

OCL Quick Reference

OCL Syntax is defined by a UML model (e.g., classes, attributes and side-effect free operations) and OCL types (see below). OCL semantics is defined by evaluating OCL expressions on an instance of the UML model. Evaluation of OCL expressions is side-effect free.

Basic types

Name	Operations (with intuitive semantics)
Boolean	<code>or, and, xor, not, =, <>, implies</code>
Integer	<code>=, <>, <, >, <=, >=, +, -, *, /, mod(), div(), abs(), max(), min(), round(), floor()</code>
Real	
String	<code>=, <>, concat(), size(), toLower(), toUpper(), substring()</code>

OclVoid and OclInvalid types

OclVoid has a single instance - null - and it conforms to all other types except OclInvalid. Any attribute call applied on null results in invalid. Any operation call applied on null results in invalid, except for the operations specified in the table on the right.

OclInvalid has a single instance - invalid - and it conforms to all other types. Any property call applied on invalid results in invalid. Any operation call applied on invalid results in invalid, except for the operations specified in the table on the right.

Collection types

There are four collection types (Set, OrderedSet, Bag, and Sequence) with Collection as the abstract supertype. OrderedSet and Sequence are ordered, while Set and OrderedSet have unique elements. All four collection types inherit operations from Collection.

Collection(T) operations	Description	Example
<code>any(expr: OclExpression): T</code>	An element in self that validates expr, null otherwise	<code>Set{0,3,0,9,1,2}->any(e e<1)=0.9</code>
<code>asBag(): Bag(T)</code>	A Bag containing all and only elements of self	<code>Set{1,2,0,3}->asBag()=Bag{2,0,1,3}</code>
<code>asOrderedSet(): OrderedSet(T)</code>	An OrderedSet containing all and only elements of self	<code>Bag{2,1,1}->asOrderedSet() = OrderedSet{1,2}</code>
<code>asSequence(): Sequence(T)</code>	A Sequence containing all and only elements of self	<code>Bag{2,1,1}->asSequence() = Sequence{1,1,2}</code>
<code>asSet(): Set(T)</code>	A Set containing all and only elements of self	<code>Bag{2,1,1}->asSet() = Set{2,1}</code>
<code>collect(expr: OclExpression): Collection(T2)</code>	A Collection containing the result of applying expr on all elements in self	<code>Set{'a', 'b'}->collect(e e.toUpper())=Set{'A', 'B'}</code>
<code>count(o: T): Integer</code>	Number of times o is in the collection self	<code>Set{3, null, 4,0}->count(null)=1</code>
<code>excludes(o: T): Boolean</code>	True iff o is not contained in self	<code>Sequence{2,3}->excludes(1)=true</code>
<code>excludesAll(c: Collection(T)): Boolean</code>	True iff no element of c is contained in self	<code>Set{2, 5, 3}->excludesAll(Set{4, 3})=false</code>
<code>excluding(o: T): Collection(T)</code>	A Collection containing all elements of self minus all occurrences of o	<code>Set{2, 5, 3}->excluding(3)=Set{2, 5}</code>
<code>exists(expr: OclExpression): Boolean</code>	True iff at least one element in self validates expr	<code>Sequence{2,3, 5,2}->exists(e e > 3)=true</code>
<code>flatten(): Collection(T2)</code>	A Collection containing all elements of self recursively flattened	<code>Set{Set{1,2},Bag{2,3,4}}->flatten()=Set{1,2,3}</code>
<code>forAll(expr: OclExpression): Boolean</code>	True iff all the elements contained in self validate expr	<code>Sequence{2,3,5,2}->forAll(e e>3)=false</code>
<code>includes(o: T): Boolean</code>	True iff o is contained in self	<code>Sequence{2,3}->includes(1)=false</code>
<code>includesAll(c: Collection(T)): Boolean</code>	True iff all elements of c are contained in self	<code>Set{2, 5, 3}->includesAll(Set{4, 3})=false</code>
<code>including(o: T): Collection(T)</code>	A Collection containing all elements of self followed by o	<code>Set{2,5}->including(3)=Set{2,5,3}</code>
<code>isEmpty(): Boolean</code>	True iff self is empty	<code>Sequence{null}->isEmpty()=false</code>
<code>isUnique(expr: OclExpression): Boolean</code>	True iff all elements contained in self evaluate to a distinct value for expr	<code>Set{2,3, 5,2}->isUnique(e e>1)=false</code>
<code>notEmpty(): Boolean</code>	True iff self contains at least one element	<code>Sequence{null}->notEmpty()=true</code>
<code>one(expr: OclExpression): Boolean</code>	True iff exactly one element in self validates expr	<code>Set{2,3, 5,2}->one(true)=false</code>
<code>product(c: Collection(T2)): Set(Tuple(first: T, second: T2))</code>	Set of Tuples which represents the cartesian product of self with c	<code>Set{3,4}->product(Set{3}) = Set{Tuple{3, 3}, Tuple{4, 3}}</code>
<code>reject(expr: OclExpression): Collection(T)</code>	A Collection with all elements of self except for those who validate expr	<code>Set{1,2,3,4}->reject(e e.mod(2)=0)=Set{1,3}</code>
<code>select(expr: OclExpression): Collection(T)</code>	A Collection with all elements of self that validate expr	<code>Set{1,2,3,4}->select(e e.mod(2)=0)=Set{2,4}</code>
<code>size(): Integer</code>	Number of elements in self	<code>Set{1,2,3,4}->size()=4</code>
<code>sortedBy(expr: OclExpression): Sequence(T)</code>	Sequence containing all elements from self sorted according to expr	<code>Set{2,1,4,3}->sortedBy(e e)=Sequence{1,2,3,4}</code>
<code>sum(): Real</code>	Sum of elements in self if they support the '+' operation, otherwise invalid	<code>Sequence{2, 4}->sum()=6</code>
Set(T) operations	Description	Example
<code>= (s: Set(T)): Boolean</code>	True iff self contains the same elements as s	<code>(Set{3, 5, 4} = Set{3, 4, 4, 5})=true</code>
<code>+(s: Set(T)): Set(T)</code>	Set containing all elements in self and not in s	<code>Set{'a', 'b', 'c'} - Set{'c', 'a'}=Set{'b'}</code>
<code>intersection(b: Bag(T)): Set(T)</code>	Bag containing all elements of self that are also contained in b	<code>Set{1,2,3}->intersection(Bag{1,2,1})=Set{1,2}</code>
<code>intersection(s: Set(T)): Set(T)</code>	Set of elements of self that are also contained in s	<code>Set{1,3}->intersection(Set{1,2})=Set{1}</code>
<code>symmetricDifference(s: Set(T)): Set(T)</code>	Set of the elements of self and s that are not present in both	<code>Set{1,3}->symmetricDiff(Set{1,2})=Set{2,3}</code>
<code>union(b: Bag(T)): Bag(T)</code>	Bag containing all elements of self and all elements of b	<code>Set{1,2}->union(Bag{2,2})=Bag{1,2,2,2}</code>
<code>union(s: Set(T)): Set(T)</code>	Set containing all elements of self and all elements of s	<code>Set{1,2}->union(Set{1,3})=Set{1,2,3}</code>
OrderedSet(T) operations	Description	Example
<code>= (os: OrderedSet(T)): Boolean</code>	True iff self contains the same elements as os regardless of element ordering	<code>(OrderedSet{3,4} = OrderedSet{1,3})=false</code>
<code>append(o: T): OrderedSet(T)</code>	OrderedSet containing all elements of self followed by o	<code>OrderedSet{1,3}->append(4)=OrderedSet{1,3,4}</code>
<code>at(i: Integer): T</code>	Element of self located at position i in the collection	<code>OrderedSet{'a', 'b'}->at(1)='a'</code>
<code>first(): T</code>	First element of self	<code>OrderedSet{'a', 'b'}->first()='a'</code>
<code>indexOf(o: T): Integer</code>	Position of o in self	<code>OrderedSet{'a', 'b'}->indexOf('a')=1</code>
<code>insertAt(i: Integer, o: T): OrderedSet(T)</code>	OrderedSet containing self with o inserted at i shifting subsequent elements	<code>OrderedSet{3}->insertAt(1,2)=OrderedSet{2,3}</code>
<code>last(): T</code>	Last element of self	<code>OrderedSet{'a', 'b'}->last()='b'</code>
<code>prepend(o: T): OrderedSet(T)</code>	OrderedSet containing o followed by all elements of self	<code>OrderedSet{1}->prepend(0)=OrderedSet{0,1}</code>
<code>subOrderedSet(s: Integer, e: Integer): OrderedSet(T)</code>	OrderedSet containing all elements of self between the positions s and e	<code>OrderedSet{3, 4, 2}->subOrderedSet(2, 3) = OrderedSet{4,2}</code>
Bag(T) operations	Description	Example
<code>= (b: Bag(T)): Boolean</code>	True iff self contains the same elements as b	<code>(Bag{3, 5, 4} = Bag{3, 4, 4, 5})=false</code>
<code>intersection(b: Bag(T)): Bag(T)</code>	Bag containing all elements of self that are also in b	<code>Bag{1,2,3}->intersection(Bag{1,2,1})=Bag{1,2}</code>
<code>intersection(s: Set(T)): Set(T)</code>	Set containing all elements of self that are also in s	<code>Bag{1,1}->intersection(Set{1,2})=Set{1}</code>
<code>union(b: Bag(T)): Bag(T)</code>	Bag containing all elements of self and all elements of b	<code>Bag{2,2}->union(Bag{2,2})=Bag{2,2,2,2}</code>
<code>union(s: Set(T)): Bag(T)</code>	Bag containing all elements of self and all elements of s	<code>Bag{2,2}->union(Set{1,2})=Bag{1,2,2,2}</code>
Sequence(T) operations	Description	Example
<code>= (s: Sequence(T)): Boolean</code>	True iff self contains the same elements as s in the same order	<code>(Sequence{3,3,4} = Sequence{3,4})=false</code>
<code>append(o: T): Sequence(T)</code>	Sequence containing all elements of self followed by o	<code>Sequence{1,3}->append(2)=Sequence{1,3,2}</code>
<code>at(i: Integer): T</code>	Element of self at the i position	<code>Sequence{'a','a','b'}->at(1)='a'</code>
<code>first(): T</code>	First element of self	<code>Sequence{'a','a','b'}->first()='a'</code>
<code>indexOf(o: T): Integer</code>	Position of o in sequence self	<code>Sequence{'a', 'b'}->indexOf('a')=1</code>
<code>insertAt(i: Integer, o: T): Sequence(T)</code>	Sequence containing self with o inserted at i shifting subsequent elements	<code>Sequence{2}->insertAt(1,2)=Sequence{2,2}</code>
<code>last(): T</code>	Last element of self	<code>Sequence{'a', 'b'}->last()='b'</code>
<code>prepend(o: T): Sequence(T)</code>	Sequence containing o followed by all elements of self	<code>Sequence{1}->prepend(0)=Sequence{0,1}</code>
<code>subSequence(s: Integer, e: Integer): Sequence(T)</code>	Sequence containing all elements of self between the positions s and e.	<code>Sequence{3, 4, 2}->subSequence(2, 3) = Sequence{4,2}</code>
<code>union(s: Sequence(T)): Sequence(T)</code>	Sequence containing all elements of self followed by all elements of s	<code>Sequence{1}->union(Sequence{2})=Seq{1,2}</code>