

## 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	<b>or</b> , <b>and</b> , <b>xor</b> , <b>not</b> , <b>=</b> , <b>&lt;&gt;</b> , <b>implies</b>
Integer	<b>=</b> , <b>&lt;&gt;</b> , <b>&lt;</b> , <b>&gt;</b> , <b>&lt;=</b> , <b>&gt;=</b> , <b>+</b> , <b>-</b> , <b>*</b> , <b>/</b> , <b>mod()</b> ,
Real	<b>div()</b> , <b>abs()</b> , <b>max()</b> , <b>min()</b> , <b>round()</b> , <b>floor()</b>
String	<b>=</b> , <b>&lt;&gt;</b> , <b>concat()</b> , <b>size()</b> , <b>toLower()</b> , <b>toUpper()</b> , <b>substring()</b>

### 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
<b>any</b> (expr: OclExpression): T	An element in self that validates expr, null otherwise	Set{0.3,0.9,1.2}->any(e e<1)=0.9
<b>asBag</b> ((): Bag(T)	A Bag containing all and only elements of self	Set{1,2,0,'3'}->asBag()=Bag{2,0,1,'3'}
<b>asOrderedSet</b> ((): OrderedSet(T)	An OrderedSet containing all and only elements of self	Bag{2,1,1}->asOrderedSet() = OrderedSet{1,2}
<b>asSequence</b> ((): Sequence(T)	A Sequence containing all and only elements of self	Bag{2,1,1}->asSequence() = Sequence{1,1,2}
<b>asSet</b> ((): Set(T)	A Set containing all and only elements of self	Bag{2,1,1}->asSet() = Set{2,1}
<b>collect</b> (expr: OclExpression): Collection(T2)	A Collection containing the result of applying expr on all elements in self	Set{'a', 'b'}->collect(e e.toUpper())=Set{'A','B'}
<b>count</b> (o: T): Integer	Number of times o is in the collection self	Set{3, null, 4,0}->count(null)=1
<b>excludes</b> (o: T): Boolean	True iff o is not contained in self	Sequence{2,3}->excludes(1)=true
<b>excludesAll</b> (c: Collection(T)): Boolean	True iff no element of c is contained in self	Set{2, 5, 3}->excludesAll(Set{4, 3})=false
<b>excluding</b> (o: T): Collection(T)	A Collection containing all elements of self minus all occurrences of o	Set{2, 5, 3}->excluding(3)=Set{2, 5}
<b>exists</b> (expr: OclExpression): Boolean	True iff at least one element in self validates expr	Sequence{2,3, 5,2}->exists(e e > 3)=true
<b>flatten</b> ((): Collection(T2)	A Collection containing all elements of self recursively flattened	Set{Set{1,2},Bag{2,3}}->flatten()=Set{1,2,3}
<b>forAll</b> (expr: OclExpression): Boolean	True iff all the elements contained in self validate expr	Sequence{2,3,5,2}->forAll(e e>3)=false
<b>includes</b> (o: T): Boolean	True iff o is contained in self	Sequence{2,3}->includes(1)=false
<b>includesAll</b> (c: Collection(T)): Boolean	True iff all elements of c are contained in self	Set{2, 5, 3}->includesAll(Set{4, 3})=false
<b>including</b> (o: T): Collection(T)	A Collection containing all elements of self followed by o	Set{2,5}->including(3)=Set{2,5,3}
<b>isEmpty</b> ((): Boolean	True iff self is empty	Sequence{null}->isEmpty()=false
<b>isUnique</b> (expr: OclExpression): Boolean	True iff all elements contained in self evaluate to a distinct value for expr	Set{2,3, 5,2}->isUnique(e e>1)=false
<b>notEmpty</b> ((): Boolean	True iff self contains at least one element	Sequence{null}->notEmpty ()=true
<b>one</b> (expr: OclExpression): Boolean	True iff exactly one element in self validates expr	Set{2,3, 5,2}->one(true)=false
<b>product</b> (c: Collection(T2)): Set(Tuple(first: T, second: T2))	Set of Tuples which represents the cartesian product of self with c	Set{3,4}->product(Set{3}) = Set{Tuple{3, 3}, Tuple{4, 3}}
<b>reject</b> (expr: OclExpression): Collection(T)	A Collection with all elements of self except for those who validate expr	Set{1,2,3,4}->reject(e e.mod(2)=0)=Set{1,3}
<b>select</b> (expr: OclExpression): Collection(T)	A Collection with all elements of self that validate expr	Set{1,2,3,4}->select(e e.mod(2)=0)=Set{2,4}
<b>size</b> ((): Integer	Number of elements in self	Set{1,2,3,4}->size()=4
<b>sortedBy</b> (expr: OclExpression): Sequence(T)	Sequence containing all elements from self sorted according to expr	Set{2,1,4,3}->sortedBy(e e)=Sequence{1,2,3,4}
<b>sum</b> ((): Real	Sum of elements in self if they support the '+' operation, otherwise invalid	Sequence{2, 4}->sum()=6
Set(T) operations	Description	Example
<b>=</b> (s: Set(T)): Boolean	True iff self contains the same elements as s	(Set{3, 5, 4} = Set{3, 4, 4, 5})=true
<b>+(s: Set(T)): Set(T)</b>	Set containing all elements in self and not in s	Set{'a', 'b', 'c'} - Set{'c', 'a'}=Set{'b'}
<b>intersection</b> (b: Bag(T)): Set(T)	Bag containing all elements of self that are also contained in b	Set{1,2,3}->intersection(Bag{1,2,1})=Set{1,2}
<b>intersection</b> (s: Set(T)): Set(T)	Set of elements of self that are also contained in s	Set{1,3}->intersection(Set{1,2})=Set{1}
<b>symmetricDifference</b> (s: Set(T)): Set(T)	Set of the elements of self and s that are not present in both	Set{1,3}->symmetricDiff(Set{1,2})=Set{2,3}
<b>union</b> (b: Bag(T)): Bag(T)	Bag containing all elements of self and all elements of b	Set{1,2}->union(Bag{2,2})=Bag{1,2,2,2}
<b>union</b> (s: Set(T)): Set(T)	Set containing all elements of self and all elements of s	Set{1,2}->union(Set{1,3})=Set{1,2,3}
OrderedSet(T) operations	Description	Example
<b>=</b> (os: OrderedSet(T)): Boolean	True iff self contains the same elements as os regardless of element ordering	(OrderedSet{3,4} = OrderedSet{1,3})=false
<b>append</b> (o: T): OrderedSet(T)	OrderedSet containing all elements of self followed by o	OrderedSet{1,3}->append(4)=OrderedSet{1,3,4}
<b>at</b> (i: Integer): T	Element of self located at position i in the collection	OrderedSet{'a', 'b'}->at(1)='a'
<b>first</b> ((): T	First element of self	OrderedSet{'a', 'b'}->first()='a'
<b>indexOf</b> (o: T): Integer	Position of o in self	OrderedSet{'a', 'b'}->indexOf('a')=1
<b>insertAt</b> (i: Integer, o: T): OrderedSet(T)	OrderedSet containing self with o inserted at i shifting subsequent elements	OrderedSet{3}->insertAt(1,2)=OrderedSet{2,3}
<b>last</b> ((): T	Last element of self	OrderedSet{'a', 'b'}->last()='b'
<b>prepend</b> (o: T): OrderedSet(T)	OrderedSet containing o followed by all elements of self	OrderedSet{1}->prepend(0)=OrderedSet{0,1}
<b>subOrderedSet</b> (s: Integer, e: Integer): OrderedSet(T)	OrderedSet containing all elements of self between the positions s and e	OrderedSet{3, 4, 2}->subOrderedSet(2, 3) = OrderedSet{4,2}
Bag(T) operations	Description	Example
<b>=</b> (b: Bag(T)): Boolean	True iff self contains the same elements as b	(Bag{3, 5, 4} = Bag{3, 4, 4, 5})=false
<b>intersection</b> (b: Bag(T)): Bag(T)	Bag containing all elements of self that are also in b	Bag{1,2,3}->intersection(Bag{1,2,1})=Bag{1,2}
<b>intersection</b> (s: Set(T)): Set(T)	Set containing all elements of self that are also in s	Bag{1,1}->intersection(Set{1,2})=Set{1}
<b>union</b> (b: Bag(T)): Bag(T)	Bag containing all elements of self and all elements of b	Bag{2,2}->union(Bag{2,2})=Bag{2,2,2,2}
<b>union</b> (s: Set(T)): Bag(T)	Bag containing all elements of self and all elements of s	Bag{2,2}->union(Set{1,2})=Bag{1,2,2,2}
Sequence(T) operations	Description	Example
<b>=</b> (s: Sequence(T)): Boolean	True iff self contains the same elements as s in the same order	(Sequence{3,3,4} = Sequence{3,4})=false
<b>append</b> (o: T): Sequence(T)	Sequence containing all elements of self followed by o	Sequence{1,3}->append(2)=Sequence{1,3,2}
<b>at</b> (i: Integer): T	Element of self at the i position	Sequence{'a','a','b'}->at(1)='a'
<b>first</b> ((): T	First element of self	Sequence{'a','a','b'}->first()='a'
<b>indexOf</b> (o: T): Integer	Position of o in sequence self	Sequence{'a', 'b'}->indexOf('a')=1
<b>insertAt</b> (i: Integer, o: T): Sequence(T)	Sequence containing self with o inserted at i shifting subsequent elements	Sequence{2}->insertAt(1,2)=Sequence{2,2}
<b>last</b> ((): T	Last element of self	Sequence{'a', 'b'}->last()='b'
<b>prepend</b> (o: T): Sequence(T)	Sequence containing o followed by all elements of self	Sequence{1}->prepend(0)=Sequence{0,1}
<b>subSequence</b> (s: Integer, e: Integer): Sequence(T)	Sequence containing all elements of self between the positions s and e.	Sequence{3, 4, 2}->subSequence(2, 3) = Sequence{4,2}
<b>union</b> (s: Sequence(T)): Sequence(T)	Sequence containing all elements of self followed by all elements of s	Sequence{1}->union(Sequence{2})=Seq{1,2}