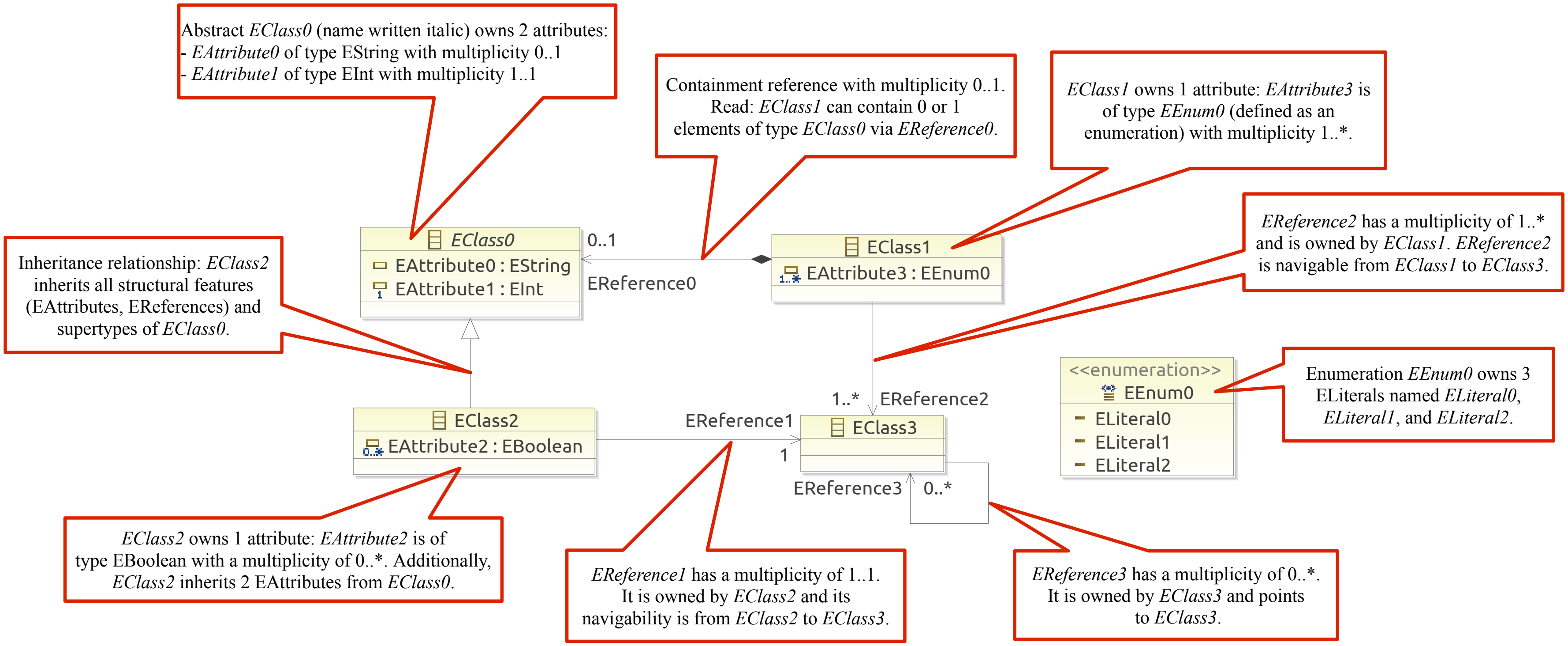
Ecore Model Reference



Abstract *EClass0* (name written italic) owns 2 attributes:
- *EAttribute0* of type EString with multiplicity 0..1
- *EAttribute1* of type EInt with multiplicity 1..1

Containment reference with multiplicity 0..1. Read: *EClass1* can contain 0 or 1 elements of type *EClass0* via *EReference0*.

EClass1 owns 1 attribute: *EAttribute3* is of type *EEnum0* (defined as an enumeration) with multiplicity 1..*.

EReference2 has a multiplicity of 1..* and is owned by *EClass1*. *EReference2* is navigable from *EClass1* to *EClass3*.

Inheritance relationship: *EClass2* inherits all structural features (EAttributes, EReferences) and supertypes of *EClass0*.

Enumeration *EEnum0* owns 3 ELiterals named *ELiteral0*, *ELiteral1*, and *ELiteral2*.

EClass2 owns 1 attribute: *EAttribute2* is of type EBoolean with a multiplicity of 0..*. Additionally, *EClass2* inherits 2 EAttributes from *EClass0*.

EReference1 has a multiplicity of 1..1. It is owned by *EClass2* and its navigability is from *EClass2* to *EClass3*.

EReference3 has a multiplicity of 0..*. It is owned by *EClass3* and points to *EClass3*.

eStructuralFeatures = all EAttributes and EReferences owned by an EClass, e.g.,
for *EClass0*: *EAttribute0* and *EAttribute1*
for *EClass1*: *EAttribute3*, *EReference0* and *EReference2*

eSuperTypes = all direct inheritance relationships of an EClass, e.g.,
for *EClass2*: *EClass0*
If there would be an inheritance relationship from *EClass0* to *EClass1*, *EClass2* would inherit all eStructuralFeatures from *EClass1*, as well: namely *EAttribute3*, *EReference0*, and *EReference2*.

An EReference with one end marked by a navigability arrow means

- that the EReference is navigable in the direction of that end and
- that the EReference is owned by the EClass at the end without the arrow.

Natural-Language Notation Reference

Structure of natural-language scenario tests:

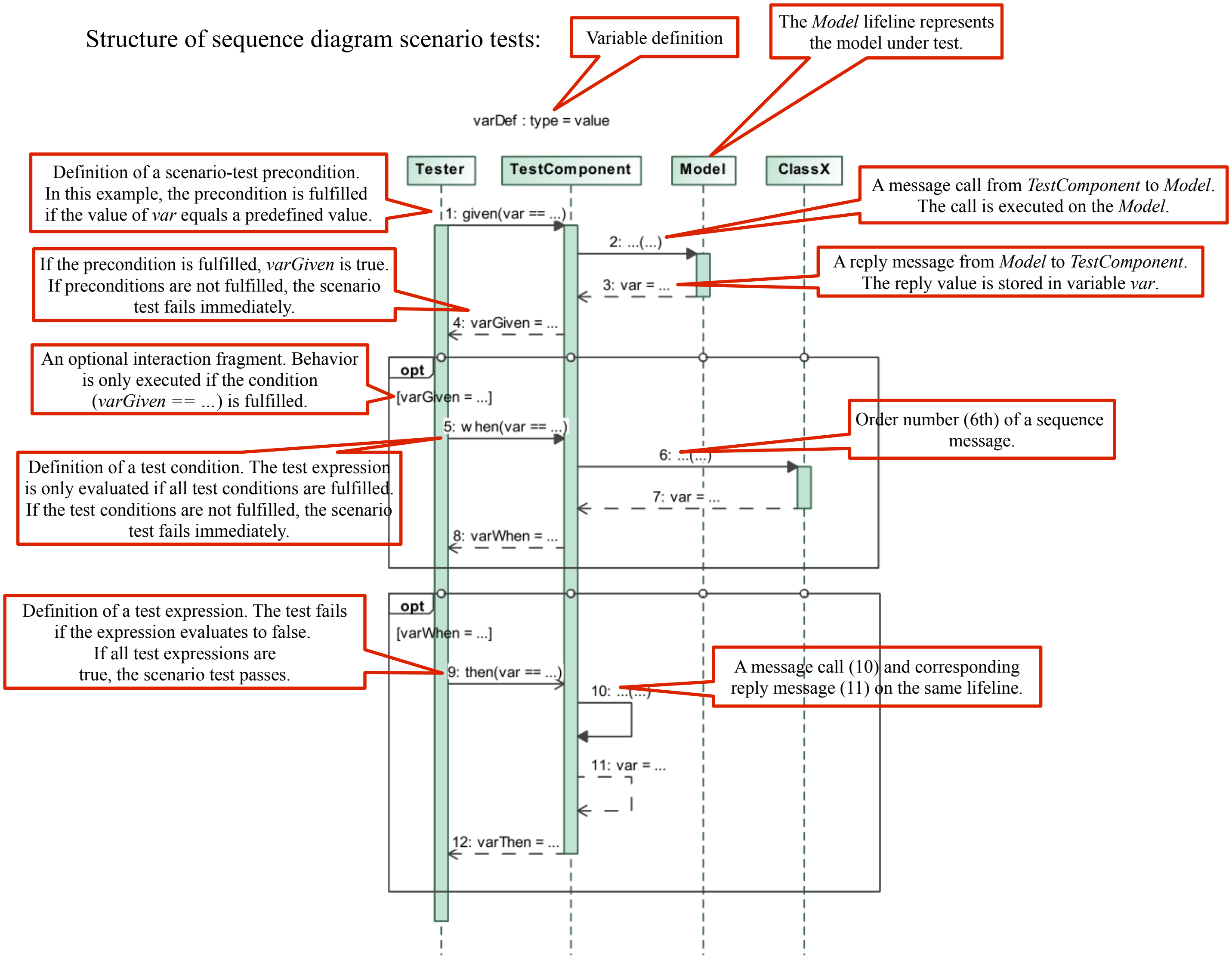
`Scenario:`	Begin of a test scenario.
`Given` `"..."`	Definition of a scenario test precondition. The scenario test is only evaluated if all preconditions are fulfilled. If preconditions are not fulfilled, the scenario test fails immediately.
`And` `"..."`	Alternative definition of a precondition.
`When` `"..."`	Definition of a test condition. The test expression is only evaluated if all test conditions are fulfilled. If test conditions are not fulfilled, the scenario test fails immediately.
`And` `"..."`	Alternative definition of a test condition.
`Then` `"..."`	Definition of a test expression. The test fails if expression is false. If all test expressions are true, the scenario test passes.
`And` `"..."`	Alternative definition of a test expression.

Language reference (sorted alphabetically):

`And` (after `Given`)	Alternative definition of a precondition.
`And` (after `When`)	Alternative definition of a test condition.
`And` (after `Then`)	Alternative definition of a test expression.
`Given`	Definition of a scenario test precondition. The scenario test is only evaluated if all preconditions are fulfilled. If preconditions are not fulfilled, the scenario test fails immediately.
`Scenario:`	Definition of a test scenario.
`Then`	Definition of a test expression. The test fails if expression is false. If all test expressions are true, the scenario test passes.
`When`	Definition of a test condition. The test expression is only evaluated if all test conditions are fulfilled. If test conditions are not fulfilled, the scenario test fails immediately.

Diagrammatic Notation Reference

Structure of sequence diagram scenario tests:



Variable definition

The *Model* lifeline represents the model under test.

Definition of a scenario-test precondition. In this example, the precondition is fulfilled if the value of *var* equals a predefined value.

A message call from *TestComponent* to *Model*. The call is executed on the *Model*.

If the precondition is fulfilled, *varGiven* is true. If preconditions are not fulfilled, the scenario test fails immediately.

A reply message from *Model* to *TestComponent*. The reply value is stored in variable *var*.

An optional interaction fragment. Behavior is only executed if the condition (*varGiven* == ...) is fulfilled.

Order number (6th) of a sequence message.

Definition of a test condition. The test expression is only evaluated if all test conditions are fulfilled. If the test conditions are not fulfilled, the scenario test fails immediately.

Definition of a test expression. The test fails if the expression evaluates to false. If all test expressions are true, the scenario test passes.

A message call (10) and corresponding reply message (11) on the same lifeline.

Language reference (sorted alphabetically):

`[exp : Expression] : Boolean`	Returns the result of evaluating the expression *exp*.
`abstract : Boolean`	Returns true if an EClass is declared abstract.
`ClassX`	Lifeline which represents an EClass named *ClassX* in the model under test.
`containment : Boolean`	Returns true if an EReference is a containment reference.
`exists(object : String) : Boolean`	Returns true if an *object* (e.g., an EClass, an EAttribute) exists. The object is selected via its name property.
`exists(exp : Expression [, exp : Expression]*) : Any`	Returns true if at least one object satisfies all conditions defined as *exp* (variable/value pairs).
`forAll(exp : Expression) : Boolean`	Returns true if all items in the collection satisfy the condition defined as *exp*.
`given(exp : Expression) : Boolean`	Definition of a scenario test precondition. The precondition is fulfilled if *exp* evaluates to true. The scenario test is only evaluated if all preconditions are

	fulfilled. If preconditions are not fulfilled, the scenario test fails immediately.
includesAll(col : Collection) : Boolean	Returns true if the collection includes all the items of collection *col*.
lowerBound : Integer	Returns the lower bound of the multiplicity interval.
Model	Lifeline which represents the model under test.
name : String	Specifies the name of an object.
opt	An optional interaction fragment. Behavior is only executed if the defined *condition* is fulfilled.
referencesTo(EClass : String) : Collection	Returns a collection of references pointing to an *EClass* (selected via its name property).
select(object : String) : Any	Returns an object (e.g., an EClass, an EAttribute) selected via its name property.
selectAllStructuralFeatures(sf : structuralFeature) : Collection	Returns a collection of structural features of type *sf* (either EAttribute or EReference) of a collection of EClasses.
selectAllSupertypes() : Collection	Returns a collection of all (direct and indirect) supertypes of an EClass.
selectClassViaReference(ref : String) : Class	Returns the EClass a reference *ref* is pointing to. The reference is selected via its name property.
selectStructuralFeatures([sf : structuralFeature]? [, exp : Expression]*) : Collection	Returns all structural features of type *sf* of an EClass which satisfies the conditions stated as *exp*. If no conditions are defined, all structural features of type *sf* are returned. If additionally no type is defined, all structural features are returned.
selectSupertype() : Class	Returns the direct supertype of an EClass.
selectTypeOf(structuralFeature : String) : Any	Returns the type of a *structuralFeature* (either EAttribute or EReference) selected via its name property.
selectTypesOf(sf : structuralFeature) : Collection	Returns a collection of types of structural features of type *sf* (either EAttribute or EReference) of a class.
size() : Integer	Returns the number of items the collection contains.
Tester	Lifeline which represents a test user.
TestComponent	Lifeline which represents a test scenario.
then(exp : Expression) : Boolean	Definition of a test expression. The test expression is fulfilled if *exp* evaluates to true. The test fails if expression evaluates to false. If all test expressions are true, the scenario test passes.
type : String	Returns the type of an object. For EReferences the type is the EClass the reference is pointing to.
upperBound : Integer	Returns the upper bound of the multiplicity interval (* ≙ -1 ≙ unlimited).
var = ... : Any	Variable *var* stores values of any type.
when(exp : Expression) : Boolean	Definition of a test condition. The condition is fulfilled if *exp* evaluates to true. The test expression is only evaluated if all test conditions are fulfilled. If test conditions are not fulfilled, the scenario test fails immediately.

Fully-structured Notation Reference

Structure of Epsilon scenario tests:

`@TestSuite`
`operation testSuite() {`
	Begin of a test suite.

`@TestCase`
`operation testCase() {`
	Begin of a test case.

`@TestScenario`	Begin of a test scenario.
`$pre ...`	Definition of a precondition.
`operation testScenario() {`	The scenario test is only evaluated if all preconditions are fulfilled. If preconditions are not fulfilled, the scenario test fails immediately.
`if (...) {`	Definition of a test condition. The test expression is only evaluated if all test conditions are fulfilled. If test conditions are not fulfilled, the scenario test fails immediately.
`assertTrue(...);`	Evaluates the test expression. The scenario test fails if expression is false. If all test expressions are true, the scenario test passes.
`} else {`	End of test expression.
`assertTrue(false);`	If test conditions are not fulfilled, the scenario test fails.
`}`	End of test condition.
`}`	End of test scenario.
`}`	End of test case.
`}`	End of test suite.

Language reference (sorted alphabetically):

`abstract : Boolean`	If true, the EClass does not provide a complete declaration and can not be instantiated. An abstract EClass is intended to be used by other EClasses (e.g., as the target of inheritance relationships).
`expression1.expression2`	Dot-notation: Either executes *expression2* on object *expression1* or returns a property *expression2* of the model element *expression1*.
`@TestSuite`	Definition of a test suite.
`@TestCase`	Definition of a test case (must be included in a test suite).
`@TestScenario`	Definition of a test scenario (must be included in a test case).
`$pre expression : Boolean`	All precondition expressions must evaluate true for the test scenario to be executed (otherwise the scenario test fails immediately).
`assertTrue(cond : Boolean)`	Fails the test if *cond* is false.
`closure(iterator | expression) : Collection`	Returns a collection containing the results of evaluating the transitive closure of the results produced by the expression on each item of the collection that is of the specified type.
`collect(iterator : Type | expression) : Collection`	Returns a collection containing the results of evaluating the expression on each item of the collection that is of the specified type.
`containment : EBoolean`	Returns the containment status of the EReference.

eLiterals : Collection	Returns a collection of all literals' names.
eStructuralFeatures : Collection	Returns a collection of all EAttributes and EReferences of an EClass.
eSuperTypes : Collection	Returns a collection of all direct EClass supertypes of an EClass.
eType : EClass	Returns the EClass (the type) of an eStructuralFeature (the type of an EReference or of an EAttribute). I.e. for EReferences: the EClass the EReference is pointing to.
exists(iterator : Type | condition) : Boolean	Returns true if there exists at least one item in the collection that satisfies the condition.
forAll(iterator : Type | condition) : Boolean	Returns true if all items in the collection satisfy the condition.
includes(item : Any) : Boolean	Returns true if the collection includes the *item*.
includesAll(col : Collection) : Boolean	Returns true if the collection includes all the items of collection *col*.
isTypeOf(type : Type) : Boolean	Returns true if the object is of the given type and false otherwise.
lowerBound : EInt	Returns the lower bound of the multiplicity interval.
Model!EClass.all() : Set	Returns all elements in the model under test (named *Model*) that are instances of type *EClass*.
name : EString	Returns the *name* property of the model element.
select(iterator : Type | condition) : Collection	Returns a sub-collection containing only items of the specified type that satisfy the condition.
selectOne(iterator | condition) : Any	Returns the first element that satisfies the condition.
Set{item1, item2, …, itemX}	Returns a unique and unordered collection.
size() : Integer	Returns the number of items the collection contains.
sum() : Real	Returns the sum of all reals/integers in the collection.
upperBound : EInt	Returns the upper bound of the multiplicity interval (* ≙ -1 ≙ unlimited).