Module java.compiler
Package javax.lang.model.element

Interface TypeElement

All Superinterfaces:
AnnotatedConstruct, Element, Parameterizable, QualifiedNameable

```
public interface TypeElement
extends Element, Parameterizable, QualifiedNameable
```

Represents a class or interface program element. Provides access to information about the class or interface and its members. Note that an enum class and a record class are specialized kinds of classes and an annotation interface is a specialized kind of interface.

While a `TypeElement` represents a class or interface *element*, a `DeclaredType` represents a class or interface *type*, the latter being a use (or *invocation*) of the former. The distinction is most apparent with generic types, for which a single element can define a whole family of types. For example, the element `java.util.Set` corresponds to the parameterized types `java.util.Set<String>` and `java.util.Set<Number>` (and many others), and to the raw type `java.util.Set`.

Each method of this interface that returns a list of elements will return them in the order that is natural for the underlying source of program information. For example, if the underlying source of information is Java source code, then the elements will be returned in source code order.

API Note:
The represented class or interface may have a reference representation (either source code or executable output). Multiple classes and interfaces can share the same reference representation backing construct. For example, multiple classes and interface can be declared in the same source file, including, but are not limited to:
- a top-level class or interface and auxiliary classes and interfaces
- a top-level class or interface and nested class and interfaces within it

In the context of annotation processing, a type element can be:
- created from the initial inputs to a run of the tool
- created from source code or class files written by a processor
- queried for in the configured environment

Since:
1.6

See Also:
DeclaredType

Method Summary

All Methods | Instance Methods | Abstract Methods | Default Methods

Modifier and Type	Method	Description
TypeMirror	asType()	Returns the type defined by this class or interface element, returning the *prototypical type* for an element representing a generic type.
List<? extends Element>	getEnclosedElements()	Returns the fields, methods, constructors, record components, and member classes and interfaces that are directly declared in this class or interface.
Element	getEnclosingElement()	Returns the package of a top-level class or interface and returns the immediately lexically enclosing element for a nested class or interface.
List<? extends TypeMirror>	getInterfaces()	Returns the interface types directly implemented by this class or extended by this interface.
NestingKind	getNestingKind()	Returns the *nesting kind* of this class or interface element.
default List<? extends TypeMirror>	getPermittedSubclasses()	Returns the permitted classes of this class or interface element in declaration order.
Name	getQualifiedName()	Returns the fully qualified name of this class or interface element.
default List<? extends RecordComponentElement>	getRecordComponents()	Returns the record components of this class or interface element in declaration order.
Name	getSimpleName()	Returns the simple name of this class or interface element.
TypeMirror	getSuperclass()	Returns the direct superclass of this class or interface element.
List<? extends TypeParameterElement>	getTypeParameters()	Returns the formal type parameters of this class or interface element in declaration order.
default boolean	isUnnamed()	**Preview.** Returns `true` if this is an unnamed class and `false` otherwise.

Methods declared in interface javax.lang.model.element.Element
accept, equals, getAnnotation, getAnnotationMirrors, getAnnotationsByType, getKind, getModifiers, hashCode

Method Details

asType

```
TypeMirror asType()
```

Returns the type defined by this class or interface element, returning the *prototypical type* for an element representing a generic type.

A generic element defines a family of types, not just one. If this is a generic element, a prototypical type is returned which has the element's invocation on the type variables corresponding to its own formal type parameters. For example, for the generic class element `C<N extends Number>`, the parameterized type `C<N>` is returned. Otherwise, for a non-generic class or interface, the prototypical type mirror corresponds to a use of the type. None of the components of the prototypical type are annotated, including the prototypical type itself.

Specified by:
asType in interface Element

API Note:
The Types utility interface has more general methods for obtaining the full range of types defined by an element.

Returns:
the type defined by this type element

See Also:
Types.asMemberOf(DeclaredType, Element),
Types.getDeclaredType(TypeElement, TypeMirror...)

getEnclosedElements

```
List<? extends Element> getEnclosedElements()
```

Returns the fields, methods, constructors, record components, and member classes and interfaces that are directly declared in this class or interface. This includes any mandated elements such as the (implicit) default constructor and the implicit `values` and `valueOf` methods of an enum class.

Specified by:
getEnclosedElements in interface Element

API Note:
As a particular instance of the general accuracy requirements and the ordering behavior required of this interface, the list of enclosed elements will be returned in the natural order for the originating source of information about the class or interface. For example, if the information about the class or interface is originating from a source file, the elements will be returned in source code order. (However, in that case the ordering of implicitly declared elements, such as default constructors, is not specified.)

Returns:
the enclosed elements in proper order, or an empty list if none

See *Java Language Specification*:
8.8.9 Default Constructor
8.9.3 Enum Members
8.10.3 Record Members

See Also:
getEnclosedElements(),
PackageElement.getEnclosedElements(),
ModuleElement.getEnclosedElements(),
Elements.getAllMembers(javax.lang.model.element.TypeElement)

getNestingKind

```
NestingKind getNestingKind()
```

Returns the *nesting kind* of this class or interface element.

Returns:
the nesting kind of this class or interface element

getQualifiedName

```
Name getQualifiedName()
```

Returns the fully qualified name of this class or interface element. More precisely, it returns the *canonical* name. For local, anonymous, and unnamed classes and anonymous classes, which do not have canonical names, an empty name is returned.

The name of a generic class or interface does not include any reference to its formal type parameters. For example, the fully qualified name of the interface `java.util.Set<E>` is "`java.util.Set`". Nested classes and interfaces use "`.`" as a separator, as in "`java.util.Map.Entry`".

Specified by:
getQualifiedName in interface QualifiedNameable

Returns:
the fully qualified name of this class or interface, or an empty name if none

See *Java Language Specification*:
6.7 Fully Qualified Names and Canonical Names
7.3 Compilation Units

See Also:
Elements.getBinaryName(javax.lang.model.element.TypeElement)

getSimpleName

```
Name getSimpleName()
```

Returns the simple name of this class or interface element. For an anonymous class, an empty name is returned. For an unnamed class, a name matching the base name of the hosting file, minus any extension, is returned.

Specified by:
getSimpleName in interface Element

Returns:
the simple name of this class or interface, an empty name for an anonymous class

See Also:
PackageElement.getSimpleName(),
ExecutableElement.getSimpleName(),
getSimpleName(),
VariableElement.getSimpleName(),
ModuleElement.getSimpleName(),
RecordComponentElement.getSimpleName()

isUnnamed

```
default boolean isUnnamed()
```

`isUnnamed` is a reflective preview API of the Java platform.
Preview features may be removed in a future release, or upgraded to permanent features of the Java platform.

Returns `true` if this is an unnamed class and `false` otherwise.

Implementation Requirements:
The default implementation of this method returns `false`.

Returns:
`true` if this is an unnamed class and `false` otherwise

See *Java Language Specification*:
7.3 Compilation Units

Since:
21

getSuperclass

```
TypeMirror getSuperclass()
```

Returns the direct superclass of this class or interface element. If this class or interface element represents an interface or the class `java.lang.Object`, then a `NoType` with kind `NONE` is returned.

Returns:
the direct superclass, or a `NoType` if there is none

getInterfaces

```
List<? extends TypeMirror> getInterfaces()
```

Returns the interface types directly implemented by this class or extended by this interface.

Returns:
the interface types directly implemented by this class or extended by this interface, or an empty list if there are none

getTypeParameters

```
List<? extends TypeParameterElement> getTypeParameters()
```

Returns the formal type parameters of this class or interface element in declaration order.

Specified by:
getTypeParameters in interface Parameterizable

Returns:
the formal type parameters, or an empty list if there are none

getRecordComponents

```
default List<? extends RecordComponentElement> getRecordComponents()
```

Returns the record components of this class or interface element in declaration order.

Implementation Requirements:
The default implementations of this method returns an empty and unmodifiable list.

Returns:
the record components, or an empty list if there are none

Since:
16

getPermittedSubclasses

```
default List<? extends TypeMirror> getPermittedSubclasses()
```

Returns the permitted classes of this class or interface element in declaration order. Note that for an interface, permitted subclasses and subinterfaces can be returned.

Implementation Requirements:
The default implementations of this method returns an empty and unmodifiable list.

Returns:
the permitted classes, or an empty list if there are none

See *Java Language Specification*:
8.1.6 Permitted Direct Subclasses
9.1.4 Permitted Direct Subclasses and Subinterfaces

Since:
17

getEnclosingElement

```
Element getEnclosingElement()
```

Returns the package of a top-level class or interface and returns the immediately lexically enclosing element for a nested class or interface.

Specified by:
getEnclosingElement in interface Element

Returns:
the package of a top-level class or interface, the immediately lexically enclosing element for a nested class or interface

See Also:
Elements.getPackageOf(javax.lang.model.element.Element)