References

Ralph Johnson's Lecture notes, Lecture 8 Object Identity http://st-www.cs.uiuc.edu/users/cs497/lectures.html

ValueWithHistory

Represents a value that changes over time

Answer its value at any point in time

Update its value at any point in time

Add to value from any point in time into the future

Instead of storing a value in a variable

Store it in a ValueWithHistory

balance := ValueWithHistory zero.

balance at: Date today put: 25.0s2.

Main Access methods

at: aDate
Return value at the given date

at: aDate add: anAmount
Add an amount to the value at aDate and all times in the future

Sample Usage

```
testAdding
   | value today yesterday tomorrow |
   today :=Date today.
   yesterday :=today subtractDays: 1.
   tomorrow := today addDays: 1.
   value := ValueWithHistory zero.
   value
      at: today add: 1;
      at: yesterday add: 2;
      at: tomorrow add: 5.
   self assert: (value at:  today ) = 3.
```

```
Smalltalk defineClass: #ValueWithHistory
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'dates valueHistory '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'

ValueWithHistory class methodsFor: 'instance creation'

zero
   ^self new initialize


ValueWithHistory methodsFor: 'initialize'

initialize
   dates := SortedSequence new.
   valueHistory := OrderedCollection new.
```

```
ValueWithHistory methodsFor: 'accessing'

at: aDate
   | index |
   index := dates startIndexOfIntervalContaining: aDate.
   index isZero ifTrue: [self error: 'date is too early'].
   ^valueHistory at: index

at: aDate add: anAmount
   | start |
   start := self indexForAccessing: aDate.
   start
      to: valueHistory size
      do: [:each | valueHistory at: each put: (valueHistory at: each) +
anAmount]

indexForAccessing: aDate
   | index |
   index := dates startIndexOfIntervalContaining: aDate.
   index isZero
      ifTrue:
         [dates add: aDate.
         valueHistory addFirst: 0.
         ^1].
   (dates at: index) = aDate ifTrue: [^index].
   dates add: aDate.
   valueHistory add: (valueHistory at: index) beforeIndex: index + 1.
   ^index + 1
```

SortedSequence

Subclass of SortedCollection with one method

```
SortedSequence>>startIndexOfIntervalContaining: anElement
   self isEmpty ifTrue: [^0].
   ^(self indexForInserting: anElement) – firstIndex
```