

## Appendix1: VisualWorks 2.5 versus Smalltalk-80

	VisualWorks 2.5	Smalltalk-80
Assignment	:=	←
Global Variables	Start with Caps	Does not care
DeepCopy	Removed	Present
Fractions	asRational	asFraction
FileName protocol	fileNamed:	named:

**Differences found throughout the lecture note's examples**

- **Classes removed from VisualWorks 2.5**

- Button
- DebuggerController
- DebuggerTextView
- DialogCompositController
- DialogController
- DialogView
- FixedThumbScrollbar
- FractionalWidgetView
- HandlerController
- ListController
- ListView
- SelectionSetInListController
- SelectionInListView
- TextItemView
- TextItemEditor
- TextController
- TextView
- PopUpMenu
- WidgetSpecification

## Appendix2: VisualWorks rules and Smalltalk Syntax

- **Capitalization rules**
  - Upper Case
    - Class names
    - Class variables and global variables
  - Lower Case
    - Method names
    - Temp variables, instance variables, class instance variables, method arguments
  - Use embedded capital letters, not underscores
- **Reserved words**
  - nil
  - true
  - false
  - self
  - super
  - thisContext
- **Operators**
  - :=
    - called 'gets' operator, used for assignment
  - ^
    - called 'returns' operator, used to return a value
- **Example**

```
name: aSymbol
name := aSymbol.
^name.
```
- **Literals**
  - use VisualWorks syntax chapter for reference here
  - Numbers
  - Characters
  - Strings
  - Symbols
  - Arrays of literals
  - Byte Arrays (notice the use of brackets)
- **Comments**
  - "Comment"
  - periods allowed within double quotes

## Appendix 3: A List of Methods for the System Classes

### Magnitude:

Creation:

Operations:

- < aMagnitude  
less than operator returns boolean
- <= aMagnitude  
less than or equal operator returns boolean
- > aMagnitude  
greater than operator returns boolean
- >= aMagnitude  
greater than or equal operator returns boolean
- between: min and: max  
returns True if object's magnitude is between min and max
- min: aMagnitude  
returns the lesser of the object and aMagnitude
- max: aMagnitude  
returns the greater of the object and aMagnitude

Magnitude->Date:

Creation:

- today  
instance representing the current date
- fromDays: dayCount  
instance representing the date dayCount days from 01/01/1901
- newDay: day month: monthName year: yearInteger  
instance representing day number of days into monthName in yearInteger
- newDay: dayCount year: yearInteger  
instance representing dayCount days into yearInteger

Operations:

- dayOfWeek: dayName  
returns index of dayName in the week, #Sunday = 0
- nameOfDay: dayIndex  
returns Symbol representing the day whose index is dayIndex
- indexOfMonth: monthName  
returns index of monthName in the year, #January = 0
- nameOfMonth: monthIndex  
returns Symbol representing the month whose index is monthIndex
- daysInMonth: monthName forYear: yearInteger  
returns Integer representing the number of days in monthName for year  
yearInteger
- daysInYear: yearInteger  
returns Integer representing the number of days in yearInteger
- leapYear: yearInteger  
returns 1 if yearInteger is a leap year, 0 otherwise
- dateAndTimeNow  
returns Array whose first element is current date, and whose second element is  
the current time
- addDays: dayCount  
returns Date that is dayCount days after object
- subtractDays: dayCount  
returns Date that is dayCount days before object
- subtractDate: aDate
- asSeconds

returns number of seconds between a time on 01/01/1901 and the same time in the receiver's day

#### Magnitude->Time:

##### Creation:

now

instance representing the current time

fromSeconds: secondCount

instance representing the time of secondCount after midnight

##### Operations:

millisecondClockValue

returns number of milliseconds since the millisecond clock was reset or rolled over

millisecondsToRun: timedBlock

returns number of milliseconds timedBlock takes to execute

timeWords

returns the number of seconds since 01/01/1901 (GMT) in 4 element byte array

totalSeconds

returns total number of seconds since 01/01/1901, correcting the time zone and daylight savings

dateAndTimeNow

returns Array whose first element is current date, and whose second element is the current time

addTime: timeAmount

returns Time that is timeAmount days after receiver

subtractTime: timeAmount

returns Date that is timeAmount before receiver

asSeconds

returns number of seconds since midnight that receiver represents

#### Magnitude->Character:

##### Creation:

value: anInteger

instance of Character which is the ASCII representation of anInteger

digitValue: anInteger

instance of Character which is the character representation of a number of radix 35- \$0 returns 0, \$A returns 10, \$Z returns 35

##### Operations:

asciiValue

returns Integer of ascii character

digitValue

returns Integer representing numerical radix

isAlphaNumeric

true if receiver is letter or digit

isDigit

true if receiver is digit

isLetter

true if receiver is letter

isLowercase

true if receiver is lowercase

isUppercase

true if receiver is uppercase

isSeparator

true if receiver is space, tab, cr, line feed, or form feed

isVowel

true if receiver is a,e,i,o,u

Magnitude->Number:

Creation:

Operations:

- + aNumber
  - returns sum of receiver and aNumber
- aNumber
  - returns difference of receiver and aNumber
- \* aNumber
  - returns result of multiplying receiver by aNumber
- / aNumber
  - returns result of dividing receiver by aNumber. If result is not a whole number, then an instance of Fraction is returned
- // aNumber
  - returns Integer result of division truncated toward negative infinity
- \ aNumber
  - returns Integer representing receiver modulus aNumber
- abs
  - returns Number representing absolute value of receiver
- negated
  - returns Number representing additive reciprocal
- quo: aNumber
  - returns quotient of receiver divided by aNumber
- rem: aNumber
  - returns remainder of receiver divided by aNumber
- reciprocal
  - returns multiplicative reciprocal (1/receiver)
- exp
  - returns e raised to the power of receiver
- ln
  - returns natural log of receiver
- log: aNumber
  - returns log base aNumber of receiver
- floorLog: radix
  - returns floor of log base radix of receiver
- raisedTo: aNumber
  - returns result of raising receiver to aNumber
- raisedToInteger: anInteger
  - returns result of raising receiver to anInteger, where anInteger must be an Integer
- sqrt
  - returns square root of receiver
- squared
  - returns receiver raised to the second power
- even
  - true if receiver is even
- odd
  - true if receiver is odd
- negative
  - true if receiver is  $\leq 0$
- positive
  - true if receiver is  $\geq 0$
- strictlyPositive
  - true if receiver  $> 0$
- sign
  - returns 1 if receiver  $> 0$ , 0 if receiver  $= 0$ . -1 if receiver  $< 0$

ceiling  
     returns result of rounding towards positive infinity  
 floor  
     returns result of rounding towards negative infinity  
 truncated  
     returns result of rounding towards zero  
 truncateTo: aNumber  
     returns result of truncating to multiple of aNumber  
 rounded  
     returns result of rounding receiver  
 roundedTo: aNumber  
     returns result of rounding receiver to nearest multiple of aNumber  
 degreesToRadians  
     returns Float of radian representation of receiver. Assumes receiver is in degrees  
 radiansToDegrees  
     returns Float in degrees of conversion of receiver. Assumes receiver is in radians  
 sin  
     returns Float of sin(receiver) in radians  
 cos  
     returns Float of cos(receiver) in radians  
 tan  
     returns Float of tan(receiver) in radians  
 arcSin  
     returns Float of arcSin(receiver) in radians  
 arcCos  
     returns Float of arcCos(receiver) in radians  
 arcTan  
     returns Float of arcTan(receiver) in radians  
 coerce: aNumber  
     casts receiver as same type as aNumber  
 generality  
     returns the number representing the ordering of the receiver in the generality  
     hierarchy  
 retry: aSymbol coercing: aNumber  
     an arithmetic operation aSymbol could not be performed, so the operation is  
     retried casting the receiver or argument to aNumber (picking the lowest order of  
     generality)

Magnitude->Number->Integer:

Creation:

Operations:

factorial  
     returns Integer representing the factorial of the receiver  
 gcd: anInteger  
     returns Integer representing the Greatest Common Denominator of the receiver  
     and anInteger  
 lcm: anInteger  
     returns Integer representing the Lowest Common Multiple of the receiver and  
     anInteger  
 allMask: anInteger  
     treat anInteger as a bit mask. Returns True if all 1's in anInteger are 1 in the  
     receiver  
 anyMask: anInteger  
     treat anInteger as a bit mask. Returns True if any on the 1's in anInteger are 1 in  
     the receiver  
 noMask: anInteger

treat an Integer as a bit mask. Returns True if none of the 1's in anInteger are 1 in the receiver  
 bitAnd: anInteger  
 returns Integer representing a boolean AND operation between anInteger and the receiver  
 bitOr: anInteger  
 returns Integer representing a boolean OR operation between anInteger and the receiver  
 bitXor: anInteger  
 returns Integer representing a boolean XOR (eXclusive OR) operation between anInteger and the receiver  
 bitAt: anIndex  
 returns the bit (0 or 1) at anIndex  
 bitInvert  
 returns an Integer which is the complement of the receiver  
 highBit  
 returns an Integer representing the index of the highest order bit  
 bitShift: anInteger  
 returns an Integer whose value (in two's-complement) is the receiver's value shifted anInteger number of bits. Negative shifts are to the right.

## Random

### Creation:

:= Random new  
 instance representation of a random number generator  
 next  
 instance of a random number. The receiver must be a random number generator, which has previously been started

### Operations:

## Collection

### Creation:

#(Object1, Object2, Object3, Object4)  
 instance representing an array containing up to 4 objects passed as arguments  
 new  
 instance representing an empty collection  
 new:  
 instance representing a collection  
 with: anObject  
 instance representing a collection containing anObject  
 with: firstObject with: secondObject  
 instance representing a collection containing firstObject and secondObject

### Operations:

add: newObject  
 adds newObject to the receiver and returns newObject  
 addAll: aCollection  
 adds aCollection to the receiver and returns aCollection  
 remove: oldObject  
 removes oldObject from the receiver and returns oldObject unless there is no object oldObject (reports an error).  
 remove: oldObject ifAbsent: anExceptionBlock  
 removes oldObject from the receiver, unless it does not exist, in which case anExceptionBlock is executed. Returns oldObject or result of anExceptionBlock  
 removeAll: aCollection

- removes all elements of aCollection from the receiver and returns aCollection, unless not all elements of aCollection were present in the receiver, in which case an error is reported.
- includes: anObject
  - returns True if anObject is an element of the receiver
- isEmpty
  - returns True if the receiver has no elements
- occurrencesOf: anObject
  - returns an Integer representing the number of occurrences of anObject in the receiver
- do: aBlock
  - evaluate aBlock for every element of the receiver
- select: aBlock
  - evaluates aBlock for every element of the receiver. Returns a new Collection containing all elements of the receiver for which aBlock evaluated to true
- reject: aBlock
  - evaluates aBlock for every element of the receiver. Returns a new Collection containing all elements for which aBlock evaluated to false
- collect: aBlock
  - evaluates aBlock for every element of the receiver. Returns a new Collection containing the results of every evaluation of aBlock.
- detect: aBlock
  - evaluates aBlock for every element of the receiver. Returns the object which is the first element in the receiver for which aBlock evaluated to true. If no object evaluated to true, an error is reported.
- detect: aBlock ifNone: exceptionBlock
  - evaluates aBlock for every element of the receiver. Returns the object which is the first element in the receiver for which aBlock evaluated to true. If no object evaluated to true, exceptionBlock is evaluated.
- inject: thisValue into: binaryBlock
  - Evaluates binaryBlock for each element of the receiver, initializing a local variable to thisValue. Returns final value of the block. BinaryBlock has two arguments.
- asBag
  - Returns a Bag with the elements from the receiver
- asSet
  - Returns a Set with the elements from the receiver
- asOrderedCollection
  - Returns an OrderedCollection with the elements from the receiver
- asSortedCollection
  - Returns a SortedCollection with the elements from the receiver, sorted to each element is less than or equal to its successor
- asSortedCollection: aBlock
  - Returns a SortedCollection with the elements from the receiver, sorted according to the argument aBlock

#### Collection->Bag

Creation:

Operations:

- add: newObject withOccurrences: anInteger

- Adds anInteger number of occurrences of newObject to the receiver, and returns newObject

#### Collection->Set

Creation:

Operations:



## Collection->Set->Dictionary and Collection->Set->IdentityDictionary

Creation:

Operations:

at: key ifAbsent: aBlock

Returns the value named by key. If the key is not present in the dictionary, returns evaluation of aBlock

associationAt: key

Returns the association named by key. If key is not present, an error is reported

associationAt: key ifAbsent: aBlock

Returns the association named by key. If key is not present, returns the evaluation of aBlock.

keyAtValue: value

Returns the name found first for value, or nil if value is not present

keyAtValue: value ifAbsent: exceptionBlock

Returns the name found first for value, or the evaluation of exceptionBlock if value is not found

keys

Returns Set representing all of the receiver's keys

values

Returns Set containing all of the receiver's values

includesAssociation: anAssociation

Returns true if anAssociation is included in the receiver

includesKey: key

Returns true if key is included in the receiver

removeAssociation: anAssociation

Removes anAssociation from the receiver. Returns anAssociation

removeKey: key

Removes key and associated value from the receiver. Returns value associated with key if key is included in the receiver, otherwise an error is reported

removeKey: key ifAbsent: aBlock

Removes key and associated value from the receiver. Returns value associated with the key if key is included in the receiver, otherwise returns the evaluation of aBlock

associationsDo: aBlock

Evaluate aBlock for each of the receiver's associations

keysDo: aBlock

Evaluate aBlock for each of the receiver's keys

## Collection->SequenceableCollection

Creation:

Operations:

atAll: aCollection put: anObject

Associate each element of aCollection with anObject.

atAllPut: anObject

Put anObject as every one of the receiver's elements

first

Returns the first element of the receiver

last

Returns the last element of the receiver

indexOf: anElement

Returns an Integer representing the index of anElement in the receiver, 0 if not present

indexOf: anElement ifAbsent: exceptionBlock

Returns an Integer representing the index of anElement in the receiver, or the evaluation of exceptionBlock if anElement is not in the receiver

indexOfSubCollection: aSubCollection startingAt: anIndex  
 If the elements of aSubCollection appear in order in the receiver, returns the index of the first element of aSubCollection in the receiver, otherwise returns 0

indexOfSubCollection: aSubCollection: startingAt: anIndex ifAbsent: exceptionBlock  
 Returns the index of the first element of aSubCollection in the receiver if the elements of aSubCollection appear in order, otherwise returns the evaluation of aBlock

replaceFrom: start to: stop with: replacementCollection  
 Associates every element of the receiver from start to stop with the elements of replacementCollection and returns the receiver. The size of replacementCollection must equal stop - start + 1.

replaceFrom: start to: stop with: replacementCollection startingAt: repStart  
 Associates every element of the receiver from start to stop with the elements of replacementCollection starting with index repStart in replacementCollection. The receiver is returned

, aSequencableCollection  
 Returns the receiver concatenated with aSequencableCollection

copyFrom: start to: stop  
 Returns a subset of the receiver starting at index start and ending an index stop

copyReplaceAll: oldSubCollection with: newSubCollection  
 Returns a copy of the receiver with all occurrences of oldSubCollection replaced with newSubCollection

copyWith: newElement  
 Returns a copy of the receiver with newElement added on to the end

copyWithout: oldElement  
 Returns a copy of the receiver without all occurrences of oldElement

findFirst: aBlock  
 Evaluates aBlock for every element of the receiver and returns the index of the first element for which aBlock evaluates to true.

findLast: aBlock  
 Evaluates aBlock for each element of the receiver and returns the index of the last element for which aBlock evaluates to true

reverseDo: aBlock  
 Evaluates aBlock for each element of the receiver, starting with the last element

with: aSequenceableCollection do: aBlock  
 Evaluates aBlock for each element of the receiver and each element of aSequenceableCollection. The number of elements in aSequenceableCollection must equal the number of elements in the receiver and aBlock must have two arguments

Collection->SequenceableCollection->OrderedCollection

Creation:

Operations:

after: oldObject  
 Returns the element occurring after oldObject, or reports an error if oldObject is not found or is the last element

before: oldObject  
 Returns the element occurring before oldObject, or reports an error if oldObject is not found or is the first element

add: newObject after: oldObject  
 Inserts newObject after oldObject into the receiver and returns newObject unless oldObject is not found, in which case an error is reported

add: newObject before: oldObject  
 Inserts newObject before oldObject into the receiver and returns newObject unless oldObject is not found, in which case an error is reported

addAllFirst: anOrderedCollection

Adds each element of anOrderedCollection to the beginning of the receiver and returns anOrderedCollection  
 addAllLast: anOrderedCollection  
 Adds each element of anOrderedCollection to the end of the receiver and returns anOrderedCollection  
 addFirst: newObject  
 Adds newObject to the beginning of the receiver and returns newObject  
 addLast: newObject  
 Adds newObject to the end of the receiver and returns newObject  
 removeFirst  
 Removes the first object from the receiver and returns it, unless the receiver is empty in which case an error is reported  
 removeLast  
 Removes the last object from the receiver and returns it, unless the receiver is empty in which case an error is reported

Collection->SequenceableCollection->OrderedCollection->SortedCollection

Creation:

sortBlock: aBlock  
 Instance representing an empty SortedCollection using aBlock to sort its elements

Operations:

sortBlock  
 Returns the block that is to be used to sort the elements of the receiver  
 sortBlock: aBlock  
 Make aBlock the block used to sort the elements of the receiver

Collection->SequenceableCollection->LinkedList

Creation:

nextLink: aLink  
 Instance of Link that references aLink

Operations:

nextLink  
 Returns the receiver's reference  
 nextLink: aLink  
 Sets the receiver's reference to be aLink  
 addFirst: aLink  
 Adds aLink to the beginning of the receiver's list and returns aLink  
 addLast: aLink  
 Adds aLink to the end of the receiver's list and returns aLink  
 removeFirst  
 Removes the first element from the receiver's list and returns it. If the list is empty an error is reported  
 removeLast  
 Removes the last element from the receiver's list and returns it. If the list is empty an error is reported

Collection->SequenceableCollection->Interval

Creation:

from: startInteger to: stopInteger  
 Instance starting with the number startInteger and ending with stopInteger, incrementing by one  
 from: startInteger to: stopInteger by: stepInteger  
 Instance starting with the number startInteger and ending with stopInteger, incrementing by stepInteger

Operations:

Collection->SequenceableCollection->ArrayedCollection

Creation:

Operations:

Collection->SequenceableCollection->ArrayedCollection->CharacterArray->String

Creation:

Operations:

< aString

Returns true if the receiver collates before aString. Case is ignored.

<= aString

Returns true if the receiver collates before aString, or is the same as aString. Case is ignored.

> aString

Returns true if the receiver collates after aString. Case is ignored.

>= aString

Returns true if the receiver collates after aString, or is the same as aString. Case is ignored.

match: aString

Treats the receiver as a pattern containing #'s and \*'s which are wild cards (# represents one character, \* represents a substring). Returns true if the receiver matches aString. Case is ignored.

sameAs: aString

Returns true if the receiver collates exactly with aString. Case is ignored.

asLowercase

Returns a String representing the receiver in all lowercase

asUppercase

Returns a String representing the receiver in all uppercase

asSymbol

Returns a Symbol whose characters are the characters of the receiver

Collection->SequenceableCollection->ArrayedCollection->CharacterArray->Symbol

Creation:

intern: aString

Returns an instance of a Symbol whose characters are those of aString

internCharacter: aCharacter

Returns an instance of a Symbol which consists of aCharacter

Operations:

Collection->MappedCollection

Creation:

Operations:

## **Stream**

Creation:

Operations:

next

Returns the next object accessible by the receiver

next: anInteger

Returns the next anInteger objects accessible by the receiver

nextMatchFor: anObject

Accesses the next object and returns true if it is equal to anObject

contents

Returns all of the objects in the collection accessed by the receiver.

nextPut: anObject

Stores anObject as the next object accessible by the receiver and returns anObject

nextPutAll: aCollection  
Store the elements in aCollection as the next objects accessible by the receiver and returns aCollection. Advances the position reference to the new object.

next: anInteger put: anObject  
Store anObject as the next anInteger number of objects accessible by the receiver and returns anObject. Advances the position reference to the new object.

atEnd  
Returns true if there are no more objects accessible by the receiver

do: aBlock  
Evaluate aBlock for each of the remaining objects accessible by the receiver

#### Stream->PositionableStream

##### Creation:

on: aCollection  
Returns an instance which streams over aCollection

on: aCollection from: firstIndex to: lastIndex  
Returns an instance which streams over a copy of a subcollection of aCollection from firstIndex to lastIndex

##### Operations:

isEmpty  
Returns true if the collection the receiver accesses has no elements

peek  
Returns the next object in the collection but does not increment the position reference

peekFor: anObject  
Does a peek, if the next object is equal to anObject, then returns true and increments the position reference, otherwise just returns false

upTo: anObject  
Returns a collection of the elements starting with the next object accessed by the receiver up to, but not including, anObject. If anObject is not an element of the remainder of the collection, then the entire remaining collection is returned.

position  
Returns the receiver's current position reference

position: anInteger  
Sets the receiver's position to anInteger. If anInteger exceeds the bounds of the collection, then an error is reported

reset  
Sets the receiver's position to the beginning of the collection

setToEnd  
Sets the receiver's position to the end of the collection

skip: anInteger  
Sets the receiver's position to the current position + anInteger

skipThrough: anObject  
Sets the receiver's position to be past the next occurrence of anObject. Returns true if anObject occurs in the collection

#### Stream->PositionableStream->ReadStream

##### Creation:

##### Operations:

#### Stream->PositionableStream->WriteStream

##### Creation:

##### Operations:

cr  
     Stores the carriage return as the next element of the receiver  
 crtab  
     Stores the carriage return and a single tab as the next elements of the receiver  
 crtab: anInteger  
     Stores a carriage return followed by anInteger number of tabs as the next elements of the receiver  
 space  
     Stores the space character as the next element of the receiver  
 tab  
     Stores the tab character as the next element of the receiver

Stream->ExternalStream

Creation:

Operations:

nextNumber: n  
     Returns a SmallInteger or LargePositiveInteger representing the next n bytes of the collection accessed by the receiver  
 nextNumber: n put: v  
     Stores v, which is a SmallInteger or LargePositiveInteger, as the next n bytes of the collection accessed by the receiver  
 nextString  
     Returns a String consisting of the next elements of the collection accessed by the receiver  
 nextStringPut: aString  
     Stores aString in the collection accessed by the receiver  
 padTo: bsize  
     Skips to the next boundary of bsize characters and returns the number of characters skipped  
 padTo: bsize put: aCharacter  
     Skips to the next boundary of bsize characters, writing aCharacter to each character skipped, and returns the number of characters skipped  
 padToNextWord  
     Skip to the next word (even) boundary and returns the number of characters skipped  
 padToNextWordPut: aCharacter  
     Skip to the next word (even) boundary, writing aCharacter to each character skipped, and returns the number of characters skipped  
 skipWords: nWords  
     Advance position reference nWords  
 wordPosition  
     Returns the current position in words  
 wordPosition: wp  
     Sets the position reference in words to wp

