Interface ValueLayout

All Superinterfaces:
MemoryLayout

All Known Subinterfaces:
AddressLayout, ValueLayout.OfBoolean, ValueLayout.OfByte, ValueLayout.OfChar, ValueLayout.OfDouble, ValueLayout.OfFloat, ValueLayout.OfInt, ValueLayout.OfLong, ValueLayout.OfShort

All Known Implementing Classes:
MemoryLayout

```
public sealed interface ValueLayout
extends MemoryLayout
permits ValueLayout.OfBoolean, ValueLayout.OfByte, ValueLayout.OfChar, ValueLayout.OfInt, ValueLayout.OfFloat, ValueLayout.OfLong, ValueLayout.OfDouble, AddressLayout
```

A layout that models values of basic data types. Examples of values modeled by a value layout are *integral* values (either signed or unsigned), *floating point* values and *address* values.

Each value layout has a size, an alignment (both expressed in bytes), a [byte order](ByteOrder.html), and a *carrier*, that is, the Java type that should be used when [accessing](MemorySegment.html) a region of memory using this layout.

This class defines useful value layout constants for Java primitive types and addresses.

API Note:
Some characteristics of the Java layout constants are platform-dependent. For instance, the byte order of these constants is set to the [native byte order](ByteOrder.html#nativeOrder), thus making it easy to work with other APIs, such as arrays and [ByteBuffer](../../java/nio/ByteBuffer.html).

Implementation Requirements:
implementing classes and subclasses are immutable, thread-safe and [value-based](../lang/doc-files/ValueBased.html).

Sealed Class Hierarchy Graph:

Since:
22

Nested Class Summary

Nested Classes

Modifier and Type	Interface	Description
static interface	ValueLayout.OfBoolean	A value layout whose carrier is boolean.class.
static interface	ValueLayout.OfByte	A value layout whose carrier is byte.class.
static interface	ValueLayout.OfChar	A value layout whose carrier is char.class.
static interface	ValueLayout.OfDouble	A value layout whose carrier is double.class.
static interface	ValueLayout.OfFloat	A value layout whose carrier is float.class.
static interface	ValueLayout.OfInt	A value layout whose carrier is int.class.
static interface	ValueLayout.OfLong	A value layout whose carrier is long.class.
static interface	ValueLayout.OfShort	A value layout whose carrier is short.class.

Nested classes/interfaces declared in interface java.lang.foreign.MemoryLayout
MemoryLayout.PathElement

Field Summary

Fields

Modifier and Type	Field	Description
static final AddressLayout	ADDRESS	An address layout constant whose size is the same as that of a machine address (size_t), byte order set to nativeOrder(), byte order set to ByteOrder.nativeOrder().
static final AddressLayout	ADDRESS_UNALIGNED	An unaligned address layout constant whose size is the same as that of a machine address (size_t), and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfBoolean	JAVA_BOOLEAN	A value layout constant whose size is the same as that of a Java boolean, byte alignment set to 1, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfByte	JAVA_BYTE	A value layout constant whose size is the same as that of a Java byte, byte alignment set to 1, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfChar	JAVA_CHAR	A value layout constant whose size is the same as that of a Java char, byte alignment set to 2, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfChar	JAVA_CHAR_UNALIGNED	An unaligned value layout constant whose size is the same as that of a Java char and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfDouble	JAVA_DOUBLE	A value layout constant whose size is the same as that of a Java double, byte alignment set to 8, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfDouble	JAVA_DOUBLE_UNALIGNED	An unaligned value layout constant whose size is the same as that of a Java double and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfFloat	JAVA_FLOAT	A value layout constant whose size is the same as that of a Java float, byte alignment set to 4, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfFloat	JAVA_FLOAT_UNALIGNED	An unaligned value layout constant whose size is the same as that of a Java float and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfInt	JAVA_INT	A value layout constant whose size is the same as that of a Java int, byte alignment set to 4, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfInt	JAVA_INT_UNALIGNED	An unaligned value layout constant whose size is the same as that of a Java int and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfLong	JAVA_LONG	A value layout constant whose size is the same as that of a Java long, byte alignment set to 8, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfLong	JAVA_LONG_UNALIGNED	An unaligned value layout constant whose size is the same as that of a Java long and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfShort	JAVA_SHORT	A value layout constant whose size is the same as that of a Java short, byte alignment set to 2, and byte order set to ByteOrder.nativeOrder().
static final ValueLayout.OfShort	JAVA_SHORT_UNALIGNED	An unaligned value layout constant whose size is the same as that of a Java short and byte order set to ByteOrder.nativeOrder().

Method Summary

All Methods | **Instance Methods** | **Abstract Methods**

Modifier and Type	Method	Description
Class<?>	carrier()	Returns the carrier associated with this value layout.
ByteOrder	order()	Returns the value's byte order.
VarHandle	varHandle()	Returns a var handle which can be used to access values described by this value layout, in a given memory segment.
ValueLayout	withByteAlignment(long byteAlignment)	Returns a memory layout with the same characteristics as this layout, but with the given alignment constraint (in bytes).
ValueLayout	withName(String name)	Returns a memory layout with the same characteristics as this layout, but with the given name.
ValueLayout	withOrder(ByteOrder order)	Returns a value layout with the same characteristics as this layout, but with the given byte order.
ValueLayout	withoutName()	Returns a memory layout with the same characteristics as this layout, but with no name.

Methods declared in interface java.lang.foreign.MemoryLayout
arrayElementVarHandle, byteAlignment, byteOffset, byteOffsetHandle, bytesize, equals, hashCode, name, scale, scaleHandle, select, sliceHandle, toString, varHandle

Field Details

ADDRESS

```
static final AddressLayout ADDRESS
```

An address layout constant whose size is the same as that of a machine address (size_t), byte alignment set to sizeof(size_t), byte order set to ByteOrder.nativeOrder().

JAVA_BYTE

```
static final ValueLayout.OfByte JAVA_BYTE
```

A value layout constant whose size is the same as that of a Java byte, byte alignment set to 1, and byte order set to ByteOrder.nativeOrder().

JAVA_BOOLEAN

```
static final ValueLayout.OfBoolean JAVA_BOOLEAN
```

A value layout constant whose size is the same as that of a Java boolean, byte alignment set to 1, and byte order set to ByteOrder.nativeOrder().

JAVA_CHAR

```
static final ValueLayout.OfChar JAVA_CHAR
```

A value layout constant whose size is the same as that of a Java char, byte alignment set to 2, and byte order set to ByteOrder.nativeOrder().

JAVA_SHORT

```
static final ValueLayout.OfShort JAVA_SHORT
```

A value layout constant whose size is the same as that of a Java short, byte alignment set to 2, and byte order set to ByteOrder.nativeOrder().

JAVA_INT

```
static final ValueLayout.OfInt JAVA_INT
```

A value layout constant whose size is the same as that of a Java int, byte alignment set to 4, and byte order set to ByteOrder.nativeOrder().

JAVA_LONG

```
static final ValueLayout.OfLong JAVA_LONG
```

A value layout constant whose size is the same as that of a Java long, byte alignment set to 8, and byte order set to ByteOrder.nativeOrder().

JAVA_FLOAT

```
static final ValueLayout.OfFloat JAVA_FLOAT
```

A value layout constant whose size is the same as that of a Java float, byte alignment set to 4, and byte order set to ByteOrder.nativeOrder().

JAVA_DOUBLE

```
static final ValueLayout.OfDouble JAVA_DOUBLE
```

A value layout constant whose size is the same as that of a Java double, byte alignment set to 8, and byte order set to ByteOrder.nativeOrder().

ADDRESS_UNALIGNED

```
static final AddressLayout ADDRESS_UNALIGNED
```

An unaligned address layout constant whose size is the same as that of a machine address (size_t), and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
ADDRESS.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

JAVA_CHAR_UNALIGNED

```
static final ValueLayout.OfChar JAVA_CHAR_UNALIGNED
```

An unaligned value layout constant whose size is the same as that of a Java char and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
JAVA_CHAR.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

JAVA_SHORT_UNALIGNED

```
static final ValueLayout.OfShort JAVA_SHORT_UNALIGNED
```

An unaligned value layout constant whose size is the same as that of a Java short and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
JAVA_SHORT.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

JAVA_INT_UNALIGNED

```
static final ValueLayout.OfInt JAVA_INT_UNALIGNED
```

An unaligned value layout constant whose size is the same as that of a Java int and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
JAVA_INT.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

JAVA_LONG_UNALIGNED

```
static final ValueLayout.OfLong JAVA_LONG_UNALIGNED
```

An unaligned value layout constant whose size is the same as that of a Java long and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
JAVA_LONG.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

JAVA_FLOAT_UNALIGNED

```
static final ValueLayout.OfFloat JAVA_FLOAT_UNALIGNED
```

An unaligned value layout constant whose size is the same as that of a Java float and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
JAVA_FLOAT.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

JAVA_DOUBLE_UNALIGNED

```
static final ValueLayout.OfDouble JAVA_DOUBLE_UNALIGNED
```

An unaligned value layout constant whose size is the same as that of a Java double and byte order set to ByteOrder.nativeOrder(). Equivalent to the following code:

```
JAVA_DOUBLE.withByteAlignment(1);
```

API Note:
Care should be taken when using unaligned value layouts as they may induce performance and portability issues.

Method Details

order

```
ByteOrder order()
```

Returns the value's byte order.

Returns:
the value's byte order

withOrder

```
ValueLayout withOrder(ByteOrder order)
```

Returns a value layout with the same characteristics as this layout, but with the given byte order.

Parameters:
`order` - the desired byte order

Returns:
a value layout with the same characteristics as this layout, but with the given byte order

withoutName

```
ValueLayout withoutName()
```

Returns a memory layout with the same characteristics as this layout, but with no name.

Specified by:
`withoutName` in interface `MemoryLayout`

Returns:
a memory layout with the same characteristics as this layout, but with no name

See Also:
`MemoryLayout.name()`

carrier

```
Class<?> carrier()
```

Returns the carrier associated with this value layout.

Returns:
the carrier associated with this value layout.

withName

```
ValueLayout withName(String name)
```

Returns a memory layout with the same characteristics as this layout, but with the given name.

Specified by:
`withName` in interface `MemoryLayout`

Parameters:
`name` - the layout name

Returns:
a memory layout with the same characteristics as this layout, but with the given name

See Also:
`MemoryLayout.name()`

withByteAlignment

```
ValueLayout withByteAlignment(long byteAlignment)
```

Returns a memory layout with the same characteristics as this layout, but with the given alignment constraint (in bytes).

Specified by:
`withByteAlignment` in interface `MemoryLayout`

Parameters:
`byteAlignment` - the layout alignment constraint, expressed in bytes

Returns:
a memory layout with the same characteristics as this layout, but with the given alignment constraint (in bytes)

Throws:
`IllegalArgumentException` - if `byteAlignment` is not a power of two

varHandle

```
VarHandle varHandle()
```

Returns a var handle which can be used to access values described by this value layout, in a given memory segment.

The returned var handle's [var type](VarHandle.html#varType) is the [carrier type](#carrier) of this value layout, and the list of coordinate types in ([MemorySegment, long](../../java/lang/MemorySegment.html), where the `MemorySegment` coordinate corresponds to the memory segment to be accessed, and the `long` coordinate corresponds to the byte offset into the accessed memory segment, at which the access occurs.

The returned var handle checks that accesses are aligned according to this value layout's [alignment constraint](#withByteAlignment).

API Note:
This method is similar, but more efficient than calling `MemoryLayout.varHandle(PathElement...)` with an empty path element array, as it avoids the creation of the var args array. The returned var handle features certain [access mode restrictions](MemoryLayout.html#access-mode-restrictions) common to all memory access var handles derived from memory layouts.

Returns:
a var handle which can be used to access values described by this value layout, in a given memory segment

See Also:
`MemoryLayout.varHandle(PathElement...)`