Module java.base
Package java.lang.constant

Interface ClassDesc

All Superinterfaces:

ConstantDesc, TypeDescriptor, TypeDescriptor.OfField<ClassDesc>

```
public sealed interface ClassDesc
extends ConstantDesc, TypeDescriptor.OfField<ClassDesc>
```

A nominal descriptor for a Class constant.

For common system types, including all the primitive types, there are predefined ClassDesc constants in ConstantDescs. (The java.lang.constant APIs consider void to be a primitive type.) To create a ClassDesc for a class or interface type, use of(java.lang.String) or ofDescriptor(String); to create a ClassDesc for an array type, use ofDescriptor(String), or first obtain a ClassDesc for the component type and then call the arrayType() or arrayType(int) methods.

Since:

12

See Also:

ConstantDescs

Nested Class Summary

Nested classes/interfaces declared in interface java.lang.invoke.TypeDescriptor

TypeDescriptor.OfField<F extends TypeDescriptor.OfField<F>>, TypeDescriptor.OfMethod<F extends TypeDescriptor.OfField<F>,M extends TypeDescriptor.OfMethod<F,M>>

Method Summary

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

Modifier and Type	Method	Description
default ClassDesc	arrayType()	Returns a ClassDesc for an array type whose component type is described by this ClassDesc.
default ClassDesc	arrayType(int rank)	Returns a ClassDesc for an array type of the specified rank, whose component type is described by this ClassDesc.
default ClassDesc	componentType()	Returns the component type of this ClassDesc, if it describes an array type, or `null` otherwise.
String	descriptorString()	Returns a field type descriptor string for this type
default String	displayName()	Returns a human-readable name for the type described by this descriptor.
boolean	equals(Object o)	Compare the specified object with this descriptor for equality.
default boolean	isArray()	Returns whether this ClassDesc describes an array type.
default boolean	isClassOrInterface()	Returns whether this ClassDesc describes a class or interface type.
default boolean	isPrimitive()	Returns whether this ClassDesc describes a primitive type.
default ClassDesc	nested(String nestedName)	Returns a ClassDesc for a nested class of the class or interface type described by this ClassDesc.

default ClassDesc	nested(String firstNestedName, String... moreNestedNames)	Returns a ClassDesc for a nested class of the class or interface type described by this ClassDesc.
static ClassDesc	of(String name)	Returns a ClassDesc for a class or interface type, given the name of the class or interface, such as "java.lang.String".
static ClassDesc	of(String packageName, String className)	Returns a ClassDesc for a class or interface type, given a package name and the unqualified (simple) name for the class or interface.
static ClassDesc	ofDescriptor(String descriptor)	Returns a ClassDesc given a descriptor string for a class, interface, array, or primitive type.
static ClassDesc	ofInternalName(String name)	Returns a ClassDesc for a class or interface type, given the name of the class or interface in internal form, such as "java/lang/String".
default String	packageName()	Returns the package name of this ClassDesc, if it describes a class or interface type.
Class<?>	resolveConstantDesc (MethodHandles.Lookup lookup)	Resolves this descriptor reflectively, emulating the resolution behavior of JVMS 5.4.3 and the access control behavior of JVMS 5.4.4.

Method Details

of

static ClassDesc of(String name)

Returns a ClassDesc for a class or interface type, given the name of the class or interface, such as "java.lang.String". (To create a descriptor for an array type, either use ofDescriptor(String) or arrayType(); to create a descriptor for a primitive type, use ofDescriptor(String) or use the predefined constants in ConstantDescs).

Parameters:

`name` - the fully qualified (dot-separated) binary class name

Returns:

a ClassDesc describing the desired class

Throws:

`NullPointerException` - if the argument is `null`

`IllegalArgumentException` - if the name string is not in the correct format

See Also:

`ofDescriptor(String)`, `ofInternalName(String)`

ofInternalName

```
static ClassDesc ofInternalName(String name)
```

Returns a ClassDesc for a class or interface type, given the name of the class or interface in internal form, such as `"java/lang/String"`.

API Note:

To create a descriptor for an array type, either use `ofDescriptor(String)` or `arrayType()`; to create a descriptor for a primitive type, use `ofDescriptor(String)` or use the predefined constants in `ConstantDescs`.

Parameters:

`name` - the fully qualified class name, in internal (slash-separated) form

Returns:

a ClassDesc describing the desired class

Throws:

`NullPointerException` - if the argument is `null`

`IllegalArgumentException` - if the name string is not in the correct format

See *Java Virtual Machine Specification*:

4.2.1 Binary Class and Interface Names

Since:

20

See Also:

`of(String)`, `ofDescriptor(String)`

of

```
static ClassDesc of(String packageName,
                    String className)
```

Returns a ClassDesc for a class or interface type, given a package name and the unqualified (simple) name for the class or interface.

Parameters:

`packageName` - the package name (dot-separated); if the package name is the empty string, the class is considered to be in the unnamed package

`className` - the unqualified (simple) class name

Returns:

a ClassDesc describing the desired class

Throws:

`NullPointerException` - if any argument is `null`

`IllegalArgumentException` - if the package name or class name are not in the correct format

ofDescriptor

```
static ClassDesc ofDescriptor(String descriptor)
```

Returns a ClassDesc given a descriptor string for a class, interface, array, or primitive type.

API Note:

A field type descriptor string for a non-array type is either a one-letter code corresponding to a primitive type (`"J"`, `"I"`, `"C"`, `"S"`, `"B"`, `"D"`, `"F"`, `"Z"`, `"V"`), or the letter `"L"`, followed by the fully qualified binary name of a class, followed by `";"`. A field type descriptor for an array type is the character `"["` followed by the field descriptor for the component type. Examples of valid type

descriptor strings include `"Ljava/lang/String;"`, `"I"`, `"[I"`, `"V"`, `"[Ljava/lang/String;"`, etc. See JVMS 4.3.2 ("Field Descriptors") for more detail.

Parameters:

`descriptor` - a field descriptor string

Returns:

a ClassDesc describing the desired class

Throws:

`NullPointerException` - if the argument is `null`

`IllegalArgumentException` - if the descriptor string is not in the correct format

See *Java Virtual Machine Specification*:

4.3.2 Field Descriptors
4.4.1 The CONSTANT_Class_info Structure

See Also:

`of(String)`, `ofInternalName(String)`

arrayType

```
default ClassDesc arrayType()
```

Returns a ClassDesc for an array type whose component type is described by this ClassDesc.

Specified by:

`arrayType` in interface `TypeDescriptor.OfField<ClassDesc>`

Returns:

a ClassDesc describing the array type

Throws:

`IllegalStateException` - if the resulting ClassDesc would have an array rank of greater than 255

See *Java Virtual Machine Specification*:

4.4.1 The CONSTANT_Class_info Structure

arrayType

```
default ClassDesc arrayType(int rank)
```

Returns a ClassDesc for an array type of the specified rank, whose component type is described by this ClassDesc.

Parameters:

`rank` - the rank of the array

Returns:

a ClassDesc describing the array type

Throws:

`IllegalArgumentException` - if the rank is less than or equal to zero or if the rank of the resulting array type is greater than 255

See *Java Virtual Machine Specification*:

4.4.1 The CONSTANT_Class_info Structure

nested

```
default ClassDesc nested(String nestedName)
```

Returns a ClassDesc for a nested class of the class or interface type described by this ClassDesc.

API Note:

Example: If descriptor `d` describes the class `java.util.Map`, a descriptor for the class `java.util.Map.Entry` could be obtained by `d.nested("Entry")`.

Parameters:

`nestedName` - the unqualified name of the nested class

Returns:

a ClassDesc describing the nested class

Throws:

`NullPointerException` - if the argument is `null`

`IllegalStateException` - if this ClassDesc does not describe a class or interface type

IllegalArgumentException - if the nested class name is invalid

nested

```
default ClassDesc nested(String firstNestedName,
                         String... moreNestedNames)
```

Returns a ClassDesc for a nested class of the class or interface type described by this ClassDesc.

Parameters:

firstNestedName - the unqualified name of the first level of nested class

moreNestedNames - the unqualified name(s) of the remaining levels of nested class

Returns:

a ClassDesc describing the nested class

Throws:

NullPointerException - if any argument or its contents is null

IllegalStateException - if this ClassDesc does not describe a class or interface type

IllegalArgumentException - if the nested class name is invalid

isArray

```
default boolean isArray()
```

Returns whether this ClassDesc describes an array type.

Specified by:

isArray in interface TypeDescriptor.OfField<ClassDesc>

Returns:

whether this ClassDesc describes an array type

isPrimitive

```
default boolean isPrimitive()
```

Returns whether this ClassDesc describes a primitive type.

Specified by:

`isPrimitive` in interface `TypeDescriptor.OfField<ClassDesc>`

Returns:

whether this ClassDesc describes a primitive type

isClassOrInterface

```
default boolean isClassOrInterface()
```

Returns whether this ClassDesc describes a class or interface type.

Returns:

whether this ClassDesc describes a class or interface type

componentType

```
default ClassDesc componentType()
```

Returns the component type of this ClassDesc, if it describes an array type, or `null` otherwise.

Specified by:

`componentType` in interface `TypeDescriptor.OfField<ClassDesc>`

Returns:

a ClassDesc describing the component type, or `null` if this descriptor does not describe an array type

packageName

`default String packageName()`

Returns the package name of this ClassDesc, if it describes a class or interface type.

Returns:

the package name, or the empty string if the class is in the default package, or this ClassDesc does not describe a class or interface type

displayName

`default String displayName()`

Returns a human-readable name for the type described by this descriptor.

Implementation Requirements:

The default implementation returns the simple name (e.g., `int`) for primitive types, the unqualified class name for class or interface types, or the display name of the component type suffixed with the appropriate number of `[]` pairs for array types.

Returns:

the human-readable name

descriptorString

`String descriptorString()`

Returns a field type descriptor string for this type

Specified by:

`descriptorString` in interface `TypeDescriptor`

Returns:

the descriptor string

See *Java Virtual Machine Specification*:

4.3.2 Field Descriptors

resolveConstantDesc

```
Class<?> resolveConstantDesc(MethodHandles.Lookup lookup)
                      throws ReflectiveOperationException
```

Description copied from interface: `ConstantDesc`

Resolves this descriptor reflectively, emulating the resolution behavior of JVMS 5.4.3 and the access control behavior of JVMS 5.4.4. The resolution and access control context is provided by the `MethodHandles.Lookup` parameter. No caching of the resulting value is performed.

Specified by:

`resolveConstantDesc` in interface `ConstantDesc`

Parameters:

`lookup` - The `MethodHandles.Lookup` to provide name resolution and access control context

Returns:

the resolved constant value

Throws:

`ReflectiveOperationException` - if a class, method, or field could not be reflectively resolved in the course of resolution

equals

```
boolean equals(Object o)
```

Compare the specified object with this descriptor for equality. Returns `true` if and only if the specified object is also a ClassDesc and both describe the same type.

Overrides:

`equals` in class `Object`

Parameters:

o - the other object

Returns:

whether this descriptor is equal to the other object

See Also:

`Object.hashCode()`, `HashMap`
