count` - the array element count.

Returns:

a segment for the newly allocated memory block.

Throws:

`IllegalArgumentException` - if `count < 0`.

allocate

```
default MemorySegment[PREVIEW] allocate(long bytesSize)
```

Allocates a memory segment with the given size.

Implementation Requirements:

the default implementation for this method calls `this.allocate(bytesSize, 1)`.

Parameters:

`bytesSize` - the size (in bytes) of the block of memory to be allocated.

Returns:

a segment for the newly allocated memory block.

Throws:

`IllegalArgumentException` - if `bytesSize < 0`

allocate

```
MemorySegment[PREVIEW] allocate(long bytesSize,
                                long bytesAlignment)
```

Allocates a memory segment with the given size and alignment constraints.

Parameters:

`bytesSize` - the size (in bytes) of the block of memory to be allocated.

`bytesAlignment` - the alignment (in bytes) of the block of memory to be allocated.

Returns:

a segment for the newly allocated memory block.

Throws:

`IllegalArgumentException` - if `bytesSize < 0`, `alignmentBytes <= 0`, or if `alignmentBytes` is not a power of 2.

newNativeArena

```
static SegmentAllocator[PREVIEW] newNativeArena(MemorySession[PREVIEW] session)
```

Creates an unbounded arena-based allocator used to allocate native memory segments. The returned allocator features a predefined block size and maximum arena size, and the segments it allocates are associated with the provided memory session. Equivalent to the following code:

```
SegmentAllocator.newNativeArena(Long.MAX_VALUE, predefinedBlockSize, session);
```

Parameters:

`session` - the memory session associated with the segments allocated by the arena-based allocator.

Returns:

a new unbounded arena-based allocator

Throws:

`IllegalStateException` - if `session` is not alive[PREVIEW].

`WrongThreadException` - if this method is called from a thread other than the thread owning[PREVIEW] `session`.

newNativeArena

```
static SegmentAllocator[PREVIEW] newNativeArena(long arenaSize,
                                                MemorySession[PREVIEW] session)
```

Creates an arena-based allocator used to allocate native memory segments. The returned allocator features a block size set to the specified arena size, and the native segments it allocates are associated with the provided memory session. Equivalent to the following code:

```
SegmentAllocator.newNativeArena(arenaSize, arenaSize, session);
```

Parameters:

`arenaSize` - the size (in bytes) of the allocation arena.

`session` - the memory session associated with the segments allocated by the arena-based allocator.

Returns:

a new unbounded arena-based allocator

Throws:

`IllegalArgumentException` - if `arenaSize <= 0`.

`IllegalStateException` - if `session` is not alive[PREVIEW].

`WrongThreadException` - if this method is called from a thread other than the thread owning[PREVIEW] `session`.

newNativeArena

```
static SegmentAllocator[PREVIEW] newNativeArena(long arenaSize,
                                                long blockSize,
                                                MemorySession[PREVIEW] session)
```

Creates an arena-based allocator used to allocate native memory segments. The returned allocator features the given block size `B` and the given arena size `A`, and the native segments it allocates are associated with the provided memory session.

The allocator arena is first initialized by allocating[PREVIEW] a native memory segment `S` of size `B`. The allocator then responds to allocation requests in one of the following ways:

- if the size of the allocation requests is smaller than the size of `S`, and `S` has a *free* slice `S'` which fits that allocation request, return that `S'`.
- if the size of the allocation requests is smaller than the size of `S`, and `S` has no *free* slices which fits that allocation request, allocate a new segment `S'`, with size `B`, and set `S = S'`; the allocator then tries to respond to the same allocation request again.
- if the size of the allocation requests is bigger than the size of `S`, allocate a new segment `S'`, which has a sufficient size to satisfy the allocation request, and return `S'`.

This segment allocator can be useful when clients want to perform multiple allocation requests while avoiding the cost associated with allocating a new off-heap memory region upon each allocation request.

The returned allocator might throw an `OutOfMemoryError` if the total memory allocated with this allocator exceeds the arena size `A`, or the system capacity. Furthermore, the returned allocator is not thread safe. Concurrent allocation needs to be guarded with synchronization primitives.

Parameters:

`arenaSize` - the size (in bytes) of the allocation arena.

`blockSize` - the block size associated with the arena-based allocator.

`session` - the memory session associated with the segments returned by the arena-based allocator.

Returns:

a new unbounded arena-based allocator

Throws:

`IllegalArgumentException` - if `blockSize <= 0`, if `arenaSize <= 0` or if `arenaSize < blockSize`.

`IllegalStateException` - if `session` is not alive[PREVIEW].

`WrongThreadException` - if this method is called from a thread other than the thread owning[PREVIEW] `session`.

prefixAllocator

```
static SegmentAllocator[PREVIEW] prefixAllocator(MemorySegment[PREVIEW] segment)
```

Returns a segment allocator which responds to allocation requests by recycling a single segment. Each new allocation request will return a new slice starting at the segment offset `0` (alignment constraints are ignored by this allocator), hence the name *prefix allocator*. Equivalent to (but likely more efficient than) the following code:

```
MemorySegment segment = ...
SegmentAllocator prefixAllocator = (size, align) -> segment.asSlice(0, size);
```

This allocator can be useful to limit allocation requests in case a client knows that they have fully processed the contents of the allocated segment before the subsequent allocation request takes place.

While the allocator returned by this method is *thread-safe*, concurrent access on the same recycling allocator might cause a thread to overwrite contents written to the underlying segment by a different thread.

Parameters:

`segment` - the memory segment to be recycled by the returned allocator.

Returns:

an allocator which recycles an existing segment upon each new allocation request.

implicitAllocator

```
static SegmentAllocator[PREVIEW] implicitAllocator()
```

Returns an allocator which allocates native segments in independent implicit memory sessions[PREVIEW]. Equivalent to (but likely more efficient than) the following code:

```
SegmentAllocator implicitAllocator = (size, align) -> MemorySegment.allocateNative(size, align, MemorySes
```

Returns:

an allocator which allocates native segments in independent implicit memory sessions[PREVIEW].