Analysis and Design of Data Structures

Massimo Benerecetti & Fabio Mogavero

containers

special

Dat

Stack

```
Top: Dat
Pop: void
Push: Dat → void
SwapTop: Dat → void
```

Dat

Queue

```
Head: Dat
Dequeue: void
Enqueue: Dat → void
```

linear

Dat

Sequence

```
GetAt: Nat → Dat
SetAt: Dat × Nat → void
Search: Dat → Nat
Exists: Dat → bool
SubSequence: Nat × Nat → Seq
IsEqual: Sequence → bool
```

collections

Dat

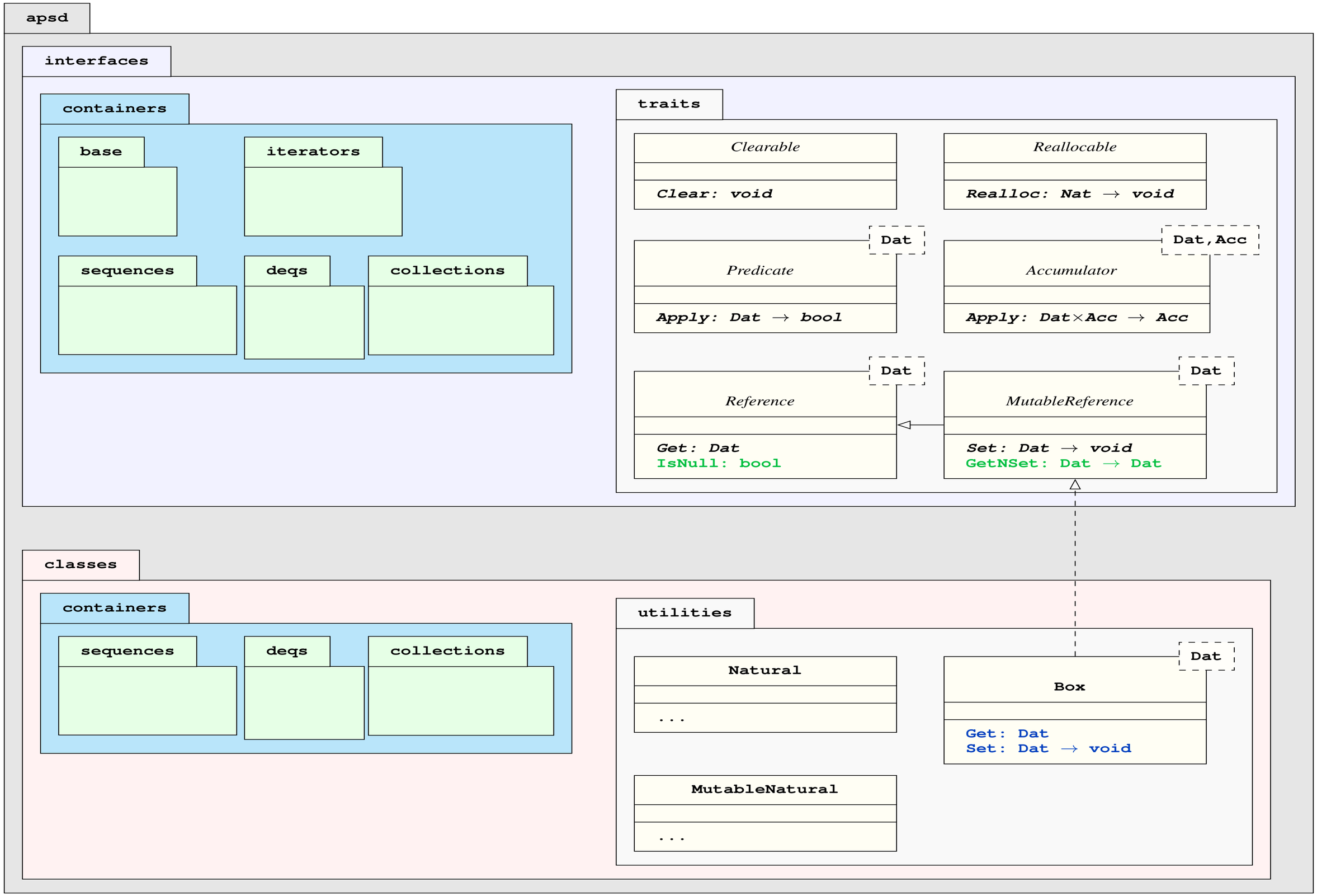
Set

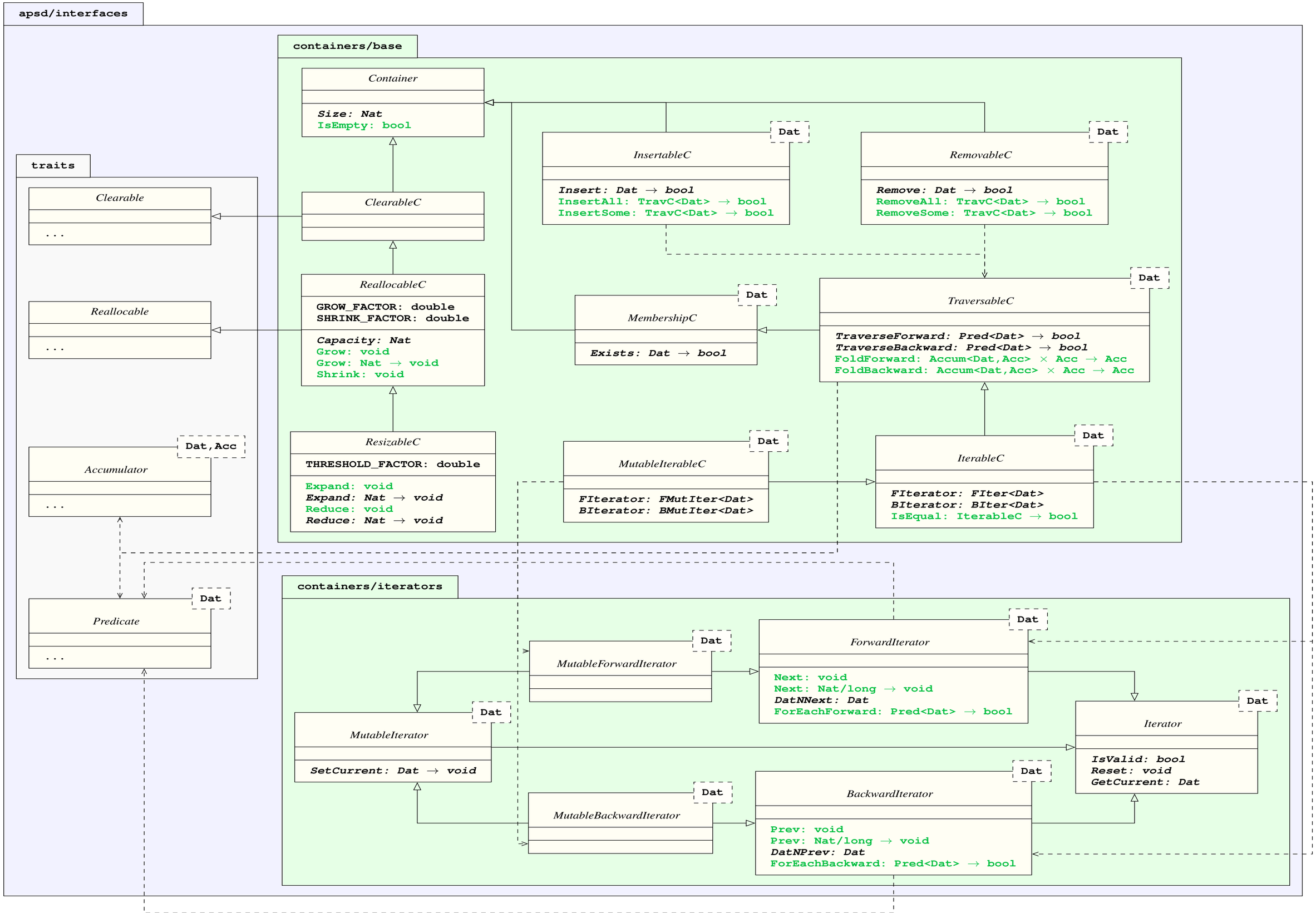
```
Exists: Dat → bool
Insert: Dat → bool
Remove: Dat → bool
Union: Set → void
Difference: Set → void
Intersection: Set → void
IsEqual: Set → bool
```

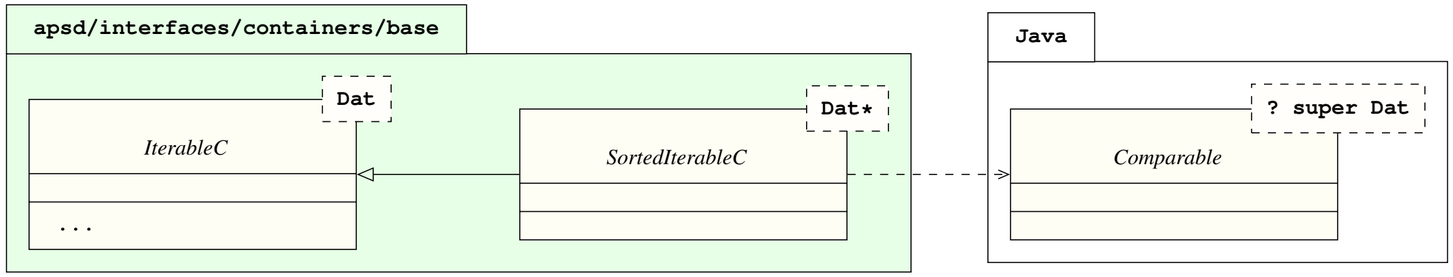
Dat*

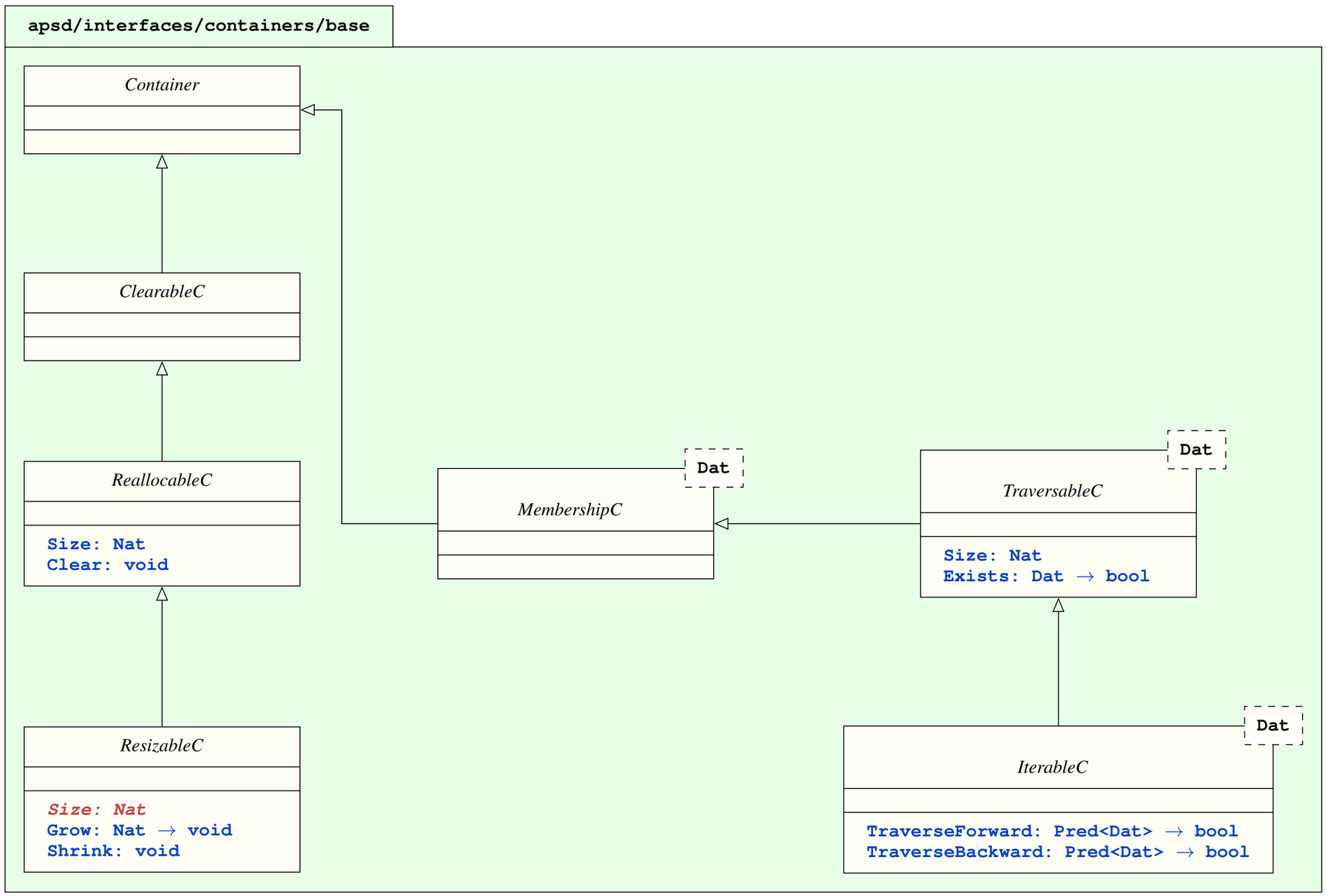
OrderedSet

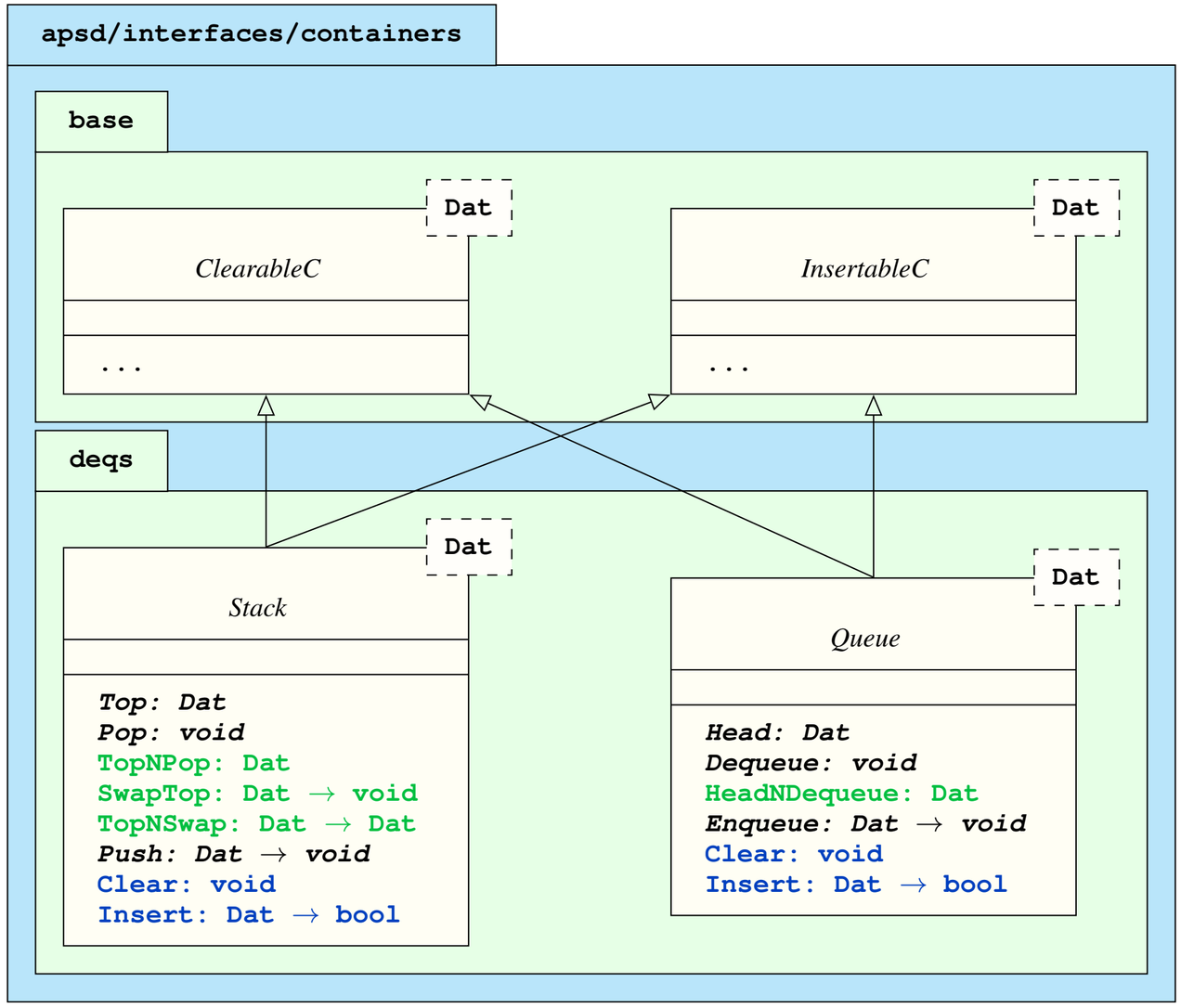
```
Min/Max: Dat
RemoveMin/Max: void
Predecessor/Successor: Dat → Dat
RemovePredecessor/Successor: Dat → void
```

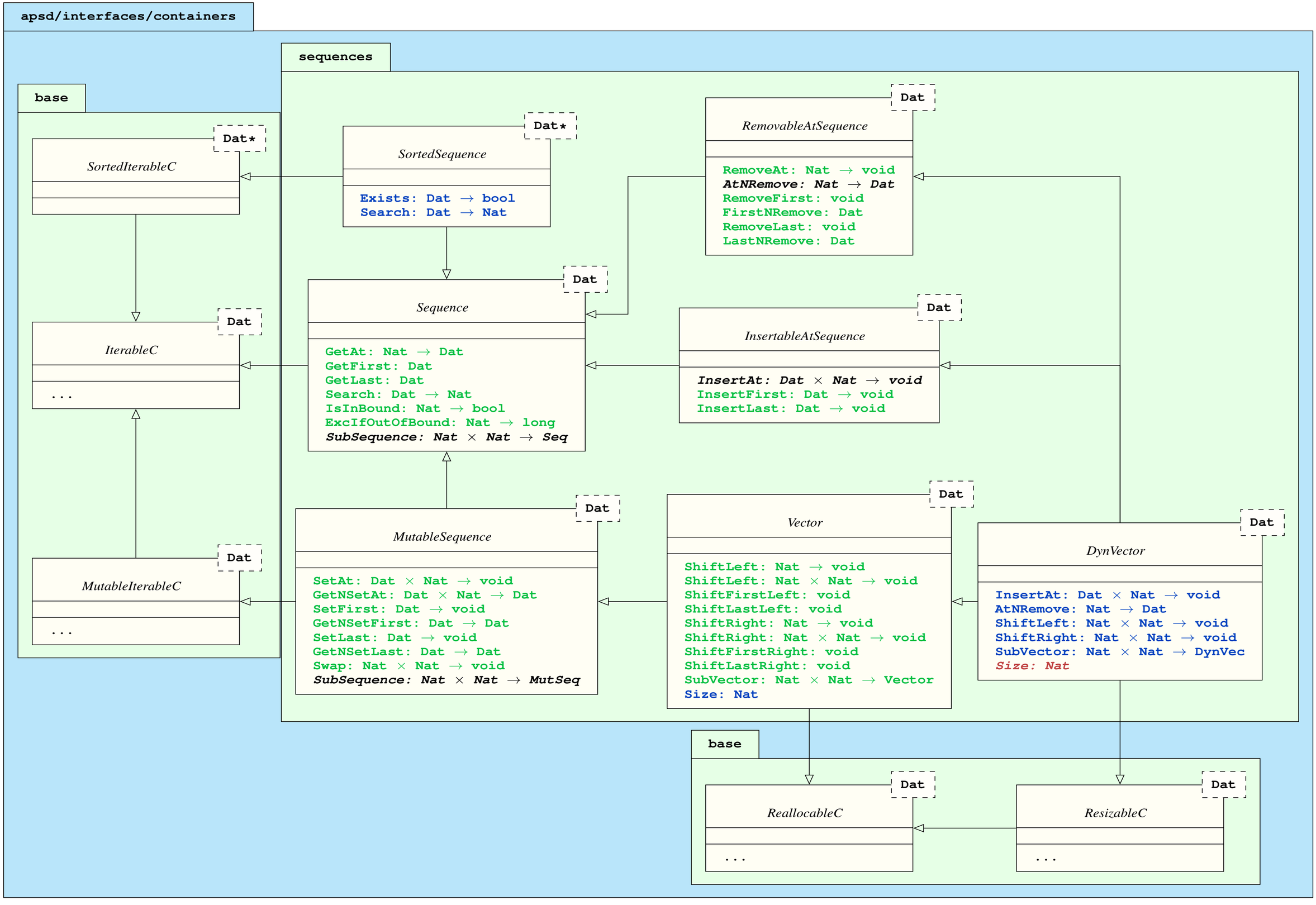

apsd
interfaces
containers
base
iterators
sequences
deqs
collections
traits
Clearable
Clear: void
Reallocable
Realloc: Nat → void
Dat
Predicate
Apply: Dat → bool
Dat,Acc
Accumulator
Apply: Dat×Acc → Acc
Dat
Reference
Get: Dat
IsNull: bool
Dat
MutableReference
Set: Dat → void
GetNSet: Dat → Dat
classes
containers
sequences
deqs
collections
utilities
Natural
...
MutableNatural
...
Dat
Box
Get: Dat
Set: Dat → void

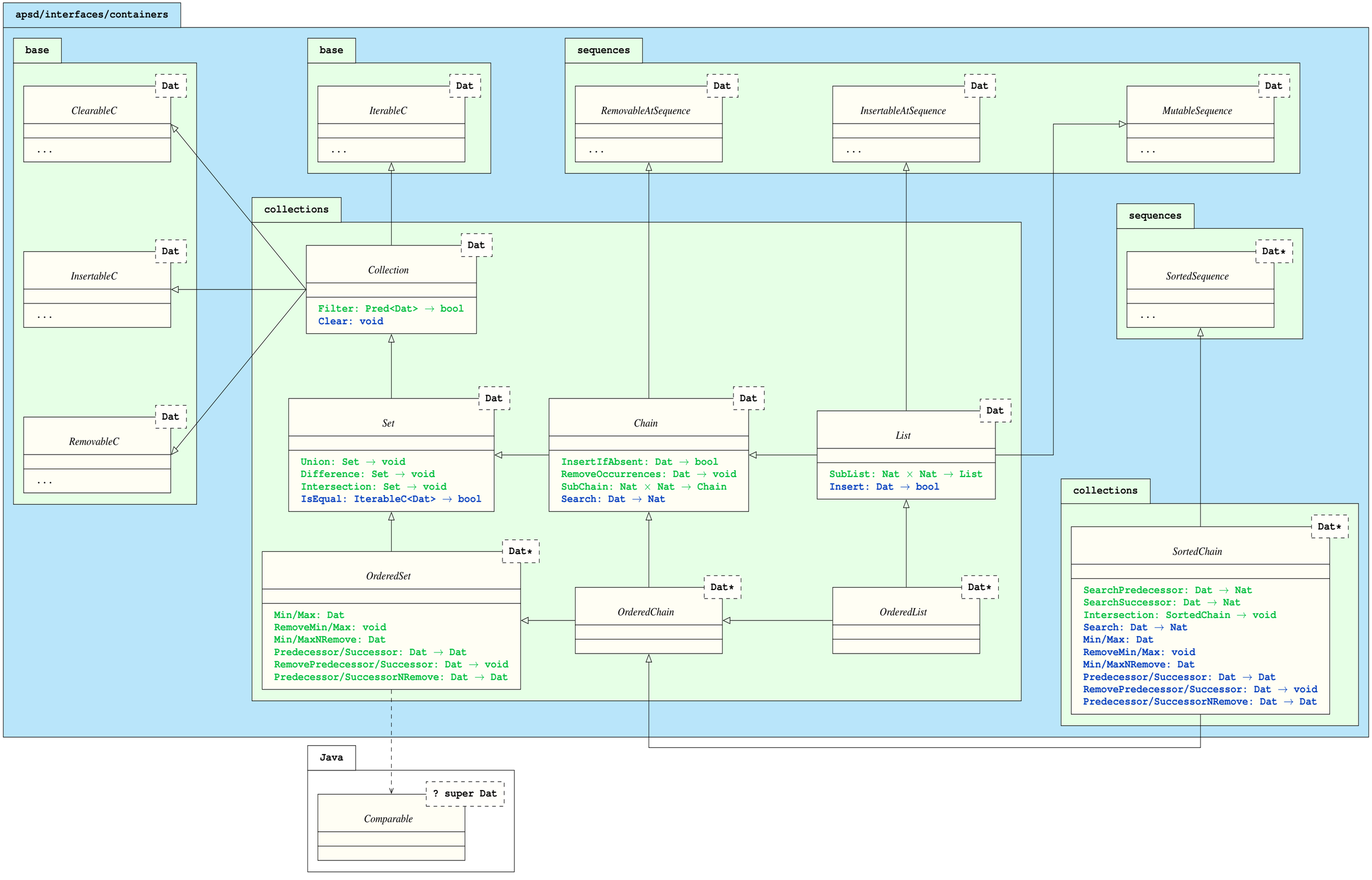

apsd/interfaces
containers/base
traits
containers/iterators
Container
Size: Nat
IsEmpty: bool
InsertableC
Dat
Insert: Dat → bool
InsertAll: TravC<Dat> → bool
InsertSome: TravC<Dat> → bool
RemovableC
Dat
Remove: Dat → bool
RemoveAll: TravC<Dat> → bool
RemoveSome: TravC<Dat> → bool
Clearable
...
ClearableC
ReallocableC
GROW_FACTOR: double
SHRINK_FACTOR: double
Capacity: Nat
Grow: void
Grow: Nat → void
Shrink: void
Reallocable
...
MembershipC
Dat
Exists: Dat → bool
TraversableC
Dat
TraverseForward: Pred<Dat> → bool
TraverseBackward: Pred<Dat> → bool
FoldForward: Accum<Dat,Acc> × Acc → Acc
FoldBackward: Accum<Dat,Acc> × Acc → Acc
Dat,Acc
Accumulator
...
ResizableC
THRESHOLD_FACTOR: double
Expand: void
Expand: Nat → void
Reduce: void
Reduce: Nat → void
MutableIterableC
Dat
FIterator: FMutIter<Dat>
BIterator: BMutIter<Dat>
IterableC
Dat
FIterator: FIter<Dat>
BIterator: BIter<Dat>
IsEqual: IterableC → bool
Predicate
Dat
...
MutableForwardIterator
Dat
ForwardIterator
Dat
Next: void
Next: Nat/long → void
DatNNext: Dat
ForEachForward: Pred<Dat> → bool
MutableIterator
Dat
SetCurrent: Dat → void
Iterator
Dat
IsValid: bool
Reset: void
GetCurrent: Dat
MutableBackwardIterator
Dat
BackwardIterator
Dat
Prev: void
Prev: Nat/long → void
DatNPrev: Dat
ForEachBackward: Pred<Dat> → bool

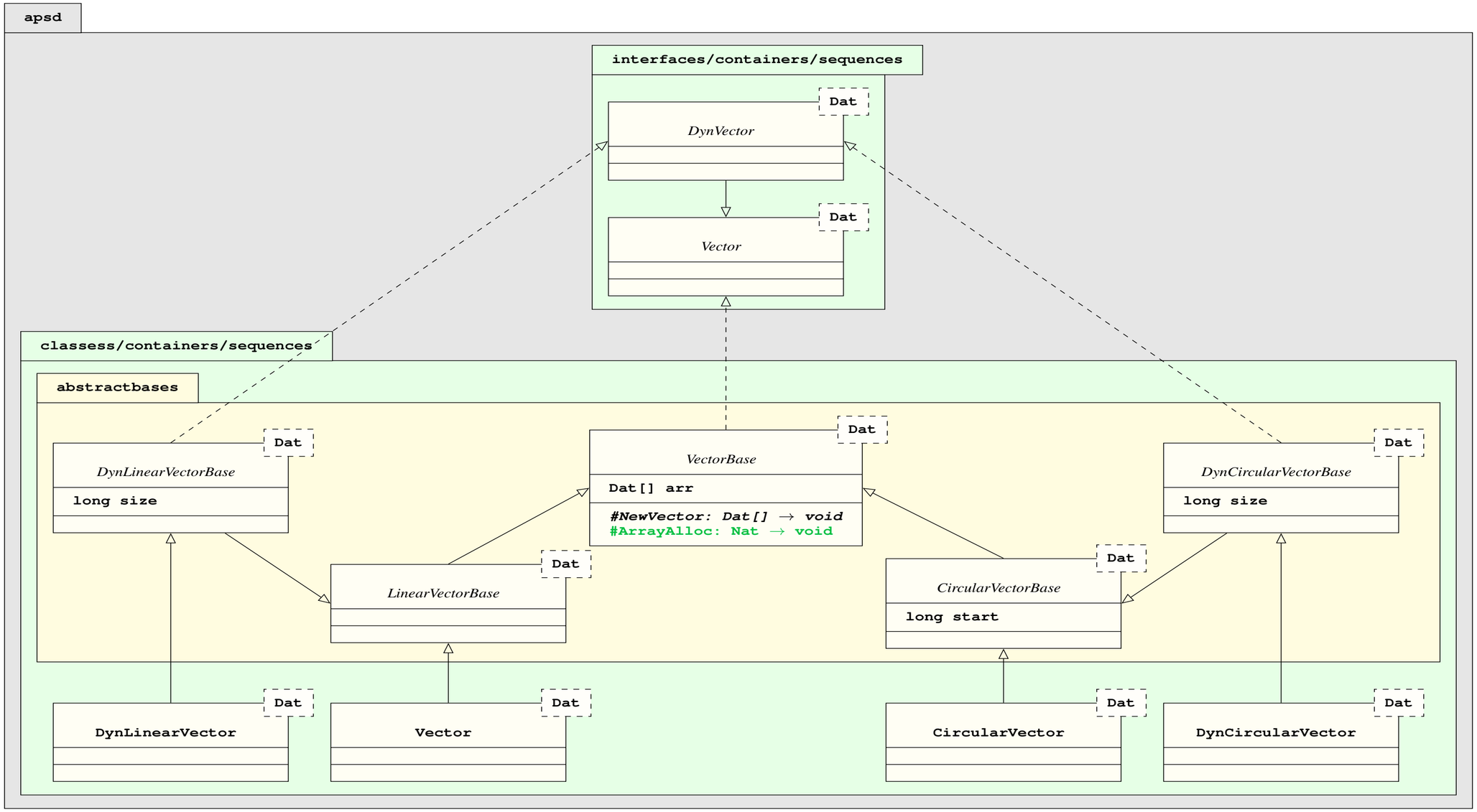

apsd/interfaces/containers/base
Dat
IterableC
...
Dat*
SortedIterableC
Java
? super Dat
Comparable


apsd/interfaces/containers/base
Container
ClearableC
ReallocableC
Size: Nat
Clear: void
ResizableC
Size: Nat
Grow: Nat → void
Shrink: void
Dat
MembershipC
Dat
TraversableC
Size: Nat
Exists: Dat → bool
Dat
IterableC
TraverseForward: Pred<Dat> → bool
TraverseBackward: Pred<Dat> → bool

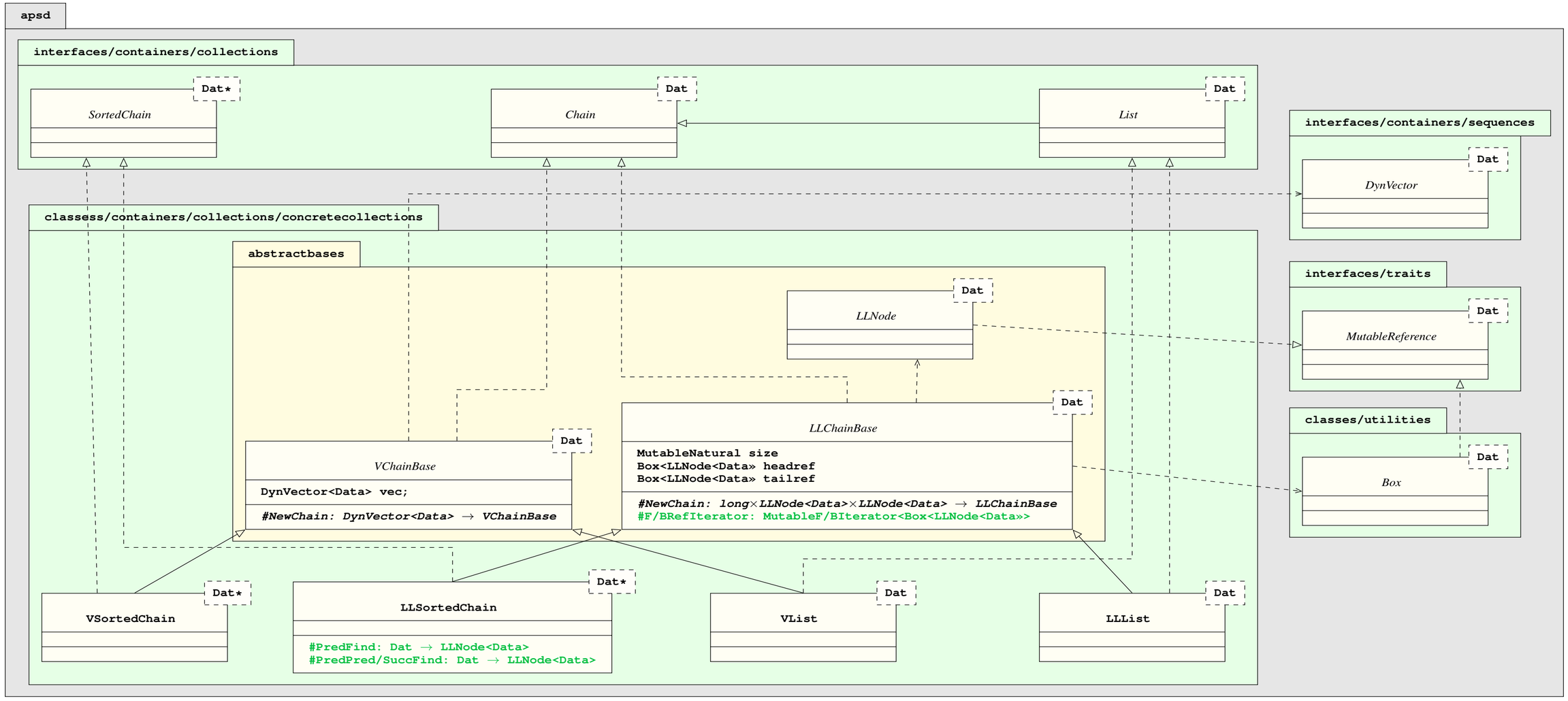

apsd/interfaces/containers
base
Dat
ClearableC
...
Dat
InsertableC
...
deqs
Dat
Stack
Top: Dat
Pop: void
TopNPop: Dat
SwapTop: Dat → void
TopNSwap: Dat → Dat
Push: Dat → void
Clear: void
Insert: Dat → bool
Dat
Queue
Head: Dat
Dequeue: void
HeadNDequeue: Dat
Enqueue: Dat → void
Clear: void
Insert: Dat → bool


apsd/interfaces/containers
sequences
base
Dat*
SortedIterableC
Dat*
SortedSequence
Exists: Dat → bool
Search: Dat → Nat
Dat
RemovableAtSequence
RemoveAt: Nat → void
AtNRemove: Nat → Dat
RemoveFirst: void
FirstNRemove: Dat
RemoveLast: void
LastNRemove: Dat
Dat
IterableC
...
Dat
Sequence
GetAt: Nat → Dat
GetFirst: Dat
GetLast: Dat
Search: Dat → Nat
IsInBound: Nat → bool
ExcIfOutOfBound: Nat → long
SubSequence: Nat × Nat → Seq
Dat
InsertableAtSequence
InsertAt: Dat × Nat → void
InsertFirst: Dat → void
InsertLast: Dat → void
Dat
MutableSequence
SetAt: Dat × Nat → void
GetNSetAt: Dat × Nat → Dat
SetFirst: Dat → void
GetNSetFirst: Dat → Dat
SetLast: Dat → void
GetNSetLast: Dat → Dat
Swap: Nat × Nat → void
SubSequence: Nat × Nat → MutSeq
Dat
Vector
ShiftLeft: Nat → void
ShiftLeft: Nat × Nat → void
ShiftFirstLeft: void
ShiftLastLeft: void
ShiftRight: Nat → void
ShiftRight: Nat × Nat → void
ShiftFirstRight: void
ShiftLastRight: void
SubVector: Nat × Nat → Vector
Size: Nat
Dat
DynVector
InsertAt: Dat × Nat → void
AtNRemove: Nat → Dat
ShiftLeft: Nat × Nat → void
ShiftRight: Nat × Nat → void
SubVector: Nat × Nat → DynVec
Size: Nat
Dat
MutableIterableC
...
base
Dat
ReallocableC
...
Dat
ResizableC
...


apsd/interfaces/containers
base
Dat
ClearableC
...
InsertableC
...
RemovableC
...
base
Dat
IterableC
...
sequences
Dat
RemovableAtSequence
...
Dat
InsertableAtSequence
...
Dat
MutableSequence
...
collections
Dat
Collection
Filter: Pred<Dat> → bool
Clear: void
Dat
Set
Union: Set → void
Difference: Set → void
Intersection: Set → void
IsEqual: IterableC<Dat> → bool
Dat
Chain
InsertIfAbsent: Dat → bool
RemoveOccurrences: Dat → void
SubChain: Nat × Nat → Chain
Search: Dat → Nat
Dat
List
SubList: Nat × Nat → List
Insert: Dat → bool
Dat*
OrderedSet
Min/Max: Dat
RemoveMin/Max: void
Min/MaxNRemove: Dat
Predecessor/Successor: Dat → Dat
RemovePredecessor/Successor: Dat → void
Predecessor/SuccessorNRemove: Dat → Dat
Dat*
OrderedChain
Dat*
OrderedList
sequences
Dat*
SortedSequence
...
collections
Dat*
SortedChain
SearchPredecessor: Dat → Nat
SearchSuccessor: Dat → Nat
Intersection: SortedChain → void
Search: Dat → Nat
Min/Max: Dat
RemoveMin/Max: void
Min/MaxNRemove: Dat
Predecessor/Successor: Dat → Dat
RemovePredecessor/Successor: Dat → void
Predecessor/SuccessorNRemove: Dat → Dat
Java
? super Dat
Comparable

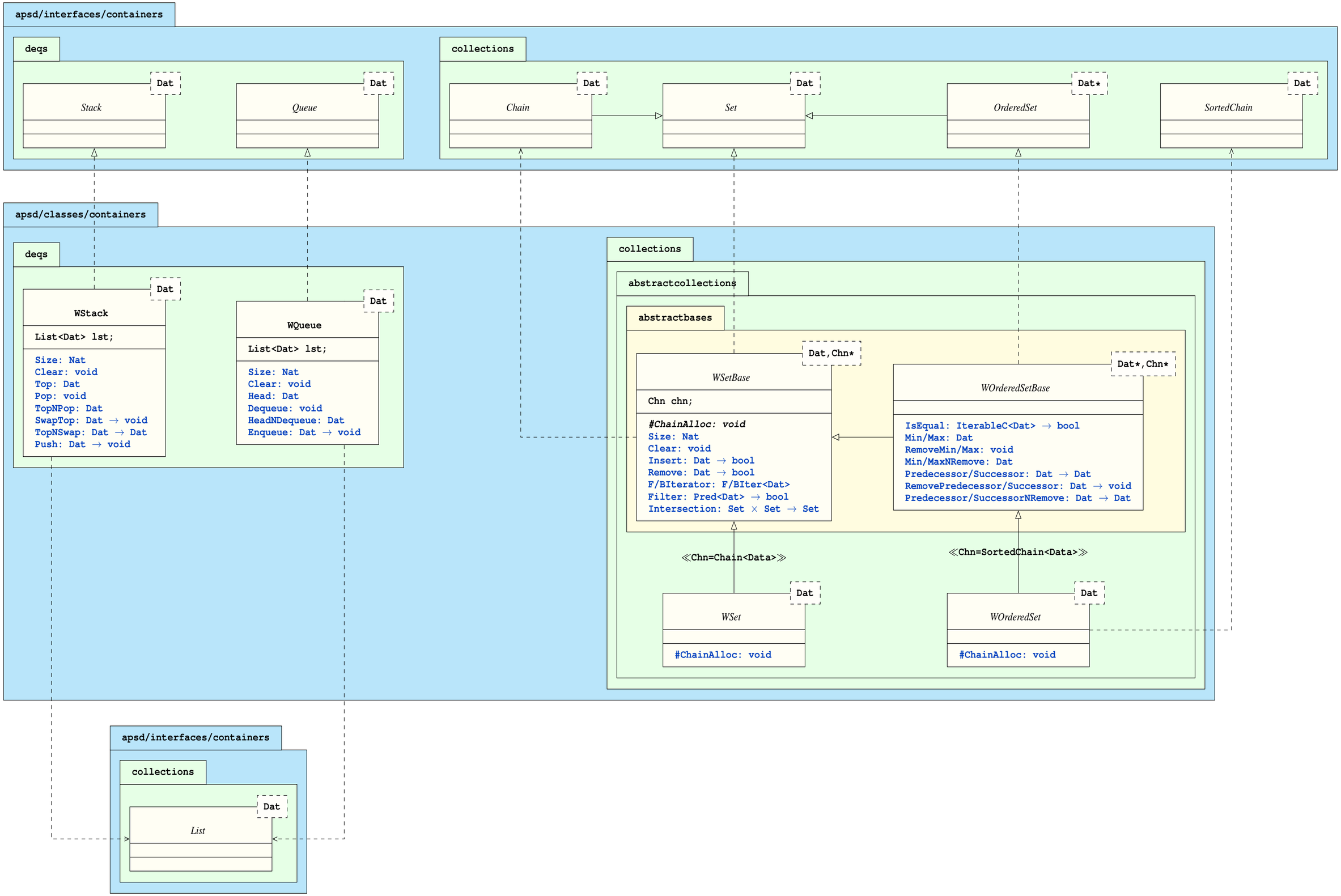


Methods	VecB	LinVecB	CirVecB	DLinVecB	DCirVecB
Size : Nat				✓	✓
Clear : void	✓			✓	✓
Realloc : Nat → void		✓	✓	✓	✓
Capacity : Nat	✓				
Expand : Nat → void				✓	✓
Reduce : Nat → void				✓	✓
F/BIterator : MutableForward/BackwardIterator	✓				
GetAt : Nat → Dat	×	✓	✓		
SetAt : Dat × Nat → void	×	✓	✓		
ShiftLeft/Right : Nat × Nat → void			✓		✓
SubSequence : Nat × Nat → MutableSequence	✓				
ArrayAlloc : Nat → void	✓		✓	✓	✓



Methods	VChainB	LLChainB
Size : Nat	✓	✓
Clear : void	✓	✓
Remove : Dat → bool	✓	✓
F/BIterator : Forward/BackwardIterator	✓	✓
GetAt : Nat → Dat	✓	
GetFirst/Last : Dat		✓
SubSequence : Nat × Nat → Sequence	✓	✓
AtNRemove : Nat → Dat	✓	✓
RemoveFirst/Last : void		✓
First/LastNRemove : Dat		✓
Filter : Pred<Dat> → bool	✓	✓

Methods	VSChain	LLSChain	VList	LLList
F/BIterator : MutableForward/BackwardIterator			✓	✓
Insert : Dat → bool	✓	✓		
Remove : Dat → bool		✓		
Search : Dat → Nat		✓		
SearchPredecessor/Successor : Dat → Nat		✓		
SetAt : Dat × Nat → void			✓	✓
SetFirst/Last : Dat → void				✓
SubSequence : Nat × Nat → MutableSequence			✓	✓
InsertAt : Dat × Nat → void			✓	✓
InsertFirst/Last : Dat → void				✓
The six methods for Predecessor and Successor		✓		
InsertIfAbsent : Dat → bool	✓	✓		
RemoveOccurrences : Dat → void	✓	✓		


apsd/interfaces/containers
deqs
Stack
Dat
Queue
Dat
collections
Chain
Dat
Set
Dat
OrderedSet
Dat*
SortedChain
Dat
apsd/classes/containers
deqs
WStack
Dat
List<Dat> lst;
Size: Nat
Clear: void
Top: Dat
Pop: void
TopNPop: Dat
SwapTop: Dat → void
TopNSwap: Dat → Dat
Push: Dat → void
WQueue
Dat
List<Dat> lst;
Size: Nat
Clear: void
Head: Dat
Dequeue: void
HeadNDequeue: Dat
Enqueue: Dat → void
collections
abstractcollections
abstractbases
WSetBase
Dat,Chn*
Chn chn;
#ChainAlloc: void
Size: Nat
Clear: void
Insert: Dat → bool
Remove: Dat → bool
F/BIterator: F/BIter<Dat>
Filter: Pred<Dat> → bool
Intersection: Set × Set → Set
WOrderedSetBase
Dat*,Chn*
IsEqual: IterableC<Dat> → bool
Min/Max: Dat
RemoveMin/Max: void
Min/MaxNRemove: Dat
Predecessor/Successor: Dat → Dat
RemovePredecessor/Successor: Dat → void
Predecessor/SuccessorNRemove: Dat → Dat
≪Chn=Chain<Data>≫
WSet
Dat
#ChainAlloc: void
≪Chn=SortedChain<Data>≫
WOrderedSet
Dat
#ChainAlloc: void
apsd/interfaces/containers
collections
List
Dat